UNIVERSITY OF CALIFORNIA, SAN DIEGO

Latent feature models for dyadic prediction

A dissertation submitted in partial satisfaction of the
requirements for the degree of Doctor of Philosophy

in

Computer Science

by

Aditya Krishna Menon

Committee in charge:

Professor Charles Elkan, Chair
Professor Gert Lanckriet
Professor Ramamohan Paturi
Professor Lawrence Saul
Professor Nuno Vasconcelos

2013

The Dissertation of Aditya Krishna Menon is approved and is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2013

DEDICATION

To my parents

EPIGRAPH

Nothing happens here,
Nothing gets done,
But you get to like it.

David McComb

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

It is an exaggeration to suggest that doing justice to everyone who helped me get to this stage of my PhD is a more difficult task than actually putting together this dissertation, but perhaps not by much. What follows is thus certainly incomplete, but I hope imputation is not too difficult.

My most obvious benefactor is my advisor, Charles Elkan. Without his guidance and support through the years, I wouldn't be writing this, and would be poorer for not having sharpened my critical thinking and writing skills (you'll have to excuse the current section when assessing the latter). I also owe debt to my undergraduate advisor, Sanjay Chawla, whose wide interests and relaxed style of research have helped shape the researcher I have tried to become.

I'd like to thank my colleagues in the AI group for their support through the years. Special mention must go to: Nakul Verma, for his intimidating knowledge, willingness to help out with matters technical and personal over whiteboards or tea, and his inimitable laconic pessimism with its resulting adages; Matus Telgarsky, for his reliable service as our resident machine learning oracle (even if we still don't understand bias terms), and his emotional support; Do-kyum Kim, for putting up with my constant interruptions, half-baked technical discussions, and unbaked advice sessions; and Ruixin Yang, for his perennial optimism, and acting as Nakul's counterweight.

I'd also like to thank my friends outside the AI group: Mohammad Al-Fares, for being an ideal role-model to an impressionable first year, and for opening my mind to Mozart and Miyazaki; Kourosh Derakshan, for proving a match to my eccentricities, and his ever positive demeanour; Gabe Doyle, for either being the finest actor in the world, or actually enjoying hours of discussion on the arts; Chaitanya Desai, for listening to my sermons on machine learning, and helping ground me through several times of uncertainty; Saurabh Prasad, for putting up with me when we first came to San Diego,

and for his contagiously relaxed attitude; Aravind Iyengar, for putting up with me when I first came to Mesa, and inspiringly impeccable conduct; and Sraavan Reddy, for his perceptive advice and support, and challenging my way of thinking.

I'd also like to thank friends from far across the sea, in particular Anthony Smith, for making me want to be a better researcher, and tireless humouring of my attempts at philosophy; and Jenny Zhu, for years of emotional support as I struggled to discern what everything means (I'm farther along than I started, but there's always further to go).

I would also of course like to thank my family for their support through the see-saw journey of the last five or so years. I hope our geographic distances can now revert to their natural state, as it was five years ago.

Finally, on a more abstract note, I'd like to think David McComb, Grant McLennan, and Stephen Morrissey for their (necessarily) unconditional support and help in the times where I most needed it.

Chapter 3, in part, contains material as it appears in "A Log-Linear Model with Latent Features for Dyadic Prediction", IEEE International Conference on Data Mining (ICDM), pages 364-373, 2010. Aditya Krishna Menon, Charles Elkan. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in part, contains material as it appears in "A Log-Linear Model with Latent Features for Dyadic Prediction", IEEE International Conference on Data Mining (ICDM), pages 364-373, 2010. Aditya Krishna Menon, Charles Elkan. The dissertation author was the primary investigator and author of this paper.

Chapter 5, in part, contains material as it appears in "Link prediction via matrix factorization", Proceedings of the 2011 European conference on Machine learning and knowledge discovery in databases - Volume Part II (ECML PKDD'11). Aditya Krishna Menon and Charles Elkan. The dissertation author was the primary investigator and author of this paper.

Chapter 6, in part, contains material as it appears in "Response prediction using collaborative filtering with hierarchies and side-information", Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '11). Aditya Krishna Menon, Krishna-Prasad Chitrapura, Sachin Garg, Deepak Agarwal, and Nagaraj Kota. 2011. The dissertation author was the primary investigator and author of this paper.

Chapter 7, in part, contains material as it appears in "Predicting labels for dyadic data", Data Mining and Knowledge Discovery, Volume 21, Number 2, pages 1384–5810, 2010. Aditya Krishna Menon and Charles Elkan. The dissertation author was the primary investigator and author of this paper.

Appendix A, in part, contains material as it appears in "Fast Algorithms for Approximating the Singular Value Decomposition", ACM Transactions on Knowledge Discovery from Data, Volume 5, Number 2, Article 13. 2011. Aditya Krishna Menon and Charles Elkan. The dissertation author was the primary investigator and author of this paper.

VITA

2006	B. Sc. Hons. in Computer Science University of Sydney, Australia
2009	M.S. in Computer Science University of California, San Diego
2013	Ph. D. in Computer Science University of California, San Diego

PUBLICATIONS

A Machine Learning Framework for Programming by Example. Aditya Krishna Menon, Omer Tamuz, Sumit Gulwani, Butler Lampson, and Adam Tauman Kalai. To Appear in ICML 2013.

Doubly Optimized Calibrated Support Vector Machine (DOC-SVM): an algorithm for Joint Optimization of Discrimination and Calibration. Xiaoqian Jiang, Aditya Krishna Menon, Shuang Wang, Jihoon Kim, and Lucila Ohno-Machado. In PLoS ONE, 2012.

Learning and Inference in Probabilistic Classifier Chains with Beam Search. Abhishek Kumar, Shankar Vembu, Aditya Krishna Menon, and Charles Elkan. In ECML-PKDD, 2012.

Predicting accurate probabilities with a ranking loss. Aditya Krishna Menon, Xiaoqian Jiang, Shankar Vembu, Charles Elkan, and Lucila Ohno-Machado. In ICML 2012.

Link prediction via matrix factorization. Aditya Krishna Menon, Charles Elkan. In Machine Learning and Knowledge Discovery In Databases - European Conference, ECML-PKDD, Proceedings Part II, 2011.

Response prediction using collaborative filtering with hierarchies and side-information. Aditya Krishna Menon, Krishna-Prasad Chitrapura, Sachin Garg, Deepak Agarwal, and Nagaraj Kota. In Knowledge Discovery and Data Mining (KDD), San Diego, California, 2011.

A log-linear model with latent features for dyadic prediction. Aditya Krishna Menon, Charles Elkan. In IEEE International Conference on Data Mining (ICDM), Sydney, Australia, 2010.

Predicting labels for dyadic data. Aditya Krishna Menon, Charles Elkan. In Data Mining and Knowledge Discovery: Special Issue on Papers from ECML-PKDD, Volume 21, Number 2, 2010.

Fast algorithms for approximating the singular value decomposition. Aditya Krishna Menon, Charles Elkan. In Transactions of Knowledge and Data Discovery: Special Issue on Large-Scale Data Mining (TKDD-LDMTA), 2010.

ABSTRACT OF THE DISSERTATION

Latent feature models for dyadic prediction

by

Aditya Krishna Menon

Doctor of Philosophy in Computer Science

University of California, San Diego, 2013

Professor Charles Elkan, Chair

Following the Netflix prize, the collaborative filtering problem has gained significant attention within machine learning, spawning novel models and theoretical analyses. In parallel, the growth of social media has driven research in link prediction, with the aim of determining whether two individuals in a network are likely to know each other. Both problems involve the prediction of label (star ratings or friendship) between a pair of entities (user-movie or user-user). We call this general problem dyadic prediction. The problem arises in several other guises: predicting student responses to test questions, military disputes between nations, and clickthrough rates of webpages on ads, to name a

few.

In general, each such domain employs a markedly different approach, obscuring the underlying similarity of the problems being solved. This dissertation aims to explore the use of a single general method, based on latent feature modelling, for generic dyadic prediction problems. To this end, we make three contributions. First, we propose a generic *framework* with which to analyze dyadic prediction problems. This lets one reason about seemingly disparate problems in a unified manner. Second, we propose a *model* based on the log-linear framework, which is applicable to each of the aforementioned problems. The model learns *latent features* from dyadic data, and estimates a probability distribution over labels. Third, we systematically explore *applications* of our latent feature model to domains such as collaborative filtering, link prediction, and clickthrough rate prediction. In all cases, we show performance comparable or superior to existing state-of-the-art methods. For clickthrough rate prediction, ours represents the first application of latent feature modelling to the problem, demonstrating the value in a single framework with which to reason about these problems. We also show that latent feature modelling is scalable to datasets with hundreds of millions of observations on a single machine (the Netflix prize dataset), and hundreds of billions of observations on a small cluster (Yahoo! ad click data).

We conclude with a discussion of future research directions, including transferring information from one network to another, and adapting to domains with extreme label sparsity.

Chapter 1

Introduction

The focus of this dissertation is the *dyadic prediction* framework. Several special cases of the framework, such as collaborative filtering and link prediction, have captured the attention of much machine learning research over recent years. However, the general framework itself is not as well-known as, say, supervised learning, despite being arguably as important. In this chapter, we will motivate the importance of the dyadic prediction problem through some examples of this framework. We then describe the contributions this dissertation makes in dyadic prediction, and give an outline of the structure of the dissertation.

1.1 Recap: the value of supervised learning

A characteristic of machine learning is the existence of precisely defined mathematical frameworks that capture a range of seemingly disparate problems. For example, a core component of machine learning is supervised learning, which comprises the problem of predicting the value of some variable(s) of interest (the *label*) based on certain other variables (the *features*). This general framework captures for example the following real-world tasks:

- **Will a person donate?** Many non-profit companies rely on public donations for

funding. To this end, solicitation campaigns are generally run, which involve for example door-to-door pledge drives. Ideally, one would like to only contact individuals who are likely to donate a substantial amount: for a given amount of time and effort, this maximizes the return for the campaign. Therefore, one would like to estimate the expected donation of an individual, based on certain characteristics e.g. whether they have donated in the past, their income level, *et cetera*. This formed the basis of the KDDCup '98 challenge[1], where machine learning methods proved successful (Zadrozny and Elkan, 2001).

- **Will a person default?** Financial institutions issuing loans to individuals necessarily impose different rules on conditions based on whether or not the individual in question is likely to be able to pay back the loan. One measure of this reliability is whether or not the individual is likely to go bankrupt in the near future. Credit-scoring agencies seek to estimate this probability based on historical information about the individuals' finances. Statistical models for this problem were explored in (Foster and Stine, 2004).

- **How many will turn up?** It is standard practice for commercial airlines to overbook flights, because it is likely that one or more passengers will be "no-shows". An important unknown to be estimated is the expected number of "no-shows" for a particular flight. Intuitively, one would like to do this based on historical flight data, and information about the passengers on the flight e.g. whether they are frequent fliers, their boarding country, *et cetera*. A machine learning model for this problem was proposed in (Lawrence et al., 2003).

- **Who's going to win?** In most forms of contest, there is a belief that the winner can be determined beforehand. This serves mostly an intellectual curiosity, and

[1] http://kdd.ics.uci.edu/databases/kddcup98/kddcup98.html

sometimes a financial one. A concrete example of such a problem is to predict the likely winner of a political election, based on historical polling data. Statistical approaches to this problem were popularized by (Silver, 2012).

Stepping away from the formalism a practitioner of machine learning is accustomed to, it is at first glance remarkable that a single method - to pick one of many contenders, logistic regression - can be used in each of these different problems. While problem- and data-specific engineering is no doubt necessary, as most of the above works attest, it is scientifically important that there is a well-studied class of models that may be used to provide an effective solution to each of these problems, and the innumerable others that can be cast in the framework of supervised learning. This framework provides a well-defined, coherent foundation on which satisfactory solutions to any of these problems may be discovered.

1.2 From supervised learning to dyadic prediction

Motivated by the generality of the supervised learning framework, we can hope to apply its techniques to any of the problems that have arisen with the proliferation of large amounts of data. However, as we shall see below, many such problems do not fit exactly into the supervised learning framework, and are instead instances of *dyadic* prediction. We explain the need for this framework with some motivating problems below.

1.2.1 Collaborative filtering and friends

Consider now a superficially similar list of problems to the one above:

- **What movies does a user like?** Among the many changes fostered by the internet is the unprecedented amount of media available to the average consumer. In fact, there are so many options that it is often difficult for someone to decide *what* to

consume. This raises a question: if we know what users have liked in the past, can we predict what else they might like? The practical importance of this problem cannot be underestimated: with businesses shifting online, recommending items to users means a greater opportunity to extract revenue. This problem is referred to as the *movie recommendation* problem, or sometimes as *collaborative filtering*[2].

- **Who knows who?** For social networking websites like Facebook and LinkedIn, complete knowledge of the social graph is attractive for many reasons, primary among them the monetary possibilities available from e.g. targetted advertising. Of course, most users are not inclined to actively add every person they know. The link prediction problem asks to predict the existence of connections between pairs of users, based on the partially observed graph.

- **Will a user click on an ad?** In computational advertising, the manager of a website needs to pick which ads to display from a large pool of candidates. Each candidate ad has associated a *bid*, being the amount the corresponding advertiser will pay the website owner when some event occurs e.g. the user clicks on the ad. It is of fundamental interest to predict the probability of a user clicking on an ad, so as to determine which ads will yield the most revenue.

- **Do two proteins interact?** In bioinformatics, knowing whether or not a pair of proteins interact helps guide models to explain biological phenomena. In principle, this knowledge can be obtained by a laboratory experiment. The downside of this is that such experiments are generally expensive to run, in terms of time and cost. Ideally, then, one would like a cheap way to predict if two proteins will interact, preferably as a probability. Experiments can then be run based on this prediction

[2]Strictly, collaborative filtering refers to a particular approach to movie recommendation. However, its success has made it not only the *de facto* approach, but also synonymous with the problem it attempts to solve.

alone.

- **How will a politician vote?** In political science, a problem that has received much attention is predicting how a senator is likely to vote on a future bill, given the past history of votes in the house. This can be useful on a small scale, to for example predict whether an important bill will get passed, but can also fit into a larger model that attempts to predict the outcome of an election.

These problems are no less interesting or practically important than the ones on the previous list. But these problems are *not*, in general, suitable fits for the supervised learning framework. To see why, recall that a basic requirement of casting a problem as one of supervised learning is to come up with a suitable feature representation. Consider the first problem, movie recommendation. A natural choice for what constitutes an example is a single rating in the database. What would constitute an appropriate feature representation for this example? The only available information is the set of past ratings for a user, which tells us something about their tastes. Apart from this, nothing is known about the user: her age, occupation, *et cetera* are unspecified. In a sense, then, our feature space and label space coincide: it appears that predicting the label for a new example is like predicting the features for the training set. This difficulty in modelling the problem as supervised learning is shared by the other problems as well.

Having shown that supervised learning is not the best way to think of these problems, we seek a framework that *does* capture them.

1.2.2 Dyadic prediction: an informal overview

Observe that in each case, we have available labels that arise from the interaction of a *pair* of entities. Indeed, in each problem, our goal is to make a prediction about the interaction between two entities, based on past history of interactions between different

entities. This interaction could be a number indicating how much one entity likes another, a categorical denoting the type of relationship the two entities have, *et cetera*. We refer to this general problem called *dyadic prediction*.

Slightly more formally, each training example consists of two entities, r and c, for which we observe a label y. Our training set therefore consists of $\{((r_i, c_i), y_i)\}_{i=1}^{n}$, and each pair (r_i, c_i) is called a *dyad*. The goal is now to predict the label for a new entity (r', c'). This seems like supervised learning, but with a crucial difference: the r's and c's are possibly only *unique identifiers*.

The movie recommendation problem is most well-known application of dyadic prediction. Each dyad here is a (user, movie) pair, and the label is the user's rating of the movie. In link prediction, each dyad here is a pair of nodes, and the associated label is some indication of the relationship between the two nodes e.g. { Friends, Not friends }. In response prediction, the dyad consists of a user and an ad, and the label is whether the user clicks on the ad, and so on.

The above makes clear that dyadic prediction is a flexible framework, capturing many important problems arising in the real world. We see that there are broad ways that we can get different incarnations of a dyadic prediction problem: the nature of the objects r, c, and the nature of the labels y. The r, c could be (i) unique identifiers, as in the case of movie recommendation, (ii) predictive features or *side-information*, as in typically the case in computational advertising, or (iii) both. Further, the objects could belong to different spaces, as is the case in most of the above examples, or they could be in the same space, such as users in a social network. Turning to the labels, they could be ordinal, such as 1-5 star ratings, or nominal, such as categories { Viewed, Purchased, Returned }.

1.3 Questions to be addressed

Of the dyadic prediction problems introduced earlier, perhaps the most well known is that of movie recommendation. Significant interest in the problem was generated by the Netflix prize (Netflix, 2006), where $1M was offered for anyone to beat Netflix's internal movie recommendation system. Machine learning models that attempt to solve the movie recommendation problem problem are known as *recommender systems*. While it is impossible to fully account for the idiosyncrasies of a person's taste, current state-of-the-art models can nonetheless give very accurate recommendations to users. The most successful class of recommender models are based on the *collaborative filtering* idea, which uses a simple intuition: similar users tend to like similar items, where "similar" has a well-defined mathematical meaning.

Given the success of collaborative filtering models for movie recommendation, one might expect similar models to be employed to solve the other dyadic prediction problems as well. In fact, most other problems employ quite different models than those popularized in the Netflix prize. For example, for recommending friends in a social network, popular methods involve applying some simple scoring functions for a pair of users, such as the number of friends they have in common. For estimating the clickthrough rates of an ad on a webpage, a popular method is to collect features for the ad and webpage, and reduce the problem to one of supervised learning. In general, we find there is no single consistent approach to modelling these problems: they are largely studied independently, and treated as distinct. This is not ideal, as it means that advances in one problem are isolated to that problem, and do not percolate to the other related problems. (While there may be good reasons to favour one type of model over another for a problem, it is scientifically important to at least be aware of alternatives.)

Based on this observation, this dissertation is concerned with addressing the

following questions related to dyadic prediction.

- *Framework.* The first question is simply: can we describe a general framework capable of capturing all the motivating problems introduced above, and other related problems? This formalism helps identify how current and future problems may be addressed using techniques used in a seemingly disparate problem that also falls under the dyadic prediction framework.

- *Modelling.* In supervised learning, we have a set of general purpose models that can address most problems in that framework: logistic regression, SVMs, *et cetera*. We saw above that the same appears to not be true of dyadic prediction. But this raises the question of whether such a model is *possible*, and if so, what it would like.

- *Applications.* The above question asks whether a suitable generic dyadic prediction model exists in principle. Equally important is the question: would such a model be useful in practice? In particular, there is a need to test any such candidate model on a range of dyadic prediction tasks. Two specific questions we can ask in this regards are:

  - How does the model compare to other state-of-the-art methods in these domains?, and
  - Can the model adapt to challenges specific to a particular problem or domain? If so, what is the procedure for doing so?

1.4 Contributions of this dissertation

This dissertation aims to provide concrete answers to each of the questions raised above. In particular, the contributions of this dissertation are as follows.

- *Framework.* We show how to unify several important problems that have been studied largely separately in the respective literatures. In particular, we will focus on:
  - Canonical dyadic prediction problems, such as collaborative filtering and link prediction. While these problems have engendered rich literatures, they have traditionally been treated as being distinct.
  - Problems generally attacked by a reduction to supervised learning, such as clickthrough rate prediction. The dyadic perspective, while sometimes acknowledged, has not led to a systematic study of the application of different models popular in other dyadic problems, like collaborative filtering.
  - Problems generally interpreted in a different way, such as within-network classification. The dyadic perspective to these problems has not to our knowledge been previously explored.
- *Modelling.*. Once defining the general dyadic prediction problem, we present a log-linear model, LFL. This is a flexible approach to dyadic prediction based on the idea of latent feature modelling. There are several salient properties of the model, such as:
  - It is applicable to a range of dyadic prediction problems, with a range of input and label types. To our knowledge, it is the first such method of its kind.
  - The ability to exploit both latent and explicit features, thus addressing for example the cold-start problem in collaborative filtering.
  - The modelling of a probability distribution over labels, which allows it to be used as a component in a larger decision making system.

- *Applications.* Our final contribution is to demonstrate that the LFL model, which is an instance of a latent feature model, enjoys good performance on a range of real-world problems and datasets. In particular, we show competitive or superior performance to state-of-the-art methods for the following dyadic prediction tasks:
  - Predicting student responses to test questions (Section 3.8.3).
  - Movie recommendation problems, such as the Netflix dataset (Section 4.8.2).
  - Clickthrough rate prediction on real Yahoo! web-traffic data (Section 6.7).
  - Predicting party affiliations of politicians based on voting records (Section 7.5.1).

1.5 Organization of this dissertation

We give an overview of the material presented in this dissertation.

First, we begin by giving a more formal description and overview of dyadic prediction in Chapter 2. We discuss the flexibility of the framework, and show how it strictly generalizes several important problems, such as collaborative filtering, matrix completion, and link prediction. This chapter includes a survey of related work in several related research areas, such as collaborative filtering, item response theory, and social network analysis.

Next, in Chapter 3, we present a generic model for dyadic prediction, the latent feature log-linear or LFL model. This model will form the basis for much of the approaches considered in subsequent chapters. We analyze the model in detail, and discuss several ways by which it may extended. We present experiments that show the richness of the model, and its basic ability to successfully model dyadic prediction tasks.

We then proceed to consider applications of the latent feature model, generally based on variants of LFL, to various instantiations of dyadic prediction. Chapter 4 begins

this focus with an application of the LFL model to collaborative filtering, arguably the canonical instantiation of dyadic prediction. Next, Chapter 5 studies the related problem of link prediction in graphs. Finally, Chapter 6 looks at an application of latent feature modelling to the response prediction problem in collaborative filtering.

The last technical contribution is Chapter 7, which studies a problem we call dyadic label prediction. This is a generalization of the within-network classification problem, wherein one needs to complete the labellings on a partially labelled graph.

We conclude the dissertation in Chapter 8 with a discussion of future research directions in dyadic prediction.

Chapter 2

Overview of Dyadic Prediction

In this chapter, we formally describe the dyadic prediction problem, and show how our formalism can capture the seemingly disparate examples of the framework outlined in Chapter 1. We then point out its relationship to other established frameworks in statistics and machine learning.

2.1 A formal definition of dyadic prediction

In dyadic prediction, our training set comprises a number of pairs or *dyads*, represented by a pair of identifiers (i, j), with a corresponding label y_{ij}. The information available about each dyad member is at least a unique identifier, and possibly some additional features or *side-information*. Informally, our goal is to learn some mapping that lets us predict the label for an unobserved dyad (i', j').

To formalize this goal, let $I, J \subseteq \mathbb{Z}_+$ be two sets of non-negative integers, $P := I \times J$ be the set of pairs of these integers, and $Z_1 \subseteq \mathbb{R}^{D_1}, Z_2 \subseteq \mathbb{R}^{D_2}$. Further, let $\mathscr{X} \subseteq P \times Z_1 \times Z_2$, and $\mathscr{Y}$ be some arbitrary set. We are provided a training set $\mathscr{T} = \{(x^{(t)}, y^{(t)})\}_{t=1}^{M}$ where each $(x^{(t)}, y^{(t)})$ is a draw, possibly without replacement, from some distribution $p(x, y)$. In dyadic prediction, we wish to learn a function $f : \mathscr{X} \to \mathscr{Y}$ that generalizes well with respect to some loss function $\ell : \mathscr{Y} \times \mathscr{Y} \to \mathbb{R}_+$ and distribution $q(x, y)$, i.e. we

want

$$\mathbb{E}_{(x,y)\sim q(x,y)}\left[\ell(f(x),y)\right]$$

to be small.

The above formalism is fairly general, so we briefly explain its key components. The set P defines all possible dyads by means of positive integer identifiers, with I and J being the identifiers for the individual dyad members. The sets Z_1, Z_2 denote the *side-information* for the individual dyad members and the dyad pair, respectively. The set $\mathscr{X}$ consists of the union of the dyads and their side-information, while $\mathscr{Y}$ is the space of possible labels. The function f returns an estimate of the label for a given dyad, and the loss ℓ measures how good this estimate is.

2.2 Example instantiations of the framework

We now give examples of how existing dyadic prediction problems can be interpreted in our framework.

2.2.1 Collaborative filtering

Arguably the canonical instantiation of dyadic prediction from a machine learning perspective is where the dyads refer to (user, item) pairs, and the labels are a user's numeric rating of an item, e.g. on a scale of 1 to 5. The task of predicting the ratings of unobserved (user, item) pairs is an example of *collaborative filtering* (Goldberg et al., 1992), where the goal is to recommend to users new item they might like, e.g. items we predict the user will rate ≥ 4 stars.

Collaborative filtering is a special case of dyadic prediction where $P = [M] \times [N]$, for some $M, N \in \mathbb{Z}_+$ denoting the number of users and movies respectively, and $\mathscr{Y}$ a space of ratings or preferences that a user can assign to an item. The space $\mathscr{Y}$ is typically,

but not necessarily, the set $[R] = \{1, 2, \ldots, R\}$ for some appropriate integer R, denoting star-ratings that the user gives to a movie. (A common setting is $R = 5$.) It is possible to have side-information in the form of the age of the user, genre of the movie, format that the user watched the movie in, and so on. For a candidate predictor $f : \mathscr{X} \to \mathscr{Y}$, a common choice of loss ℓ is squared error,

$$\ell(f(x), y) = (f(x) - y)^2,$$

which is precisely the error measure used in the Netflix prize.

Collaborative filtering generally fits into the *transductive learning* framework (Gammerman et al., 1998): we have some fixed training set $\mathscr{T}$ of the entries with a single label for each x value, assuming a user can rate a movie at most once. So, this training set may be considered a sample without replacement from some underlying probability distribution $p(x, y)$. We wish to predict the ratings for $\mathscr{T}' = \mathscr{X} \setminus \mathscr{T}$, i.e. all unobserved pairs, so that from these the ones with highest predicted ratings can be recommended.

2.2.2 Link prediction

The (unweighted) link prediction problem (Liben-Nowell and Kleinberg, 2003) is where we have as input a partially observed graph, in the sense that for some pairs of nodes, it is unknown whether or not there exists an edge between them. (Note that this is distinct from knowing that there is no edge between the node pair.) The goal is simply to predict the link status for all such node pairs. Applications of this problem include recommending friends in a social network (Liben-Nowell and Kleinberg, 2003), and predicting interactions between pairs of proteins (Szilágyi et al., 2005). Link prediction is a sub-problem in the bigger field of social network analysis (SNA). As the name suggests, SNA aims to derive some understanding from a social network (Wasserman and Faust,

1994). Classically, this involves making qualitative deductions or inferences from the network, such as determining the reasons why people decide to become friends or not.

For a graph over a set of nodes $\mathscr{V}$, link prediction is a dyadic prediction problem where $P = [|\mathscr{V}|] \times [|\mathscr{V}|]$, and $\mathscr{Y}$ is $\{0,1\}$ in the unweighted case. Unlike collaborative filtering, there is a constraint that both dyad members belong to the same space, i.e. nodes in the graph, which intuitively necessitates some modification in the modelling approach. It is generally the case that there is some side-information present for the dyad members, and possibly for the edges as well. For example, in a social network, we may have features specific to each user (age, gender, *et cetera*), and features specific to each pair of user (the time that they initiated a connection, the number of times they have communicated, *et cetera*). In terms of loss functions, it is common to employ some form of *ranking* metric, such as precision at K, or the area under the ROC curve.

In more general incarnations, link prediction is essentially identical to dyadic prediction; we discuss this in Section 2.4.3.

2.2.3 Response prediction

Response prediction (Agarwal, 2008a; Mao, 2010) is a fundamental problem in computational advertising, where we wish to estimate the clickthrough rate of an ad if it is shown on a specific webpage. This may be seen as a type of dyadic prediction where the dyads are (webpage, ad) pairs. Here, we again have $P = [M] \times [N]$ for some fixed M, N denoting the number of webpages and ads respectively. The label set $\mathscr{Y} = \{0,1\}$, denoting whether or not the ad is clicked on the page. Note that unlike the previously described problems, here, we can expect a dyad to be associated with multiple labels, corresponding to different times and contexts in which the display happens. (This can be seen as an artifact of not explicitly treating time as an entity.) These contexts are captured by the side-information for the dyad. It is generally desired to accurately model

the clickthrough rates as probability estimates (Sculley, 2010), and so a popular choice of loss function is the log-loss, or Bernoulli log-likelihood:

$$\ell(f(x), y) = -y \log f(x) - (1-y) \log(1 - f(x)).$$

2.2.4 Item response theory

Item response theory (IRT) is concerned with modelling responses of set of individuals to test questions (Hambleton et al., 1991). The aim of such analysis is to help with tasks such as designing future tests, comparing results across different test sets, and so on (Hambleton et al., 1991). This is a dyadic modelling problem, where each dyad is a (person, question) pair. It is common for there to be side-information that describes the questions, for example in terms of their subject areas. Unlike the above problems, the classical focus in IRT is less on prediction, and more on modelling the observed data well to perform hypothesis testing. Therefore, it is not customary to think of there being a loss function we wish to optimize. Nonetheless, after collaborative filtering, IRT is arguably the field with the biggest range of dyadic models.

2.3 Generality of the framework

We highlight some salient points of the above setup, which give a sense of the generality of the dyadic prediction problem. We separate our discussion into the nature of the train and test distributions, the label space, and the side-information.

2.3.1 Train and test distributions

Perhaps the biggest difference between dyadic prediction and supervised learning setting is that the training data may *not* be independent and identically distributed (iid). This is of course true in the case of sampling without replacement, as is the case in

settings such as collaborative filtering. Even if we sample with replacement, however, both the identical distribution and independence conditions may fail. First, it will not be true in general that the distributions $\Pr[y|x=(i,j)]$ and $\Pr[y|x=(i,j')]$ will be identical. In collaborative filtering for example, we do not expect the rating distribution for two different movies j and j' to be the same for a given user i. Second, it will also not be true in general that the samples $((i,j),y)$ and $((i,j'),y)$ are independent. Taking the collaborative filtering example again, we expect the ratings the same user assigns to two movies to be a product of the user's taste, and so knowledge of one rating gives us more information about the other. This is related to the fact that in multilabel learning problems, the individual elements of a label vector are assumed to contain information about one another; we discuss the connection to dyadic prediction in Section 2.4.1.

As a further complication, the train and test distributions often do not match, that is $p(x,y) \neq q(x,y)$. This is classically known as the sample selection bias problem (Heckman, 1979), and it tends to be the norm in these datasets. The sample selection bias problem can be understood by imagining the existence of an additional *selector* variable $s \in \{0,1\}$ (Zadrozny, 2004). We assume that the relationship between training and test distributions is $p(x,y) \propto \Pr[s=1|x,y]q(x,y)$, i.e. that the training samples as drawn from $\Pr[x,y|s=1]$, but the test examples are drawn from $\Pr[x,y]$. The nature of $\Pr[s=1|x,y]$ quantifies the nature of the selection bias, and suggests strategies to cope with the same. For example, if it is true that $\Pr[s=1|x,y]=\pi_y$, so that it is only the base rate of the labels that changes in train and test distribution, one can apply correction schemes (given knowledge about the true base rate $\Pr[y]$). Unfortunately, in many dyadic prediction problems, it is likely the case that s depends on both x and y. For example, in datasets that arise in collaborative filtering, users tend to provide ratings for movies that they enjoy (Marlin et al., 2007). One can also imagine that users tend to provide ratings for movies that they watched recently. So, one plausible assumption as to the nature of

dyad selection is

$$\Pr[s = 1|x, y] \propto \exp\left(\frac{y}{T(x)}\right),$$

where $T(x)$ denotes the time since some starting point that the user watched the movie. As this function depends on both x and y, the train and test distributions will not match. Intuitively, the unrated movies will not only tend to have lower ratings, but also comprise older movies (as these are more likely to have been seen a long time ago). (We discuss the issue more in Appendix B.4.)

As a final note, we observe that the *cold-start* setting is implicit in our setup. This refers to the problem of making a prediction for a dyad (i, j) where either i or j does not appear in the training set. This problem has long been a concern in collaborative filtering (Schein et al., 2002), where it is common to have to make predictions for a user who has just joined a system, and so does not have any historical ratings from which to infer her tastes.

2.3.2 Label space

The label set $\mathscr{Y}$ may be varied along at least two dimensions. First, the nature of the ordering of the elements in $\mathscr{Y}$ is flexible: most generally, it could be an unordered set, e.g. $\{$ Friend, Colleague, Family Member $\}$. It could be on an ordinal scale where there is some partial order over the elements, e.g. $\{$ Dissatisfied, Neutral, Satisfied $\}$, or be a numeric or cardinal set where the difference between outcomes has a specific numeric value e.g. $\mathbb{R}$ or $\{1, 2, \ldots, 5\}$. The choice of appropriate loss function of course depends on the nature of the elements of $\mathscr{Y}$.

Second, the dimensionality of $\mathscr{Y}$ is not constrained to be unity. This means that each dyad can possess *multiple* labels, analogous to multilabel learning problems in supervised learning. In link prediction, this setup is also known as a *multirelational* problem (Zheleva et al., 2010). As an example, if our dyads are (user, page) with the

label being a tag that a user applies to a page, the space $\mathscr{Y}$ could be the powerset $\mathscr{P}(\{\text{Politics, Sports, Entertainment,}\ldots\})$.

2.3.3 Side-information

Side-information is optional in our framework, because we can set $Z_1, Z_2 = \varnothing$, the empty set. In this setting, the only information available is the dyad identities I. In such cases, we need to perform "feature-less" learning, and so the problem looks like an unfamiliar departure from standard supervised learning. We discuss the connection between the frameworks in more detail in Section 2.4.1.

When side-information is present, note that we only defined two sets, one for individual dyad member features, another for dyad (interaction) features. These features can be of different dimensionality, because e.g. we may have a rich representation for each user and movie in a collaborative filtering problem, but for their interaction only know the time that the user rated the movie. Note that the features for the individual dyad members may themselves be of different length, because we can just pad the features in the smaller space with zeros. It should be clear that this has the same effect as explicitly considering two separate spaces.

Finally, note that the same dyad (i, j) may occur multiple times in the train and test sets, each time possessing a different label and/or side-information. To see an example of how this arises, consider e.g. the problem of predicting if an ad is clicked on a webpage. Here, we can think of each dyad as occurring multiple times when the particular (page, ad) pair is seen by a particular user[1]. Each such event may clearly have a different label, since different users may choose to click on the ad or not. Further, the side-information for the dyad may include features describing the user. In this case, the

[1]Conceptually, we can think of the interaction as occurring due to the *triplet* (page, ad, user). However, it may be the case that we do not have a reliable mechanism to associate a unique identifier to each user. In this case, since the identity of the user is absent, we effectively collapse the third dimension into multiple entries for a dyadic problem.

features across different events will again be different.

2.4 Relationship to existing frameworks

In this section, we point out connections and differences between dyadic prediction and other frameworks in statistics and machine learning.

2.4.1 Supervised learning

Ostensibly, the formalism of Section 2.1 is nothing more than a special case of supervised learning. Recall that in supervised learning, we have some input space $\mathscr{X}$ (generally $\mathbb{R}^D$), a label space $\mathscr{Y}$ (generally a subset of $\mathbb{R}$), and a training set comprising M samples from some distribution $p(x,y)$ over $\mathscr{X} \times \mathscr{Y}$. Our goal is to learn a function $f : \mathscr{X} \to \mathscr{Y}$ that generalizes well with respect to some loss $\ell : \mathscr{Y} \times \mathscr{Y} \to \mathbb{R}_+$, in the sense of $\mathbb{E}_{(x,y)\sim p(x,y)}\ell(f(x),y)$ being small. The dyadic prediction problem as defined in 2.1 corresponds to a specific assumption on the nature of $\mathscr{X}$: namely, it says that our input space comprises pairs of non-negative integers, and optionally (more familiar) feature representations for the dyad members and the dyad.

Of course, a mathematical connection between the problems does not mean that models from supervised learning can be applied as-is to dyadic prediction. The biggest reason is that, in the case where there is no side-information, the learning problem is "feature less": the only information we have about a label is the identity of the dyad members that it corresponds to. (Another reason is that, as mentioned before, the data may not satisfy the iid assumption.) This certainly is motivation to disregard the connection between the two problems, and study different solutions to dyadic prediction. But in fact, it is possible to *construct* features from the training data, and then apply supervised learning techniques, as we discuss in Section 2.5.2.

It is also possible to interpret dyadic prediction as a more specific type of super-

vised learning problem, namely, multilabel learning (Tsoumakas and Katakis, 2007). In particular, we can imagine that each dyad member i has a J dimensional label vector, $(y_{i1}, y_{i2}, ..., y_{iJ})$, representing the labels of its interaction with all possible dyad members in J. The dyadic prediction problem arises when this label vector is partially observed: the goal of predicting the labels (i, j) for any $i \in I, j \in J$ is then nothing but predicting these missing labels. Indeed, one can represent the problem as being one of filling in the entries of an incompletely observed matrix $X \in \mathscr{Y}^{|I| \times |J|}$. This leads to an equivalent interpretation of the problem comprising matrix completion, which we now discuss.

2.4.2 Matrix completion

Matrix completion (Johnson, 1990) is the following task. Let $X \in \mathscr{Y}^{M \times N}$ be some matrix over a set $\mathscr{Y}$. At training time, only some subset $\mathscr{O} \subseteq [M] \times [N]$ of entries in X are known, and the rest are missing. The goal at test time is to fill in the missing entries, namely to predict X_{ij} for $(i, j) \notin \mathscr{O}$. Clearly, this task is impossible unless we impose some structure on the underlying matrix X. When $\mathscr{Y} = \mathbb{R}$ and X is low-rank, a remarkable result of (Candès and Recht, 2009) shows that for a randomly chosen $\mathscr{O}$ of modest size, with high probability we can reconstruct X *exactly*. Even in the case of X being low-rank plus (sufficiently low) noise, strong reconstruction guarantees are possible (Keshavan et al., 2009).

Matrix completion is an instance of transductive dyadic prediction, where our underlying space $\mathscr{X} = [M] \times [N]$, and there is no side-information. The training set $\mathscr{T}$ comprises the observed pairs $(i, j) \in \mathscr{O}$ and their corresponding labels X_{ij}. The general form of dyadic prediction can be seen as a richer version of the matrix completion problem, for the following reasons:

- Most work in matrix completion implicitly assumes that the data is numeric, so that $\mathscr{Y} = \mathbb{R}$. This is true for example of all guarantees we are aware of for approximate

or exact recovery. This does not capture settings where the outcomes are nominal, or multidimensional.

- As a consequence of the above, matrix completion focusses on loss functions applicable to numerical outcomes, most commonly squared loss. Dyadic prediction can be defined for other loss functions, such as for example *ranking* losses, which we discuss in Section 5.3. (Of course, this shift away from standard losses is potentially at the cost of having generalization bounds.)

- In matrix completion, it is assumed that the only data available is the set of indices and the corresponding labels. Side-information in the form of features for rows, columns and entries is not considered. This information potentially complements the raw matrix itself, and thus may be essential for good predictive performance, especially in regimes of high sparsity ($|\mathcal{O}| \ll M \cdot N$).

- In dyadic prediction, each dyad may have multiple labels, not just a single one. This corresponds to the data being a *tensor* rather than a matrix. While conceptually this extension is not difficult, computationally, the mature techniques in matrix completion rely on operations such as eigendecompositions, which are not as well-studied for higher order tensors as they are for matrices.

- In dyadic prediction, a dyad may occur several times in the training and/or test set, each time with multiple labels. That is, instead of a matrix, the appropriate generic structure for dyadic data is a table, where each cell comprises multiple entries.

2.4.3 Weighted link prediction

Matrix completion is closely related to weighted link prediction, which is defined as follows. We assume the existence of a graph over some set of nodes $\mathcal{V}$. For some set $\mathcal{Y}$, let $L \in \mathcal{Y}^{|\mathcal{V}|\times|\mathcal{V}|}$ denote the label on an edge between any two pairs of nodes. At

training time, we are given the labels for some set $\mathcal{O} \subseteq |\mathcal{V}| \times |\mathcal{V}|$ of pairs of nodes. Our goal is to predict L_{ij} for $(i, j) \notin \mathcal{O}$. Clearly, this is equivalent to (generalized) matrix completion of the matrix L. Similarly, matrix completion of the matrix X can be cast as a weighted link prediction problem where we construct a node for each row and column, and set the label matrix $L = X$. Given the relationship between matrix completion and dyadic prediction discussed earlier, we see that dyadic prediction is thus closely related to link prediction also.

For much the same reasons as outlined for matrix completion, we consider dyadic prediction to be a more general task than link prediction. In particular, most instantiations of link prediction, the underlying graph is undirected, unweighted, and possesses only a single edge between a pair of nodes. This said, it is certainly correct to consider dyadic prediction as a link prediction problem on an arbitrary multigraph, potentially with side-information for the nodes and/or edges.

2.4.4 Random effects models and ANOVA

Random effects models arise in scenarios where our data possesses a number (often two) of categorical variables. This is sometimes referred to as "grouped data". We can think of our dataset as comprising instances (i, j, y), where $i \in [M], j \in [N]$ encode the group identities for which the corresponding observation (or label) is y. A simplified definition of a random effects model is one where there is a parameter associated with each group, and this parameter is assumed to be a random variable. (A more detailed description is provided in Appendix C.) That is, we assume there are parameters $\alpha \in \mathbb{R}^M, \beta \in \mathbb{R}^N$ which have some probability distribution imposed on them. A classical instantiation of a random effects model is the analysis of variance (ANOVA)

framework (Maxwell and Delaney, 2003). A two-way ANOVA model is

$$y_{ij} = \mu + \alpha_i + \beta_j + \varepsilon_{ij} : \alpha_i \sim \mathcal{N}(0, \sigma_\alpha^2), \beta_j \sim \mathcal{N}(0, \sigma_\beta^2), \varepsilon_{ij} \sim \mathcal{N}(0, \sigma_\varepsilon^2), \qquad (2.1)$$

where we write y_{ij} for the label corresponding to the interaction of groups (i, j). Such a model may be applied to dyadic prediction tasks, where we think of the identities of the dyad members as being those of different groups. Indeed, in collaborative filtering problems, this model is identical to the bias-only approach (Feuerverger et al., 2012). The result is a random effects model, as it keeps a separate parameter for each group which is assumed to be subject to random variation.

Another connection between ANOVA and dyadic prediction exists. ANOVA can be seen as an instance of linear regression where all covariates are categorical (Montgomery, 2000, Chapter 3.9), and are encoded with a one-hot mechanism, sometimes referred to as "dummy variables" (Suits, 1957). Specifically, if there are M groups each with N observations, then the two-way model of Equation 2.1 is equivalent to

$$y_{ij} = \mu + \begin{bmatrix} \alpha \\ \beta \end{bmatrix}^T \begin{bmatrix} e_i \\ e_j \end{bmatrix} + \varepsilon_{ij},$$

where $e_i \in \{0,1\}^M$ and $e_j \in \{0,1\}^N$ are bitvector encodings of the group and the observation respectively. We will see in Section 2.6.3 that an identical encoding scheme underlies a generic approach to dyadic prediction.

2.5 Overview of dyadic prediction models

We survey some prominent approaches to dyadic prediction. An exhaustive survey is impossible due to the myriad number of fields where dyadic problems arise, and the various approaches within these fields. Note that not all approaches have been

applied in all fields. For example, in response prediction, until recently the state of the art has been feature-based models. The popularity of a method within a field can be thought to be both cultural, as well as informed by domain-specific challenges and constraints.

We also note that in some fields, the main goal is not prediction but rather gaining some insight, or testing some hypothesis. This difference in end-goals is similar to that between machine learning and classical statistics (Breiman, 2001). Nonetheless, essentially the same models may be employed for both tasks, and so we include such approaches in our discussion below.

2.5.1 Unsupervised models

In link prediction, a popular class of models is based on assuming some simple generative model of a graph, and using this to compute scores that tell us the chance of a linking forming between any pair of nodes. Generally, this generative model is based on certain topological properties of the graph, such as the degrees of the nodes, the set of neighbours for a node, and so on. For example, we may posit that the probability of two nodes having an edge is proportional to the number of neighbours they share. Then, given a partial snapshot of a graph, for any two nodes (i, j) where the edge status is unknown, our prediction is

$$\Pr[y = 1|(i, j)] \propto \sum_{k \in \mathcal{V}} L(i,k)L(j,k).$$

Other popular scores are the Adamic-Adar (Adamic and Adar, 2003) and Katz score (Lu and Zhou, 2010). These models have also been applied to collaborative filtering problems (see e.g. (Chiluka et al., 2011)), but relatively sparingly; alternate prediction rules based on characteristics of rating data have been more popular, e.g. (Lemire and Maclachlan, 2005). A limitation of these models is that their assumed generative processes tend to be rigid, in the sense of not being parametrized. Nonetheless, they tend to perform well in practical settings of link prediction (Liben-Nowell and Kleinberg, 2003; Lu and Zhou,

2010).

A related class of models are so-called neighbourhood methods, which were a popular early solution to collaborative filtering (Resnick et al., 1994; Herlocker et al., 1999; Sarwar et al., 2001). The idea here is to consider the equivalent weighted bipartite graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ for the training set. We now posit that the weight of an edge for nodes (i, j) is

$$L_{ij} = \mu_j + \sum_{k \in \mathscr{N}(j)} \alpha_{jk}(L_{ik} - \mu_k),$$

where $\mu_j = \sum_{i:(i,j)\in\mathscr{O}} y_{ij} / \sum_{i:(i,j)\in\mathscr{O}} 1$ is the average label corresponding to the node j, α_{jk} represents the similarity between nodes j, k, and $\mathscr{N}(j)$ is some notion of a neighbourhood of j. Generally, the similarity scores are also derived from the data via measures such as Pearson correlation of the labels corresponding to nodes j and k, and the neighbourhood for a node is chosen to be the K most similar nodes according to this measure. As with the above models for link prediction, the generative process of the bipartite graph has no parameters with these choices, and is thus inflexible. A richer model would attempt to learn the α weights from training data, and indeed this turns out to be closely connected to latent feature models (Hu et al., 2008; Lawrence and Urtasun, 2009), which we shortly discuss. It is also possible to learn the weights by thinking of the problem in terms of a graphical model (Defazio and Caetano, 2012).

2.5.2 Feature-based models

We saw in Section 2.4.1 that dyadic prediction can be thought of as a special type of supervised learning. If we have side-information, we can ignore the dyad members' identities and just look to apply supervised learning on the features. Indeed, this is precisely what bilinear regression, discussed in Section 2.6.3, does. While a reasonable solution, intuitively it discards information present in the dyad members' identities, which

is sub-optimal. For example, in collaborative filtering, it would mean that two users with the same feature vectors would get identical predictions, even if they have extremely differing tastes. A better alternative is to use a random effects regression model (see Appendix C), wherein each dyad member gets their own feature vector. Such models are popular in item response theory (Rasch, 1961), response prediction (Agarwal et al., 2010a), and are a component of some collaborative filtering models (Agarwal et al., 2010b; Khanna et al., 2012).

In the setting where there is no side-information, the above strategy does not work. However, if we can *construct* features from this data, we can apply supervised learning models. Therefore, we can ask what a reasonable feature generation scheme is. Arguably the simplest strategy is to transform each dyad $x = (i, j)$ via the representation $\phi(x) = \begin{bmatrix} e_i & e_j \end{bmatrix}$, where $e_i \in \{0,1\}^M$ denotes the standard basis vector with a 1 at the ith position. If one were to apply a standard linear model to this representation, we would essentially learn weights specific for each user and movie. However, such a model would miss out on the interactions between the two. This is discussed more in Section 3.2.3.

In collaborative filtering, an alternate feature construction strategy that has been explored (see e.g. (Crammer and Singer, 2001)) is to take some subset $\mathscr{S}$ of users who have made a lot of ratings, and perform a mean-shift, so that the average rating of a user is 0. Next, we replace their missing ratings with 0, and for every user $u \notin \mathscr{S}$, perform a separate learning task. There is a single example corresponding to every movie the user u has rated in the training set. For each training example, the features are the ratings of all users in $\mathscr{S}$ for the movie, and the label is u's rating of the movie. The goal is then to predict the ratings that u will give on the unrated movies, where the features are again the ratings given to that movie by the users in $\mathscr{S}$. A related strategy (see e.g. (Shashua and Levin, 2002)) is to treat the problem as M separate learning tasks, one for each user, and use the labels of other user as features. In particular, for a fixed i, we assume that each

known entry $(i, j) \in \mathcal{O}$ constitutes a training example. The features for this example can simply be set to the labels for all other users associated with this movie, $M_{i'j}$ for $i' \neq i$. As some of these may not be present in the training set, our feature set comprises missing data, for which one coping mechanism is to perform some form of feature imputation. There are a few problems with this approach. First, treating each learning task separately is time-consuming. Second, we have to come up with a heuristic to fill in missing entries when constructing features. Third, by making each learning task completely separate, we are possibly losing a lot of shared information among the row objects.

Other domain-specific feature generation schemes are possible. For example, in the context of collaborative filtering, (J.S. Kong, 2012) uses as features the number of paths in the bipartite user-movie graph that have a given like/dislike configuration. The intuition for such a scheme is that a configuration such as { Like, Dislike, Like, Dislike } tells us how many users there are that vehemently disagree with a reference user's tastes.

2.5.3 Clustering models

A natural strategy to dyadic prediction is assuming that the dyad members belong to certain *clusters*, and that their cluster memberships influence the label for their interaction. These are known as *blockmodels* in psychometrics and social network analysis (Lorrain and White, 1971). If we place a probability distribution over the clusters, which corresponds to a soft clustering of the dyad members, we end up with so-called *stochastic blockmodels* (Fienberg and Wasserman, 1981; Snijders and Nowicki, 1997; Nowicki and Snijders, 2001). Recent advancements in these models have been allowing for infinite number of clusters (Kemp et al., 2006), and simultaneous membership in multiple clusters (Airoldi et al., 2008). A related line of work in collaborative filtering is the design of *co-clustering* models (Agarwal and Merugu, 2007; Shan and Banerjee, 2010). A generalization of blockmodels and coclustering is the aspect model (Hofmann

et al., 1999), where it is assumed that each *dyad* itself has a cluster membership.

An appealing property of these models is that they produce interpretable results, in the form of probabilistic cluster memberships for the entities in the dyad. A limitation of basic versions of these models is that they are potentially too coarse-grained, in that they treat all entities within a cluster identically.

2.5.4 Latent feature models

A powerful approach to dyadic prediction is based on learning *latent features*. We give special attention to this type of model, as they have been shown to give state-of-the-art performance. We specifically focus on *matrix factorization* methods, which are the most popular latent feature approach for dyadic prediction. In the machine learning community, this approach is best known for being a key ingredient in the Netflix prize (Koren et al., 2009). For simplicity, say that $\mathscr{Y} = \mathbb{R}$, and ignore any available side-information. The factorization model for such a dyadic prediction problem posits that

$$\Pr[y|x = (i, j); \theta] = \mathscr{N}(y; u_i^T v_j, \sigma_\varepsilon^2).$$

where $u_i, v_j \in \mathbb{R}^k$ for some $k \in \mathbb{Z}_+$. That is, each dyad member i, j is associated with a *latent feature vector*, and the label arising from the interaction between these members is determined by the similarity between the corresponding vectors. In the case of collaborative filtering, these latent feature vectors have a natural interpretation as users' and items' propensities towards certain features (e.g. whether the item is a status symbol, whether the user likes foreign films, *et cetera*).

Factorization methods can be seen as extensions of the classical singular value decomposition (SVD; see Appendix A). The SVD says that every matrix $X \in \mathbb{R}^{M \times N}$

admits a decomposition of the form

$$X = U\Sigma V^T,$$

where Σ is a diagonal matrix with nonnegative entries. By absorbing $\Sigma^{1/2}$ into each of U, V, we see that the above may be rewritten as $X = \tilde{U}\tilde{V}^T$. Thus, *any* collection of unidimensional interactions X between a pair of entities may be exactly modelled by assocating a latent vector to each entity, and taking the dot-product. In the presence of missing data, we do not know the entire matrix X, which means the SVD cannot be applied as-is. Factorization methods thus try to find the best factorization of the form $X = UV^T$ that agrees with the observed data. This may also be interpreted as the maximum likleihood estimate under an appropriate probabilistic model of the data (Appendix B).

We now provide a survey of existing latent feature approaches to dyadic prediction. Most machine learning research on dyadic prediction has been in the field of collaborative filtering, and so our survey is biased towards this field. Nonetheless, we do wish to emphasise that many models in collaborative filtering have been explored in fields outside of machine learning. Our aim in reviewing these related fields, such as social network analysis and item response theory, is to give some concrete examples of these rediscoveries.

Collaborative filtering

The approach of learning latent factors to make predictions in collaborative filtering data has been very successful. The earliest realizations of this idea that we are aware of include (Lee et al., 1995; Pryor, 1998; Sarwar et al., 2000). However, it was only until the Netflix prize (Netflix, 2006) that the approach was accepted to be the best single method. In particular, the famous blog post of Simon Funk (Funk, 2006) was

arguably responsible for resurrecting interest in the latent feature method. The core of his approach may be seen as a MAP estimate of probabilistic PCA, with training performed by SGD.

Subseqent to this, many enhancements to the basic latent feature model were proposed for the Netflix prize. For example, (Koren, 2008) showed that augmenting the latent feature for a user based on the movies they have rated can give a boost in performance. We largely ignore such developments: though practically important, they are arguably specialized to the characteristics of a specific dataset, or more generally to a specific application of dyadic prediction (namely collaborative filtering). We focus instead on more generic extensions to the basic factorization idea.

Bayesian extensions of the simple matrix factorization models have proved very successful in collaborative filtering. The probabilistic matrix factorization (PMF) model (Salakhutdinov and Mnih, 2007) formalized how probabilistic PCA may be applied to collaborative filtering. They proposed a quasi-Bayesian scheme of learning regularization parameters automatically, rather than relying on cross-validation, by attempting to find MAP estimates of them as part of the learning process. A complete Bayesian analysis of the same model was done later in (Salakhutdinov and Mnih, 2008). Here, the predictive distribution of missing ratings given the observed ratings was directly modelled by integrating out all parameters, rather than seeking point estimates. An intermediate of sorts between these models was proposed in (Lawrence and Urtasun, 2009), where only one set of latent features were integrated out, and the other set estimated by the MAP procedure. It was further shown how to extend this to allow for a *nonlinear* factorization, by exploiting the relationship between PCA and Gaussian processes. Further extensions of this Bayesian approach have been studied, e.g. (Mackey et al., 2010), which combines the factorization with a clustering model.

Another broad extension has been the use nonparametric models that allow an

infinite number of latent features. An early successful model of this type is maximum-margin matrix factorization (MMMF) (Srebro et al., 2004), which additionally applies the maximum margin principle to modelling. Here, the training objective is

$$\min_{\hat{X}} \sum_{(i,j)\in\mathscr{O}} \ell(X_{ij},\hat{X}_{ij}) + \lambda||\hat{X}||_{\Sigma},$$

where $\mathscr{O}$ is the set of observed entries in X, ℓ is an ordinal extension of the hinge-loss, and $||\cdot||_{\Sigma}$ is the trace norm of a matrix (see Appendix B). Training this model requires solving an SDP, which is not very scalable. In (Rennie and Srebro, 2005), the parametric special case was considered, where the number of latent features was fixed. This is at the cost of non-convexity, but experimental results in (Rennie and Srebro, 2005) show that the mode is able to find good local optima.

Two different nonparametric matrix factorization models were proposed in (Yu et al., 2009). In the first, instead of optimizing over the factors U,V, one optimizes over the kernel matrix $K = VV^T$. Indeed, the standard factorization objective is shown to be equivalent to

$$\min_{\hat{X},K\succ 0} ||X-\hat{X}||_F^2 + \gamma_1 \mathrm{tr}[\hat{X}K^{-1}\hat{X}^T] + \gamma_2 \mathrm{tr}[K].$$

An EM algorithm was proposed to solve the above objective. The second method is a Bayesian version of the above, where the variable $\hat{X}$ is integrated out entirely, so that the optimization is only over K.

There are other ways to introduce latent features beyond matrix factorization. Restricted Boltzmann machines (RBMs) have enjoyed some success for collaborative filtering (Salakhutdinov et al., 2007). The joint probability over input x, label y and

hidden units h for an RBM is log-linear,

$$\Pr[x, y, h; \theta] = \frac{\exp \Psi(x, y, h)}{\sum_{x', y', h'} \exp \Psi(x, y, h)}.$$

Here, Ψ is some linear function of its inputs. This model also has latent features, but not in the form of matrix factors. Rather, we assume that there are a number of latent binary features that describe the data. We learn the probability distributions of these features. As a consequence of not modelling user and item specific latent features (at least in the general case), we cannot make simple conditional independence assumptions. Instead, what we have to do is impose relationships between different entries of the matrix, so that e.g. all nodes corresponding to the same user have a link between them. This makes the modelling task more difficult, but efficient approximations for training are possible (Truyen et al., 2009).

Item response theory

Recall that item response theory (IRT) is concerned with the responses of individuals to certain questions, such as student responses to examination questions. We briefly go over some standard IRT models which can be interpreted as learning latent features; see e.g. (Reckase, 2009) for a more thorough reference. We begin with the classical Rasch model (Rasch, 1961), which is applicable for responses that are either correct ($y = 1$) or incorrect ($y = 0$). The model for person i getting question j correct is

$$\Pr[y = 1 | (i, j); \theta] = \sigma(u_i - d_j).$$

Here, u_i represents some latent characteristic of the the person i (their "ability"), and d_j a corresponding characteristic of the question j (its "difficulty"). The model says that if the person's ability vastly exceeds that of the question, they will almost certainly

get the question right. The model is most uncertain about questions whose difficulty is exactly on pair with the ability of the person, i.e. the question tests the limits of their knowledge. We note that the model can be seen as a type of random-effects logistic regression (Appendix C), or equivalently an application of logistic regression with the dummy feature representation $\phi((i,j)) = \begin{bmatrix} e_i & e_j \end{bmatrix}$, as discussed in Section 2.5.2.

The Rasch model is sometimes referred to as a one-parameter logistic (1PL) model. An extension to this is the two-parameter logistic model (2PL), which is

$$\Pr[y_{ij} = 1|\theta] = \sigma(v_j \cdot (u_i - d_j)),$$

The difference to the Rasch model is thus the addition of a question-dependent scaling factor, v_j, which affects the influence of the disparity between the person's ability and the question's difficulty. This can also be interpreted as a one dimensional logistic PCA model (Schein et al., 2003), with a per-item bias term.

We note that multidimensional extensions of the above models exist, although they appear not as popular. For example, the multidimensional 2PL model is

$$\Pr[y_{ij} = 1|\theta] = \sigma\left(\sum_{k=1}^{K} u_{ik} \cdot v_{jk} - d_j\right).$$

These models can also be generalized to the case of non-binary responses. The choice of model depends on the nature of response. One possibility is that the responses come from a Likert scale, e.g. $\{$ Dissatisfied, Neutral, Satisfied $\}$. The partial credit model (Masters, 1982) can be seen as an extension of the Rasch model for such responses, and models the probability of individual i responding to item j with option r as

$$\Pr[y_{ij} = r|\theta] \propto \exp\left(u_i - \sum_{r'=1}^{r} d_j^{(r')}\right).$$

Compared to the standard Rasch model, we see that we keep a series of difficulty thresholds for each possible grade r, representing the fact that it may not be difficult to get a question partially correct, but very difficult to get it perfectly correct. A different approach is the Graded Response Model (Samejima, 1970),

$$\Pr[y_{ij} = r|\theta] = \sigma(u_i \cdot v_j - d^{(r)}) - \sigma(u_i \cdot v_j - d^{(r+1)}),$$

which can be seen as as a latent version of the cumulative logistic regression model for ordered labels (Agresti, 2010, p. 47).

For the case where the responses are not ordered, e.g. multiple-choice questions, one possible model is the nominal response model (Bock, 1972),

$$\Pr[y_{ij} = r|\theta] \propto \exp(v_j^{(r)} \cdot (u_i - d_j^{(r)})),$$

which can be seen as an extension of the 2PL model, with response-specific item difficulties and scaling factors. A multidimensional extension was studied in (Bolt and Johnson, 2009), and is

$$\Pr[y_{ij} = r|\theta] \propto \exp\left(\sum_{k=1}^{K} u_{ik} v_{jk}^{(r)} - d_j^{(r)}\right),$$

where now both $u_i, v_j^{(r)} \in \mathbb{R}^K$.

Social network analysis

While we focus on latent feature approaches to social network analysis, many other approaches are possible; see e.g. (Goldenberg et al., 2010) for a more detailed survey of other approaches.

The first latent space model for social network analysis was proposed in (Hoff

et al., 2001). Here, the model for a connection existing between actors i and j is

$$\Pr[y_{ij} = 1|\theta] = \sigma\left(\frac{u_i^T u_j}{||u_j||} + w^T x_{ij} + \mu\right),$$

where $u_i, u_j \in \mathbb{R}^K$ for some $K \in \mathbb{Z}_+$, and $x_{ij} \in \mathbb{R}^D$ is a vector of covariates (or features, or side-information) for the dyad (i, j). The parameters $\theta = (U, w, \mu)$ are all learned from a training set using MCMC sampling. This is related to logistic PCA, with accomodation for dyad side-information. It is also seen to be similar to the 2PL model from item response theory. One difference is that the dot-product is computed between the latent vector for i and the normalized latent vector for j. This can be seen as the projection of j's latent vector onto that of i. Another difference is that the method handles missing data by only modelling observed edges in the network.

Various extensions of the above model have been studied. One line of work has been to extend the applicability of the model. For example, (Handcock et al., 2007) looks to impose a clustering prior on the u_i's, which encodes the belief that actors in a network tend to form groups or cliques. Another line of work has been to change the role of the latent parameters in the model. A subsequent model in (Hoff, 2003) drops the projection of u_j, and instead uses $u_i^T u_j$. Note that this model is only suitable for symmetric social networks. This was further extended in (Hoff, 2007), which uses the model

$$\Pr[y_{ij} = 1|\theta] = \sigma\left(u_i^T \Lambda u_j + w^T x_{ij} + \mu\right)$$

where $\Lambda \in \mathbb{R}^{K \times K}$ is a diagonal matrix which is additionally learned. In the case of a directed network, the paper mentions (but does not explore) the model

$$\Pr[y_{ij} = 1|\theta] = \sigma\left(u_i^T v_j + w^T x_{ij} + \mu\right).$$

Such a model has been successfully applied in domains such as political science, where an important problem is modelling disputes between nations (Ward et al., 2007). A weakness of this approach, however, is that it does not exploit the fact that u_i and v_i are expected a-priori to be similar to each other, by consequence of belonging to the same actor in the network. A recent extension (Li et al., 2011) attempts to correct this via the model

$$\Pr[y_{ij} = 1|\theta] = \sigma\left(u_i^T u_j + u_i^T v_j + \mu\right),$$

which can be seen to combine the models for the directed and undirected case. However, this model does not study the case of missing links.

Another extension to the above latent space models has been the use of an *infinite* number of latent features. For example, (Miller et al., 2009) assumes that each u_i is an infinite dimensional binary vector, and imposes an Indian Buffet Process prior on the U matrix.

Other fields

In the social sciences, the social relations model (Kenny and Voie, 1984) is a a popular model for qualitative analysis of the interaction between "actors" (individuals). In particular, the model assumes the strength of interaction y between actors i and j can be modelled as

$$y_{ij} = \mu + a_i + b_j + e_{ij}.$$

The goal is to estimate the inter- and intra-group variances. This can also be seen as a two-way ANOVA model, and depending on further modelling assumptions, a random-effects model. Note that we don't estimate e_{ij} directly. We generally assume all data is present. In collaborative filtering, (Li and Yeung, 2011) looks to adapt this model to the missing data scenario through a probabilistic framework.

In political science, several of the latent space models from social network analysis have been applied for dyadic problems, such as the analysis of military disputes between nations (Ward et al., 2007), and voting patterns of politicians (Clinton et al., 2004). Generally, Bayesian estimation is standard, as there is concern with interpretation of the learned model and hypothesis testing.

A dyadic modelling problem arising in many fields is the analysis of labels provided by multiple experts, not all of which agree with each other. For example, in epidemiology, the problem arises when one can perform a number of tests on a patient, each of which may give a different assessment of her condition (Joseph et al., 1995). Here, the dyad is the (rater, item) pair, with there potentially being features describing each item and/or rater. The major goal here is generally to either extract the underlying ground truth label for an item, or to determine the reliability of each rater, rather than assess how a rater will rate a future (or unrated) item. (The recent work of (Raykar et al., 2009) discusses a solution to this classification problem.) An early formalization of the problem looked at the case of nominal labels, and used the EM algorithm to learn the reliability of each rater (Dawid and Skene, 1979). Related models have been devised and applied in biomedical informatics (Hripcsak and Heitjan, 2002; Rzhetsky et al., 2009), epidemiology (Joseph et al., 1995), remote sensing (Smyth et al., 1994). Models based on latent features (sometimes called latent *traits*) have been studied in psychometric instantiations of this task (Uebersax and Grove, 1993), and have been imported to crowdsourcing applications (Whitehill et al., 2009; Welinder et al., 2010; P. Welinder, 2010; Liu and Wang, 2012).

The above represent only the fields where dyadic problems are commonly studied. There are innumerable others where specific dyadic problem arise, and latent feature models have been applied: to name a few, models of distances in computer networks (Mao and Saul, 2004), ranking of football teams based on results of their games with

each other (Park and Newman, 2005), and economic analysis of individuals' choices such as that between several brands (McFadden, 2001; Görür et al., 2006) have been analyzed with latent feature approaches.

Comment on the differences between the fields

The above discussion makes clear that essentially identical models have been applied in many different fields. These connections motivate testing advances in one field to another. For example, see (Winters et al., 2005) for an application of collaborative filtering methods to IRT. In machine learning, the collaborative filtering literature is often the most familiar point of reference, so it is prudent to ask what differences, if any, there are between the models as applied in these different fields. The conceptual difference is that, as mentioned earlier, collaborative filtering models are concerned with *prediction*, which in item response theory and social network analysis, say, the goal is on *modelling*. Concretely, this translates to the following differences in the models.

- The datasets are generally assumed to possess no missing data, e.g. in IRT, all individuals provide responses to all questions. This is of course patently not the case in collaborative filtering. There are training procedures that remove this assumption, but they often involve imputing the missing values (e.g. (Finch, 2008), (Patz and Junker, 1999)), which is not feasible on large datasets, or are based on heuristics (Lord, 1974).

- Many studies focus on a single dataset without performing cross-validation and/or a train-test split. In collaborative filtering models, as with most machine learning research, such procedure is endemic.

- As the focus is on reliable estimation of parameters from the data, issues such as asymptotic behaviour of estimation schemes (efficiency and consistency) are

important concerns. In collaborative filtering models, generalization ability is the main concern.

- Generally, low-dimensional models are employed (whether latent or explicit), so that the results may be interpreted. In collaborative filtering models, the number of latent dimensions is often a large constant.

- Training procedures for these models are generally expensive to run on large datasets. In collaborative filtering, techniques from large-scale learning are employed to scale to dataset sizes as faced by industry applications.

2.6 Analysis of the latent feature approach

We analyze the latent feature approach, and in particular factorization methods, in more detail. We begin with a discussion of its strengths and weaknesses. Then, we look at the relationship between latent feature models and existing models.

2.6.1 Strengths and weaknesses

As discussed in Section 2.5.4, factorization models are motivated by the SVD of a matrix. This decomposition is guaranteed to always exist for a fully observed matrix, and enjoys certain optimality properties as low-rank approximation (see Appendix A). Strictly, these do not hold in dyadic prediction problems, where there is missing data. Nonetheless, this is often taken as an explanation for the effectiveness of factorization methods. From a more basic perspective, by virtue of keeping a parameter for each dyad member, factorization methods are capable of producing very fine-grained predictions. Compared to a clustering method, for example, with a factorization we can account for the fact that two entities may be similar in general, but have very different behaviour in some regions of the data. For example, in collaborative filtering, two users may have

largely similar taste, except that they are completely opposite when it comes to horror movies. As a result of this "personalization", the factorization method tends to produce high accuracy solutions.

One issue with factorization models is that they are not as scalable as some other methods, in particular unsupervised or pure feature based methods. In particular, while fast training schemes such as stochastic gradient descent may be applied (see Section 2.6.2) in single-machine settings, for web-scale data that requires working with a cluster, while still feasible, training can become a bottleneck. We do note that there has been recent work on parallelizing the training of factorization models (Gemulla et al., 2011; Recht and Ré, 2011; Khanna et al., 2012). Another potential challenge is that the number of parameters to be estimated is linear in the number of dyad members, which may be too large to store in memory in commercial applications. One proposal to combat this is (Karatzoglou et al., 2010), which uses a hashing mechanism to reduce the storage requirement.

Another issue worth noting is that the parameters in a factorization model depend only on the dyad identity data, and *not* on any side-information. This has been observed to not be a major issue in applications like collaborative filtering (Pilászy and Tikk, 2009), where easily collected side-information tends not to be very predictive of the label, but this may not be true in general.

2.6.2 Training latent feature models

We now discuss some options in training factorization models. In its most basic form, such a model may be represented by the objective

$$\min_{U,V} \frac{1}{|\mathscr{O}|} \sum_{(i,j)\in\mathscr{O}} \ell(y_{ij}, u_i^T v_j) + \lambda \Omega(U,V), \tag{2.2}$$

for an appropriate loss function ℓ and regularizer Ω.

The first point to note is that the problem is non-convex: there are many local optima corresponding to matrices that are not merely isomorphic upto rotation (Srebro and Jaakkola, 2003). Since the objective is non-convex, we have to resort to finding some local optimum. A natural way to perform the optimization is criss-cross regression (Gabriel and Zamir, 1979), also known as alternating least squares in the case where ℓ is the square loss. The idea is simply to optimize U with V fixed, and then optimize V with U fixed, and so on. Each sub-optimization is a classification or regression problem, and may be performed independently of the others (i.e. the overall optimization becomes embarrassingly parallel).

Another optimization strategy is stochastic gradient descent (SGD), which was popularized in the Netflix prize after a famous post of Simon Funk (Funk, 2006). (Similar pair-at-a-time learning algorithms based on the SVD have been considered independently (Gorrell, 2006).) An advantage of SGD is that it generalizes faster than batch methods (Bottou and Bousquet, 2007), and is very simple to implement, with a low memory overhead. Some studies have found ALS to perform better than SGD on the same objective (Zhou et al., 2008), but using larger values of K.

It is possible to train factorization models using a more sophisticated objective. For example, one can look to explicitly learn the unobserved labels by treating them as additional parameters, and use expectation-maximization (EM). It is also possible to follow a Bayesian framework, and integrate out one or more sets of parameters. Appendix B.1.3 these alternate methods.

2.6.3 Connections to other models

We discuss the relationship between factorization methods and some existing models for dyadic data.

Supervised learning models

Earlier, we explained how some instantiations of dyadic prediction can be seen as a type of "feature-less" learning. If we consider Equation 2.2 when U is fixed, we see that it reduces to a standard multivariate supervised learning problem. (The same holds for fixed V.) Thus, factorization methods attempt to simultaneously discover predictive features and weights on these features, solely from the label information.

Random effects models

The latent feature approach to dyadic prediction can also be seen as a random effects model, introduced in Section 2.4.4: writing such a model as

$$y_{ij} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \varepsilon_{ij} : \alpha_i \sim \mathcal{N}(0, \sigma_\alpha^2), \beta_j \sim \mathcal{N}(0, \sigma_\beta^2), \gamma_{ij} = u_i^T v_j, \varepsilon_{ij} \sim \mathcal{N}(0, \sigma_\varepsilon^2),$$

the group-interaction parameters γ are random effects. This interpretation was introduced in (Hoff, 2005), but more generally, the use of a multiplicative term in ANOVA dates back to at least (Fisher and Mackenzie, 1923).

Clustering models

Clustering models can be seen as a special case of (possibly probabilistic) latent feature models, where the cluster membership vector for a dyad element is treated as its latent feature representation. Typically, clustering models are based on a single cluster membership generative process, so that a dyad element may belong to one and only one group for the purposes of a particular interaction. This is not the case for factorization models.

Bilinear regression

Recall the *fixed design* linear regression problem in statistics: given a design matrix $X \in \mathbb{R}^{n \times D}$ and a vector $y \in \mathbb{R}^n$ drawn from $\mathcal{N}(Xw, \sigma^2 I)$, find an estimator $\hat{w}$ of w. Note that X here is assumed fixed and known, and is not subject to random variation. The classical estimator of $\hat{w}$ is the ordinary least squares estimator, defined as

$$\hat{w} = \operatorname*{argmin}_w ||y - Xw||_2^2.$$

The two dimensional, or dyadic, extension of this problem is called *bilinear regression* (Gabriel, 1998). We are given design matrices $X \in \mathbb{R}^{n \times D_1}$ and $Z \in \mathbb{R}^{m \times D_2}$ representing features for the rows and columns of a matrix respectively. We further have a matrix $Y \in \mathbb{R}^{n \times m}$ for which we assume $\text{vec}(Y) \sim \mathcal{N}(\text{vec}(XWZ^T), \sigma^2 I)$, where $\text{vec}(\cdot)$ denotes concatenation of columns. The goal is to construct an estimator $\hat{W} \in \mathbb{R}^{D_1 \times D_2}$ of W. The analogue of the OLS estimator is

$$\hat{W} = \operatorname*{argmin}_W ||Y - XWZ^T||_F^2.$$

We point out a simple connection between latent feature models and bilinear regression (Section 2.6.3). This observation is not new, and dates back to at least (Gabriel, 1998). However, the fact seems under-appreciated in some of the modern literature, for example in collaborative filtering. We also note that essentially the same mathematical observation is made in (Rendle, 2010), but without identifying the relationship to bilinear regression.

The connection is derived as follows. Given a target matrix Y, recall that bilinear regression assumes the existence of feature sets X, Z for the rows and columns respectively. Consider the setting where we do not have explicit covariates for the rows and

columns. We can nonetheless set $X = I_n$ and $Z = I_m$ and attempt to proceed. Intuitively, we are using the identity of each row and column as a feature. The underlying model then becomes $\text{vec}(Y) \sim \mathcal{N}(\text{vec}(W), \sigma^2 I)$. Without further assumptions, the model is thus underspecified, because we have a trivial solution of $W = Y$. To make progress, we must assume there is some special structure to W that allows sharing of parameters. One possible assumption is that W has rank r, and so may equivalently be expressed as $W = AB^T$ for $A \in \mathbb{R}^{n \times r}$ and $B \in \mathbb{R}^{m \times r}$. This results is the model $\text{vec}(Y) \sim \mathcal{N}(\text{vec}(AB^T), \sigma^2 I)$, with the goal now being to get estimators $\hat{A}, \hat{B}$. Using the same least-squares objective as above, we get

$$(\hat{A}, \hat{B}) = \underset{A,B}{\text{argmin}} ||Y - AB^T||_F^2,$$

which is nothing but a standard latent factor model.

2.6.4 A comment on the independence assumption

Recall that in Section 2.3.1, we stated that dyadic data is generally *not* iid. At first glance, then, it seems that a model like Equation 2.2 should be invalid, because the loss function is evaluated independently for each dyad. If the objective may be motivated probabilistically as a MAP estimate, for example the MAP estimate under a Bernoulli model with a Gaussian prior on the parameters U, V, does this not imply an independence assumption? In particular, we are assuming

$$\Pr[\{y_{ij}\}_{(i,j) \in \mathcal{O}} | U, V] = \prod_{(i,j) \in \mathcal{O}} \Pr[y_{ij} | u_i, v_j].$$

The explanation for the apparent paradox is that the above encodes *conditional* independence, if we interpret U, V as being random variables. The labels may by themselves still be conditionally dependent, however.

A natural question, then, is whether such a conditional independence assumption

is justified. In general, the answer is yes, due to a theorem of Aldous and Hoover (Aldous, 1985). The Aldous-Hoover theorem is an extension of de Finetti's theorem, which works on exchangeable random variables, to exchangeable *arrays*. Recall that, informally, de Finetti's theorem states that an infinite sequence of exchangeable random variables may be written as a mixture of iid sequences (see e.g. (Aldous, 1985, pg 19), (Schervish, 1995, pg 28), (Smith, 1984), (Kingman, 1978)). This is used as justification of Bayesian modelling for exchangeable data, as we can think of the mixing weights as being prior probabilities on different parameters. The Aldous-Hoover theorem, informally, states that the same holds when we consider infinite sequences of pairs of random variables, or arrays. More formally (see e.g. (Teh and Roy, 2009)):

Theorem 1 (Aldous-Hoover theorem)**.** *Let* $\{X_{ij}\}_{i,j=1}^{\infty}$ *be an infinite row-and-column-exchangeable matrix. Then, there exist random variables* Θ, U, V *such that*

$$(\forall m,n \in \mathbb{N})\ \Pr[X_{11} = x_{11}, ..., X_{mn} = x_{mn}] = \int \prod_{i=1}^{m} \prod_{j=1}^{n} \int \int \Pr[X_{ij} = x_{ij} | \Theta, u_i, v_j] dv_j du_i d\Theta.$$

Above, the random variables U, V play a similar role to the ones that we use in our latent feature model.

Chapter 3

LFL: a Log-Linear Model for Dyadic Prediction

In this chapter, we propose LFL, a log-linear model for generic dyadic prediction tasks. The model learns latent features from dyadic data to model a probability distribution over all possible outcomes for a given dyad. We analyze various characteristics of the model, and in particular contrast it to the factorization approaches described in the previous chapter. Experimental results confirm that the model is computationally simple to train, is sufficiently expressive, and that it performs favourably compared to existing models.

3.1 Motivation: a generic dyadic prediction model

In the previous chapter, we saw examples of several applications of dyadic prediction arising in different fields. Each application had a slew of models studied in the respective literatures. Given the similarity between these problems - for example, between collaborative filtering and link prediction in graphs - a natural question is whether a single model, or an instantiation of a general family of models, can perform well on these range of applications. This is conceptually appealing, but also opens the opportunity for improved modelling: once we establish that the same general model can be applied

to two or more problems, then extensions to the model inspired by developments in a particular field can also be transferred to other problems.

Our aim in this chapter is to design such a model. As a guiding principle, we would ideally like to meet three goals:

- *Flexibility*. The basic aim is to construct a model that can attack a range of dyadic prediction problems. Thus, we would like our model to be sufficiently generic as to make minimal assumptions about the nature of the data provided to it.

- *Accuracy*. A model capable of addressing many problems is only of use if it is competitive with state-of-the-art models for those problems.

- *Simplicity*. While not necessary, it is certainly attractive if the model is simple, because it reduces the barrier of adoption in other fields and in real-world applications.

In this chapter, we will attempt to meet these goals with a *log-linear* model. We will now describe a first attempt at designing such a model.

3.2 A first attempt at a log-linear model

In this section, we describe a simple log-linear model that can be applied to dyadic prediction. As will be discussed, the model has limited expressivity. Nonetheless, it serves as a useful starting point for our design.

3.2.1 Log-linear models in general

Given an observation $x \in \mathscr{X}$ and target variable $y \in \mathscr{Y}$, a log-linear model represents the conditional distribution of y given x via

$$\Pr[y|x;\theta] = \frac{\exp\left[\sum_{d=1}^{D} w_d f_d(x,y)\right]}{\sum_{y'} \exp\left[\sum_{d=1}^{D} w_d f_d(x,y')\right]}.$$

Here, the model parameters are $\theta = \{w \in \mathbb{R}^D\}$, a vector of real-valued weights. The functions $f_d : \mathscr{X} \times \mathscr{Y} \to \mathbb{R}$ are called *feature functions*, and measure different interactions between the inputs and labels. The most important design decision that goes into using a log-linear model is choosing sensible f_d's, so as to capture useful relationships in the training data. Essentially the only constraint on the feature functions is that they must depend on y, because otherwise they do not contribute to the probability. An important special case of a log-linear model is multinomial logistic regression. Here, for input $x \in \mathscr{X}$ and label $y \in \mathscr{Y}$, we construct feature functions so that

$$\Pr[y|x;\theta] = \frac{\exp\left[w^{(y)}x + \mu^{(y)}\right]}{\sum_{y' \in \mathscr{Y}} \exp\left[w^{(y')}x + \mu^{(y')}\right]}.$$

Here, the model parameters are $\theta = \{(w^{(y)}, \mu^{(y)}) : y \in \mathscr{Y}\}$, that is, we keep a separate set of weights and a bias term for each outcome in $\mathscr{Y}$.

Essentially the only assumption placed on $\mathscr{Y}$ above is that it is a finite set. This makes the log-linear framework a very general modelling option. Indeed, in addition to capturing multinominal logistic regression, note that it is possible for $\mathscr{Y}$ to be multidimensional, which encodes the multilabel learning problem. Further, $\mathscr{Y}$ may also possess additional structure, such as the case where it comprises of variable-length sequences. In such cases, however, it makes sense from a computational and statistical point to tie together the various parameters for each $y \in \mathscr{Y}$ by exploiting the structure of $\mathscr{Y}$. Indeed, for sequence data, linear chain conditional random fields (Lafferty et al., 2001) can be seen as a log-linear model with a specific choice of parameter tying to ensure tractability.

We now show how to apply the log-linear framework to dyadic prediction problems.

3.2.2 Applying the log-linear framework to dyadic prediction

For simplicity, suppose we have a dyadic prediction task with no side-information. Then, our inputs x is merely a pair (i, j) of integer identities. In Section 2.6.3, we suggested that one way to deal with such data is to use the bitvector encoding $\phi(x) = \begin{bmatrix} e_i & e_j \end{bmatrix}$, where $e_i \in \{0,1\}^{|I|}$ is the ith standard basis vector, with a 1 only at position i, and similarly for e_j. Applying a multinomial logistic regression on this feature representation yields

$$\Pr[y|x = (i, j); \theta] = \frac{\exp\left[a_i^{(y)} + b_j^{(y)} + \mu^{(y)}\right]}{\sum_{y'} \exp\left[a_i^{(y')} + b_j^{(y')} + \mu^{(y')}\right]}. \tag{3.1}$$

The model parameters are $\theta = \{(a_i^{(y)}, b_j^{(y)}, \mu^{(y)}) : i \in I, j \in J, y \in \mathscr{Y}\}$. Plainly, for each possible outcome y, the model associates a weight for each dyad member, as well as a global weight. For example, in collaborative filtering, we keep a set of parameters for each possible rating: one parameter that intuitively measures the frequency of that rating across all users and movies, and one parameter each that reflects the deviation from this overall frequency for a particular user and movie. This can be considered a random intercept model (Kreft and de Leeuw, 1998), a log-linear form of ANOVA (Maxwell and Delaney, 2003), or a random effects model if we placed a probability distribution on the parameters a, b (see Appendix C).

Revisiting the modelling goals introduced in the previous section, we see that the above model is certainly simple, and inherits the flexibility of the log-linear framework. To complete the specification of the model, one should describe how to learn the weights and use it to make predictions. But in fact, it turns out that there is a fundamental limitation in the model, which obviates such discussion.

3.2.3 A weakness of the model: the propensity problem

A problem with the model of Equation 3.1 is that there is no interaction between the dyad members i and j when modelling the probability of y. Consequently, the model only learns *propensities* of row and column items towards a particular outcome, and leads to poor ranking of examples (Vert and Jacob, 2008). To see this, for a fixed i, consider dyads of the form (i, j). For some fixed outcome y, consider the ranking of the j's according to the value of $\Pr[y|x = (i, j)]$. In the case of collaborative filtering, this corresponds to ranking all movies for a fixed user based on how likely the user is to give a rating of y. Note that the ranking depends solely on $b_j^{(y)}$, and so is *independent* of i. This means that we get the *same* ranking for the set of dyads (i', j) for every $i' \neq i$ – all the model manages to learn is the *propensity* towards any particular label. In the movie rating example, this means that all users have the same ranking over movies. This is an undesirable property for a dyadic prediction model.

This suggests that our initial attempt at a log-linear model is flawed, in the sense of having limited expressiveness. We now look at how we can enrich the model in a natural way.

3.3 LFL: a log-linear model with latent features

In this section, we enrich the basic log-linear model to address the propensity problem. We present the resulting model in some generality, and discuss how it can be trained and used to make predictions.

3.3.1 Adding latent features to the log-linear model

The limitation of the previous model can be traced back to our choice of feature representation, $\phi(x) = \begin{bmatrix} e_i & e_j \end{bmatrix}$. This representation inherently does not encourage interaction between the dyad members, because the information about them is merely

concatenated. A natural way to try to encourage such interaction is to consider a cross-product representation, $\phi(x) = e_i \otimes e_j$, where $\otimes$ denotes the Kronecker product. The resulting model is

$$\Pr[y|x = (i,j);\theta] \propto \exp\left(W_{ij}^{(y)}\right), \tag{3.2}$$

where, perhaps unsurprisingly, we end up with a separate set of weights for each dyad (i,j) and outcome y. This model has the opposite problem, in that it is *too* powerful: in particular, it is not hard to see that the weights will end up being set so as to memorize the training labels for each dyad. Further, it has no means of generalizing to dyads that are not observed in the training set, which is clearly undesirable.

To get around this problem, we assume that for each y, the matrix $W^{(y)}$ is *low rank*. Specifically, we assume that for each y, there are matries $U^{(y)} \in \mathbb{R}^{K\times M}, V^{(y)} \in \mathbb{R}^{K\times N}$, vectors $a^{(y)} \in \mathbb{R}^M, b^{(y)} \in \mathbb{R}^N$, and a scalar $\mu^{(y)} \in \mathbb{R}$, such that

$$W^{(y)} = (U^{(y)})^T V^{(y)} + a^{(y)}\mathbf{1}_N^T + \mathbf{1}_M (b^y)^T + \mathbf{1}_M \mathbf{1}_N^T \cdot \mu^{(y)}. \tag{3.3}$$

In other words, our probability model is

$$\Pr[y|x = (i,j);\theta] \propto \exp\left((u_i^{(y)})^T v_j^{(y)} + a_i^{(y)} + b_j^y + \mu^{(y)}\right). \tag{3.4}$$

The difference to the basic log-linear model of Equation 3.1 is the first term, which involves a product of factors specific to the dyad members i and j respectively. By virtue of this term, which effectively ties together the various $W_{ij}^{(y)}$ parameters, we no longer suffer from the propensity problem. We call this model the *latent feature log-linear* or *LFL* model.

The set of parameters in this model is $\theta = \{(U_{ki}^{(y)}, V_{kj}^{(y)}, a_i^{(y)}, b_j^{(y)}, \mu^{(y)}) : i \in \mathscr{I}, j \in \mathscr{J}, k \in [K], y \in \mathscr{Y}\}$. Note that $K \in \mathbb{Z}_+$ here is some constant, which we refer to as the

number of *latent features*. The total number of parameters is therefore $O(|\mathscr{Y}| \cdot (|\mathscr{I}| + |\mathscr{J}|) \cdot K)$. We will discuss ways to reduce this in Section 3.5.1.

3.3.2 Exploiting side-information

An appealing feature of the log-linear framework is that it can seamlessly incorporate side-information. Suppose that the dyad (i, j) has feature vectors $p_i \in \mathbb{R}^{D_1}, q_j \in \mathbb{R}^{D_2}$ for the individual dyad members, and $r_{ij} \in \mathbb{R}^D$ for the dyad itself. We can now consider the augmented feature representation $\phi(x) = \begin{bmatrix} e_i \otimes e_j & s_{ij} \end{bmatrix}$, where $s_{ij} = \begin{bmatrix} p_i & q_j & r_{ij} \end{bmatrix}$. It turns out that the resulting probability model is

$$\Pr[y|x = (i, j); \theta] \propto \exp\left((u_i^{(y)})^T v_j^{(y)} + (w^{(y)})^T s_{ij} + a_i^{(y)} + b_j^y + \mu^{(y)}\right), \tag{3.5}$$

where $w^{(y)} \in \mathbb{R}^{D+D_1 \cdot D_2}$ is an additional set of weights for the explicit features. We note that the model is essentially unchanged even if we modify the precise meta-feature representation s_{ij}. We will discuss some other choices for this representation in Section 3.5.3.

3.3.3 Training the model

Given a training set $\{(x^{(t)}, y^{(t)})\}_{t=1}^T$ of labelled dyads, with $x^{(t)} = (i^{(t)}, j^{(t)})$, we would like to find the LFL parameters $\theta = \{(U_{ki}^{(y)}, V_{kj}^{(y)}, a_i^{(y)}, b_j^{(y)}, \mu^{(y)})\}$ that best explain this data, in the sense of maximizing its likelihood. But since there are potentially a large number of weights, it is beneficial to additionally regularize the weights. Many choices are possible, but we will focus on the canonical ℓ_2 penalty. We thus minimize the

regularized negative conditional log-likelihood (CLL):

$$\hat{\theta} = \operatorname*{argmin}_{\theta} L(\theta) + \lambda\Omega(\theta) \text{ , where} \tag{3.6}$$

$$\begin{aligned} F(\theta) &= \frac{1}{T}\sum_{t=1}^{T} -\log \Pr[y^{(t)}|x^{(t)};\theta] \\ &= \frac{1}{T}\left(\sum_{t=1}^{T} \log\left[\sum_{y\in\mathscr{Y}} \exp(W^{y}_{i,j})\right] - W^{y^{(t)}}_{i^{(t)},j^{(t)}}\right), \\ \Omega(\theta) &= \frac{1}{2}\sum_{y\in\mathscr{Y}} \left(||U^{(y)}||_F^2 + ||V^{(y)}||_F^2 + ||a^{(y)}||_2^2 + ||b^{(y)}||_2^2\right) \end{aligned}$$

where $W^{(y)}_{ij}$ is as defined in Equation 3.3.

Many choices are possible for optimizing the above objective, as the optimization is unconstrained and the objective is differentiable. Perhaps the simplest option is to perform stochastic gradient descent, where we just consider the loss with respect to a single training example. That is, we compute

$$\tilde{\nabla} F(\theta) = -\frac{\partial}{\partial\theta}\log \Pr[y^{(t')}|x^{(t')};\theta] - \lambda\theta,$$

where t' is a random index drawn from 1 to T. An advantage of this choice is not only scalability, but fast generalization (Bottou and Bousquet, 2007). See Appendix D for more discussion of stochastic gradient descent. We will discuss some practical details with this optimization procedure in Section 3.7.3.

Note that above, we penalize all components of θ equally with the regularization scaling λ. But it is plausible that the U and V weights have slightly different penalties. It is especially plausible that the weights for side-information be penalized differently.

Therefore, in practice, it is beneficial to use the more generic regularizer

$$\Omega'(\theta) = \sum_{y \in Y} \left(\frac{\lambda_U}{2}(||U^{(y)}||_F^2 + ||a^{(y)}||_2^2) + \frac{\lambda_V}{2}(||V^{(y)}||_F^2 + ||b^{(y)}||_2^2) \right).$$

3.3.4 Making predictions

A strength of our model, as we shall discuss, is that it models the probability distribution over labels for a given dyad. This can be useful to determine if e.g. a dyad is too difficult to accurately predict. However, we sometimes need a hard prediction or classification from a model. Let $\text{Pred}(x;\theta)$ denote our model's prediction for the dyad x given the parameters θ. The simplest choice of prediction is the mode of the probability distribution,

$$\text{Pred}(x;\theta) = \underset{y}{\text{argmax}}\ \Pr[y|x;\theta].$$

In fact, if one wishes to minimize the dyad 0-1 misclassification error on test data, then this is the optimal prediction. In the case where the train and test distributions match, the proof of this mirrors the one for supervised learning. Briefly, letting $\eta(x,y) = \Pr[y|x]$ be the true conditional label probability, the generalization error with respect to 0-1 misclassification loss for a model $\hat{y} : \mathscr{X} \to \mathscr{Y}$ is

$$\mathbb{E}_x \mathbb{E}_{y|x} \mathbf{1}[\hat{y}(x) \neq y] = \mathbb{E}_x \sum_{y \in \mathscr{Y}} \eta(x,y) \cdot \mathbf{1}[\hat{y}(x) \neq y].$$

For a fixed x, it is not hard to see that the optimal prediction $\hat{y}(x)$ must be $\text{argmax}_y \eta(x,y)$, since this term carries the most weight in the objective.

3.4 Analysis of the LFL model

We now analyze the LFL model proposed above in more detail, beginning with a discussion of some of its strengths and weaknesses.

3.4.1 Strengths and weaknesses of the LFL model

The log-linear framework explicitly models the probability distribution over labels for a given dyad. This can be important in some applications, because these probabilities can be used in a decision theoretic framework. For example, they may be used to optimize utility functions that depend on unknown costs (Zadrozny and Elkan, 2001). In Chapter 4, we will discuss how probabilities may be useful in collaborative filtering applications. In Chapter 6, we will discussion how probabilities are useful in computational advertising applications.

The only assumption our model makes about $\mathscr{Y}$ is that its outcomes are finite and mutually exclusive. Consequently, in contrast to most dyadic prediction models, it can handle non-ordinal and multidimensional $\mathscr{Y}$. These settings are common in many real-world applications of dyadic prediction. For example, an online store may information about customers' interactions with their products, with possible outcomes being $\{$viewed, purchased, returned$\}$. Of course, there are situations where $\mathscr{Y}$ is infinite, such as $\mathscr{Y} = \mathbb{R}$. Other models are more appropriate in this setting, but we note that many commonly studied applications of dyadic prediction, such as movie rating prediction, involve finite outcomes.

As we saw above, it is easy for the model to handle side-information. This is beneficial in real world applications of dyadic prediction, where one has to address the cold-start problem, where we have to make predictions for a dyad member that does not appear in the training set. Intuitively, this problem does not have a non-trivial solution

without extra information and/or prior knolwedge. Side-information is one source of extra information that can help partially solve this problem. The LFL model is flexible enough to make use of either identities alone, side-information alone, or both.

The fact that the model is amenable to stochastic gradient training means that it is scalable, certainly in the single-machine setting, and possibly in the cluster setting as well. (See also the discussion in Section 2.6.1.) We do note that, like all methods based on a bilinear factorization, the training objective is non-convex. In practical applications, it appears that this non-convexity does not introduce many "bad" local optima, and performance is comparable to convex methods based on trace norm regularization (Srebro et al., 2004; Avron et al., 2012).

3.4.2 Different perspectives on the model

The LFL model can be seen as an application of the log-linear framework to dyadic prediction. But there are other ways to think of the model also, which may prove useful when designing extensions to it. Below, we outline three alternate perspectives.

A factorization perspective

We show how the LFL model can be thought of as type of matrix factorization. The model of Equation 3.2 may be rewritten as

$$\log \frac{\Pr[y|x=(i,j);\theta]}{\Pr[y'|x=(i,j);\theta]} = W_{ij}^{(y)} - W_{ij}^{(y')},$$

where each $W^{(y)}$ is by Equation 3.3 a low-rank matrix. That is, we model the pairwise log-odds by the difference of two low-rank matrices. Compare the above to multinomial logistic regression, where we have

$$\log \frac{\Pr[y|x;\theta]}{\Pr[y'|x;\theta]} = (w^{(y')} - w^{(y')})^T x.$$

Indeed, when side-information is present, it can be thought of as providing an extra linear discriminant, as in the standard setting of log-linear models and logistic regression. Specifically, suppose that we have side-information in the form of a vector s_{ij} for a dyad (i, j). Then, the resulting model is an additive combination of the logistic regression and low-rank model of the pairwise log-odds:

$$\log \frac{\Pr[y|x=(i,j);\theta]}{\Pr[y'|x=(i,j);\theta]} = W_{ij}^{(y)} - W_{ij}^{(y')} + (w^{(y)} - w^{(y')})^T s(x).$$

As discussed in Section 3.5.2, it is useful to define a base class for the probability model, which involves setting $W^{(y_{\text{base}})} = \mathbf{0}$ for some arbitrary outcome y_{base}. Then, observe that the log-odds relative to this base class are exactly modelled as a low-rank matrix, since $\log \frac{\Pr[y|x=(i,j);\theta]}{\Pr[y_{\text{base}}|x=(i,j);\theta]} = W_{ij}^{(y)}$.

It is also possible to interpret the model as a kind of *tensor* factorization. Let

$$P_{ijy} := \Pr[y|x=(i,j);\theta] = \frac{\exp(W_{ij}^{(y)})}{\sum_{y'} \exp(W_{ij}^{(y')})}$$

denote a $I \times J \times |\mathscr{Y}|$ tensor. Then, the LFL model says that

$$\log \frac{P_{ijy}}{P_{ijy_{\text{base}}}} = W_{ij}^{(y)} = \sum_{k=1}^{K} u_{ik}^{(y)} v_{jk}^{(y)}.$$

That is, we are implicitly defining a factorization of the tensor P in terms of parameters $\{(U^{(y)}, V^{(y)})\}_{y \in \mathscr{Y}}$. This interpretation suggests alternate decompositions of $W^{(y)}$, which will explore in Section 3.5.1.

A supervised learning perspective

As a complement to the factorization perspective above, we now show how the LFL objective may be interpreted in terms of supervised learning. Rewrite the objective

function of Equation 6.3, where for simplicity we drop bias terms:

$$\min_U \left[\sum_{i \in I} \sum_{y \in \mathscr{Y}} \frac{\lambda}{2} ||u_i^{(y)}||_2^2 + \min_V \left\{ \sum_{j \in J} \sum_{y \in \mathscr{Y}} \frac{\lambda}{2} ||v_j^{(y)}||_2^2 + \sum_{t=1}^{T} \log \frac{\exp(W_{i^{(t)},j^{(t)}}^{(y^{(t)})})}{\sum_{y' \in \mathscr{Y}} \exp(W_{i^{(t)},j^{(t)}}^{(y')})} \right\} \right].$$

Recall that the W terms are linear in both U and V. Thus, for fixed U, the optimization with respect to V can be seen as an instance of regularized multinomial logistic regression, with the U's playing the role of the features. The same holds true for fixed V, with the roles merely being reversed. This is characteristic of factorization methods, which can be seen as simultaneously learning a predictive latent feature representation, *and* weights on these features.

A probabilistic perspective

Recall that in Section 3.3.3, we presented the training objective as a form of regularized log-likelihood maximization. Equivalently, we can think of the objective as finding the maximum a posteriori (MAP) estimate under the probabilistic model

$$y|x = (i,j), \theta \sim \text{Discrete}(\{p_{ij}^{(y)} : y \in \mathscr{Y}\})$$
$$\theta \sim \mathscr{N}(0, \lambda^{-1} I),$$

where

$$P_{ij}^{(y)} = \frac{\exp(W_{ij}^{(y)})}{\sum_{y'} \exp(W_{ij}^{(y')})}$$

given $W_{ij}^{(y)}$ as defined in Equation 3.3.

The discrete distribution is equivalent to a multinomial with a single trial. Thus, we observe that the label for each dyad (i, j) is assumed to follow a multinomial whose parameters are constrained to be low-rank. This is as first glance reminscent of the use of

latent variables for topic modelling. However, as we discuss in Section 3.6.3, the two approaches are subtly different.

3.4.3 Do we get meaningful probabilities?

A crucial property of using log-likelihood as our objective function is that it will recover the true probabilities $\Pr[y|x]$ in the limit of infinite samples. To see this, consider the generalization error of an estimator $f : \mathscr{X} \times \mathscr{Y} \to [0,1]$, which aims to return the probability estimate for a label y on instance x:

$$\mathbb{E}_x \mathbb{E}_{y|x}[-\log f(x,y)] = \mathbb{E}_x \left[\sum_{y \in \mathscr{Y}} -\eta(y;x) \log f(x,y) \right],$$

where $\eta(x,y) := \Pr[y|x]$, the true conditional class probability. For a fixed x, the optimal prediction can be checked to be $f(x,y) = \eta(x,y)$, i.e. the optimal estimator is to exactly return the true probabilities. This fact can be seen as a consequence either of using the log-loss, which is *Fisher-consistent* or *proper* (Buja et al., 2005) in the sense of precisely recovering $\eta(x,y)$ in the infinite sample limit, or simply of using maximum likelihood estimation for a well-defined probabilistic model for the labels (Zhang, 2004b).

Observe that the above did not make any special assumptions on $\mathscr{Y}$. It holds if the labels are binary, nominal, multidimensional, and so on. The nature of the labels can be thought to affect the rate of convergence, however (e.g. one would expect a slower convergence rate for multidimensional problems).

3.5 Extensions and variations on the LFL model

Having presented the basic components of our latent feature model, we now discuss some important extensions to it, which can improve performance in practice.

3.5.1 Alternate factorizations

Recall our initial specification for the weights $W^{(y)}$ from Equation 3.3, for simplicity ignoring the bias terms:

$$W^{(y)} = (U^{(y)})^T V^{(y)}.$$

This factorization keeps a separate weight for each possible outcome $y \in \mathscr{Y}$. The decomposition must have *some* dependence on y, else it cancels when we form the probability model. In the above, both U and V depend on y. We can still maintain a dependence on y while cutting down the number of parameters, simply by setting $U^{(y)} = U$, yielding:

$$W^{(y)} = U^T V^{(y)}.$$

The resulting model is of course less powerful, by virtue of having fewer parameters, but may be easier to fit for the same reason. Intuitively, it is reasonable that in many applications, label-specific weights are not needed for both U, V. For example, in collaborative filtering, it is plausible that a user's representation in latent space is independent of the label under consideration, while movies have label specific weights, reflecting different traits that matter for different ratings.

A further reduction in parameters is possible. Note that setting $V^{(y)} = V$ above is not allowed, because then the term $U^T V$ would cancel in the probability model. Instead, we can set

$$V^{(y)} = V \odot (\lambda^{(y)} \mathbf{1}_N^T)$$

for some scalar $\lambda^{(y)} \in \mathbb{R}$. The decomposition becomes

$$W_{ij}^{(y)} = \sum_{k=1}^{K} u_{ik} v_{jk} \lambda_k^{(y)}, \tag{3.7}$$

which is reminiscent of the CANDECOMP model for tensor factorization (Carroll and Chang, 1970). Indeed, this relationship arises from thinking of the data as comprising an $I \times J \times |\mathscr{Y}|$ tensor, as described in Section 3.4.2. Note however that unlike a standard tensor factorization, the LFL model keeps an explicit probability distribution over the elements of $\mathscr{Y}$, which comprise the third mode of the tensor. This lets it keep track of the fact that in nominal settings, for example, one and only one entry per row-column pair (i, j) is active.

As noted, the original factorization of 3.3 is the most general, and hence most powerful. The subsequent simplifications attempt to reduce the risk of overfitting, while still providing a rich enough model. Whether or not this is possible is of course problem dependent. We will empirically compare these schemes in a collaborative filtering context in Section 4.8.1.

3.5.2 Fixing a base class

In multinomial logistic regression, it is standard to fix one label outcome to be the "base" against which all other outcomes are measured. The reason is as follows. Recall that the multinominal logistic regression model is

$$\Pr[y|x;\theta] \propto \exp((w^{(y)})^T x).$$

Observe that translating $w^{(y)}$ by any vector v yields an identical solution in terms of the probability model. This is known as an *identifiability* problem: there is no unique global optimum to the likelihood under this model. To correct for this, it is generally assumed that $w^{(y_{\text{base}})} = \mathbf{0}$ for some fixed $y_{\text{base}} \in \mathscr{Y}$, so that the model may be written as

$$\Pr[y|x;\theta] = \frac{\exp((w^{(y)})^T x)}{1 + \sum_{y \neq y_{\text{base}}} \exp((w^{(y)})^T x)}.$$

For the LFL model, using the general expression for W_{ij}^{y} from Equation 3.3, we note that there is no similar identifiability problem for the U or V parameters. The reason is that since both depend on the outcome y, a translation of either yields a distinct model, e.g.

$$(u_i^{(y)} + c)^T v_j^{(y)} = (u_i^{(y)})^T v_j^{(y)} + c^T v_j^{(y)},$$

where the second term depends on y and so does not cancel with the denominator. However, if we use the factorizations introduced above in Section 3.5.1, where only one term depends on y, the identifiability problem arises. In such cases, as with multinomial logistic regression, we fix one class as the base. Note that in the case of binary $\mathscr{Y}$, this reduces the model to a simple form. For example, using the factorization of Equation 3.7, we get

$$\Pr[y = 1 | x = (i, j); \theta] = \frac{1}{1 + \exp(-u_i^T \Lambda v_j)},$$

which is analogous to logistic regression with latent features.

3.5.3 Finer-grained weights for side-information

In Section 3.3.2, we showed a simple way to incorporate side-information into the LFL model. This relied on a particular choice of feature representation s_{ij} for the dyad (i, j). In particular, we merely concatenated the dyad member specific explicit features. This is potentially sub-optimal for the same reason the basic log-linear model of Section 3.2 was: if we ignore the latent features, the weights on the explicit features will not learn a rich ranking over dyads. To overcome this, we can consider the alternate feature representation

$$s_{ij} = \begin{bmatrix} p_i \otimes q_j & r_{ij} \end{bmatrix}.$$

This turns out to be equivalent to a bilinear regression model (Section 2.6.3) for the dyad member specific features, as we can write

$$(w^{(y)})^T s_{ij} = p_i^T \tilde{W}^{(y)} q_j.$$

Another extension is to have a separate weight vector for each dyad member:

$$\Pr[y|x=(i,j);\theta] \propto \exp\left((u_i^{(y)})^T v_j^{(y)} + (w_j^{(y)})^T p_i + (w_i^{(y)})^T q_j + a_i^{(y)} + b_j^y + \mu^{(y)}\right).$$

This is akin to a random effects model (Appendix C). Compared to the basic model of Equation 3.5, we see that the term $(w^{(y)})^T p_i$ is replaced by $(w_j^{(y)})^T p_i$. Thus, each dyad member j is allowed to have potentially *different* weights on the side-information. Of course, a concern is that we may not have enough data to sufficiently model these fine-grained differences. Thus, as with a random-effects model, it is advisable to add an offset that captures the global effects.

3.6 Comparison to existing models

While the LFL model for dyadic prediction is to our knowledge new, it certainly has precedents and connections to existing models. We now describe its relation to this previous work.

3.6.1 PCA and probabilistics variants

As discussed in Sections 3.3.1 and 3.4.2, a key component of LFL is the use of low-rank factorization. Perhaps the first contrast to be made is to classical principal component analysis (PCA), which models a matrix $X \in \mathbb{R}^{I \times J}$ as a low-rank factorization

$$X_{ij} = u_i^T v_j + \mu.$$

One difference is immediate, namely, the fact that we work in the log-odds domain. Specifically, in the case where $\mathscr{Y} = \{0,1\}$, fixing $y = 0$ as the base class and defining a matrix P by $P_{ij} = \Pr[y = 1|(i,j)]$, the basic version of LFL model becomes

$$P = \sigma(U^T V),$$

where $\sigma(\cdot)$ is the sigmoid function, or equivalently,

$$\log \frac{P_{ij}}{1 - P_{ij}} = u_i^T v_j.$$

Thus, we see that unlike PCA, we do not directly impose a rank constraint on the raw input data. This is akin to the difference between linear and logistic regression in supervised learning.

While the above shows the model is distinct to classical PCA, as stated it is identical to *logistic* PCA, which defines a probabilistic version of PCA for Bernoulli random variables (Schein et al., 2003). The binary version of LFL is slightly different for a couple of reasons, however. First, it provides a means to learn from covariates for the rows, columns and/or cells of the matrix. Second, it explicitly models bias terms for the row and column, which have been found to be beneficial for predictive performance in applications such as collaborative filtering (Koren and Bell, 2011). Third, the training procedure for LFL is much simpler, as it involves SGD on the penalized likelihood. Fourth, our model seamlessly handles missing data, without requiring EM or similar. This said, we do not wish to overstate these points, and prefer instead to think of LFL as a generalization of this model.

Further generalizations of PCA exist for multinomial observations (Buntine, 2002), and more generally members of an exponential family (Collins et al., 2001).

However, as noted in Section 3.4.2, the LFL model strictly provides a *tensor* rather than matrix factorization. In particular, for models such as (Buntine, 2002), each row of the matrix is assumed to be a draw from a multinomial, whereas in LFL, each *dyad* is assumed to be a draw from a multinomial. Aside from this, all the differences between LFL and logistic PCA hold for these models also.

3.6.2 Statistical network models

In the binary case, LFL is also similar to some statistical models that have been proposed in the social network analysis literature, for example (Hartzel et al., 2001; Coull and Agresti, 2003; Hoff, 2006; Meeds et al., 2006; Miller et al., 2009). The mathematical form of these models is nearly identical to ours in the binary case. In particular, our use of a linear combination of latent features and side-information is inspired by these papers. However, there are some differences. First, our model addresses the general dyadic prediction task, with the dyad members belonging to the same space as a special case. (In Chapter 5, we will explore the latter problem more.) Second, our treatment of the model is not fully Bayesian. Thus, our training procedure is considerably simpler than the MCMC scheme used in these papers. Third, as our emphasis is on predictive accuracy rather than qualitative analysis, we will explore in Chapters 4 and 5 different training objectives beyond just log-likelihood maximization.

3.6.3 Other models

The LFL model is similar to the nominal response model of (Bock, 1972), proposed for item response theory. However, there are sveral important differences, such as our use of a range of different factorizations or scoring functions, the use of *multiple* latent features rather than just one, our ability to exploit side-information, our use of regularization to prevent overfitting, and our training procedure relying on SGD.

In Section 3.4.2, we showed how the LFL model can be interpreted as the result of a generative process comprising a draw from a multinomial. We note that this is distinct to the canonical application where a similar model is adopted, namely, topic modelling. Here, the goal is to model the occurrences of words in a collection of documents. This is ostensibly a dyadic modelling problem, where each dyad is a (document, word) pair, with the label being binary, indicating whether or not the word appears in the document. (This is also known as *co-occurrence* data (Dagan et al., 1994).) As dyadic modelling allows for multiple occurrences of a dyad, this can handle the case of multiple occurrences of a word in a document. However, it is crucial in topic modelling that the label for a dyad is binary. It is not possible to directly handle cases of nominal or otherwise non-binary labels[1], for example preference ratings in collaborative filtering. Indeed, the document-word (count) matrix relies on this fact, because otherwise the aggregation procedure is not obvious. To see this another way, consider a mixture of unigrams model for the words $w_1, \ldots, w_N$ in a document d:

$$\Pr[w_1, \ldots, w_N | d] = \sum_z \Pr[z|d] \prod_{n=1}^{N} \Pr[w_n|z],$$

where z here denotes the latent mixture component. The distribution $\Pr[w_n|z]$ is a multinomial, representing a draw from the underlying vocabulary. If we think of documents as being users and words as being movies, what this model gives us is only a way to generate a set of movies for a user, being the movies that the user likes, say. It does not directly give a way to model a quantity specific to *both* a word w_n and the document d. (Indeed, the fact that LFL does this is why we were able to interpret it as a tensor factorization in Section 3.4.2.) Thus, while ostensibly similar, there are important differences to the type of dyadic prediction problem we consider here.

[1]There do exist modifications of topic models for such settings, however (Marlin, 2004; Porteous et al., 2008).

3.7 Experimental design

We now discuss the design of our experiments that test the efficacy of the above model.

3.7.1 Aims of the experiments

Given the generality of the model proposed above, an exhaustive experimental comparison is difficult. For one, as noted earlier, each prominent instance of dyadic prediction has a range of different methods that are popular. Therefore, we will defer detailed comparisons to the relevant chapters that study these problems in more detail. At this stage, we wish to answer three questions:

- *Are local optima a concern?* The objective for the LFL model is non-convex. A natural question is whether these a significant concern, or whether the model tends to have many "good" local optima. We study this with experiments on a synthetic dataset, where we study the ability of the model to recover the true class conditional probabilities.

- *Is the model powerful?* Certainly a minimum requirement from any dyadic prediction model is that it be powerful enough to accurately predict labels for unseen dyads. We study this with experiments on a basic matrix completion task.

- *Is the model flexible?*, and *Does it scale?* Finally, we see whether the model is flexible enough to adapt to the requirements of a real-world problem, and whether it can meet the scale of such a problem. We study this with experiments on an item response theory dataset, which includes side-information.

3.7.2 Hyperparameter selection procedure

We have explained how our model may be trained using a gradient following method, such as SGD. In principle, this is simple: for appropriate tuning or hyperparameters, we compute the gradients with respect to each example one at a time, and update our parameters. In practice, the task of determining appropriate hyperparameters is not so simple, for the basic reason that there are many of them, each of which has a range of sensible values. In particular, we observe that there are *model specific* hyperparameters in the strength of regularization, and the strategy for initializing the parameters, which generally involves specifying their scales. In addition to these, there may be *optimization specific* hyperparameters, such as the learning rate and the number of iterations or epochs to perform. For each of these hyperparameters, it is beneficial to keep separate values for each different type of model parameter: the latent feature weights U and V, the bias terms a and b, and so on. (Using a single value for all parameters generally gives significantly sub-optimal results.) We thus find ourselves having to search over a grid for 10-15 hyperparameters. This is computationally infeasible on even moderately sized datasets.

One way to speed up the grid search is to subsample the dataset used for model selection. This is useful, and we did employ this on larger datasets. But even with subsampling, the cost of an exhaustive search can be prohibitive. To overcome this, we found it useful to resort to a strategy inspired by (Bergstra and Bengio, 2012), who address a similar problem that arise in neural networks. Suppose we have N hyperparameters, with grid ranges $\{G_i\}_{i=1}^{N}$. An exhaustive grid search involves evaluating the model for every hyperparameter setting in $\mathscr{P} = \times_{i=1}^{N} G_i$. As a computationally tractable alternative, we pick some $n \ll |\mathscr{P}|$ hyperparameters $\{p_i\}_{i=1}^{n}$ uniformly at random from $\mathscr{P}$. Note that each p_i is an N-tuple, comprising the settings for each hyperparameter. For each such p_i,

we perform a type of coordinate descent: we cycle through each hyperparameter one at a time, and then perform a one-dimensional grid search to try to find a better setting for it. By picking n so as to encourage reasonable diversity in the search space, we can expect this procedure to provide a reasonable approximation to the exhaustive grid search.

We have found the above scheme to provide good tuning parameter settings at a fraction of the time required for an exhaustive grid search. As potential future work, it would be interesting to evaluate direct search techniques such as compass search and the Nelder-Mead method (Kolda et al., 2003) for hyperparameter optimization.

3.7.3 Practical details on training procedure

We identified that there are a few potentially important ingredients in training the LFL model. We discuss two of these in more detail, as we have found them to have a significant impact on performance: the choice of learning rate, and the scheme to initialize the model parameters.

The general SGD update equation at the tth iteration is (see Section D):

$$\theta_{t+1} = \theta_t - \eta_t \nabla \ell(x_{i(t)}, y_{i(t)}; \theta_t) - \eta_t \cdot \nabla r(\theta_t).$$

The sequence of learning rates $\{\eta_t\}$ is generally varied according to a *schedule function*, which specifies the value of η_t in terms of the initial learning rate η_0, and the current iteration t. To prove theoretical convergence, one needs the sequence of learning rates to satisfy $\sum_{t=1}^{\infty} \eta_t = \infty, \sum_{t=1}^{\infty} \eta_t^2 < \infty$. In practice, several different choices of schedule are popular, such as:

- *Constant schedule*. Here, we set $\eta_t = \eta_0$ for every t, so that the learning rate does not change at all. An issue with this is that we will likely have difficulty at later stages of the optimization, in particular the risk of oscillating around the optimum.

- *Exponential dampening*. Here, we set $\eta_t = c^t \eta_0$ for some constant $0 < c < 1$ that must be specified. This scheme has been successful in collaborative filtering applications (Koren, 2008; Yang et al., 2011).

- *Inverse dampening*. Here, we set $\eta_t = \frac{\eta_0}{t}$. This satisfies the theoretical requirement of rate of decay of learning rates.

- *Inverse square root dampening*. Here, we set $\eta_t = \frac{\eta_0}{\sqrt{t}}$.

For any of the above schedules, it is also possible to apply the dampening at the end of every *epoch* (pass of the training data), rather than at the end of every parameter update. This is equivalent to considering the meta-schedule $\tilde{\eta}_t = \eta_{\lfloor t/N \rfloor}$.

For the initialization of parameters, a standard choice is to draw each parameter from $\mathcal{N}(0, \sigma^2)$, where $\sigma \in \{10^{-2}, 10^{-1}, 10^0\}$. For the LFL model, we found it useful to initialize weights so that the initial predicted label distributions are close to the overall empirical label distribution. That is, for every dyad (i, j), we would like the initial distribution over labels to approximately match the overall distribution over labels. To accomplish this, consider the scoring function $W_{ij}^{(y)} \propto \exp(u_i^T \Lambda^{(y)} v_j)$. (It is not hard to derive similar initializations for other scoring functions.) We can set

$$\Lambda^{(y)} = \ln \frac{c^{(y)}}{c^{(y_{\text{base}})}} + \varepsilon,$$

where $c^{(y)}$ denotes the empirical occurrence of the label y, and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. (Recall that y_{base} refers to the base class, for which we set $\Lambda^{(y_{\text{base}})} = 0$.) To see the effect of this choice, note that in the noiseless case,

$$\exp(W_{ij}^{(y)}) = \left(\frac{c^{(y)}}{c^{(y_{\text{base}})}} \right)^{u_i^T v_j}.$$

So, if we pick u, v so that $u_i^T v_j = 1$, the unnormalized probability for the label y will be $c^{(y)}/c^{(y_{\text{base}})}$, and so the normalized probability will exactly match the empirical frequency of label y. To pick such u, v, one can set e.g.

$$u_i = \frac{1}{\sqrt{k}}\mathbf{1} + \varepsilon_i$$
$$v_i = \frac{1}{\sqrt{k}}\mathbf{1} + \varepsilon_j$$

where again $\varepsilon_i, \varepsilon_j$ are Gaussian noise.

3.7.4 Implementation details

We implemented our model in MATLAB. For the first two experiments, we trained using LBFGS (Nocedal and Wright, 2006) as our optimizer. This does not scale to very large datasets, but is a reasonable solution for medium sized problems. We wish to emphasise that there is no reason why stochastic gradient descent cannot be applied for all experiments: the only disadvantage is that it requires tuning of the learning rate. Note that the final experiment demonstrates the scalability of the model on a large dataset, and here we train using stochastic gradient descent.

Since the optimization problem is nonconvex, there is variability in our final accuracy given the initialization of weights: different initializations lead to potentially different local optima. We account for this by constructing a meta-model, which runs our model some fixed number of times (generally 3 or 5) with different random initializations, and then reports the average prediction from each such run as the meta prediction. Note that as the model is generally not identifiable, it does not make sense to directly average the model parameters from different runs.

All experiments were run on a 2.7GHz Core i7 machine with 12 GB of RAM.

3.8 Experimental results

We now present our results that address in turn each of the main questions highlighted in the previous section.

3.8.1 Are local optima a concern?

We ran experiments on a synthetic dataset to check that we can learn from nominal data, and that it is possible to find good local optima of the objective function, despite its non-convexity. We constructed a matrix whose entries were in $\mathscr{Y} = \{1,2,3\}$. Each of these can be thought of as an index into some set of categories, e.g. {bought, viewed, returned}. We picked the entries of the matrix M by sampling

$$M_{ij} \sim \text{Disc}\left(\left\{\frac{\exp((u_i^{(y)})^T v_j^{(y)})}{\sum_{y' \in \mathscr{Y}} \exp((u_i^{(y')})^T v_j^{(y')})}\right\}_{y \in \mathscr{Y}}\right),$$

where there are $k = 5$ latent factors for the U and V matrices. We drew each element of U, V from Unif$([-3,3])$. We set some fraction of entries to be unobserved, which were used for testing, and let the remaining entries form the training set. The goal of training is to maximize the log-likelihood, which means we try to discover good values for U, V. Two parameters that will have an influence on the quality of the learned model are the size of the matrix, n, and the retention rate i.e. the fraction of entries kept for training.

An important question is how to assess the quality of our learned model. The simplest measure is 0-1 accuracy: we penalize any entry for which we do not predict exactly the right value. But such a result is only meaningful if we know the scale against which we measure 0-1 accuracy. Since we are explicitly sampling from a probability distribution with known parameters, we can find the Bayes error rate for the data. We compute the Bayes error for a single matrix entry by $1 - \max_y p(y|i,j)$. The mean Bayes

error rate gives us an idea of how difficult the problem is. If we are able to get close to this rate in terms of 0-1 error, our model is doing well.

The other question of interest is how well a method for ordinal labels performs on this dataset. Intuitively, because such a method imposes an artifical structure on the outcomes, it will be difficult to learn a good model.

Our results are presented in Table 3.1. We see that for varying choices of n and the retention rate, we are able to learn a model that has high 0-1 accuracy. Our error rate is closest to the Bayes error when our training set is large: this is intuitive, because we expect the learning task to be simpler with access to more samples. Another promising result is that the accuracy of the log-linear model increases with the size of the training data, *despite* the increase in missing data.

Table 3.1. Accuracy of LFL model on synthetic nominal dataset.

n	k	Retention	Bayes	Log-linear
500	5	80%	4.8%	7.7%
500	5	50%	4.8%	8.2%
500	5	25%	4.8%	12.0%
1000	5	80%	4.8%	5.9%
1000	5	50%	4.8%	6.3%
1000	5	25%	4.8%	8.1%
1500	5	80%	4.8%	5.5%
1500	5	50%	4.8%	5.9%
1500	5	25%	4.8%	7.0%

3.8.2 Is the model powerful?

We ran an experiment on the usps dataset of handwritten digits USPS (2010), following (Meeds et al., 2006). The point of this experiment is to show that our model is powerful enough to learn something meaningful from the data. To begin, we focus on digits $1, 2$ or 3 and pick 100 random examples of each digit. The images are binarized so that the pixel values are either $+1$ or -1. For 16 of these images, we chop off their

bottom half by replacing the pixel values with 0. We then view the dataset as a 300×256 matrix where the 0 pixel values are treated as missing entries. The goal is to fill in the missing entries, or equivalently, to reconstruct the bottom half of the 16 corrupted digits. We can view this problem as an instance of dyadic prediction: we treat each digit as our row object, and each pixel position as our column object.

We applied our latent feature log-linear model on this dataset using 3 latent factors[2] and mild regularization ($\lambda = 0.5$). We optimized MAE as the objective function using LBFGS, which was feasible given the small size of this dataset. Our results are shown in Figure 3.1. In the figure, the top row shows the original versions of the 16 corrupted images. The middle row shows the data as presented to the training algorithm, with the bottom halves chopped off. The last row shows the predictions made by our model. (For this row, even the top half is the model prediction.) We see that it is able to accurately reconstruct most of the images. (Quantitatively, it achieves an area under the ROC curve of 0.9122.) One exception is an image whose true digit is 3, but which our model reconstructs as 1. Looking at the corrupted version of the image, this behaviour is very understandable: it is diffcult for a human to deduce the correct digit in this instance, because there is not enough information in the top half.

3.8.3 Application to IRT: does the model work in practice?

We report results on a real-world item response theory dataset, from the Kaggle "What Do You Know" challenge (Kaggle Inc., 2012). The challenge involved a dataset from GrockIt[3], comprising student responses to examination questions. The data comprises the responses of 138993 students to 6045 questions. In total, there are 4266137 responses in the training set, each of which is annotated with a label denoting whether the student answered the question correctly or not. For questions that are answered

[2]Recall that we include bias weights, meaning we are working with 5 dimensional vectors.
[3]https://grockit.com/

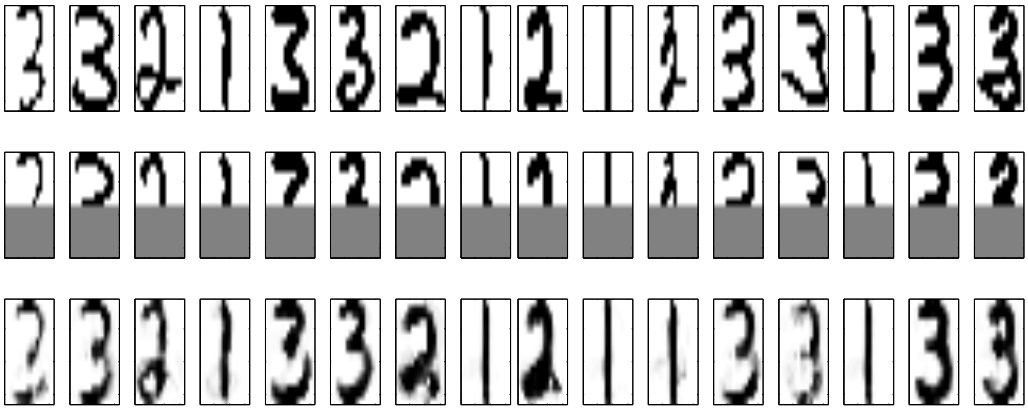


Figure 3.1. Results on USPS dataset. **Top**: Random images chosen for corruption. **Middle**: Images with bottom half removed. **Bottom**: Reconstructed images using log-linear model, rank $k = 3$.

incorrectly, it is additionally known if the reason is because the question was skipped or due to a timeout. The task is to predict whether or not a student will correctly answer a new question in the future. This is an instance of dyadic prediction where the dyads are (student, question) pairs, and the labels are whether or not the student answered the question correctly.

During the contest, a leaderboard was maintained to rank teams. This leaderboard was based on held-out data known as the *public* test set. When the competition concluded, the winner was chosen on a separate set, known as the *private* test set. The performance metric on both sets was *capped* Bernoulli log-likelihood, defined as

$$\ell(y,\hat{y}) = -y\log C(\hat{y}) - (1-y)\log(1-C(\hat{y}))$$

where the capping function $C(x) = \min(\max(x, 0.01), 0.99)$. The contest description refers to the metric as *binomial capped deviance* (BCD), and we follow suit henceforth. The baseline provided in the contest was the `lme4` model in `R`, which implements a binary Rasch model, or equivalently a linear mixed-effects model (Bates, 2010). We hencefort

refer to this as the "IRT benchmark".

We look to see how the basic LFL model fares compared to the baseline, as well as compared to other models that were employed in the competition. We implemented the basic LFL model with $k = 10$ latent features on this dataset, including bias terms for students and questions. Explicitly, our predicted probability for student i answering question j correctly is

$$\Pr[y = 1 | x = (i, j); \theta] = \sigma(u_i^T v_j + a_i + b_j + \mu)$$

where $u_i, v_j \in \mathbb{R}^{10}$. We optimized the LFL model for Bernoulli log-likelihood[4] with ℓ_2 regularization of all parameters. We only performed one dataset-specific action, namely, we removed all dyads where the response was incorrect due to a skip or timeout. This is because the test set does not include such outcomes, and we found performance to worsen otherwise. Apart from this, no special effort was made to tune the model to the details of this dataset.

Table 3.2 gives the results of methods on both the public and private test sets. We see that the LFL model manages to reliably outperform the provided IRT baseline on both sets. While the difference in BCD scores are $\sim$0.5%, the final standing of this model is rank #38 out of 241 on the public, and #33 on the private leaderboard. We also see that adding side-information gives a small boost in performance on the public test, but essentially none on the private set. Figure 3.2 shows the top 15 weights learned on the individual features. We see that the weights are generally not that large, indicating that most of the predictive power comes from the latent feature component. This is not surprising, as the side-information only provides basic information as to the nature of the question being answered. Also as expected is the fact that using just the side-information

[4]It is in principle possible to optimize for the BCD directly, but we found almost all our model predictions to be in the range $[0.01, 0.99]$, so this will likely have marginal additional impact.

alone as input to logistic regression performs significantly poorer than the LFL method without side-information. Comparing the learned weights on the features in this setting to those when we include latent features (Figure 3.3), we see that generally latent features absorb much of the signal that is otherwise captured by question specific features, in particular features such as the number of players involved in answering a question and whether the question was part of a competitive game.

A more careful use of side-information in LFL, along with blending with other methods, managed to finish at rank #4 on the public and #5 on the private leaderboard (Anil, 2012). Our goal here is merely to demonstrate that our log-linear framework is powerful enough to be useful in real-world settings, and that with minimal tuning it is competitive with state-of-the-art approaches for particular dyadic prediction problems.

Table 3.2. BCD scores on the public and private test sets, GrockIt dataset.

Model	**Public test score**	**Private test score**
IRT Benchmark	0.25663	0.25766
Logistic regression	0.30511	0.30420
LFL	0.25562	0.25633
LFL + Side-information	**0.25543**	**0.25632**

We also note that with 4M dyads in the training set, the GrockIt dataset is large enough to cause scalability problems for batch methods. (We will later study performance on several orders of magnitude larger datasets.) Our use of stochastic gradient descent for optimization obviates this issue: our MATLAB implementation of LFL took around 1 second to complete an epoch, and converged in less than 10 epochs. Further, and more importantly, the models learned even after a few epochs achieved good generalization performance. Figure 3.4 shows a learning curve on the training and test set created for validation purposes. We see that within a couple of epochs, the model finds a solution that generalizes well. Further epochs improve performance, but only marginally. The learning curve also shows that while the training and test scores follow the same general trend,

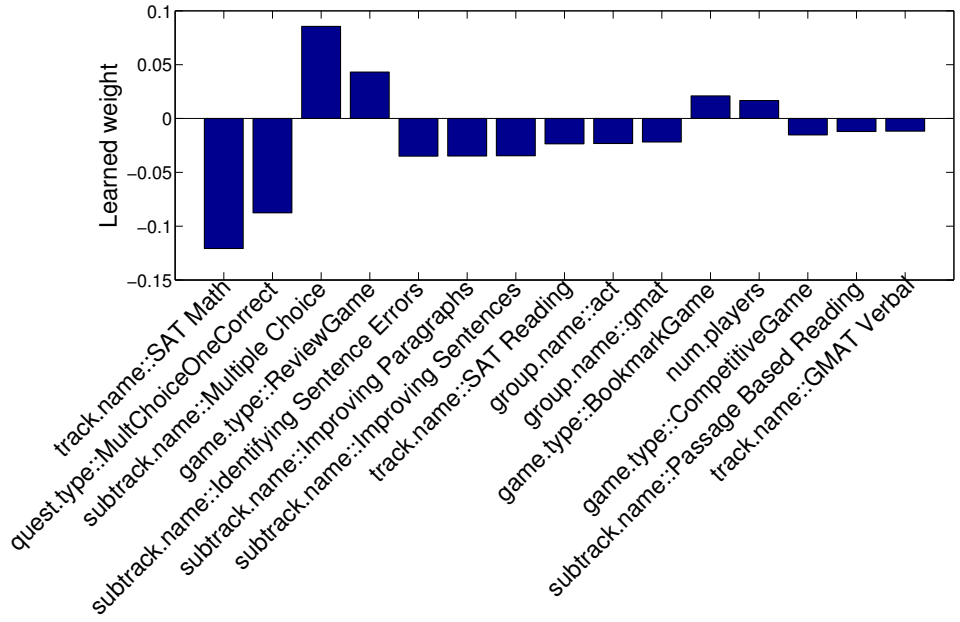


Figure 3.2. Weights learned on explicit features as offset to latent features, GrockIt dataset.

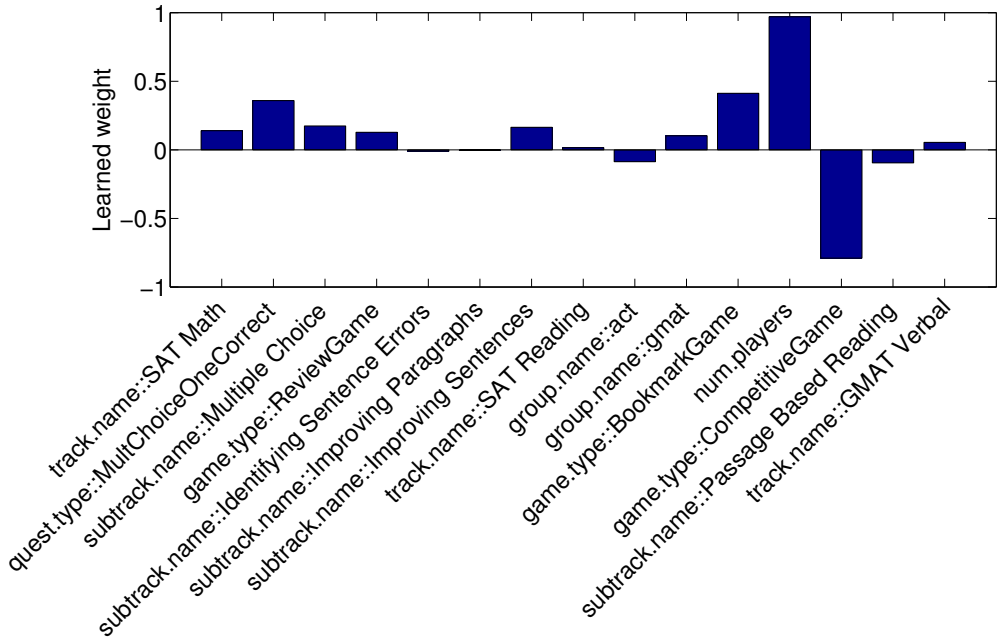


Figure 3.3. Weights learned on explicit features without latent features, GrockIt dataset.

there is a noticeable gap between them. This reflects that the train and test distributions are not identical. We found it to be difficult to reduce this gap with regularization without compromising the raw test set BCD score.

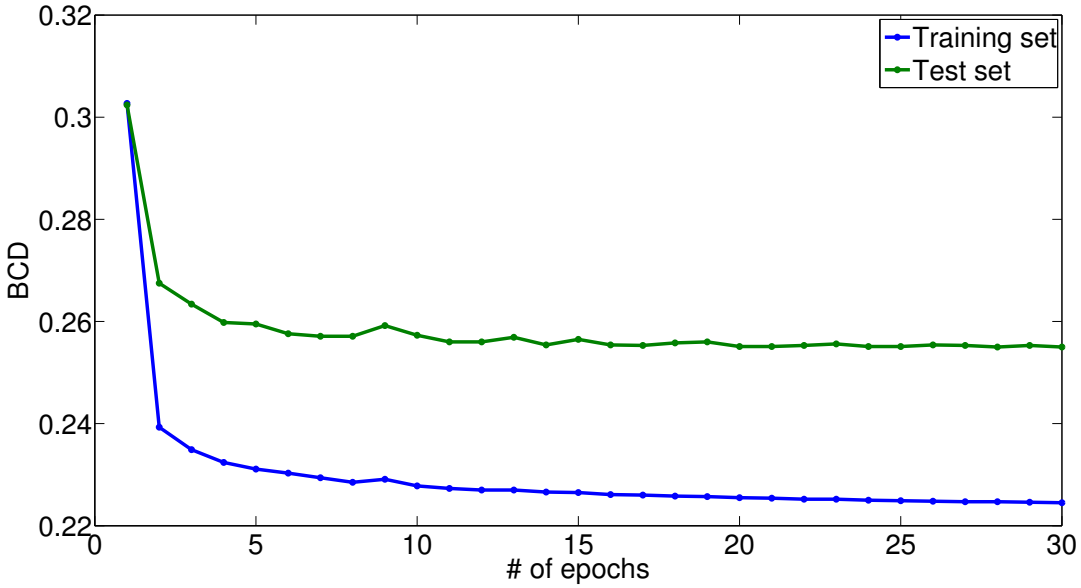


Figure 3.4. Learning curve for LFL model on GrockIt dataset.

3.9 Conclusion

In this chapter, we have presented LFL, a log-linear model for dyadic prediction. The salient features of this model are:

- It learns latent features for each dyad member to explain their corresponding labels, endowing it with strong predictive power.
- It models the entire probability distribution of labels for a particular dyad, which may be desirable in some applications.
- Side-information can be incorporated seamlessly, if it is present.
- Minimal assumptions are placed on the nature of the labels, making it applicable to a range of dyadic prediction tasks.

- It can be trained with stochastic gradient descent and so is scalable to datasets with several million observations.
- It is conceptually simple, which reduces the barrier for adoption in real-world applications.

Our experimental results show that we can learn from nominal and binary data. We now turn to studying in detail its applicability to several important instantiations of dyadic prediction, beginning with collaborative filtering.

3.10 Acknowledgements

Chapter 3, in part, contains material as it appears in "A Log-Linear Model with Latent Features for Dyadic Prediction", IEEE International Conference on Data Mining (ICDM), pages 364-373, 2010. Aditya Krishna Menon, Charles Elkan. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Modelling Rating Distributions: Application to Collaborative Filtering

In this chapter, we show how the LFL model may be applied to collaborative filtering problems. An advantage over existing approaches is that we model the distribution of ratings for each (user, movie) pair. This allows us to measure the uncertainty in the model's predictions, which is useful in practical applications of recommender systems. We explain why the basic LFL model of the previous section may be sub-optimal for collaborative filtering data, and propose modelling changes that address these issues. Experimental results show that the model's predictive accuracy is competitive on benchmark collaborative filtering datasets, and that the framework can comfortably handle cold-start problems. Further, the rating distributions learned by the model allow for qualitative insights into ratings data.

We begin the chapter by studying some reasons why the LFL framework is appealing for collaborative filtering problems.

4.1 Advantages of LFL for collaborative filtering

Recall that in collaborative filtering, our dyads comprise (user, movie) pairs, with the labels being star ratings e.g. $\mathscr{Y} = \{1, 2, \ldots, 5\}$. We discuss two key advantages that

the LFL model provides for such data: the ability to model a distribution over the ratings, and the ability to make predictions in the cold-start scenario.

4.1.1 Modelling rating distributions

A common approach to modelling collaborative filtering data is to treat $\mathscr{Y} \subseteq \mathbb{R}$, and thus reduce the problem to one of regression. This approach lets one leverage the vast body on literature on that topic for matrices, such as probabilistic PCA and trace norm regularization (see Appendix B). For a given dyad, such models return a single number that represents the predicted score. These scores are generally real numbers in the interval $[y_{\min}, y_{\max}]$ (e.g. $[1,5]$ in the case of 1-5 star ratings), and so may be used to rank dyads meaningfully.

Methods based on this principle have been very successful in collaborative filtering. However, conceptually, reducing the prediction for a dyad to a single number loses potentially useful information. In particular, by just returning a number, we do not have any sense of the *uncertainty* the model has in the prediction. There are many possible reasons for such uncertainty: there may be only a few ratings for a movie, a user may have strongly idiosyncratic prior ratings, and so on. Such information is potentially important when using the model's predictions to actually make recommenations. As a simple example, consider the case of two dyads (u,m) and (u,m'), both involving the same user. For the first, suppose that we can determine that our model is equally confident that the rating is 5 or 2 stars. For the second, suppose the model is equally confident of the rating being either 3 or 4 stars. In both cases, the expected rating is 3.5, and if forced to return a single score as predicted rating, this is a reasonable prediction. But intuitively, if we were to recommend one of the two movies to the user, we would likely choose the second one: with the first, there is greater perceived risk that the user will not like the movie at all. (Formally, the variance of the rating distribution for the first movie is

greater than that of the second.) Therefore, in general, possessing such information about the uncertainty in a prediction is important.

The precise mechanism by which a model keeps track of the (un)certainty in its predictions depends on the nature of the model. For example, if the model is based on constructing a neighbourhood of similar dyads, the characteristics of the neighbourhood will largely determine the confidence in a prediction. Our focus will be on modelling the *rating distribution*, $\Pr[y|x;\theta]$, from which we may derive e.g. the variance for a given dyad. We observe that modelling an explicit rating distribution is useful when considering the use of a model in a larger system. For example, suppose that we wish to contact users with suggested movie recommendations. This contact likely has a price, such as the perceived cost of annoyance of the user, or a monetary cost if we physically mail out the recommendations. Of course, there is also a corresponding return associated with such a recommendation, and this depends on how much the user likes the recommendation that was provided. In this case, we would only want to send out recommendations where the *expected* return exceeds the (estimated) cost: so for example, for a fixed x, we may be interested in maximizing

$$\mathbb{E}_{y|x}\left[\max(0, R(x,y) - c(x,y))\right],$$

where $R(x,y)$ is the return for the dyad x if the rating is y, and $c(x,y)$ is the associated cost. Observe that if we have a model for the rating probabilities $\Pr[y|x]$, we can look to directly maximize the above quantity.

Having established the value of keeping a rating distribution, we note that this is precisely what the LFL model of the previous chapter maintains: we can model $\Pr[y|x]$ for each $y \in \mathscr{Y}$ using Equation 3.4. With such a distribution, the estimated variance for a

dyad is

$$\begin{aligned} V_{ij} &= \mathbb{E}[y^2] - (\mathbb{E}[y])^2 \\ &= \sum_{y \in \mathscr{Y}} y^2 \Pr[y|x=(i,j);\theta] - \left(\sum_{y \in \mathscr{Y}} y \Pr[y|x=(i,j);\theta] \right)^2. \end{aligned}$$

As discussed above, such a distribution may also be used as a component in a larger decision making process. In the experiments (Section 4.8), we provide some qualitative analysis of collaborative filtering data through these distributions.

4.1.2 Addressing the cold-start problem

The cold-start problem is where we have to make predictions for a user or movie that does not appear in the training set, meaning they have no ratings from which we can infer something about their personality. As discussed in Section 3.4.1, intuitively, making progress here requires some additional information: if we truly know nothing about a user or movie, it is hard to see how we could make a prediction that is not simply some form of average over all other users and/or movies. An example of potentially useful extra information is side-information for the users and movies. In this setting, we may train our model to learn weights for both the latent and explicit features, as described in Section 3.3.2. Suppose that at test time, we have to predict the label associated with some dyad (i, j), where the user i does not appear in the training set. Let $s_{ij} \in \mathbb{R}^D$ be some vector of side-information for the (user, movie) pair. Recall that the predicted probability under the LFL model is

$$\Pr[y|x=(i,j);\theta] \propto \exp((u_i^{(y)})^T v_j^{(y)} + (w^{(y)})^T s_{ij}).$$

Since we assume that we have side-information available even for the cold-start user, the second term above may be computed with no issues. Thus, all that remains is to specify how to set the latent feature vector $\{u_i^{(y)}\}_{y \in \mathscr{Y}}$. There are several simple choices for this that may make sense. The first choice is to set each $u_i^{(y)} = \mathbf{0}$, so that the prediction for the dyad is *only* determined by the side-information. This choice can be interpreted as the logical conclusion of following the probabilistic interpretation of LFL 3.4.2, in particular, the use of a zero-mean Gaussian prior on all parameters: in the absence of any data, the optimal value for a parameter will of course be the prior mean, which in this case is $\nvDash$.

A simple tweak to the above strategy is to instead use the *average* of all other (warm-start) users' latent feature vectors. That is, we pick

$$u_i^{(y)} = \frac{\sum_{i' \in \mathscr{W}} u_{i'}^{(y)}}{|\mathscr{W}|},$$

where $\mathscr{W}$ denotes the set of warm-start users in the data. As a result of this choice, the first term in the probability model is replaced with the average predicted rating of the model across the warm-start users. This can be expected to be a more reasonable offset for the second term, rather than 0. A further tweak is to exploit the fact that we expect users with similar explicit features to be more indicative of the cold-start user's taste. That is, we can consider weighting the average based on some measure of similarity between two users:

$$u_i^{(y)} = \frac{\sum_{i' \in \mathscr{W}} S_{ii'} u_{i'}^{(y)}}{\sum_{i' \in \mathscr{W}} S_{ii'}},$$

where $S_{ii'}$ measures the similarity between user i and i'; for example, we can use the cosine similarity, $S_{ii'} = \frac{x_i^T x_{i'}}{||x_i||_2 ||x_{i'}||_2}$. This can be seen as a simple approximation to more sophisticated cold-start correction schemes based on imposing a Markov random field prior on the latent features (Agarwal et al., 2009; Gantner et al., 2010a).

4.2 Is LFL appropriate for collaborative filtering?

Above, we have argued that the LFL model has some important characteristics that make it attractive for collaborative filtering problems. However, this discussion was at an abstract level, and ignored some potential issues that arise when we attempt to train such a model on this data. We now specifically discuss such issues, which stem from an important characteristic of labels in collaborative filtering problems.

4.2.1 Ordinal nature of collaborative filtering labels

The labels in collaborative filtering are star ratings, with the number of stars denoting the degree of preference a user has for a movie. Consequently, the labels can be thought to lie, at a minimum, on a Likert scale (Likert, 1932), with the ratings being realizations of an underlying scale such as $\{$ "Hate", "Dislike", "Neutral", "Like", "Love" $\}$. Such a scale is *ordinal* in the sense that there is a total order $\preceq_\pi$ defined on its elements. So, "Hate" $\preceq_\pi$ "Dislike" $\preceq_\pi$ "Neutral", or equivalently, a rating of 1 star $\preceq_\pi$ 2 stars $\preceq_\pi$ 3 stars. In general, an ordinal scale does not ascribe numeric meaning to its outcomes, and so differences between the outcomes may not be meaningful. If we further assume a numeric structure, the result is an *interval* scale, according to the nomenclature of (Stevens, 1946). This means that the difference between 1 and 3 stars (equivalently "Hate" and "Neutral") is twice the difference between 1 and 2 stars (equivalently "Hate" and "Dislike").

Most collaborative filtering methods assume that the ratings like on an interval scale. This is reflected by the loss functions commonly used for training and evaluation, the mean absolute and mean square error (MAE and MSE), which for a prediction of $\hat{y}$ for a true rating of y are $|y-\hat{y}|$ and $(y-\hat{y})^2$ respectively. (The Netflix prize used the RMSE, or root mean square error, to rank contestants.) Strictly, the interval scale is not

justified in general (Jamieson, 2004), and the loss functions may not reflect the real-world penalties of misprediction. For example, predicting 2 stars for a movie that a user really rates 1 stars may not be as bad as predicting 3 stars for a movie the user really rates 4 stars, because, in the latter case, we might not recommend a movie to the user, thus losing a potential opportunity to please them. Nonetheless, given the widespread use of this reduction to an interval scale, it appears that it is a reasonable enough approximation to be useful. (The MSE in particular has been shown to be reasonably correlated with the fraction of items with the top K predicted ratings that the user actually enjoys (Koren, 2010).)

We now discuss whether the LFL model is suited for labels on an ordinal or interval scale. This discussion hinges on two characteristics of the model: the prediction rule, and the log-likelihood objective used for training.

4.2.2 Is predicting the mode appropriate?

Recall the basic LFL prediction rule suggested in Section 3.3.4,

$$\text{Pred}(x;\theta) = \operatorname*{argmax}_{y \in \mathscr{Y}} \Pr[y|x;\theta].$$

This prediction rule is certainly sensible for collaborative filtering problems: it predicts the rating we believe to be the most likely. As discussed in Section 3.3.4, if our objective is to minimize 0-1 error, this is indeed the optimal prediction. But in collaborative filtering problems, such an objective is intuitively too harsh: we will be content with a prediction that is close to the true rating. Thus, it is more common to consider absolute or squared error, which as discussed earlier implicitly assume that the ratings lie on an interval scale. In case of squared error, the optimal prediction is the *expected* rating under

the probability distribution:

$$\begin{aligned}\text{Pred}(x;\theta) &= \mathbb{E}_{y\sim\Pr[y|x;\theta]}[y] \\ &= \sum_{y\in\mathscr{Y}} y\cdot\Pr[y|x;\theta]. \end{aligned} \tag{4.1}$$

In the case of absolute error, the optimal error is the *median* rating under the probability distribution:

$$\begin{aligned}\text{Pred}(x;\theta) &= \text{median}(\Pr[y|x;\theta]) \\ &= \frac{1}{|\mathscr{Y}|}\sum_{y\in\mathscr{Y}} y\cdot\mathbf{1}[\Pr[y|x;\theta]\geq 0.5]\cdot\mathbf{1}[\Pr[y|x;\theta]\leq 0.5].\end{aligned}$$

In general, it is sensible to tailor the predictions of the model to the loss function that is used for evaluation. We will mostly focus on squared error as our objective, and so will largely work with the prediction of Equation 4.1.

4.2.3 Is maximizing log-likelihood appropriate?

The LFL objective of Equation 3.6 was based on maximizing the log-likelihood of the training data. This loss function ignores the ordering amongst the labels in the following sense: when faced with the choice between two competing models, it will not favour the one that makes more sense from an ordinal point of view. Formally, the loss on some labelled dyad $((i,j),y)$ is

$$-\log\frac{\exp(W_{ij}^{(y)})}{\sum_{y'\in\mathscr{Y}}\exp(W_{ij}^{(y')})} = W_{ij}^{(y)} - \log\sum_{y'\in\mathscr{Y}}\exp(W_{ij}^{(y')}).$$

where W are the model parameters. Now consider a different solution W_0 where we swap the weights for some pair of outcomes $y_1, y_2 \neq y$ i.e. $W_0^{(y_1)} = W^{(y_2)}, W_0^{(y_2)} = W^{(y_1)}$. It is clear that the loss stays identical, because the second term is a disjoint sum over

the outcomes. This means that the loss will identically penalize the solutions W and W_0. In collaborative filtering, this is not appropriate. Suppose that the true label $y = 5$ stars, and say that $y_1 = 1, y_2 = 4$. This means that the model is indifferent to whether our current probability estimates are concentrated around 1 star or 5 stars; both mistakes are treated equally. Intuitively, though, we expect that concentration around 1 star should be penalized much stronger than concentration around 4 stars.

Having said this, the above argument does not paint the whole picture, because it only considers the loss for a single example. For a training set $\{((i^{(t)}, j^{(t)}), y^{(t)})\}$, the loss can be written

$$\sum_{y \in \mathscr{Y}} \sum_{t: y^{(t)} = y} W^{(y)}_{i^{(t)}, j^{(t)}} - \sum_t \log \sum_{y' \in \mathscr{Y}} \exp(W^{(y')}_{i^{(t)}, j^{(t)}}).$$

As before, the second term is oblivious to the ordering of the weights for each y. That is, we can swap $W^{(y)}$ and $W^{(y')}$, and the term remains unchanged. However, doing so has an impact on the first term, and so the loss is no longer unchanged. To revisit the example above, if we swapped $W^{(1)}$ and $W^{(4)}$, this would have no effect on the loss for an example whose label is 5, but it would have an impact on all examples whose loss is 1 or 4.

In general, then, not explicitly taking into label ordinality in the objective can be mitigated if we have sufficiently many training examples. This fact should not be surprising, because in Section 3.4.2, we showed that log-likelihood optimization recovered the true class conditional probabilities. This fact is true regardless of what the outcomes represent, and so in particular holds for ordinal $\mathscr{Y}$.

The above analysis suggests that it is sensible to use the basic LFL model trained according to Equation 3.6 to collaborative filtering data. (In fact, the related restricted Boltzmann machine model (Salakhutdinov et al., 2007) similarly performs log-likelihood

optimization. We will discuss this model more in Section 4.6.) Nonetheless, in practice, modifying the model to be cogniscent of the ordinal scale may bring benefits. One reason is that in the ordinal case, we may be able to exploit the fact that we believe there to be significant shared structure amongst each of the matrices $W^{(y)}$. In particular, we may be able to reduce the number of parameters we need to estimate. Another reason is that in practice, training data may be sparse at a user- or movie-level. This introduces the risk that we do not have enough examples to sidestep the non-identifiability concern raised above. Motivated by this, we now look at ways to modify LFL for collaborative filtering problems.

4.3 Modifying LFL for collaborative filtering problems

There are at least two basic ways to exploit the special structure of collaborative filtering labels: changing the training objective, and changing the underlying model itself. We discuss both options in turn.

4.3.1 Modifying the training objective function

Suppose we wish to minimize squared error. Then, we saw above that the optimal prediction for a dyad x is $\mathbb{E}_{y|x}[y]$. This suggests that when training the model itself, we can look to minimize the squared error between this prediction and the true label:

$$\begin{aligned}\ell(y;\theta) &= (y - \mathbb{E}_{y' \sim \Pr[y|x;\theta]}[y'])^2 \\ &= \left(y - \sum_{y' \in \mathscr{Y}} y' \cdot \Pr[y'|x;\theta]\right)^2. \end{aligned} \tag{4.2}$$

Such a loss function completely avoids the issue faced by log-likelihood on a per-example basis. Recall that for log-likelihood, given any set of weights W, the solution W_0 where $W_0^{(y_1)} = W^{(y_2)}, W_0^{(y_2)} = W^{(y_1)}$ incurred equal loss on a single example. Using

squared error, however, this is no longer true. The reason is that the value of $\mathbb{E}[y]$ changes when we switch from W to W_0, and so if W_0 results in a solution where this expected rating deviates more from the true rating, it will be penalized higher.

While squared error is the most popular objective in the collaborative filtering literature, other options like absolute error are also possible. Extending the above idea to different loss functions poses challenges. In the case of absolute error, for example, we would ideally like to optimize for the median being close to the true rating:

$$\ell(y;\theta) = |y - \text{median}(\Pr[y|x;\theta])|.$$

However, an issue with this choice is that the model is no longer simple to optimize, as the median is a non-differentiable function of θ. Though sub-optimal, one option is to plug-in the expected value here also, and hope that the model is nonetheless able to find a reasonable fit:

$$\ell(y;\theta) = |y - \mathbb{E}_{y' \sim \Pr[y|x;\theta]}[y']|.$$

Such a model may be justified in cases where the median and mean of the rating distributions are close to each other.

We emphasise that one advantage of methods such as LFL is the ability to directly optimize the objective of interest. This is not true of some more sophisticated methods, for example those that attempt to integrate out some subset of parameters. We will see in experiments on benchmark datasets (Section 4.8.2) that the ability to directly optimize for the objective used for evaluation can give significant performance gains.

4.3.2 Modifying the underlying model

To see how we might change the underlying model for collaborative filtering data, recall that the basis for LFL was multinomial logistic regression (MLR). We can consider

approaches for making MLR exploit ordinal data, and see whether they may be adapted to the dyadic setting. One such approach is the ordered stereotype model (Anderson, 1984), which models

$$\Pr[y|x;\theta] \propto \exp(\phi^{(y)}(w^T x) + b^{(y)}),$$

where the ϕ's are ordered to match the order imposed on $\mathscr{Y}$, i.e. if $y_1 \prec y_2$, then $\phi^{(y_1)} \leq \phi^{(y_2)}$. The weights are then learned by maximizing the log-likelihood. The effect of this ordering constraint is that a unit increase in some feature x_d causes an change in $\Pr[y|x]$ that depends on the relative position of y in $\mathscr{Y}$ (Agresti, 2010). Observe that the model can be thought of as a constrained MLR, where it is assumed that $w^{(y)} = \phi^{(y)} \cdot w$.

We can apply the stereotype model to the LFL model, by noting that the analogue of $w^{(y)}$ above is just the factorized weight matrix $W^{(y)}$. A natural analogue of the model is to assume that there is a scalar $\phi^{(y)}$ for each $y \in \mathscr{Y}$, so that $W^{(y)} = \phi^{(y)} \cdot W$, or

$$W^{(y)} = \phi^{(y)} \cdot (U^T V).$$

In such a model, we essentially keep a single set of latent features, as done in standard collaborative filtering models, and get probability estimates by attaching a single scaling factor to the scores $U^T V$. Intuitively, this may be unable to faithfully model the underlying probabilities, as it constrains the prediction space. To enrich it, at least two options are possible. One is to think of the previous model as being

$$W^{(y)} = U^T(\phi^{(y)} I_K)V,$$

which suggests a natural extension, namely,

$$W^{(y)} = U^T \Phi^{(y)} V,$$

where $\Phi^{(y)}$ is a $K \times K$ diagonal matrix. The ordering constraint specifies that $\Phi^{(y)} - \Phi^{(y')} \succeq 0$ if $y \succ y'$. In fact, this is nothing but the alternate decomposition discussed in Section 3.5.1, except that here, we have an ordering constraint on the ϕ parameters.

Another option is to adapt a suggestion proposed in (Anderson, 1984), which is to have a multidimensional decomposition:

$$W^{(y)} = \sum_{l=1}^{L} \phi_l^{(y)} \cdot ((U^{(l)})^T V^{(l)}).$$

Here, we keep L separate user and movie weights, and for have a scalar $\phi_l^{(y)}$. If $L \ll |\mathscr{Y}|$, this model does not increase the number of parameters over the original factorization. Overall, though, this approach is less compact than the previous one, which also has a natural interpretation as a tensor factorization. For this reason, in general we prefer the previous approach to modifying the model.

4.3.3 Which approach is better?

We have seen two distinct approaches to exploit the ordinal structure of ratings. It is worth asking what potential benefits and drawbacks each approach has. Broadly, there are two competing tensions when applying the LFL model to collaborative filtering data: obtaining good predictive performance with respect to the error metric, and obtaining meaningful probability estimates of each of the ratings. In general, the two approaches can be seen as solutions that place opposite emphasis on these goals.

As we noted above, a limitation of modifying the training objective is that its sensibility is tied to using square loss. While this is the most common loss function used in the literature, one can imagine settings where this loss is not appropriate. A more fundamental issue is that this approach may not fulfil a basic motivation of applying the LFL model to collaborative filtering data, namely, obtaining a reasonable probability

distribution over the ratings. Recall that we argued in Section 4.2.3 that when using log-likelihood, we will get meaningful probabilities even in the case of ordinal labels. However, the same guarantee may not hold in Equation 4.2. The reason is the following. Consider some candidate set of weights $\{W^{(y)}\}$, with resulting dyad probability estimates $\{p^{(y)}\}$. By virtue of using squared error, we are guaranteed to match the true *expected* rating given sufficient data:

$$\sum_{y \in \mathscr{Y}} y p^{(y)} = \sum_{y \in \mathscr{Y}} y \eta(y;x),$$

where $\eta(y;x) = \Pr[y|x]$ is the true conditional label probability. The case of $p^{(y)} = \eta(y;x)$ is certainly sufficient for the above to hold, but it is not necessary. In particular, if we modify the probability estimates such that the expected rating matrix $\sum_{y \in \mathscr{Y}} y p^{(y)}$ is unchanged, we will incur the same loss, and thus the solution is unchanged. Therefore, there is in principle a risk that the probability distributions obtained by this method are not reliable, in the sense of only modelling the expectation correctly. In *practice*, such a situation may be unlikely, because the translation space is constrained by the low rank assumption on the $W^{(y)}$'s, and possible sharing of parameters between the same. Nonetheless, this is an issue to keep in mind.

A potential limitation of modifying the model is that it relies on reducing the number of parameters. This introduces the risk of underfitting, in the sense of not being able to faithfully capture the individual rating distributions. (Of course, it is worth noting that when $|\mathscr{Y}|$ is large, such as the KDDCup 2010 dataset for which $|\mathscr{Y}| = 100$, an accurate probability model may be difficult to learn.) Further, optimizing log-likelihood potentially introduces a mismatch between the training and test objective. If for example the model is to be evaluated based on squared error, it is intuitively sub-optimal to

optimize for a different objective during training[1].

We note that in principle, either of the above extensions may be extended to the case of genuinely ordinal rather than interval data, using a reduction of ordinal to binary classification (Li and Lin, 2007).

4.4 Analysis of the model

We give some alternate perspectives on our model, and comment on some of our modelling choices.

4.4.1 Matrix factorization perspective

We can contrast our loss in Equation 4.2 to the standard one used in latent feature models for collaborative filtering. Such models keep a single pair of matrices U, V and make the prediction

$$\text{Pred}(x = (i, j); \theta) = u_i^T v_j.$$

By contrast, taking the probability model of Equation 3.4, our prediction is

$$\begin{aligned}\text{Pred}(x = (i, j); \theta) &= \sum_{y \in \mathscr{Y}} y \cdot \frac{\exp(W_{ij}^{(y)})}{\sum_{y'} \exp(W_{ij}^{(y')})}. \\ &= \sum_{y \in \mathscr{Y}} y \cdot \frac{\exp((u_i^{(y)})^T v_j^{(y)})}{\sum_{y'} \exp((u_i^{(y')})^T v_j^{(y')})}.\end{aligned}$$

That is, we model each label with a linear combination of *multiple* factorized matrices. A direct comparison between the predictive power of the models is difficult: for a start, each of our constituent matrices, $P_{ij}^{(y)} = \frac{\exp(W_{ij}^{(y)})}{\sum_{y''} \exp(W_{ij}^{(y'')})}$, is not necessarily low rank, but

[1] If the model is correctly specified, and we are able to learn the true probability distributions, this is not an issue: we will predict the true expected rating at test time, and thus incur minimal squared error. However, in reality we do not expect our model to be correctly specified, and so the best solution according to log-likelihood may not be the best solution according to squared loss.

$\log P^y / P^{y_{\text{base}}}$ is. We do note that our model enforces shared structure amongst each of these terms, by virtue of a common denominator. Therefore, compared to a standard latent feature model that has a $K \cdot |\mathscr{Y}|$ parameters (so that both models have the same raw number of parameters), our model may induce a simpler search space.

Perhaps more importantly, we believe that our model is better specified than the standard latent feature model. The latter assumes that a user directly comes up with a score for a movie based on certain unobserved characteristics of both entities. Our model, on the other hand, assumes that the user actually determines how much she believes a movie deserves a particular rating – which may represent her uncertainty as to whether she likes or loves a movie, for example – and when forced to a pick a single number to quantify this distribution, chooses the mean.

4.4.2 Are rating-specific weights meaningful?

Observe that if we apply the standard factorization of the W matrix from Equation 3.3, we keep a separate set of weights for each rating y. This means that each movie, for example, has a different set of latent features for each rating. This is in contrast to standard latent feature models for collaborative filtering, which keep only a single set of weights for each user and movie. The existence of such rating-specific weights can be plausible. For example, the characteristics that make a user rate a movie 1 star may be quite different from the characteristics that make him rate it 5 stars: it is not necessary that a 1 star movie simply have negative weights for each of the 5 start latent features. It is largely an empirical question whether this is borne out in practice.

Observe that adapting the stereotype model to the dyadic setting keeps a single set of weights per user and movie, as in a standard latent feature model. However, each latent feature k is assumed to have a different influence $\Phi_{kk}^{(y)}$ depending on the rating, which has a similar flavour to the above.

4.5 Further extensions of the model

We now discuss some further extensions and variations to the LFL model, so as to potentially further improve performance in collaborative filtering tasks.

4.5.1 Anchor points for prediction

We have thus far considered the prediction to be the expected rating under our probability distribution. This in turn relies on having reasonable estimates for the individual ratings, which may not always be simple. A particular case where this is so is when $|\mathscr{Y}|$ is large. Then, reliably estimating the probability of each individual rating is intuitively difficult given limited training data. We may nonetheless be able to learn an accurate scoring function based on some subset $\mathscr{Y}' \subseteq [\mathscr{Y}_{\min}, \mathscr{Y}_{\max}]$, via the modified prediction function

$$\text{Pred}(x;\theta) = \sum_{y \in \mathscr{Y}'} y\text{Score}(y,x;\theta),$$

where

$$\text{Score}(y, x = (i,j);\theta) \propto \exp(W_{ij}^{(y)}).$$

In the above score function, the number of parameters depends on $|\mathscr{Y}'|$ rather than $|\mathscr{Y}|$; in particular, the normalizer for the score only considers the elements of $\mathscr{Y}'$. If $\mathscr{Y}' = \mathscr{Y}$, we get the standard prediction function. But when this is not the case, the predictions will of course differ. We can think of the elements of $\mathscr{Y}'$ as *anchor points* for the prediction. As a special case, suppose that $\mathscr{Y}' = \{a := \min \mathscr{Y}, b := \max \mathscr{Y}\}$. Then, fixing a base class, the prediction is just

$$\text{Pred}(x;\theta) = a + (b-a) \cdot \frac{1}{1+\exp(-u_i^T v_j)}.$$

This is exactly the heuristic proposed in (Salakhutdinov and Mnih, 2007) to constrain the predictions for a latent feature method to lie in the range of the ratings (referred to as "logistic PMF" in that paper). Thus, the LFL method for collaborative filtering can be seen as a principled generalization of this idea.

Observe that if $\mathscr{Y}' \subseteq \mathscr{Y}$, then there is still a probabilistic interpretation to the model: we are just setting most weights to be $-\infty$. When this is not the case, we are just learning a good approximation to the expected rating for each dyad.

4.5.2 Other approaches to capturing ordinal structure

We discussed the stereotype model as one possible way to extend multinomial logistic regression so to as handle ordinal data. There are several other options in the literature. A classic solution due to (McCullagh, 1980) is the *proportional odds* model. Here, instead of directly modelling $\Pr[y|x]$, we instead model $\Pr[y \leq y'|x]$ for each $y' \in \mathscr{Y}$ as a binary logistic regression, with only the bias term depending on y':

$$\Pr[y \leq y'|x;\theta] \propto \exp(w^T x + b^{(y')}).$$

To ensure coherence, the bias terms are assumed to be ordered according to the order on $\mathscr{Y}$. This model is similar to the stereotype model considered earlier, except that it assumes that the label only determines the anchor point of the probability, as that the hyperplanes for each label are identical. This undimensional assumption may be restrictive in some cases.

In the machine learning literature, one proposed solution is ordistic regression (Rennie, 2005), where we change the nature of the exponents in the probability model, making them resemble Gaussians. An interesting extension of our work would be to consider the effect of changing the underlying probability model to take ordinality into

account, and see its impact on predictive performance. However, in this work our interest is to construct a simple model that is both general (i.e. works for nominal and ordinal data) and powerful (i.e. makes accurate predictions).

4.5.3 Incorporating collaborative filtering specific extensions

A vast body of work in the field has concentrated on ways to extend basic factorization methods to improve their accuracy. This is a laudable goal that has spurred several interesting research directions. Most of these extensions may be incorporated into our model, as they involve modifications of the underlying scoring function, which in our case is $W_{ij}^{(y)}$. For example, (Salakhutdinov and Mnih, 2007) suggests representing a user's latent factor vector as

$$u_i = \tilde{u}_i + \frac{1}{|\{j : (i,j) \in \mathscr{O}\}|} \sum_{j:(i,j)\in\mathscr{O}} w_j,$$

where w_j is a vector for movie j, which is distinct from the standard movie vector v_j. This model corresponds to expressing a belief that the set of movies a user chose to rate directly affects the latent weights for that user. Similarly, as in the related (Koren and Sill, 2011) model, extensions proposed in SVD++ (Koren, 2008) may be easily incorporated. We leave exploration of these extensions for future work. While important, of more immediate interest are the goals we stated at the beginning of the chapter, namely, learning probabilities for the ratings, taking into account ordinal structure, and exploiting side-information.

An extension to the *regularizer* is also possible. In (Weimer et al., 2008), it is suggested to apply the regularization inversely proportional to the (square of the) number of observed entries for a particular dyad member. This ensures that the penalty for dyad members that have appeared only infrequently is larger than for those that have

appeared frequently, which is sensible because we expect to overfit more for members for which we have only limited data. Interestingly, the opposite conclusion was drawn in (Salakhutdinov and Srebro, 2010), which recommended regularizing users with fewer ratings more. In practice, we have found the latter regularization scheme to give a significant boost to performance on collaborative filtering datasets.

4.6 Comparison to existing models

We discuss how the basic LFL model and the modified versions proposed in Section 4.3 are related to other approaches for collaborative filtering. Recall that two motivations for using LFL, outlined in Section 4.1, were the ability to model a rating distribution, and the ability to handle cold-start scenarios. We will discuss related work in both these problems.

4.6.1 Comparison to existing models for rating distributions

The problem of explicitly modelling a rating distribution in collaborative filtering problems has not received significant attention. Nearly all matrix factorization models for collaborative filtering operate under the assumption that the input entries are real, and thus do not provide an explicit probability distribution over the ratings. Indeed, prior to the publication of our model, we are not aware of any similar approach that deviates from the interval scale assumption. Subsequent to our work, there have been methods with similar goals, and we begin by discussing them. We then discuss other related models.

Ordinal regression models

The recent OrdRec model (Koren and Sill, 2011) can be seen as an application of the idea of ordinal logistic regression models, in particular the proportional odds model, to our framework. Indeed, the authors' MultMF method is nothing but a special case of

the LFL model for general nominal labels. Compared to our model, one difference is that the authors train by log-likelihood maximization, rather than optimizing for squared loss. It is of interest whether the latter could improve predictive performance, while still producing a reasonable probability distribution. In (Paquet et al., 2012), the authors proposed a probabilistic version of an ordinal classification model. This is similar to OrdRec, but the treatment is Bayesian, and so the model is trained by Gibbs sampling and variational methods rather than SGD. We note that neither of the above models explore how to incorporate side-information.

Restricted Boltzmann Machines (RBMs)

Restricted Boltzmann Machines (RBMs) were proposed for collaborative filtering in (Salakhutdinov et al., 2007). RBMs also model the rating distribution for a dyad, but do so akin to the generic LFL model for nominal labels; that is, they do not attempt to exploit ordinal structure. Mathematically, for an input $x \in \mathscr{X}$ and output $y \in \mathscr{Y}$, an RBM assumes the existence of a hidden vector $h \in \{0,1\}^K$ such that

$$\Pr[x, y, h; \theta] \propto \exp(\Psi(x, y, h; \theta))$$

where the *energy function* Ψ takes a linear form,

$$\Psi(x, y, h; \theta) = h^T W x + h^T U y + b^T x + c^T y + d^T h.$$

Note that the joint distribution over (x, y) alone may be computed via marginalization

$$\Pr[x, y; \theta] = \sum_{h \in \{0,1\}^K} \Pr[x, y, h; \theta],$$

where in general computing the summation requires $O(2^K)$ time. Training is performed by maximizing the (joint) log-likelihood, which requires the use of contrastive divergence as the gradients are difficult to compute exactly (Salakhutdinov et al., 2007).

In the collaborative filtering setting, (Salakhutdinov et al., 2007) proposed to use for the user i the feature representation $\sum_{j \in R_i} e_j$, where R_i is the set of movies that the user rated. It turns out that for such a choice, the energy function for the user i is

$$\Psi(x, y, h; \theta) = \sum_{k=1}^{K} h_k \left(a_k + \sum_{j \in R_i} W_{jk}^{(y_j)} \right) + \sum_{j \in R_i}^{m} \mu_j^{(y_j)} - \log \sum_{y'} \exp \left(\mu_j^{(y')} + \sum_{k=1}^{K} h_k W_{jk}^{(y')} \right).$$

That is, as with a latent feature model, we keep a matrix $W^{(y)} \in \mathbb{R}^{N \times K}$ for each rating y. We additionally have a global bias a_k and $b_k^{(y)}$ for the kth latent feature, as well as a movie-label bias $\mu_j^{(y)}$. Note that the variables h_k are not directly estimated, but are instead marginalized over. While having a hidden unit (or latent feature) interpretation, the RBM is not a matrix factorization model. This explains why it has been found to be an important ingredient in constructing an ensemble for good performance.

The RBM is similar in flavour to the LFL model we have presented in this Chapter, most obviously in the use of a log-linear model for the joint distribution over the data and hidden variables. However, there are some important differences between the approaches. First, the RBM is a *generative* model, and training is performed by maximizing the joint likelihood $\Pr[x, y; \theta]$. By contrast, LFL is a discriminative model, and only specifies the distribution $\Pr[y|x; \theta]$. The RBM therefore attempts to solve more complex modelling task, and this extra difficulty may be unnecessary. Indeed, for large datasets, one can expect discriminative classifiers to have better accuracy than generative ones (Ng and Jordan, 2001). In principle, an RBM could be trained discriminatively (Larochelle and Bengio, 2008), but to our knowledge the resulting model has not been assessed in a collaborative filtering context.

Second, training in the RBM is much more involved than with the LFL model. The RBM requires the use of contrastive divergence, whereas LFL may be trained with SGD. The former is generally much more computationally expensive. If we can achieve comparable performance with both models, we would therefore favour the LFL model, especially on large datasets.

Third, the RBM as proposed in (Salakhutdinov et al., 2007) does not explicitly consider any mechanism to deal with the ordinal nature of collaborative filtering ratings. A follow up work, (Truyen et al., 2009) proposed changing the objective function to take the order of the labels into account. The underlying model here is a general Boltzmann machine, however – meaning that the hidden units are no longer disconnected – and so training and inference is even more involved than with the RBM model. By contrast, in this Chapter we have discussed simple schemes by which the LFL model may try to exploit the ordinal nature of collaborative filtering ratings.

Fourth, to the best of our knowledge, models based on the RBM have not been extended to incorporate side-information. Therefore, they are as-is not applicable to cold-start scenarios.

Probabilistic models

As noted in Section 3.6.1, LFL can be seen as a generalization of logistic PCA when applied to binary data. There has been some related work on using exponential family generalizations of PCA for collaborative filtering data, which we now discuss.

A logistic PCA model for collaborative filtering was applied in (Kozma et al., 2009). In particular, the authors convert a ratings matrix $X \in [R]^{I \times J}$ into a collection of R binary indicator matrices $\{X^{(r)}\}_{r \in [R]}$, with $X^{(r)}_{ij} = \mathbf{1}[X_{ij} = r]$. Then, logistic PCA is applied on each of the $X^{(r)}$ matrices. Compared to the LFL model, we see that this is akin to a one-versus-all approach, rather than a direct model for the ratings. The

latter can be expected to be a better model for the underlying probability in general, as the one-versus-all approach effectively ignores the normalization term when modelling multiclass data. (Indeed, the method in (Kozma et al., 2009) is reported to do slightly worse than a standard, non-logistic latent feature model.) Further, the model does not attempt to exploit any ordinality in the labels: it is trained to maximize log-likelihood, rather than square error, for example. In our experiments, we will see that while in principle exploiting ordinality is not necessary, it practice it can have a significant impact on performance.

(Delannay and Verleysen, 2008) proposed a generalized linear model framework for collaborative filtering problems. In particular, the authors explored the use of a binomial rather than Gaussian distribution to model ratings data:

$$X_{ij} \sim \text{Bin}(R, \sigma(u_i^T v_j)),$$

so that the generation of a rating for user i and movie j is seen to be the result of R "preference trials", with a fixed success probability. By itself, this does not give an explicit rating distribution for a dyad, but rather a way to model the success probability for a "preference trial". Compared to the LFL model, we see that one difference is that we allow for the success probability for each "preference trial" to be distinct. This is arguably a more realistic model of the generative process.

The model of (Lawrence and Urtasun, 2009) similarly does not provide an explicit rating distribution, but does provide a means of assessing the uncertainty in its predictions. This is a result of the model employing a type of Gaussian process, so that it has a predictive distribution for each dyad. The model is however based on the simple generative process that underlies other factorization models, except that it integrates out part of the model parameters. Thus, as argued, we believe that the LFL model is better

specified for collaborative filtering tasks. Further, there may be other uses of the rating distribution beyond just measuring uncertainty, e.g. to measure the similarity between users.

4.6.2 Comparison to existing schemes for cold-start correction

As discussed in Section 3.6.2, the use of a linear combination of latent and explicit features in LFL is identical to that employed by several statistical models for social networks. This approach has also been used independently in the collaborative filtering literature, e.g. (Yang et al., 2011). Arguably the most fruitful alternate direction for cold-start modelling has been based on placing a prior on the latent features for users and movies, which we now discuss.

Prior based methods

The earliest such work we are aware of is the RLFM model of (Agarwal and Chen, 2009), which for a user i with side-information x_i assumes a prior

$$u_i \sim \mathcal{N}(Wx_i, \sigma^2).$$

That is, in the absence of data, we find a linear mapping from the explicit to latent feature space. This was extended to an arbitrary regression prior in (Zhang et al., 2011; Khanna et al., 2012). While (Agarwal and Chen, 2009) proposed to use Bayesian inference on this model, we can contrast the equivalent MAP regularizer

$$\Omega(U;W) = \sum_{i \in I} ||u_i - Wx_i||_2^2$$

to the standard $\sum_{i \in I} ||u_i||_2^2$, which conservatively (but possibly non-optimally) pushes the latent weights to 0. The model of (Yang et al., 2011) employed a similar MAP estimate

in conjunction to a linear combination of latent and explicit features, thus attempting to get the best of both worlds.

Other choices for a probabilistic prior are possible, depending on the nature of the available side-information. For example, (Agarwal et al., 2009) imposes a Markov Random Field prior on the latent features,

$$\Pr[u] \propto \prod_{i,i'} e^{-\lambda_1 S_{ii'} ||u_i - u_{i'}||_2^2} \cdot \prod_i e^{-\lambda_2 ||u_i||_2^2},$$

where $S_{ii'}$ is the similarity between users i and i' derived from the side-information, e.g. using the cosine similarity $\frac{x_i^T x_{i'}}{||x_i||_2 ||x_{i'}||_2}$. For a cold-start user i, this means that the latent feature estimate u_i is the weighted average of the latent vectors of its nearest neighbours, as measured in explicit feature space. This is related to the model used in (Gantner et al., 2010b), which learns a mapping from side-information to latent features using, amongst other methods, k-nearest neighbour. More sophisticated priors based on Gaussian and Dirichlet processes have also been explored (Zhou et al., 2012; Porteous et al., 2010).

In general, we do not wish to claim that our approach is superior to any of these schemes. It is not clear that either scheme entirely subsumes the other: we note for example that it is not clear how to employ some of the above methods when there is only side-information for the (user, movie) pair, rather than just the user or the movie. Nonetheless, as these prior-based methods have largely been explored subsequent to our initial work on the model, and in future work it would be interesting to systematically explore integrating this idea into our framework. Indeed, our mechanism for setting the latent vector for cold-start entities in Section 4.1.2 can be seen as a simple version of this idea. On a related note, in Chapter 6, we will see how certain hierarchical information can be used to alleviate the cold-start problem, although the main goal is to improve modelling for warm-start entities that happen to have very few observations.

Kernel based methods

Another approach to exploiting side-information has been to consider a kernel function amongst users, say, and make that depend on the side-information in addition to the latent features (Lawrence and Urtasun, 2009; Abernethy et al., 2009). For example, in (Lawrence and Urtasun, 2009), the kernel function between users i and i' is

$$K(i,i') = \exp\left(-\gamma \cdot \left(||x_i - x_{i'}||_2^2 + ||u_i - u_{i'}||_2^2\right)\right).$$

This kernel is used when training and making predictions with the model, and arises as a result of marginalizing out the latent vectors V for movies. This is an elegant mechanism for exploiting side-information, but appears to have not been as popular as the prior-based methods. This is possibly because of the perceived computational cost of working with a kernel of users and movies.

4.7 Experimental design

Having discussed how the LFL model may be modified for collaborative filtering data, we now check how the resulting model performs empirically.

4.7.1 Aims of the experiments

At a high level, our experiments look to test whether the LFL method is indeed suitable for collaborative filtering problems, and to test the claimed salient features of the model for this problem domain. To this end, we present results for the following sets of experiments:

- *Does the choice of scoring and training scheme affect performance?* We give a comparison of the different variants of the LFL model, such as different scoring functions and training objectives.

- *How does it compare on benchmark datasets?* We provide results on benchmark collaborative filtering datasets, where we demonstrate that we perform comparably with state-of-the-art methods.

- *Can we handle the cold-start problem?* We demonstrate that we can exploit side-information to address the cold-start setting.

- *Does the model learn meaningful probabilities?* Here, we show the qualitative value of estimating probabilities.

4.8 Experimental results

We now present experimental results that address the aims stated in the previous section.

4.8.1 Does the choice of scoring and training scheme affect performance?

We look to see whether any of the scoring schemes for $W^{(y)}$ that we have introduced work better than others - specifically, we will look at the schemes where we have separate per-rating weights for users and movies ($(u^{(y)})^T v^{(y)}$), a per-rating weight for users only ($(u^{(y))})^T v$), a per-rating scaling factor on the latent features ($u^T \Lambda^{(y)} v$), and an ordered version of the latter. Further, we compare optimizing log-likelihood and squared error.

We compare these variants on the MovieLens 100K dataset. We used the provided 80%–20% train-test splits to evaluate performance in terms of RMSE. Table 4.1 summarizes the results. We see that there is a significant advantage in directly optimizing for squared error, regardless of the scoring scheme used. Amongst the scoring schemes, the differences are not significant, but as expected the richest one, which keeps a separate set

of weights for users and movies, does the best. We also note that enforcing an ordering constraint on the matrix $\Lambda^{(y)}$ for a scoring scheme $u^T\Lambda^{(y)}v$, as done in the stereotype model, does not appear to give significant improvements over using arbitrary weights for $\Lambda^{(y)}$.

Table 4.1. Comparison of different training schemes for LFL in terms of RMSE, Movielens 100K dataset. "LL" denotes log-likelihood, "MSE" denotes mean squared error.

	Scoring			
Training	$(u^{(y)})^T v^{(y)}$	$(u^{(y)})^T v$	$u^T\Lambda^{(y)}v$	$u^T\Lambda^{(y)}v$ ordered
LL	0.9087 ± 0.0163	0.9052 ± 0.0172	0.9010 ± 0.0161	0.9004 ± 0.0203
MSE	0.8717 ± 0.0146	0.8717 ± 0.0145	0.8685 ± 0.0145	0.8702 ± 0.0149

4.8.2 Comparison on benchmark datasets

We now present results on some benchmark collaborative filtering datasets. Before doing so, we emphasise that our goal is not to be the single best method for a collaborative filtering task. Instead, we wish to show that our model is at a minimum competitive with existing methods for collaborative filtering problems, despite being strictly more general.

Results on 1M movielens

We report results on the 1M movielens dataset[2]. While the dataset does not have a pre-defined train-test split, for comparison to previous work, we use a reasonably commonly used split due to (Marlin, 2004), where 5000 random users are chosen and a single rating is withheld for the test set[3].

We ran several variants of the LFL model on this dataset, using both log-likelihood and mean absolute error as the training objective. For comparison, we report the results of several baselines from (Srebro et al., 2004), primarily comprising clustering models

[2] Available at http://www.grouplens.org/node/73

[3] This split is available at http://people.csail.mit.edu/jrennie//matlab/marlin.mat.

such as a mixture of multinomials and extensions to the same. Briefly, these are an SVD method trained with EM (wSVD), a mixture of multinomial models (MixMulti), a model based on user clustering (Aspect) and fully generative extensions of the same (URP, Attitude).

We also report the results for maximum-margin matrix factorization (MMMF), and a Gaussian process approach to matrix factorization (GPMF) (Lawrence and Urtasun, 2009). Finally, we report results of a basic matrix factorization model, which following the literature we call SVD. We use two training objectives - squared and mean absolute error (the latter denoted by the suffix "-MAE") - and the prediction $\text{Pred}(x = (i, j); \theta) = u_i^T v_j$. For the LFL method, we tried several different scoring functions, and optimized for log-likelihood, squared error, and mean absolute error (with suffixes "-LL", "-Square", and "-MAE"). For the SVD and LFL methods, we chose $k = 8$ latent features, as based on informal tests this appeared to give the best performance.

Following most previous work, we report the *normalized* mean absolute error (NMAE) as our performance measure, which is the mean absolute error divided by the constant 1.6 (representing the error of a random guesser). We found it beneficial to round predictions of the SVD and LFL methods to the nearest integer for this metric, as both by default make a real-valued prediction. While the LFL method can in principle report the median of the learned probability distribution, we found this to work slightly worse.

Table 4.2 summarizes the results. We make the following observations. First, both the SVD and LFL methods comfortably outperform the clustering baselines. This is not surprising, given that latent feature methods have been shown to be the best single method for collaborative filtering tasks. It is interesting to note that it also outperforms the wSVD baseline, which uses EM in conjunction with SVD to impute the missing ratings. This imputation procedure is seen to work much worse than directly modelling the observed entries.

Second, both SVD and LFL (for most choices of loss and scoring function) are superior to MMMF. We note that the results for the MMMF method were using $k = 100$ latent features, but our methods need only $k = 8$ latent features. We conjecture that the reported results for MMMF may also be attained with significantly smaller k. There are two possible explanations for the improved performance: the fact that we directly optimize for the MAE, rather than a surrogate, and that we round predictions to the nearest integer.

Third, the LFL method performs comparably with SVD. This shows that the LFL method does not sacrifice accuracy in order to solve the more general problem of estimating a rating distribution. We do note that, interestingly, the LFL method performs well even when optimizing log-likelihood directly when using the scoring function $u_i^T \Lambda^{(y)} v_j$, and not exploiting any ordinal structure in the labels. This indicates that accurate modelling of the probability distribution need not be at odds with good predictive performance. Of further interest is the fact that optimizing for squared error often yields better results than optimizing for MAE directly for all of the scoring functions. A possible explanation for this is that the non-differentiability of the MAE objective, which is known to hamper theoretical convergence rates (Bertsekas, 1999). Another explanation is that our choice of prediction, $\mathbb{E}[y]$, is optimal for squared error and not for absolute error.

Fourth, we see that the results of the GPMF method are superior to both SVD and LFL, but not significantly when the latter are optimized for MAE directly. (In particular, the differences are not within standard error.) The slightly improved performance is not surprising, since in the latter, the user weights are integrated out, and so only the movie weights need to be learned. The results suggest that this quasi-Bayesian approach can help mitigate the sparsity in ratings data. The fact that the lift over SVD-MAE is not significant shows the value in directly optimizing for the objective of interest, which is

difficult in approaches such as GPMF that crucially rely on the objective function being squared error.

Finally, we note that we are able to outperform all existing methods with a linear combination of the predictions of the SVD and LFL methods, denoted SVD + LFL-MAE. The fact that this simple combination works so well illustrates the complementary nature of the predictions made by the LFL and basic latent feature models. We beleve that a more sophisticated ensemble involving bagged versions of these models, possibly with varying numbers of latent features, can further reduce the NMAE, but we did not explore this.

Results on EachMovie

Next, we report results on the EachMovie dataset[4], which comprises 2811718 ratings for 74424 users on 1648 movies. The dataset does not have a standard split, but a similar evaluation procedure to the MovieLens 1M dataset, due to (Marlin, 2004), is popular: we pick 30000 random users who have at least 20 ratings, and then withhold a single rating for each user to create a test set. This process is repeated three times. Unlike the MovieLens 1M dataset, the exact train-test splits used by prior work is not to our knowledge available, and so we perform this procedure ourselves.

We evaluated the same methods as used on the MovieLens 1M dataset. Additionally, we also report results for the nonparametric versions of SVD and probabilistic PCA from (Yu et al., 2009), denoted "NSVD" and "NPCA" respectively. Table 4.3 summarizes the results. We see that as before, both the SVD and LFL methods significantly outperform the clustering and neighbourhood baseline methods (top panel). The LFL method outperforms MMMF for most scoring functions when optimized for squared and absolute error. As before, we find that optimizing for squared error generally is no worse

[4]http://www.grouplens.org/node/76

than optimizing for absolute error, and in some cases is significantly better.

Unlike the MovieLens results, we observe that the gap between the LFL method and most of the nonparametric and/or Bayesian methods is significant. This illustrates the value of these schemes for coping with sparsity in the rating matrix. We emphasise that we do not expect the LFL method to outperform these schemes in general, and that the fairer comparison is to other parametric, non-Bayesian methods like MMMF.

Another difference to the MovieLens dataset is the surprisingly strong performance of the SVD method when optimized for MAE. This method manages to outperform the GPMF method, as well as the nonparametric SVD and PCA methods. The latter two are in principle strictly better than training with a fixed number of latent features. However, they crucially rely on the loss function being squared error, and indeed, we find that they do manage to outperform the SVD method when it is trained for squared error. Our results show that directly optimizing for absolute error gives equally significant gains over optimizing squared error with a fixed k. (Interestingly, we are not aware of the use of this method as a benchmark in prior studies.)

Finally, we see that as before, a linear combination of the SVD and LFL methods is the best performing method, outperforming the previously reported bagged version of MMMF.

Results on Netflix dataset

The Netflix prize dataset (Netflix, 2006) is easily the most popular and established benchmark collaborative filtering dataset. It comprises the ratings of 480189 users on 17770 movies in the form of 1–5 star ratings. The overall dataset is split into several components, which were progressively released as the competition progressed. The components are as follows: there is a *training* set of 99072112 ratings, a *probe* set of 1408395 ratings, a *quiz* set of 1408342 ratings, and a *test* set of 1408789 ratings.

Collectively, the quiz and test sets are referred to as the *qualifying* set; these sets were used to rank contestants during the competition. The probe set is also sometimes referred to as the *validation* set, as its intention was to allow contestants to tune hyperparameters of their learning algorithms.

We would like to assess the performance of the LFL method compared to established baselines on this dataset. It is impossible to perform an exhaustive comparison to other collaborative filtering models, given their scope and range. We instead focus on some representative latent feature models, given that they are the closest in spirit to the LFL model. First, we trained a basic matrix factorization model, SVD, as defined in the previous section. This model is identical to the probabilistic matrix factorization approach of (Salakhutdinov and Mnih, 2007), without the use of adaptive regularization strengths proposed in that paper.

Next, we report results of a restricted Boltzmann machine (RBM) as applied to collaborative filtering (Salakhutdinov et al., 2007), given its similarity to the LFL model. Due to the complexity involved with correctly implementing this model, we used a publicly available RBM implementation provided by a competitor in the Netflix prize[5]. As reference, we also report the results of Netflix's Cinematch baseline, and the winning solution of the Netflix prize. The former model turns out to be not especially difficult to beat by a considerable margin, but the latter less so, as it involves the blending of several hundreds of individual models (including RBMs and SVD models).

Finally, we report results for Bayesian probabilistic matrix factorization (BPMF) (Salakhutdinov and Mnih, 2008). The salient feature of this model is that it does not require searching for regularization strengths, as the regularization is integrated out with Bayesian inference. Of course, this comes at a computational cost, and the model is trained using Monte Carlo sampling. The results we report are from the paper (Mackey

[5] Available at https://code.google.com/p/nprizeadditions/

et al., 2010), which evaluated BPMF for a range of k values on the quiz, test and qualifying sets.

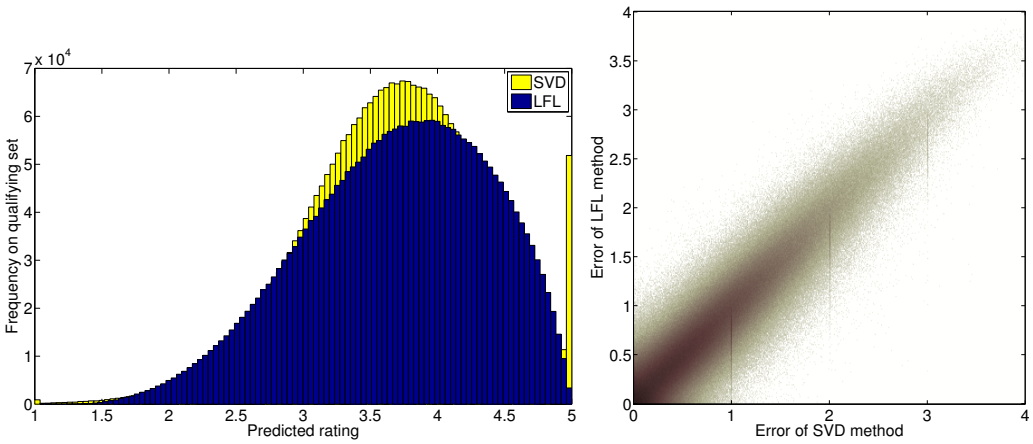
For the LFL model, we tried the various scoring functions, as well as optimizing for both log-likelihood and square loss.

For model selection, we performed 5-fold cross-validation on (a subsample of) the training set. We then evaluated the performance of these parameters on the probe set. Then, we retrained on the combination of the training and probe set, and evaluated performance on the quiz and test set both separately and in conjunction. We find the differences in performance on the three sets to not be dramatic, although of course in a contest accounting for these small differences is important. (All research published prior to the conclusion of the Netflix prize reported results on the probe and/or quiz set, as the test set was of course only released subsequently.)

The existence of an established train-test split for this dataset allows for easy comparison between models. However, some care is still needed to ensure a fair comparison. For example, the latent feature methods we study all have as a parameter the number of latent features, k. But the various models have vastly different parameterizations, and may behave differently with respect to the choice of k. Further, the raw # of parameters varies across models for a fixed k. For example, the LFL model stores a separate set of weights for each rating, and so for a fixed k would have strictly more parameters than a standard latent feature method. We attempt to ameliorate this issue by reporting results, where feasible, for a number of different instantiations of each model with different values of k, rather than picking a single one.

Our results are in Table 4.4. We note the following main findings. First, we see that the basic LFL model performs competitively with the SVD model. For lower k, the LFL model outperforms SVD, indicating that there is value in estimating the rating distribution, rather than directly modelling the ratings themselves. We further note that

the nature of the predictions made by the LFL method, and consequently the errors, are different than that of the SVD method. Figure 4.1a compares the histogram of predictions both methods make on the qualifying set. We see that the predictions of the SVD are roughly Gaussian in nature, with noticeable spikes at the boundaries: these are a result of truncation to the range $[1,5]$, and indicate that a non-neglible fraction of dyads are given raw predictions that lie outside this range. The scores of the LFL method, by contrast, are slightly skewed towards higher ratings, which matches the empirical rating histogram of the Netflix data. If we compare the errors of the two methods (Figure 4.1b), as expected from the commensurate RMSEs, they are largely similar, although consistently (if slightly) smaller for the LFL method.



(a) Predictions. **(b)** Errors.

Figure 4.1. Comparison of the predictions of the SVD and LFL methods, Netflix qualifying set.

Second, we see that the LFL model trained with log-likelihood maximization performs significantly worse than that trained to minimize squared loss. This corroborates our earlier claim that it is important for the model to exploit the ordinal structure of labels in collaborative filtering data.

Third, we observe that the RBM method tends to underperform with a small

number of hidden units. For it to achieve commensurate performance with the $k = 10$ SVD model requires between 30 to 60 hidden units. This finding is consistent with that of (Smolensky, 1986), where it was noted that an extension to the RBM for Netflix data, which exploited knowledge about which dyads were in the test set, slightly outperformed SVD, and significantly outperformed the basic RBM.

Fourth, we note that of the methods considered, the BPMF method appears to give the best performance. This is not surprising, as it can be seen as strictly generalizing the SVD method, by virtue of performing full Bayesian inference, in particular by directly modelling the predictive distribution of the hidden dyads. This good performance comes at a computational cost, however, as BPMF requires MCMC sampling to train. On this note, we wish to point out that training with the LFL model is fast, due to the use of stochastic gradient descent. For a model with $k = 10$ latent features, for example, a single epoch takes around 3 minutes. In total, we found that less than 10 epochs were needed for convergence. In (Salakhutdinov and Mnih, 2008), the authors report a time of 188 hours to achieve the reported results for $k = 60$ latent features. Similarly, the RBM model is much slower to train, taking around 6 hours to train for $k = 10$ hidden units, with a single iteration requiring roughly 12 minutes. This slower runtime is by virtue of requiring contrastive divergence to perform training.

Finally, we note that good as the performance of the individual methods we report are, they are very far from the RMSE of the winning method. It is also of interest that to break the top 1000 in the public leaderboard (on the quiz set), one needs an RMSE of 0.9172, which our higher rank methods do improve upon, but only slightly. This is not surprising, as the public nature of discussions by contestants led to latent feature methods, in particular variants of SVD and RBMs, being used either as-is or as part of an ensemble by many contestants. We do note that Netflix did not end up implementing the winning solution due to the perceived complexity for relatively little gain over simpler

individual models[6].

4.8.3 Results in the cold-start setting

As discussed earlier, side-information allows us to make predictions for dyads (i, j) where one or both of i, j did not appear in the training set, which is known as the cold-start regime. Here, we present results showing that our model is able to incorporate side-information, and use it to make useful predictions in the cold-start setting.

Results on Movielens 100K

We took the Movielens 100K dataset, and randomly discarded 50 users from the training set. We then trained a rank 5 model and made predictions on a test set consisting of the ratings of those 50 users. The baseline method we compare to is where we simply predict the mean from the learned model. That is, for the dyad (i, j) where i was not present during training, our prediction

$$F(i, j) = \frac{1}{n_j} \sum_{(i', j) \in \mathscr{T}} F(i', j),$$

where n_j is the number of users in the training set who have rated movie j. The baseline model is slightly more sophisticated than just predicting the mean from the training set, because it incorporates the learned latent weights for the movie when making a prediction.

The side-information we used was the user's age, gender and occupation, and the movie's genre. These were treated as categorical variables and encoded as multiple binary features. We also tried using interactions between the two sets of features - this is similar to applying a degree 2 polynomial kernel - but this did not dramatically improve performance.

[6] http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html

When training with side-information, we are learning weights for the latent features and the given covariates for the users and movies. This means we learn more parameters than the case with no side-information, and so the number of local optima increases. Further, these two groups of parameters reside in quite different spaces, and thus require different strengths of regularization to get a solution with good generalization. We can of course just tune the regularization parameter so as to avoid these issues, but it can still be difficult to escape the local optima. One heuristic to improve accuracy of the final model is the following idea: first train the model without any side-information, and get the appropriate weight vector. Then, use this as initialization to the model with side-information weights included. By this we mean we initialize the latent weights for the second model using the (locally) optimal values learned by the first one. We found that in the second optimization, better results were obtained when the latent weights were frozen for the second optimization i.e. we treat them as fixed, and just optimize the weights for the side-information. This is a form of block coordinate descent, where we optimize over two groups of variables by optimizing over each one in turn.

In our experiments, we considered the following scenarios:

i the standard setting, where there are no cold-start users or movies,

ii cold-start for users, but known movies,

iii a "warm-start" setting, where the test set also contains dyads where both members were observed in training, and

iv full cold-start, where both users and movies are unobserved.

For (ii), we took the test set for (i) and then discarded all movies that occur in it from the training set.

Table 4.5 summarizes our results. It is evident that learning with side-information significantly improves performance in the cold-start setting. When using mean value predictions from the standard latent feature model, we do much worse on the cold-start data than on the standard test set, where we have observed each user and movie at least once. With side-information, by contrast, we manage to do almost as well on the test set as we do in the standard setting. As expected, side-information helps in the warm-start setting also. This shows that we do not overfit with respect to the side-information: we still make good predictions for users and movies that were previously observed during training.

Comfortingly, side-information gives better MAE when tested in the standard setting i.e. no cold-start users or movies. This means that taking side-information into account can give slightly better predictions than if we just learn latent features, which is as expected.

Results on Movielens 1M

Following (Gunawardana and Meek, 2009), we simulate a cold-start setting in the Movielens 1M dataset as follows: we pick a random set of 2962 movies to be in the training set (viz. 80% of all movies), and keep the rest of the movies in the test set. We repeated this process 3 times, yielding 3 train-test sets where in each case, the test set comprises of only cold-start movies. We use the side-information provided for this dataset, which for users comprises their age, gender and occupation, and for movies their genre.

As baselines, we consider two types of methods. First, we have feature independent methods, where we randomly predict the rating ("Random"), and predict the mean training set rating for all dyads ("Global Mean"). Next, we have two feature dependent methods, where we train linear and logistic regression on the concatenation of the user

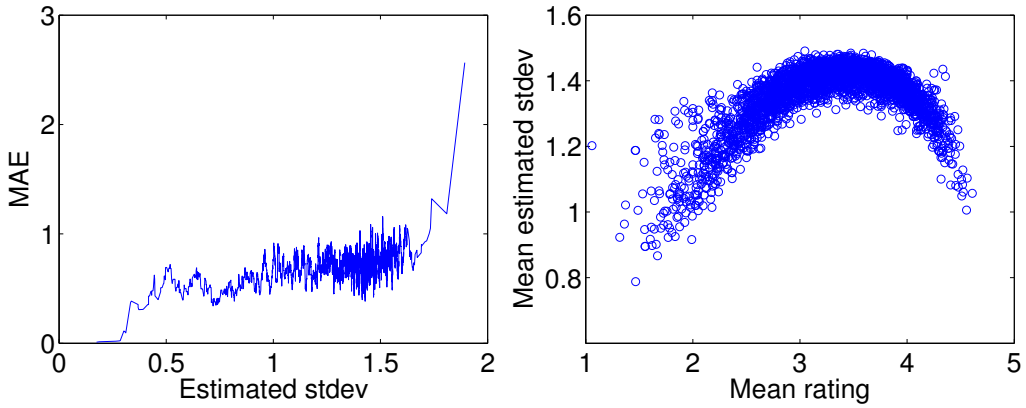
and movie feature vectors to try and predict the test set ratings. These methods are compared to LFL trained with and without explicit features.

The results are shown in Table 4.6. We make the following observations. First, the explicit features here are not especially predictive of ratings, as they give performance that is barely better that of the naïve mean prediction. Second, we see that LFL with explicit features alone manages to significantly outperform predicting the global median rating. This is due to the fact that the predictions for each user in this regime amount to the average predicted rating amongst all movies in the training set. Third, we see that there is no significant value in adding explicit features to the latent features, but, as expected, they do not hamper performance.

4.8.4 Are the learned probabilities meaningful?

We look to qualitatively analyze the rating distributions learned by the LFL model. We ran the model with $k = 10$ latent features on the Movielens 1M data. One interesting summary statistic for a rating distribution is its standard deviation. Intuitively, this reflects the model's uncertainty in its predictions. Figure 4.2a compares the standard deviation of the rating distributions to the absolute error of the model's prediction to the true rating. We see that towards the extremities, the uncertainty corresponds in the expected way to the observed error. In particular, for highly uncertain dyads, the model tends to attain a high error. In between these extremes, there is a noisy, but slight, correlation between the two, with higher uncertainty generally meaning a higher error.

There are many possible explanations for why the model is uncertain for a particular dyad. One possibility is that the movie in question is extremely polarizing, so that reliably predicting if a user will like it is a challenge. To test this, we define the uncertainty of each movie to be the average uncertainty of all dyads involving that movie. In Figure 4.2b, we compare the result with the average rating for each movie. As



(a) Error. **(b)** Number of ratings.

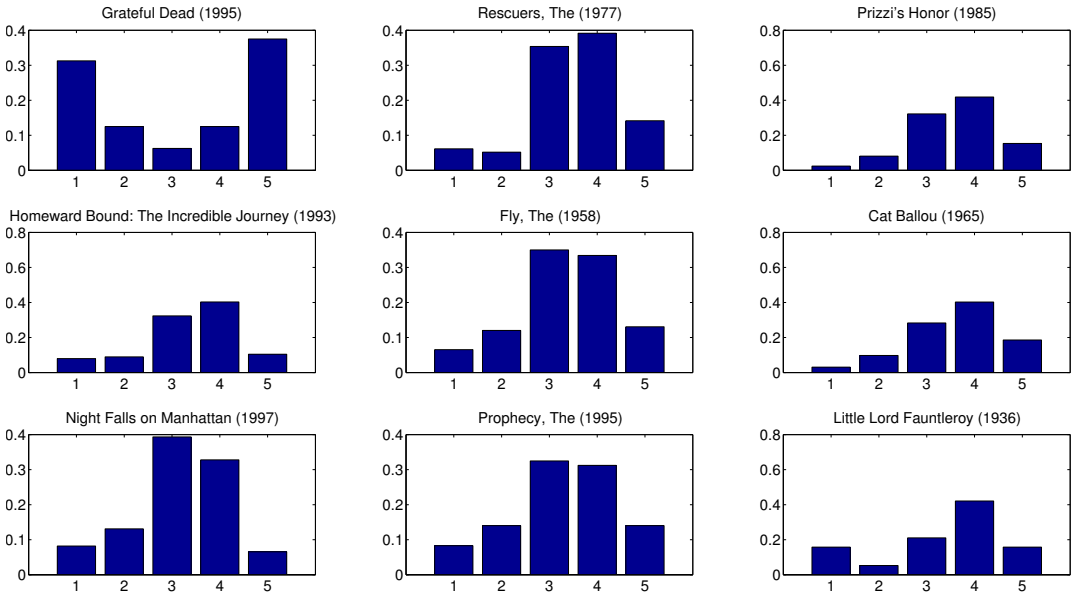
Figure 4.2. Relationship between estimated standard deviation and data characteristics, MovieLens 1M dataset.

expected, we see that movies with extremely high or low average ratings tend to have low uncertainty, as they are easy to predict. Table 4.7 lists some of these movies, most of which tend to be universally panned. The highest uncertainty is for movies with an average rating of just over 3 out of 5 stars, some of which are again listed in Table 4.7.

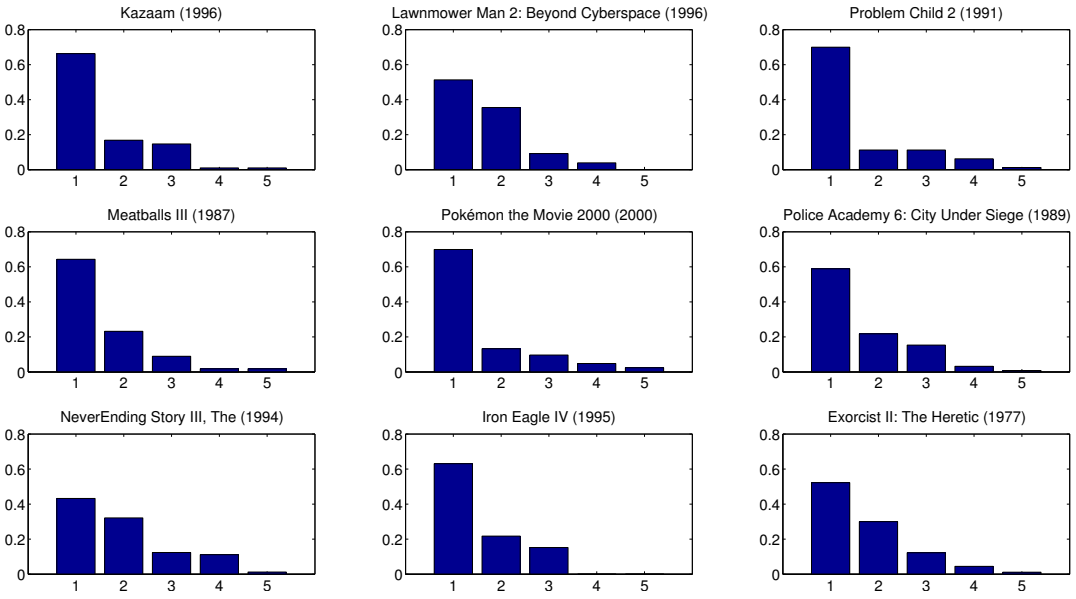
To more directly assess if polarization is correlated with the model's uncertainty, Figures 4.3a and 4.3b look at the empirical rating distributions for the high and low variance movies, computed across all users. As expected, we see that the low variance movies tend to not have many dissenting ratings from the mean, whereas the high variance ones sometimes do: in particular, note the rating distribution for the "Grateful Dead" movie, where roughly as many people give it 1 and 5 stars.

4.9 Conclusion

Collaborative filtering is a special case of dyadic prediction where the labels are measurements on an ordinal scale, representing some degree of preference a user has for an item. Matrix factorization approaches underly the most popular (and accurate) approaches to this problem. In this Chapter, we have described how the LFL model



(a) High variance.



(b) Low variance.

Figure 4.3. Sample of empirical rating distributions for movies with extreme variances, MovieLens 1M dataset.

introduced in the previous Chapter may be adapted to this task. We explained how the model may be used to assess the uncertainty in its predictions, by virtue of keeping a distribution over the ratings, and also how it can address the cold-start problem. Empirical results on benchmark collaborative filtering datasets, including the Netflix prize dataset, show the model has favourable performance compared to state of the art methods. We also demonstrated some qualitative insights that can be gained by studying the rating distribution that it learns.

4.10 Acknowledgements

Chapter 4, in part, contains material as it appears in "A Log-Linear Model with Latent Features for Dyadic Prediction", IEEE International Conference on Data Mining (ICDM), pages 364-373, 2010. Aditya Krishna Menon, Charles Elkan. The dissertation author was the primary investigator and author of this paper.

Table 4.2. Comparison of normalized MAE scores for different models, Movielens 1M dataset. Entries marked with a * indicate results reported in the published literature, as opposed to the result of code we ran ourselves.

Model	**NMAE**
wSVD $k = 10^*$	0.4886 ± 0.0065
Pearson NN *	0.4539 ± 0.0010
MixMulti*	0.4444 ± 0.0007
URP*	0.4341 ± 0.0023
Aspect*	0.4339 ± 0.0023
Attitude*	0.4320 ± 0.0055
SVD $k = 8$	0.4142 ± 0.0026
SVD-MAE $k = 8$	0.4085 ± 0.0046
MMMF $k = 100^*$	0.4156 ± 0.0037
GPMF $k = 10^*$	0.4052 ± 0.0011
GPMF RBF $k = 10^*$	0.4026 ± 0.0020
LFL-LL $(u_i^{(y)})^T v_j^{(y)}$ $k = 8$	0.4279 ± 0.0088
LFL-LL $u_i^T v_j^{(y)}$ $k = 8$	0.4110 ± 0.0061
LFL-LL $u_i^T \Lambda^{(y)} v_j$ $k = 8$	0.4129 ± 0.0023
LFL-LL $u_i^T \Lambda^{(y)} v_j$ ord $k = 8$	0.4290 ± 0.0027
LFL-Square $(u_i^{(y)})^T v_j^{(y)}$ $k = 8$	0.4074 ± 0.0029
LFL-Square $u_i^T v_j^{(y)}$ $k = 8$	0.4097 ± 0.0031
LFL-Square $u_i^T \Lambda^{(y)} v_j$ $k = 8$	0.4035 ± 0.0039
LFL-MAE $(u_i^{(y)})^T v_j^{(y)}$ $k = 8$	0.4063 ± 0.0012
LFL-MAE $u_i^T v_j^{(y)}$ $k = 8$	0.4170 ± 0.0014
LFL-MAE $u_i^T \Lambda^{(y)} v_j$ $k = 8$	0.4117 ± 0.0056
MMMF $k = 100$ + Bagging*	0.4029 ± 0.0027
SVD + LFL-MAE $k = 8$	$\mathbf{0.4004 \pm 0.0013}$

Table 4.3. Comparison of normalized MAE scores for different models, EachMovie dataset. Entries marked with a * indicate results reported in the published literature, as opposed to the result of code we ran ourselves.

Model	**NMAE**
wSVD $k = 20^*$	0.4562 ± 0.0032
Pearson NN *	0.4886 ± 0.0014
MixMulti*	0.4557 ± 0.0012
URP*	0.4442 ± 0.0013
Aspect*	0.4573 ± 0.0007
Attitude*	0.4520 ± 0.0016
SVD $k = 10$	0.4265 ± 0.0022
SVD-MAE $k = 10$	0.4154 ± 0.0018
MMMF $k = 100^*$	0.4397 ± 0.0006
GPMF $k = 20^*$ Linear	0.4209 ± 0.0017
GPMF $k = 20^*$ RBF	0.4179 ± 0.0018
NSVD*	0.4548 ± 0.0005
NPCA*	0.4257 ± 0.0005
LFL-LL $(u_i^{(y)})^T v_j^{(y)}$ $k = 10$	0.4583 ± 0.0008
LFL-LL $u_i^T v_j^{(y)}$ $k = 10$	0.4229 ± 0.0029
LFL-LL $u_i^T \Lambda^{(y)} v_j$ $k = 10$	0.4230 ± 0.0018
LFL-Square $(u_i^{(y)})^T v_j^{(y)}$ $k = 10$	0.4470 ± 0.0006
LFL-Square $u_i^T v_j^{(y)}$ $k = 10$	0.4327 ± 0.0016
LFL-Square $u_i^T \Lambda^{(y)} v_j$ $k = 10$	0.4254 ± 0.0027
LFL-MAE $(u_i^{(y)})^T v_j^{(y)}$ $k = 10$	0.4362 ± 0.0023
LFL-MAE $u_i^T v_j^{(y)}$ $k = 10$	0.4406 ± 0.0022
LFL-MAE $u_i^T \Lambda^{(y)} v_j$ $k = 10$	0.4250 ± 0.0030
MMMF $k = 100$ + Bagging*	0.4287 ± 0.0021
SVD + LFL-MAE $k = 10$	$\mathbf{0.4134 \pm 0.0018}$

Table 4.4. Comparison of RMSEs for different models, Netflix dataset. Entries marked with a * indicate results reported in the published literature, as opposed to the result of code we ran ourselves. "Unknown" means we were unable to find a published result for the exact component of the dataset.

Model	**Probe set**	**Quiz set**	**Test set**	**Qualifying set**
Cinematch*	0.9474	0.9514	0.9525	Unknown
RBM $k = 10$	0.9536	0.9524	0.9532	0.9528
RBM $k = 30$	0.9315	0.9293	0.9304	0.9299
RBM $k = 60$	0.9229	0.9198	0.9191	0.9204
RBM $k = 100$	0.9185	0.9148	0.9141	0.9154
SVD $k = 10$	0.9274	0.9232	0.9239	0.9236
SVD $k = 30$	0.9172	0.9120	0.9129	0.9124
SVD $k = 60$	0.9147	0.9097	0.9105	0.9101
SVD $k = 100$	0.9138	0.9084	0.9091	0.9088
BPMF $k = 15$*	Unknown	0.9117	0.9125	0.9121
BPMF $k = 30$*	Unknown	0.9044	0.9049	0.9047
BPMF $k = 60$*	Unknown	0.9004	0.9001	0.9002
BPMF $k = 120$*	Unknown	0.8953	0.8958	0.8956
Variational Bayes $k = 10$*	0.9242	Unknown	Unknown	Unknown
Variational Bayes $k = 30$*	0.9141	Unknown	Unknown	Unknown
NSVD*	Unknown	0.9216	Unknown	Unknown
NPCA*	Unknown	0.8926	Unknown	Unknown
LFL-LL $u_i^T \Lambda^{(y)} v_j$ $k = 10$	0.9457	0.9418	0.9430	0.9424
LFL-Square $u_i^T v_j^{(y)}$ $k = 10$	0.9268	0.9223	0.9230	0.9226
LFL-Square $u_i^T v_j^{(y)}$ $k = 30$	0.9194	0.9148	0.9152	0.9150
LFL-Square $u_i^T v_j^{(y)}$ $k = 60$	0.9174	0.9129	0.9134	0.9131
LFL-Square $u_i^T v_j^{(y)}$ $k = 100$	0.9180	0.9125	0.9129	0.9127
LFL-Square $u_i^T \Lambda^{(y)} v_j$ $k = 10$	0.9238	0.9192	0.9198	0.9195
LFL-Square $u_i^T \Lambda^{(y)} v_j$ $k = 30$	0.9209	0.9167	0.9171	0.9169
LFL-Square $u_i^T \Lambda^{(y)} v_j$ $k = 60$	0.9190	0.9142	0.9148	0.9145
LFL-Square $u_i^T \Lambda^{(y)} v_j$ $k = 100$	0.9173	0.9131	0.9134	0.9133
Winning solution*	Unknown	0.8554	0.8567	Unknown

Table 4.5. Comparison of MAE with and without side-information.

Setting	No side-info MAE	Side-info MAE
Standard, all users	0.7162 ± 0.0054	$\mathbf{0.7063 \pm 0.0000}$
Cold-start users	0.8039 ± 0.0000	$\mathbf{0.7118 \pm 0.0208}$
Warm-start users	0.7455 ± 0.0046	$\mathbf{0.7232 \pm 0.0112}$
Cold-start users + movies	0.9608 ± 0.0000	$\mathbf{0.7451 \pm 0.0196}$

Table 4.6. Comparison of performance in cold-start case, Movielens 1M dataset.

Method	Test set MAE	Test set RMSE
Random	1.5180 ± 0.0027	3.5887 ± 0.0182
Global Mean	0.8657 ± 0.0095	1.2430 ± 0.0167
Linear regression	0.8652 ± 0.0108	1.1491 ± 0.0149
Logistic regression	0.8621 ± 0.0155	1.1655 ± 0.0168
LFL Latent	0.8105 ± 0.0088	1.0843 ± 0.0068
LFL Latent + Explicit	$\mathbf{0.8091 \pm 0.0087}$	$\mathbf{1.0674 \pm 0.0085}$

Table 4.7. Movies with extreme variances, MovieLens 1M dataset.

Lowest variance	**Highest variance**
Kazaam	Grateful Dead
Lawnmower Man 2: Beyond Cyberspace	The Rescuers
Problem Child 2	Prizzi's Honor
Meatballs III	Homeward Bound: The Incredible Journey
Pokemon the Movie 2000	The Fly

Chapter 5

Application to Link Prediction

This chapter studies the effectiveness of the latent feature approach for the link prediction problem. We first make a case for matrix factorization to serve as a foundation for a general purpose link prediction model, and show how to adapt the LFL model for this purpose. We show how unlike previous approaches, the LFL model is capable of handling graphs beyond ones that are unweighted and undirected. Further, in the case of unweighted graphs, we propose a novel mechanism to overcome the class imbalance problem, based on the idea of optimizing for a *ranking loss*. Experimentally, we show that the LFL method significantly outperforms several popular link prediction methods on both weighted and unweighted graphs.

5.1 Link prediction: overview and existing models

Before proceeding, we define the link prediction problem formally, and study some existing approaches to the problem.

5.1.1 Problem definition

Link prediction is the problem of predicting the presence or absence of edges between nodes of a graph. Broadly, there are two types of link prediction that have been studied in the literature: (i) *structural*, where the input is a partially observed graph, and

we wish to predict the status of edges for unobserved pairs of nodes (Clauset et al., 2008), and (ii) *temporal*, where we have a sequence of fully observed graphs at various time steps as input, and our goal is to predict the graph state at the next time step (Liben-Nowell and Kleinberg, 2003). Both problems have important practical applications, such as predicting interactions between pairs of proteins and recommending friends in social networks respectively. This chapter will focus on the structural link prediction problem, and henceforth, we will use the term "link prediction" to refer to the structural version of the problem.

Formally, structural link prediction has as input a partially observed graph with adjacency matrix $G \in \{\mathscr{Y} \cup \text{“?”}\}^{n \times n}$, where $\mathscr{Y}$ is the set of possible edge weights (or labels), and "?" denotes a missing or unknown status link. (Note that an unknown status link is *not* the same as a link that is *known* to not exist, i.e. an edge with weight 0.) In some cases, we may have additional side-information for edges and/or individual nodes. Our goal is to exploit all these sources of information to make predictions for the missing entries.

An important special case of the above is the case of binary labels, $\mathscr{Y} = \{0, 1\}$, with 0 denoting a *known absent* link, and 1 denotes a *known present* link. A further special case is where the underlying graph is symmetric, so that $G = G^T$. This case of unweighted, undirected graphs is the most common setting studied in the literature, and we will similarly place extra attention to this regime.

5.1.2 Desiderata for a link prediction model

Having defined the general link prediction problem, we identify some important desiderata that a model should ideally address.

First, a model should ideally be able to integrate information available both in the topological structure of the graph, as well as the side-information for nodes and/or

edges. The graph topology and the side information potentially encode different types of information: the topological structure can be useful when there is a clear underlying process that describes the linking behaviour, and side-information can be useful in predicting links when a node is only sparsely connected. Thus, combining both sources of information can be expected to give the best performance.

Second, in the case of binary labels, link prediction datasets are characterized by extreme *imbalance*: the number of edges known to be present is often significantly less than the number of edges known to be absent. (For non-binary datasets, the challenge is of course exacerbated if one considers the problem as a series of one-versus-all relations.) For example, in the PowerGrid dataset (discussed in Section 5.4.2), there is 1 link for every 1850 non-links. Class imbalance potentially can potentially hamper the effectiveness of a model that would otherwise be suitable on balanced data.

Third, we noted above that the case of unweighted and undirected graphs is the one most commonly studied in the literature. However, in practice, this is by no means the most realistic setting. Certainly there are many settings where the graph of interest is directed, such as a network of trust relationships between individuals. In terms of the edge weights, for one, the labels in a network may be a set of nominal outcomes, e.g. $\{$ friend, colleague, family member $\}$. Further, many real-world networks are multirelational, so that $\mathscr{Y}$ is multidimensional. While it is of course possible to model these relations independently, ideally we would like to exploit shared structure amongst the relations. Thus, we claim that a general purpose link prediction method should be applicable beyond the case of edges that are undirected and with binary weights.

Fourth, as common with other dyadic prediction problems we have studied thus far, many real-world applications of link prediction involve large graphs both in terms of the number of nodes and non-missing edges. It is thus imperative that models be computationally efficient if they are to be usable in real-world scenarios.

We now review existing methods for link prediction, and then analyze how well they meet the above desiderata.

5.1.3 Existing link prediction methods

At a high level, existing link prediction models fall into two categories: unsupervised and supervised. Unsupervised models compute scores for pairs of nodes based on topological properties of the graph (Lu and Zhou, 2010), such as the set of neighbours of a node. Table 5.1 summarizes the scoring rules for some popular unsupervised models. We introduced these in Section 2.5.1, as a canonical example of an unsupervised approach to dyadic prediction. Recall that one conclusion we drew there is that these models use predefined scores that are invariant to the specific structure of the input graph, and thus do not involve any learning.

Table 5.1. Examples of some popular unsupervised scoring rules for link prediction. We use $\mathrm{N}(i,j)$ to denote the set of common neighbours, and $\deg(i) = |N(i)|$ to denote the degree of a node.

Method	**Scoring rule**
Common Neighbours (CN)	$\hat{G}_{ij} = \lvert N(i) \cap N(j) \rvert$
Preferential Attachment (PA)	$\hat{G}_{ij} = \deg(i) \cdot \deg(j)$
Adamic-Adar (AA)	$\hat{G}_{ij} = \sum_{k \in N(i) \cap N(j)} \frac{1}{\log \deg_k}$
Katz	$\hat{G}_{ij} = (I - \beta G)^{-1}_{ij} - I_{ij}, \beta \in \mathbb{R}_+$
Shortest Path (SHP)	$\hat{G}_{ij} =$ length of shortest path from i to j

Supervised models, on the other hand, attempt to be directly predictive of link behaviour by learning a parameter vector θ via

$$\min_{\theta} \frac{1}{|\mathscr{O}|} \sum_{(i,j) \in \mathscr{O}} \ell(G_{ij}, \hat{G}_{ij}(\theta)) + \Omega(\theta), \tag{5.1}$$

where $\hat{G}_{ij}(\theta)$ is the model's predicted score for the dyad (i,j), $\ell(\cdot,\cdot)$ is a loss function, and $\Omega(\cdot)$ is a regularization term that prevents overfitting. (We note that as in previous

chapters, the set of observed dyads is denoted by $\mathcal{O} = \{(i,j) : G_{ij} \neq \text{“?”}\}$, and we use $\mathcal{O}_i$ to mean the observed dyads involving the ith node.) The choice of these terms depends on the type of model. We list some popular approaches.

Feature-based models

Suppose each node i in the graph has an associated feature vector $x_i \in \mathbb{R}^d$. Suppose further that each edge (i,j) has a feature vector $z_{ij} \in \mathbb{R}^D$. Then, we may instantiate Equation 5.1 via

$$\hat{G}_{ij}(w,v) = L(f_D(z_{ij};w) + f_M(x_i,x_j;v)) \tag{5.2}$$

for appropriate scoring functions $f_D(\cdot), f_M(\cdot,\cdot)$ acting on dyadic and monadic features respectively, and a link function $L(\cdot)$. Both linear (Wang et al., 2007) and nonlinear (Hasan et al., 2006; Beck et al., 2000) choices of $f_D(\cdot), f_M(\cdot,\cdot)$ have been considered. In the linear case, it is standard to let $f_D(z_{ij};w) = w^T z_{ij}$ and $f_M(x_i,x_j;v) = (v^{(1)})^T x_i + (v^{(2)})^T x_j$, where $v^{(1)} = v^{(2)}$ iff the graph is undirected. The loss is typically either square- or log-loss, and the regularizer typically $\frac{\lambda_w}{2}||w||_2^2 + \frac{\lambda_v}{2}||v||_2^2$.

Note that we can compute multiple unsupervised scores between pairs of nodes (i,j), and treat these as comprising a feature vector z_{ij}. In this sense, such supervised methods can be expected to strictly dominate the unsupervised methods in terms of predictive accuracy.

Graph regularization models

In graph regularization models, we assume the existence of node features $x_i \in \mathbb{R}^d$, based on which we construct a kernel $K_{ii'jj'}$ that compares the node pairs (i,j) and (i',j'). We compute the predicted adjacency matrix $\hat{G}$ by constraining that values in this matrix should vary smoothly according to K. Thus K acts as a graph regularizer, a popular approach in semi-supervised learning (Zhu and Ghahramani, 2002). In the framework of

Equation 5.1, we have

$$\Omega(\hat{G}) = \frac{\lambda}{2} \sum_{i,i',j,j'} K_{ii'jj'}(\hat{G}_{ij} - \hat{G}_{i'j'})^2 + \frac{\mu}{2} \sum_{(i,j)\notin\mathscr{O}} \hat{G}_{ij}^2$$

The above is called *link propagation* (Kashima et al., 2009; Raymond and Kashima, 2010).

The performance of such methods depends on the choice of kernel K, which is pre-specified and not learned from the data. In this sense, regularization methods share a weakness of the unsupervised methods, in that the final scoring function is not explicitly tuned to the characteristics of the data.

Latent class models

Latent clas models assign each node of the graph to a class, and use the classes to predict the link structure. (Batagelj et al., 1999) assumes that nodes interact solely through their class memberships. It is possible to extend this to allow nodes to have membership in multiple classes (Airoldi et al., 2008). These models are largely Bayesian, and so strictly are not directly expressible in the empirical loss framework of Equation 5.1. Nonetheless, they do learn a matrix of probabilities $P \in \{0,1\}^{C\times C}$, where C is the number of classes, and this is done by placing appropriate priors on P, which may be viewed as a form of regularization.

Latent feature models

Here, we treat link prediction as a matrix completion problem, and factorize G as $L(U\Lambda U^T)$ for some $U \in \mathbb{R}^{n\times k}$, $\Lambda \in \mathbb{R}^{k\times k}$ and link function $L(\cdot)$. Each node i thus has a corresponding latent vector $u_i \in \mathbb{R}^k$, where k is the number of latent features. In the

setup of Equation 5.1, we have

$$\hat{G}_{ij}(U,\Lambda) = L(u_i^T \Lambda u_j). \tag{5.3}$$

The regularizer $\Omega(U,\Lambda) = \frac{\lambda_U}{2}||U||_F^2 + \frac{\lambda_L}{2}||\Lambda||_F^2$ usually. As discussed in Section 2.5.4, several Bayesian methods along these lines have been developed for network analysis (Hoff, 2003; Miller et al., 2009; Mørup et al., 2010), and in related problems in fields like political science (Ward et al., 2007). In the machine learning literature, computationally efficient frequentist versions of these latent feature models have been studied in (Menon and Elkan, 2010b; Yang et al., 2011).

Note that latent feature models are *not* the same as just computing the singular value decomposition (SVD) of the adjacency matrix with unknown status and known absent edges collapsed into a single class. The latter has been proposed independently as a complement to unsupervised scoring rules (Liben-Nowell and Kleinberg, 2003), but has potentially very different behaviour. Latent feature methods only attempt to model the known present and known absent edges in the graph, with regularization to prevent overfitting; no effort is spent in modelling the unknown status edges. The solutions of the two models are thus very different.

5.1.4 Do existing methods meet the desiderata?

We now study how existing methods meet the desiderata we discussed earlier. We go through each point in turn. Our analysis is summarized for some representative link prediction methods in Table 5.2.

Incorporating topological and side-information

The need for fusing topological and side-information has been an issue that has received a reasonable amount of attention in the literature. Perhaps the most popular way

to do this is by constructing explicit features for edges based on topological structure, as discussed above, thus reducing the problem to one of pure feature-based learning. It is common to use a nonlinear classifier such as a kernel SVM (Hasan et al., 2006) or decision tree (Lichtenwalter et al., 2010) to exploit nonlinear interactins between the different types of resulting features. While this is a simple approach to the problem, it is potentially limited, since the only way topological structure is exploited is through the unsupervised scores, which may not capture all relevant information about a node and/or edge.

As discussed in Section 3.6.2, latent feature methods like (Hoff, 2003) do incorporate side information. However, to our knowledge the value of using pairwise interactions in these models (Section 3.5.3) has not been explored.

Overcoming imbalance

Relatively little attention has been paid to the class imbalance problem for binary link prediction, beyond its impact on performance metrics. The *de facto* performance measure for link prediction tasks is the use of area under the ROC curve (AUC) as, unlike standard 0-1 accuracy, AUC is not influenced by the distribution of the classes. However, few models attempt to explicitly account for imbalance in their optimization or prediction. One simple solution that has been employed is undersampling the set of training dyads so that the classes are more balanced (Hasan et al., 2006; Lichtenwalter et al., 2010). This has the disadvantage of necessarily throwing away information, and thus potentially increasing the variance in the resulting model. In (Doppa et al., 2010), the imbalance problem is treated as being one of cost-sensitive classification with *unknown* costs. The costs are tuned the costs via cross-validation, and these are used to train the model. A disadvantage of this is of course that we add an extra cross-validation step to our workflow.

Non-binary labels

Most previous models and studies have focussed on the case of link prediction on undirected and unweighted graphs (Lu and Zhou, 2010). In particular, the unsupervised methods all operate on this assumption. While weighted variants of some of these scoring functions do exist for the case of numeric weights (Murata and Moriyasu, 2007; Wind and Mørup, 2012), there is not to our knowledge as clear a consensus on which of these is best way to handle this generalization.

Similarly, latent feature methods have been mostly studied for the case of binary edge weights, and undirected edges. Some work has looked at the case of multidimensional $\mathscr{Y}$ where each dimension is still binary (Miller et al., 2009). In principle this can handle nominal labels as well, and would correspond to a one-versus-rest approach.

We note that in principle, feature-based methods can operate on any set of edge weights, with appropriate modification of the underlying classification or regression model. However, this capability has not been studied in detail, to our knowledge.

Scalability

In terms of scalability, most simple methods are easy to compute on large graphs. For example, methods based on topological scores mostly require only simple, localized operations on the adjacency matrix, and so can be computed even when the graph as a whole is very large. Scores that look at long-range relationships between node pairs, such as the Katz measure, are harder to compute, but may be efficiently approximated (Dunlavy et al., 2011). The slightly more (computationally) involved feature-based methods are also scalable in general, as they rely on mature (and hence optimized) supervised learning methods for their training. Undersampling to overcome imbalance also reduces the number of training examples for such methods (Lichtenwalter et al., 2010). Unfortunately, the most powerful methods, based on latent classes and features,

do not generally scale to large graphs. This is because they rely on Bayesian inference, and in particular are most often learned through MCMC sampling, which can take a long time to converge. Thus, such models have generally been applied to small-scale graphs.

Table 5.2. A comparison of various link prediction methods. The *'s for "Scalable?" indicate methods that perform some pre-processing on the data.

Class	Method	Side-Info?	Imbalance?	Scalable?
Unsupervised	Adamic-Adar (Lu and Zhou, 2010)	No	No	Yes
	Katz (Lu and Zhou, 2010)	No	No	Yes
Feature-based	Toplogical feats (Hasan et al., 2006)	Optional	No	Yes
	CCP (Doppa et al., 2010)	Optional	Yes	Yes*
	HPLP (Lichtenwalter et al., 2010)	Optional	Yes	Yes*
Graph regularizer	LP (Kashima et al., 2009)	Required	No	No
	ELP (Raymond and Kashima, 2010)	Required	No	Yes
Latent class	MMSB (Airoldi et al., 2008)	No	No	No
	IBP (Miller et al., 2009)	Optional	No	No
	IMRM (Mørup et al., 2010)	Optional	No	No
Latent feature	Random effects (Hoff, 2003)	Optional	No	No
	Basic LFL	Optional	No	Yes
	Our model	Optional	Yes	Yes

5.2 Applying the LFL model to link prediction

We would like a model that has the same expressiveness as existing methods, while directly addressing the above challenges. Our proposal is to extend the latent feature approach to do so, and in particular to apply the LFL framework to link prediction.

5.2.1 Does LFL meet the desiderata?

By interpreting link prediction as an instance of dyadic prediction, we may apply the basic LFL model,

$$\Pr[y|x=(i,j);\theta] \propto \exp(W_{ij}^{(y)})$$

with an appropriate factorization of $W_{ij}^{(y)}$. This is of course related to the latent feature methods described in the previous section; see Section 3.6.2 for more discussion of the differences. We would like assess the efficacy of this approach to link prediction. The main reason for pursuing a latent feature method is that it is perhaps the richest of the models we have discussed. Indeed, latent feature models can be seen as a generalization of latent class models, which may be thought of as learning a binary matrix $U \in \{0,1\}^{n \times C}$, where C is the number of classes, and predicting UWU^T for a matrix W of inter-class link scores. Latent features can also be viewed as a much richer way of exploiting topological information than popular unsupervised measures described in the previous section. By construction, the latent feature approach exploits the graph topology so as to be maximally predictive of link behaviour. Thus in general, one would expect its scores to be more accurate than any single unsupervised method. A further conceptual advantage over unsupervised scores is that the learned u_i's let us make qualitative analyses of the nodes in the graph; for example, they may be used to visualize the structure of the graph.

This said, we did establish in the previous section that existing latent feature methods do not address all the desiderata introduced in Section 5.1.2. We note that the LFL method is scalable by virtue of relying on SGD for training. With regards to side-information, Sections 3.3.2 and 3.5.3 proposed a linear combination of latent and explicit features. This ability to augment latent with explicit features has a pleasant consequence, namely that we can combine latent features with the results of any other link prediction model. Suppose another model returns scores $\hat{G}_{ij}$ for the dyad (i, j). Then, we can treat this as being a dyadic feature z_{ij} in the above framework, and learn latent features that fit the residual of these scores. In general then, the latent feature approach has a natural mechanism by which any predictive signal can be incorporated, whether it is an explicit feature vector or model predictions. However, a caveat is in order: it is not necessary that combining latent features with another model will improve

performance on test data. If the latent features learn similar structure to the other model, then combining the two cannot be expected to yield better results.

The remaining questions are whether the LFL method can handle different types of graphs, and whether it can overcome the class imbalance problem. We discuss these issues in turn.

5.2.2 Handling generic graphs: undirected, directed, multirelational

In this section, we show how the LFL model is flexible enough to accomodate a range of graph types. In terms of labels, the ability to model nominal $\mathscr{Y}$ is automatic, and indeed was claimed as a salient point of the model in Section 3.4.1. However, some care is needed when exploiting the directionality of edges, and multidimensional $\mathscr{Y}$. We now discuss the modelling extensions required for these settings.

Undirected graphs

First, we consider the setting where the input graph is undirected. We allow the weights $\mathscr{Y}$ to be any discrete set, not necessarily binary. To apply the LFL model in such cases, we need to enforce symmetry in our predictions: it should be the case that $\Pr[y|x=(i,j);\theta] = \Pr[y|x=(j,i);\theta]$. Equivalently, we need to pick our factorization of W so that $W^{(y)}$ is a symmetric matrix for each $y \in \mathscr{Y}$. It is clear that none of the factorizations discussed in Chapter 3 will satisfy this in general, for the simple reason that they keep a separate set of weights u_i and v_j for the dyad members i and j. A simple modification is to keep a single set of weights for both dyad members. Then, one possible appropriate probability model is

$$\Pr[y|x=(i,j);\theta] \propto \exp(u_i^T \Lambda^{(y)} u_j),$$

where $\Lambda^{(y)}$ is a diagonal matrix. One interpretation of this model is that each node i has a latent representation $u_i \in \mathbb{R}^k$, and each possible edge label has is influenced by these latent features as encoded by $\Lambda^{(y)}$. We note that as with the general LFL model, different scoring mechanisms are possible, such as

$$\Pr[y|x=(i,j);\theta] \propto \exp((u_i^{(y)})^T \Lambda u_j^{(y)}),$$

however for clarity we focus only on the preceding decomposition.

Observe that in the case of binary labels, setting $y=0$ as the base class gives

$$\Pr[y=1|x=(i,j);\theta] = \frac{1}{1+\exp(-u_i^T \Lambda u_j)},$$

which can be seen as an application of a sigmoidal link $L(\cdot)$ and a scoring function $u_i^T \Lambda u_j$ in the general model of Equation 5.3. (Recall that we similarly studied the binary label case for a general dyadic prediction problem in Section 3.5.2.)

Directed graphs

In a directed graph, we no longer need worry about there enforcing any constraints on our predictions. In this sense, the generic LFL model using any decomposition for $W^{(y)}$ will be sensible, for example

$$\Pr[y|x;\theta] \propto \exp(u_i^T \Lambda^{(y)} v_j)$$

where again $\Lambda^{(y)}$ is diagonal. The interpretation of this is that each node has a separate set of weights for its outgoing (u_i) and incoming (v_j) edges, or equivalently for "sender" and "receiver" roles, reflecting potentially different linking behaviour in these regards.

While correct in principle, the drawback with using the generic LFL model is

that there is no sharing of information between the two sets of weights: they might as well belong to different nodes. Intuitively, we would *a priori* expect some similarity between the two sets of weights, reflecting the fact that they belong to the same node. Such prior information may be especially important on sparse real-world graphs, where there is insufficient data to accurately estimate both the sender and receiver profiles of a node. To incorporate this property, one possibility is to keep a single set of weights for each node, but weigh the features differently based on whether the node is used as a "sender" or "receiver". In particular, consider the same model as for the undirected case,

$$\Pr[y|x;\theta] \propto \exp(u_i^T \Lambda^{(y)} u_j)$$

except that $\Lambda^{(y)}$ is *not* constrained to be diagonal or symmetric. It is clear that this model can thus capture asymmetric relationships, while only estimating a single set of parameters to characterize a node.

An intermediate solution between the above and the generic LFL model is to posit that there is some underlying latent representation identity each the nodes, on top of which we allow for sender- or receiver-specific identity. That is, we enforce the prior,

$$\begin{aligned} u_i|z &\sim \mathcal{N}(z, \sigma^2 I) \\ v_j|z &\sim \mathcal{N}(z, \sigma^2 I) \\ z &\sim \mathcal{N}(0, \sigma^2 I) \end{aligned}$$

for any choice of factorization of W. We can achieve much the same effect by explicitly modelling two separate components to the probability model:

$$\Pr[y|x;\theta] \propto \exp(u_i^T \Lambda^{(y)} u_j + \alpha_i^T \Gamma^{(y)} \beta_j)$$

where both $\Lambda^{(y)}, \Gamma^{(y)}$ are symmetric, so that the first component captures the direction-independent nature of the relationship, and the second component fits the (presumably direction-specific) residue. We note that a similar factorization model was recently proposed in (Li et al., 2011) for the case of unweighted graphs, and was reported to perform better than standard factorization.

Multirelational graphs

In the case where $\mathscr{Y}$ is multidimensional, either of the appropriate LFL models above (depending on whether the graph is symmetric or not) are applicable as-is, but are potentially prone to overfitting. The reason is that we have prescribed keeping a separate set of (factorized) weights $W^{(y)}$ for each $y \in \mathscr{Y}$. In the case of multirelational data, $|\mathscr{Y}|$ is exponential in the number of relations. Thus, while the generic model is mathematically correct, it is unlikely that one sees sufficiently many examples of each tag combination to learn such weights reliably. (In multilabel learning, this is akin to keeping a separate set of weights for every possible combinations of individual tags.) To overcome this, we can consider analogues of solutions to multilabel learning problems. The first is to learn separate models for each relation. This is known as the binary relevance baseline in multilabel learning (Tsoumakas and Katakis, 2007). Suppose we have R binary relations, with outcomes $y_1, \ldots, y_R$. Then, for any $r \in \{1, \ldots, R\}$, we can model

$$\Pr[y_r = 1 | x = (i, j); \theta] = \sigma((u_i^{(r)})^T v_j^{(r)}),$$

where $\sigma(\cdot)$ denotes the sigmoid function. The number of parameters here is $O(R)$ rather than $O(2^R)$ in the naïve model.

As with other models we have now seen, the above does not induce any sharing amongst each of the parameters. This is potentially inadvisable in sparse data regimes. Therefore, we favour a different approach that shares parameters amongst the relations,

while allowing for relation-specific modelling:

$$\Pr[y_r = 1 | x = (i, j); \theta] = \sigma(u_i^T \Lambda^{(r)} v_j).$$

Here, the node-specific latent vectors attempt to capture global, relation-independent characteristics, while $\Lambda^{(r)}$ captures the relation-specific component.

The above model is very similar, but subtly distinct to the approach in the previous sections for modelling generic nominal labels. There, we defined probability models for $\Pr[y|x = (i, j); \theta]$ for each $y \in \mathscr{Y}$. Here, we separately model the probability of each binary tag, $\Pr[y_r = 1|x = (i, j); \theta]$. One could apply such a model even for generic nominal labels: we can think of the label y as a bitvector $e_y \in \{0, 1\}^{|\mathscr{Y}|}$. This would correspond to a one-versus-rest probability model, rather than a direct nominal probability model. In general, such an approach may not be able to fully capture the underlying nominal probability distribution.

Further modelling choices are also available, by virute of simple probabilistic setup of the LFL model. For example, we can look to apply a variant of the probabilistic classifier chain method for multilabel learning (Dembczynski et al., 2010), which models

$$\Pr[y_1, \ldots, y_R | x; \theta] = \prod_{r=1}^{R} \Pr[y_r | x, y_1, \ldots, y_{r-1}; \theta],$$
$$\Pr[y_r = 1 | x, y_1, \ldots, y_{r-1}; \theta] = \sigma(w^T \begin{bmatrix} x & y_1 & \ldots & y_{r-1} \end{bmatrix}).$$

That is, we use the chain rule for probabilities to decompose the original joint tag probability, and then for each tag learn a logistic regression model that depends on the preceding tags. In the dyadic setting, this translates to

$$\Pr[y_r | x = (i, j), y_1, \ldots, y_{r-1}; \theta] = \sigma(\left(u_i^T v_j + \sum_{r'=1}^{r-1} \alpha_{r'} y_{r'} \right).$$

5.3 Overcoming class imbalance for unweighted graphs

We now discuss how to adapt the LFL model to deal with class imbalance in the unweighted graph setting. First, it is worth asking: why do imbalanced classes pose a problem in the first place? There are at least two reasons for this:

- with fewer examples of one class, it is more difficult to infer reliable patterns,
- it is standard to train models to optimize for 0-1 accuracy, which can be made very high by trivially predicting everything to belong to the dominant class; hence, most models are prone to yield biased results.

We note that the LFL model is not immune to these problems when trained by maximizing the log-likelihood, or equivalently using the log-loss, which may be seen as a convex approximation to the 0-1 loss. Therefore, we must consider how to modify the model to account for imbalance.

A standard strategy to overcome imbalance in supervised learning is undersampling (Chawla et al., 2004), where we randomly throw out examples from the dominant class till the classes are more reasonably balanced. A disadvantage of undersampling is that it necessarily throws out information in the training set. Further, it is not clear to what ratio we can undersample without compromising the variance of our learned model.

An alternative is based on the following observation: in imbalanced classification problems, we often measure performance using the area under the ROC curve (AUC), which is agnostic to the distribution of classes. Intuitively, to get good test set AUC, it makes sense to *directly optimize* for AUC on the training set. This implicitly overcomes the imbalance problem, while simultaneously attempting to get the best AUC score on test data. This idea appears under-explored in the supervised learning literature, possibly

due to the perceived complexity of optimizing for AUC. However, it is possible to optimize for AUC even on very large datasets (Sculley, 2009). To do this, we begin with the pairwise SVMRank framework (Joachims, 2002). Consider a binary classification scenario with training set $\{(x_i, y_i)\}$, and let $P = \{i : y_i = 1\}$ and $N = \{i : y_i = 0\}$. The empirical AUC of a linear classifier with weight w is

$$\hat{\mathscr{A}} = \frac{1}{|N||P|} \sum_{i \in P} \sum_{j \in N} \mathbf{1}[w^T x_i > w^T x_j].$$

The problem of maximizing AUC can be cast as

$$\min_w \frac{1}{|N||P|} \sum_{i \in P} \sum_{j \in N} \mathbf{1}[w^T (x_i - x_j) < 0].$$

The above is a binary classification task on $\{(x_i - x_j, 1)\}_{(i,j) \in \mathscr{Q}}$, where $\mathscr{Q} = \{(i, j) : y_i = 1, y_j = 0\}$. Thus, to maximize the AUC, we can replace the indicator function above with a regularized loss function:

$$\min_w \frac{1}{|N||P|} \sum_{(i,j) \in \mathscr{Q}} \ell(w^T (x_i - x_j), 1) + \Omega(w).$$

The above can be optimized efficiently using stochastic gradient descent, where at each iteration we randomly pick a *pair* of examples and compute the gradient with respect to them (Sculley, 2009). We can directly translate this idea to matrix factorization for link prediction. However, there is a subtlety in how we decide what pairs of examples to consider. Suppose $\mathscr{O}^+, \mathscr{O}^-$ are the sets of known present and known absent dyads respectively. Then, there are two ways in which we can construct pairs of nodes.

Per-node pairs. Here, we consider (known present, known absent) pairs that share one node in common. This can be thought as applying the AUC loss for every row

of the G matrix. The optimization is

$$\min_{U,\Lambda} \frac{1}{|\mathscr{D}|} \sum_{i=1}^{n} \frac{1}{|\mathscr{O}_i^+||\mathscr{O}_i^-|} \sum_{j \in \mathscr{O}_i^+, k \in \mathscr{O}_i^-} \ell(u_i^T \Lambda (u_j - u_k), 1) + \Omega(U),$$

where $\mathscr{D}$ is set of all (i, j, k) triplets where $j \in \mathscr{O}_i^+, k \in \mathscr{O}_j^-$. In the case of logistic loss, the above model is similar to BPR (Rendle et al., 2009), although the motivations of the two models are different: BPR was proposed to deal with implicit feedback (or positive only) collaborative filtering datasets. In the above, we are treating the known present links as the "positive feedback", and only links in $\mathscr{O}_i^-$ as being "unspecified feedback". Further, we combine the learned latent features with side information via bilinear regression.

All pairs. Here, we consider (known present, known absent) pairs that *need not* share a node in common. This can be thought as applying the AUC loss globally on every dyad in G. The optimization is

$$\min_{U,\Lambda} \frac{1}{|\mathscr{O}^+||\mathscr{O}^-|} \sum_{(i,j) \in \mathscr{O}^+, (i',j') \in \mathscr{O}^-} \ell(u_i^T \Lambda u_j - u_{i'}^T \Lambda u_{j'}, 1) + \Omega(U).$$

The choice between the two schemes depends on whether we ultimately evaluate AUC on a per-node or global basis. This choice in turn is largely problem specific; for example, in a social network we would like each individual user to have a good ranking, whereas in a protein-protein interaction network, our interest is in which unobserved dyads are worth performing further analysis on. Regardless of the choice of scheme, we are faced with the problem of having to learn from a large number of examples, one that is superlinear in the number of observed dyads. However, in practice, only a fraction of a single epoch of stochastic gradient descent is needed to achieve good results; we will discuss this more in our experiments.

In summary, our final model optimizes for AUC directly, with regularization to prevent overfitting. We linearly combine side information via a bilinear regression model. Assuming we follow the per-node ranking scheme, and assuming per-node and per-dyad side information x_i and z_{ij} respectively, the objective is:

$$\min_{U,\Lambda,V,w,b} \frac{1}{|\mathscr{O}|} \sum_{i=1}^{n} \sum_{j \in \mathscr{O}_i^+, k \in \mathscr{O}_i^-} \ell(L(u_i^T \Lambda (u_j - u_k) + b_i + b_j + x_i^T V x_j + w^T z_{ij}), 1) + \frac{\lambda_U}{2} ||U||_F^2 + \frac{\lambda_\Lambda}{2} ||\Lambda||_F^2 + \frac{\lambda_V}{2} ||V||_F^2 + \frac{\lambda_w}{2} ||w||_2^2, \tag{5.4}$$

where the link function $L(\cdot)$ and loss $\ell(\cdot,\cdot)$ are user-specified, and $\Lambda = I$ if the graph is symmetric. Further, if required, we factorize the bilinear weights $V = D + A^T B$ for diagonal D and arbitrary A, B, and learn A, B, D.

5.4 Experimental design

We present an experimental comparison of our model to other link prediction methods in two scenarios: binary edges, and nominal edges.

5.4.1 Aims of the experiments

The primary questions we wish to address with the experiments are as follows.

- *Does LFL improve on existing methods?* Clearly, a minimum requirement of a new link prediction model is that it performs favourably compared to existing approaches to the problem. We attempt to answer this question with experiments on a range of datasets.

- *Does the ranking loss overcome class imbalance?* For binary labels, we proposed the use of a ranking loss as our training objective to overcome the class imbalance

problem. While theoretically sensible, it remains to be verified that in practice, this approach improves performance compared to standard models.

- *Can we successfully handle different graph types?* We claimed that an advantage of LFL is its ability to handle graphs beyond ones that are unweighted and undirected. We test the performance of LFL on a couple of graphs with nominal labels, and contrast its performance to some state-of-the-art methods for these datasets.

We now describe the datasets used to evaluate the various methods.

5.4.2 Description of datasets

As mentioned earlier, the case of binary labels or unweighted graphs has been the major focus of the link prediction literature. We ran experiments on a range of datasets from this domain. We separately ran experiments on data with nominal edges, although this type of data is relatively less popular. We describe both classes of dataset in turn.

Datasets with binary labels

We used datasets from a range of applications of link prediction:

- **Computational biology**.
  - Protein-protein interaction data from (Tsuda and Noble, 2004), denoted "Prot-Prot". Each protein has a 76 dimensional feature vector.
  - Metabolic pathway interaction data for *S. cerevisiae* provided in the KEGG PATHWAY database (Yamanishi et al., 2005), denoted "Metabolic". Each node has three sets of features: a 157 dimensional vector of phylogenetic information, a 145 dimensional vector of gene expression information, and a 23 dimensional vector of gene location information.

- **Bibliographic networks**.
  - The co-author network of authors at the NIPS conference (Roweis, 2002), denoted "NIPS". Each node has a 14035 dimensional bag-of-words feature vector, being the words used by the author in her publications. We performed latent semantic indexing (LSI) to reduce the number of features to 100.
  - The co-author network of condensed-matter physicists (Lichtenwalter et al., 2010), denoted "Condmat". Following (Lichtenwalter et al., 2010), we consider node-pairs that are 2 hops away in the first five years of the data. There is no side information in this problem.
- **Other networks**.
  - A network of military disputes between countries (Ghosn et al., 2004), denoted "Conflict". Following (Ward et al., 2007), we considered all disputes in the period 1990–2000. The graph is directed, with an edge originating from the conflict initiator. To allow for comparison with baseline methods, we only report results on the symmetrized version of the data, whether two countries have a link if either initiated conflict with the other. Each node has 3 features, comprising the country's population, GDP and polity, and additionally each dyad has 6 features, including e.g. the countries' geographic distance.
  - The US electric powergrid network (Watts and Strogatz, 1998), denoted "PowerGrid". There is no side information in this problem. This dataset is challenging because of the extreme imbalance (1 link for every 1850 non-links), and the nature of its linking behaviour: we expect two nodes to be linked if they are nearby geographically, which is a latent feature that may be difficult to infer.

Table 5.3. Properties of datasets used in experimental comparison.

Dataset	Nodes	$\lvert\mathscr{O}^+\rvert$	$\lvert\mathscr{O}^-\rvert$	+ve:−ve ratio	Average degree
Prot-Prot	2617	23710	6,824,979	1 : 300	9.1
Metabolic	668	5564	440,660	1 : 80	8.3
NIPS	2865	9466	8,198,759	1 : 866	3.3
Condmat	14230	2392	429,232	1 : 179	0.17
Conflict	130	320	16580	1 : 52	2.5
PowerGrid	4941	13188	24,400,293	1 : 1850	2.7

Table 5.3 summarizes the dataset sizes in terms of number of nodes and known present/known absent dyads. Condmat is the only dataset where some edges have genuinely missing status even at test time (corresponding to node pairs more than 2 hops away). Note that we do not undersample any of the datasets. We see that PowerGrid is the most imbalanced dataset, while also having a small average degree.

Datasets with nominal labels

For nominal data, we report results on the Alyawarra dataset (Denham, 1971), which contains kinship relations between people of the alyawarra tribe in Australia. The network comprises 104 nodes, with each edge between a node pair having one of 26 possible labels, denoting different kinship relations. We can view the task as multi-relational, so that each possible relation defines a separate matrix, or as a multi-category task where the number of outcomes is the number of relations. We chose the latter because there is only one kinship relation observed between two people (the outcomes are mutually exclusive).

5.4.3 Evaluation methodology

To evaluate the models on datasets with binary edges, we kept aside a fixed fraction of the observed dyads $\mathscr{O}$ for training various models, and evaluated AUC on a test set comprising the remaining dyads. This process was repeated 10 times, and we

report the average test set AUC. We used two training split ratios: for the datasets with side information (Prot-Prot, Metabolic, NIPS and Conflict), we used 10% of dyads for training, and for the others (Condmat, PowerGrid), we used 90% of dyads for training. On the latter two datasets a 10% split would cause most nodes to have no known present links in the training set, making it difficult to make predictions based solely on the topological structure. To tune the model, we swept over a grid of regularization parameters and learning rates for stochastic gradient descent, and evaluated the average AUC across the 10 splits. We report results for the parameters selected by the grid search. The number of latent features, k, was informally picked to be 30 for all datasets except Conflict, where only $k = 5$ were needed to achieve good performance.

On the nominal datasets, essentially the same procedure was performed, except that we report the average AUC for each of the possible edge labels. In other words, for evaluation purposes we treat the data as a series of one-verus-rest relations, and average the AUC across each of these relations. The precise evaluation procedure for the two datasets is described in the respective sections.

5.5 Experimental results

We now present experimental results for datasets with binary and nominal edges.

5.5.1 Results for binary edges

We divide the experimental results into three parts, each considering a different type of method. Methods with scores in **bold** have the highest score amongst methods being compared in that table. Methods that additionally have a **star** * are the best performing across all tables. In both cases, we only consider differences in AUC that are greater than the standard deviation across the splits.

Do latent features improve on unsupervised scores?

We first report results on the following methods, which only exploit topological information:

- *Unsupervised scores*. We used Adamic-Adar (AA), Preferential Attachment (PA), Shortest-Path (SHP) and Katz, which are popular scores that perform well on a range of graphs (Lu and Zhou, 2010). We also ran linear regression on all unsupervised scores (Sup-Top), which attempts to find a weighted combination of the scores with better performance.

- *Raw SVD*. We computed the raw SVD on the adjacency matrix, treating known absent and unknown status edges as being one and the same.

- *Factorization*. We used the factorization model of Equation 5.3 using square-loss and identity link (Fact-Sq), and log-loss with sigmoid link (Fact-Log).

- *Unsupervised scores as input to factorization*. Here, we used all the unsupervised scores as features for each dyad, and fed them into a factorization model trained with square-loss (Fact+Scores).

Our results are given in Table 5.4. (The table is split into two halves for visual clarity.) We make the following observations:

- Individual unsupervised scores are always outperformed by factorization methods by around 5%–25%. In most cases, factorization also outperforms a supervised combination of such scores. This suggests that in general, latent features better exploit topological information by virtue of directly optimizing to be predictive of link behaviour.

Table 5.4. Test AUC scores for methods based on topological features alone.

Dataset	AA	PA	SHP	Katz	Sup-Top
Prot-Prot	0.564 ± 0.005	0.750 ± 0.003	0.726 ± 0.005	0.727 ± 0.005	0.754 ± 0.003
Metabolic	0.524 ± 0.005	0.524 ± 0.005	0.626 ± 0.004	0.608 ± 0.007	0.628 ± 0.001
NIPS	0.512 ± 0.002	0.543 ± 0.005	0.517 ± 0.003	0.517 ± 0.003	0.542 ± 0.007
Condmat	0.567 ± 0.014	0.716 ± 0.026	0.673 ± 0.018	0.673 ± 0.017	0.720 ± 0.020
Conflict	0.507 ± 0.008	0.546 ± 0.024	0.512 ± 0.014	0.512 ± 0.014	$\mathbf{0.695 \pm 0.076}$
PowerGrid	0.589 ± 0.003	0.442 ± 0.010	0.659 ± 0.015	0.655 ± 0.016	$\mathbf{0.708 \pm 0.062}^*$

Dataset	SVD	Fact-Sq	Fact-Log	Fact+Scores
Prot-Prot	0.635 ± 0.003	$\mathbf{0.795 \pm 0.005}$	$\mathbf{0.793 \pm 0.002}$	$\mathbf{0.793 \pm 0.005}$
Metabolic	0.538 ± 0.017	$\mathbf{0.696 \pm 0.001}$	$\mathbf{0.695 \pm 0.001}$	$\mathbf{0.696 \pm 0.002}$
NIPS	0.512 ± 0.031	$\mathbf{0.612 \pm 0.007}$	$\mathbf{0.610 \pm 0.008}$	$\mathbf{0.613 \pm 0.019}$
Condmat	0.629 ± 0.051	$\mathbf{0.810 \pm 0.020}^*$	$\mathbf{0.822 \pm 0.025}^*$	$\mathbf{0.812 \pm 0.020}^*$
Conflict	0.541 ± 0.094	$\mathbf{0.692 \pm 0.040}$	$\mathbf{0.692 \pm 0.039}$	$\mathbf{0.689 \pm 0.042}$
PowerGrid	0.691 ± 0.026	0.637 ± 0.012	0.675 ± 0.017	$\mathbf{0.751 \pm 0.020}^*$

- In some cases, combining multiple topological features worsens test set AUC. The reason is likely twofold: first, a linear combination of weights may be too simplistic to leverage their combined power. Second, linear regression may underperform due to the imbalance of the data, as noted in Section 5.3.

- In most cases, combining latent features and unsupervised scores does not improve performance. This indicates that the unsupervised scores do not capture sufficiently complementary information to the latent features.

- As conjectured in Section 5.2.1, raw SVD performs much worse than the factorization methods on the datasets using 10% of dyads for training, as it treats all missing edges as being known absent.

- Amongst the two factorization approaches, the choice of loss function generally does not influence results significantly. For both losses, we required no more than 10 epochs to converge on any dataset.

To further see the advantage of the latent feature approach, we consider the effect of changing the number of observed edges in the training set. In principle, even with a few

number of edges, the latent feature approach should be able to find some latent structure in the linking pattern between nodes, and use this to make reasonable predictions. On the other hand, unsupervised methods potentially suffer, as the neighbourhood for an individual node tends to get very sparse, and so raw statistics computed on it tend to be noisy. We confirmed this by picking the Conflict dataset, and varying the amount of observed edges in the training set from 10% to 90%. We then compared the performance of the LFL and Katz methods. Figure 5.1 shows the test set AUCs for both methods. We see that as expected, with fewer edges in the training set, the performance of both methods degrades. However, the LFL method is still able to perform reasonably even in the 10% setting, achieving an AUC of 0.692. By contrast, the performance of the Katz model degrades more sharply, and in the 10% setting is only barely above random guessing, with an AUC of 0.512. This confirms that unsupervised methods, by virtue of being based on a fixed rule, are more sensitive to the nature of the data distribution.

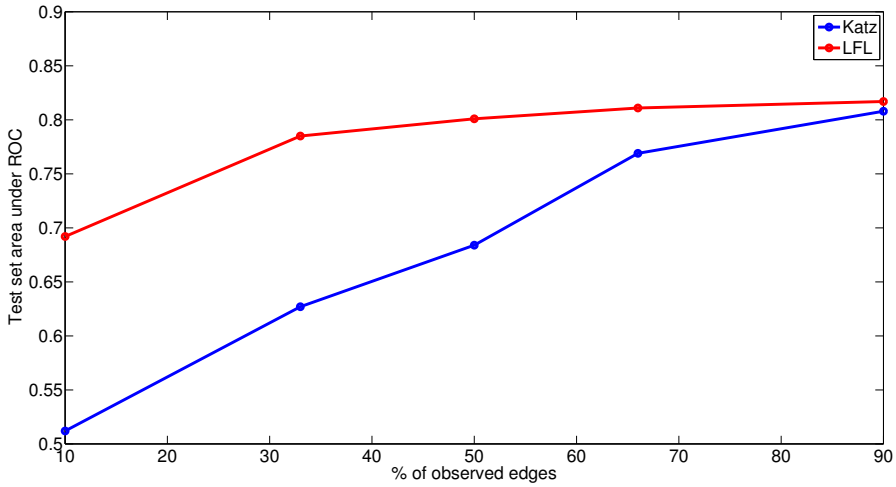


Figure 5.1. Comparison of the LFL and Katz methods as the fraction of observed edges in training changes.

How predictive is side information?

Next, we tried several methods that use side information $x_i \in \mathbb{R}^d$ for each node i:

Table 5.5. Test AUC scores for methods based on explicit features. Condmat and PowerGrid are not included because they do not have side information.

Dataset	Similarity	LP	ELP	ULR	BLR
Prot-Prot	0.680 ± 0.002	0.771 ± 0.002	0.740 ± 0.003	0.670 ± 0.002	0.776 ± 0.006
Metabolic	0.605 ± 0.002	0.719 ± 0.001	0.659 ± 0.010	0.694 ± 0.007	0.725 ± 0.012
NIPS	$\mathbf{0.953 \pm 0.000}$	0.767 ± 0.004	0.929 ± 0.010	0.611 ± 0.007	$\mathbf{0.951 \pm 0.002}$
Conflict	0.577 ± 0.008	0.614 ± 0.016	0.648 ± 0.029	$\mathbf{0.869 \pm 0.029}^*$	$\mathbf{0.891 \pm 0.017}^*$

Dataset	Fact+LP	Fact+BLR
Prot-Prot	0.789 ± 0.003	$\mathbf{0.813 \pm 0.002}^*$
Metabolic	0.701 ± 0.002	$\mathbf{0.763 \pm 0.006}^*$
NIPS	0.885 ± 0.032	0.945 ± 0.003
Conflict	0.693 ± 0.046	$\mathbf{0.890 \pm 0.017}^*$

- *Raw similarity*. As a baseline, we use the cosine similarity $\frac{x_i^T x_j}{||x_i|| ||x_j||}$ as the predicted score for the node-pair (i, j).

- *Link propagation*. We use Link Propagation (LP) with the "sum kernel" as defined in (Kashima et al., 2009), using cosine similarity as our base measure. We specifically use a special case of LP described in (Kashima et al., 2009) that can be efficiently implemented in MATLAB using Lyapunov functions. We also tried an approximation to this method, Exact Link Propagation (ELP) (Raymond and Kashima, 2010).

- *Regression*. We apply unilinear (ULR) and bilinear (BLR) regression on the feature vectors. Both methods did *not* use unsupervised scores as input, and were trained with square-loss.

- *Combinations*. We combined the factorization model with Link Propagation (Fact+LP) and bilinear regression (Fact+BLR).

We present the results in Table 5.5. (Condmat and PowerGrid are not included because they do not possess side information.) We observe the following:

Table 5.6. Test AUC scores for methods optimized with ranking loss.

Dataset	Fact-Rank	Fact-Rank-Global	BLR-Rank	Fact+BLR-Rank
Prot-Prot	0.798 ± 0.001	0.794 ± 0.001	0.785 ± 0.003	$\mathbf{0.806 \pm 0.003}$
Metabolic	0.705 ± 0.007	0.706 ± 0.006	$\mathbf{0.764 \pm 0.007}^*$	$\mathbf{0.765 \pm 0.007}^*$
NIPS	0.609 ± 0.008	0.605 ± 0.007	0.949 ± 0.002	$\mathbf{0.956 \pm 0.002}^*$
Condmat	$\mathbf{0.814 \pm 0.019}^*$	$\mathbf{0.826 \pm 0.019}^*$	N/A	N/A
Conflict	0.690 ± 0.042	0.686 ± 0.042	$\mathbf{0.885 \pm 0.018}^*$	$\mathbf{0.886 \pm 0.021}^*$
PowerGrid	0.723 ± 0.015	$\mathbf{0.754 \pm 0.014}^*$	N/A	N/A

- Bilinear regression is better than plain factorization on all datasets but Prot-Prot, which indicates that it is difficult to infer latent structure from the observed data that is more predictive than the given side information. This is not surprising given the sparsity of known present edges in the datasets.

- On Prot-Prot and Metabolic, jointly learning latent features and a bilinear regression model gives better performance than doing either individually. This suggests that despite the general superiority of explicit over latent features, the two can have complementary characteristics.

- The factorization model does not benefit from incorporating the output of LP. In fact, we find the test set AUC decreases on the NIPS dataset. On most datasets, LP had training AUC close to 1, suggesting that it is difficult to learn latent features on top of these scores without overfitting.

- Unilinear regression is always outperformed by bilinear regression, usually by a significant margin. This shows that the propensity problem with unilinear regression has important practical implications.

- Bilinear regression always outperforms both variants of Link Propagation.

Does the ranking loss overcome imbalance?

Finally, we check whether directly optimizing for AUC helps overcome imbalance. We apply the model of Equation 5.4 using square-loss (Fact-Rank), and consider an alternative where the ranking is defined over all dyads (Fact-Rank-Global) (see Section 5.3 regarding the per-node versus global ranking). We also optimize the BLR and Fact+BLR models of the previous section with the per-node form of ranking loss (BLR-Rank and Fact+BLR-Rank). On datasets with many dyads, the ranking losses only required a small fraction of a single epoch to converge (e.g. on PowerGrid, 1%–5% of an epoch for per-node ranking, and 0.01%–0.05% for global ranking). From the results in Table 5.6, we note that:

- For factorization methods, the ranking loss is dramatically superior to the regression losses on the PowerGrid dataset, which is the most imbalanced and has a small average degree. On other datasets, the differences are more modest, but the ranking loss is always competitive with the regression losses, while requiring significantly fewer dyads for convergence.

- For bilinear regression, optimizing with a ranking loss gives better performance than square loss on Prot-Prot and Metabolic.

- Jointly learning latent features and a bilinear regression model with a ranking loss performs at least as well as optimizing with square loss.

- Overall, we do see that the gains from using a ranking loss are not as significant as one might expect, especially given the high degree of imbalance on some of the datasets. In fact, recent theoretical work has shown that *proper* losses, such as log-loss, approximately maximize the AUC (Kotlowski et al., 2011; Agarwal, 2012). This would explain the largely equitable performance of the basic LFL

method. Nonetheless, we do note that the convergence of the optimization with ranking loss tends to be faster than that of the basic LFL method.

5.5.2 Results for nominal edges

We now present results for datasets where the graph edges are nominal.

Predicting kinship between individuals

In the Alyawarra dataset, each dyad (i, j) comprises one (and only one) of 26 possible labels, denoting the kinship relation between the individuals i and j. This can be thought of as a multiclass (or nominal) link prediction prediction. There are two ways to model such data. One is to share weights across all relations (a "global" model). Another is to learn separate weights for each relation (a "single" model), analogous to training a one-versus-all for multiclass classification.

We compare the predictive performance our method to the nonparametric binary latent feature model of (Miller et al., 2009). The paper shows that its Bayesian model outperforms two clustering- or block-models: the infinite relation model (IRM) (Kemp et al., 2006), and its mixed membership extension, mixed membership stochastic block-model (MMSB) (Airoldi et al., 2008). We follow the error metric used in (Miller et al., 2009), namely, the average of the AUCs of each class (or relation). The paper (Miller et al., 2009) reports results for a 80%–20% train-test split of the data. We do not have access to the exact split, but we generated 25 such random splits, and report the average AUC score over all the splits. (We comment that the error metric is potentially unstable due to the extreme rarity of some of the label classes, some of which appear less than 6 times in the 104×104 matrix. Therefore, we used a stratified split that ensured it was possible to train and evaluate the model with respect to every label class.)

The results in Table 5.7 show that our LFL method outperforms the latent class

models of MMSB and IRM[1], and slightly outperforms the nonparametric latent feature method in (Miller et al., 2009). The difference in AUC values is not significant, but the only point we wish to make is that our simple method is competitive with a state-of-the-art method that is more involved to optimize. (Indeed, even with LBFGS optimization, training takes around 10 seconds total for the "single" relation case, where we need to learn a model for 26 separate instantiations of a 104×104 matrix.) We in fact expect that the results for the nonparametric latent feature method and our method to be similar, since the essential differences are that the former constraints the latent features to be binary, and uses Bayesian inference of its parameters. Our results show that for some problems, a simple MAP estimate of parameters may be sufficient to get equivalent predictive performance, while also significantly helping scalability.

Table 5.7. AUC of various methods on alyawarra dataset.

Method	**Test set AUC**
MMSB global	0.9143 ± 0.0097
IRM global	0.8943 ± 0.0300
IBP global	0.9183 ± 0.0108
MMSB single	0.9005 ± 0.0022
IRM single	0.9310 ± 0.0023
IBP single	0.9443 ± 0.0018
LFL single	$\mathbf{0.9459 \pm 0.0119}$

5.6 Conclusion

In the chapter, we proposed to solve structural link prediction problems in multi-graphs using the LFL model. In the case of binary labels, we showed how to train the model with a *ranking loss* to overcome the imbalance problem that is common in link

[1] As noted several times by now, our focus in this thesis is on predictive performance. This is not the explicit aim of the latent class methods, although they have been used for this predictive task also. In particular, by virtue of being fully probabilistic, the parameters discovered latent class methods have a clear meaning as representing the probability of membership in one of several latent classes, which may be useful in some applications.

prediction datasets. Empirically, we find that the LFL method significantly outperforms popular unsupervised scores, such as Adamic-Adar and Katz. We find that it is possible to learn useful latent features on top of explicit features, which can give better performance than either model individually. Finally, we observe that optimizing with a ranking loss can improve AUC performance over a standard regression loss. Overall, on six datasets from widely different domains, some possessing side information and others not, our proposed method (Fact-BLR-Rank from Table 5.6 on datasets with side information, Fact-Rank on the others) has equal or better AUC performance (within statistical error) than previously proposed methods.

5.7 Acknowledgements

Chapter 5, in part, contains material as it appears in "Link prediction via matrix factorization", Proceedings of the 2011 European conference on Machine learning and knowledge discovery in databases - Volume Part II (ECML PKDD'11). Aditya Krishna Menon and Charles Elkan. The dissertation author was the primary investigator and author of this paper.

Chapter 6

Predicting Clickthrough Rates: Application to Response Prediction

Response prediction is a fundamental problem in computational advertising, where one needs to estimate the clickthrough rate of an ad when it is shown on a webpage. Existing methods for response prediction are based either on training standard classifiers on explicit features for pages and ads, or on statistical smoothening over baseline estimates. In this chapter, we show to interpret the problem as one of dyadic prediction, and propose a novel approach to the problem based on latent feature modelling. However, a standard application of models as used in collaborative filtering is not appropriate: compared to other dyadic prediction problems we have studied, such as collaborative filtering, there are a couple of key requirements in response prediction: (i) the predicted CTRs to be meaningful probabilities, and (ii) our goal is to not only fill-in the missing CTRs in this matrix, but also *smoothen* the CTRs for which there is limited historical data. We discuss how to meet these challenges, and in particular present a latent feature model that can exploit *hierarchical information* about pages and ads. Experiments on three real-world datasets show that our model improves on the performance of state-of-the-art response prediction models, and demonstrates the model's scalability.

6.1 Background and related work

We begin by giving some background on the response prediction problem, including a discussion of some related work, to serve as the foundation for our subsequent modelling approach.

6.1.1 The response prediction problem

Online advertising involves the interaction between two entities (Agarwal, 2008b): *publishers* (e.g. AOL, Yahoo), who own and manage webpages, and *advertisers* (e.g. Pepsi, Nike), who have products that they wish to market to users of the publishers' webpages via ads. A fundamental monetary paradigm in this form of advertising is *pay-per-click*. Here, each advertiser places a bid for their ad to be placed on a publisher's webpage, and pays the bid amount to the publisher only if the ad was selected to be displayed (referred to as a *view*) and subsequently clicked by a user. Amongst many competing ads, the publisher chooses to display the ad with the highest *expected revenue*, which is the ad's bid amount multiplied by the probability that it is clicked. This probability is known as the clickthrough rate (CTR) of an ad, and its reliable estimation is crucial for a publisher to maximize revenue. The task of estimating this probability is known as *response prediction*, or *CTR estimation*.

While the above assumes that the monetary paradigm is that an advertiser pays a publisher for every user click, recently, new response types have been adopted by advertisers. For example, in one scheme, the payout happens not when a user clicks on an ad, but only when she performs an action (called a *conversion*) after the ad is displayed. The action could be buying a product, filling out a form, *et cetera*. Accurate estimation of the probability for all response types is crucial to select the ad with highest expected revenue. "Response prediction" is therefore a blanket term that includes prediction for all

three types of user response. However, the specific type of conversion does not change the modelling problem, and so for the rest of the chapter, we will assume we are dealing with click events for simplicity.

6.1.2 Challenges in response prediction

Response prediction is a challenging problem for at least two reasons. First, the majority of ads have limited or no past history on a particular publisher page. This *sparsity* obviously hinders simple CTR estimation, and necessitates a principled way to exploit correlations in the data. Second, as most ads are clicked very infrequently, we have to predict the probability of a *rare event*, which is very challenging.

In both situations, it is beneficial to exploit the following useful fact: pages and ads can often be organized into pre-defined *hierarchies*. For example, the set of all ads can be categorized according to the advertiser who made the ad. Each advertiser in turn typically instruments a series of campaigns in which to display the ads, each with differing markets and goals. This means that there is a tree-like structure to the ads: Root $\rightarrow$ Advertiser $\rightarrow$ Campaign $\rightarrow$ Advertisement. See Figure 6.1 for a simple example of an ad hierarchy. (Notice that hierarchies are not necessarily trees, since an ad may be shown as part of multiple campaigns, for example.) Such hierarchies encode correlations between the CTRs. For example, the CTRs of all ads from the same campaign tend to correlated with one another. So, if a particular (page, ad) pair has only a few views, but the siblings of the ad have many views, then the reliability of its estimate can be increased by "borrowing" from the siblings' probability estimates in some principled manner. Thus, this type of information can be very useful in deriving reliable probability estimates when historical data is limited.

We now look to give more formal definitions of the terms introduced above, before proceeding to a discussion of related work.

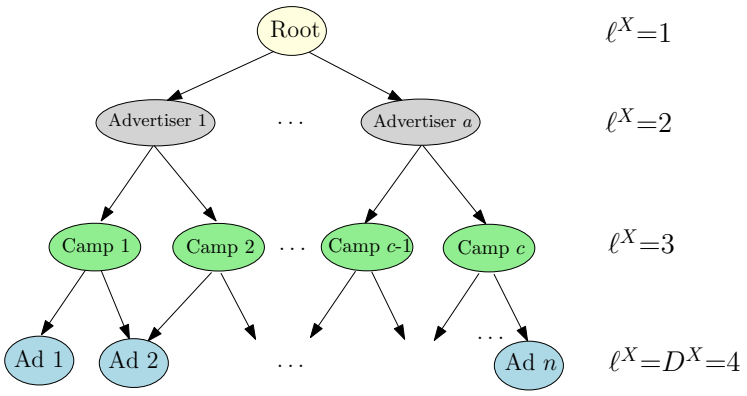


Figure 6.1. Hierarchical structure for advertisers.

6.1.3 Formal definitions

Formally, response prediction is the following task. We are interested in predicting probabilities given the interaction between (web)pages $\mathscr{P} = \{1,\ldots,m\}$ and ads $\mathscr{A} = \{1,\ldots,n\}$, where each entity is represented by a unique identifier. Given a page i and an ad j, we would like to predict $P_{ij=}\Pr[\text{Click}|i,j]$. The available historical data consists of matrices $C,V \in \mathbb{Z}^{m\times n}$, representing the number of *clicks* and *views* respectively that have been observed for particular (page, ad) pairs. Specifically, V_{ij} tells us how many times ad j was shown on publisher's ith page, and C_{ij} says how many times it was clicked once shown. By definition, we must have $C_{ij} \leq V_{ij}$.

We use $\mathscr{H}^X$ to denote a *hierarchy* over objects in the set X. A hierarchy is a directed graph $(\mathscr{V},\mathscr{E})$ with a *level* based structure: there is an injective function $\ell^X : \mathscr{V} \to \{1,\ldots,D^X\}$, where D^X is the *depth* of the hierarchy, such that a node $u \in \mathscr{V}$ can only have edges to nodes $v \in \mathscr{V}$ with $\ell^X(v) = \ell^X(u)+1$. Further, every node in levels $\{1,\ldots,D^X-1\}$ has at least one outgoing edge. The leaf nodes in level D^X correspond to the elements of X. Finally, we constrain that there is only one node $v \in \mathscr{V}$ with $\ell^X(v) = 1$, which we call the *root node*. Figure 6.1 represents an example of such a hierarchy over ads, where $D^X = 4$. We let $|\mathscr{H}^X| = |\mathscr{V}|$ denote the number of nodes in the hierarchy

$\mathscr{H}^X$.

We refer to the set of parents for a node u by $\text{Par}(u)$. A hierarchy is a *tree* if and only if every non-root node has exactly one parent. Any two nodes u, u' with a common parent are called *siblings*. We refer to the set of paths from the root node to a node u by $\text{Path}(u)$.

6.1.4 Existing models

Consider the problem of predicting $P_{ij} = \Pr[\text{Click}|\text{View}; i, j]$. Given historical data, arguably the simplest solution is the maximum likelihood estimate (MLE), P^{MLE}:

$$P_{ij}^{\text{MLE}} := \begin{cases} \frac{C_{ij}}{V_{ij}} & \text{if } V_{ij} \neq 0 \\ ? & \text{else.} \end{cases} \tag{6.1}$$

The "?" denotes that the MLE is undefined when $V_{ij} = 0$, since we have no historical data for the particular (page, ad) pair. Thus, the MLE is unusable if $V_{ij} = 0$. Similarly, if $V_{ij} \neq 0$ but is small, then the MLE is very noisy: for example, if an ad has been shown 5 times on a page and received 0 clicks, a CTR estimate of 0 is intuitively too extreme. As these two cases are the majority in practice, we need a different estimate $\hat{P}$ that *smoothens* the MLE to make it more reliable. There are three properties any estimator $\hat{P}$ should satisfy:

(i) it should provide estimates when $V_{ij} = 0$,

(ii) it should provide smoothened estimates when $V_{ij} \neq 0$ but is small, and

(iii) it should output meaningful probabilities in $[0, 1]$. Note however that the MLE is *consistent*, meaning that as $V_{ij} \to \infty$, $P_{ij}^{\text{MLE}} \to P_{ij}$. This implies that any estimator $\hat{P}_{ij}$ should ideally *converge to the MLE when V_{ij} is sufficiently large*.

Existing learning methods for response prediction can be categorized as feature-based or MLE-based. Feature-based methods build prediction models based on explicit features of an ad and a page, also known as *side-information*. These could include the textual content of an ad, its placement on the webpage, *et cetera*. Many existing feature based methods build prediction models using the logistic regression family (Richardson et al., 2007; Graepel et al., 2010). MLE-based methods smoothen the raw MLE via statistical models of clicks and views, with a popular choice being the Gamma-Poisson model (Agarwal et al., 2009, 2010a).

As discussed in the previous section, hierarchical structure encodes useful prior knowledge about CTRs. Hierarchical information has been successfully used in previous work on response prediction. (Agarwal et al., 2007) enforces relationships between estimates for (page, ad) pairs using a Markov model based on the hierarchy. The state-of-the-art LMMH model in (Agarwal et al., 2010a) stores a separate weight ϕ_{ij} for each pair of nodes (i, j) in the page and ad hierarchies, and the click probability is modelled as a product of weights for all pairs of nodes on the respective hierarchy paths for i and j. The key idea here is *fallback*: if a (page, ad) pair (i, j) has an insufficient number of views, then the model sets ϕ_{ij} to 1, causing a fall back onto the probability estimates of parent nodes.

6.2 From collaborative filtering to response prediction

We now discuss the dyadic prediction interpretation of response prediction. We then outline our strategy for applying a latent feature method to the method, given its success in other dyadic prediction tasks.

6.2.1 A dyadic interpretation of response prediction

Recall that the goal in response prediction is to construct an estimator $\hat{P}$ of the CTR that satisfies the three requirements outlined in Section 6.1.4. If we treat the matrix P^{MLE} as our input, then requirement (i) asks us to fill in the missing entries of this matrix. Immediately then, we see that response prediction can be thought of as a type of matrix completion. Strictly, it is more suitable to think of it as an instance of dyadic prediction, or a general matrix completion problem, as each cell for a particular webpage and ad combination can have multiple observations, and there exists predictive features for webpages, ads, and their interaction. Thus, like collaborative filtering say, we have the problem of trying to predict the "affinity" of webpages to ads, where the measure of affinity here is the clickthrough rate.

The dyadic prediction interpretation is useful because it lets us consider applying techniques that have been successful for similar tasks. In particular, we will be interested in applying latent feature methods, which we have previously shown to be useful in applications such as collaborative filtering and link prediction. At a high level, the intuition of applying such a model to response prediction is that "similar" ads will demonstrate "similar" CTRs on a given page, where "similar" is measured with respect to the learned latent space.

However, as we shall discuss, a mere black-box application of latent feature models as popular in collaborative filtering is insufficient. Fundamental modifications are required to deal with the specific challenges in response prediction data. We now provide an overview of the modelling changes needed, before discussing our approach in detail.

6.2.2 Overview of our latent feature model

There are two main ingredients to our enhanced latent feature model. First, we present a simple *confidence weighted* latent feature approach to response prediction. This

model deals with the fact that there may be multiple labels per dyad. The model also uses both latent features as well as side-information, and uses an iterative procedure to leverage the complementary nature of these features. Second, we provide a principled way to incorporate hierarchies into matrix factorization methods for collaborative filtering. The modelling details here are very different from how hierarchies are used in existing response prediction models, since our approach is based on a different foundation. Table 6.1 provides a summary of the components of our final model.

Table 6.1. Summary of the various components of the model used in this chapter.

Method	**Section**	**Key idea**
Confidence-weighted factorization	Section 6.3.1	$P_{ij}^{\mathrm{MF}} = \sigma(u_i^T v_j)$ with confidence determined by C_{ij}, V_{ij}
Fusion with side-information	Section 6.4.1	$P_{ij}^{\mathrm{SI}} = \sigma(u_i^T v_j + w^T x_{ij})$
Iterative refinement	Section 6.4.2	$P_{ij}^{\mathrm{MLE}} \rightarrow P_{ij}^{\mathrm{SI}}$ and confidence weighted factorization repeated
Hierarchy regularization	Section 6.5.1	$u_i \sim \mathcal{N}(u_{\mathrm{Par}(i)}, \lambda_u^{-1} I)$ $v_j \sim \mathcal{N}(v_{\mathrm{Par}(j)}, \lambda_v^{-1} I)$
Agglomerate fitting	Section 6.5.2	$u_i \sim \mathcal{N}(u_{\mathrm{Par}(i)}, \lambda_u^{-1} I)$ $v_j \sim \mathcal{N}(v_{\mathrm{Par}(j)}, \lambda_v^{-1} I)$; $C, V \rightarrow C^{\mathrm{Agg}}, V^{\mathrm{Agg}}$
Residual fitting	Section 6.5.3	$u_i \rightarrow \sum_{u \in \mathrm{Path}(i)} u_u, v_j \rightarrow \sum_{v \in \mathrm{Path}(j)} v_u$
Hybrid hierarchical model	Section 6.5.4	Hierarchy regularization + Agglomerate fitting + Residual fitting

Both ingredients of our approach are novel. To our knowledge, the only prior work on using collaborative filtering for response prediction is (Banerjee and Ramanathan, 2008), but it uses mixture-model approaches that have been superseded by matrix factorization. While (Banerjee and Ramanathan, 2008) concludes that these mixture-models do not deal with the rare event problem in response prediction data, our model demonstrates state-of-the-art performance. There has been work on exploiting hierarchical information in the CF literature, but only using neighbourhood models (Ziegler et al., 2008, 2004), and

so the modelling details are again completely different. Note also that such approaches are not applicable to response prediction for the same reasons that apply to standard factorization methods, which we discuss in Section 6.3.

We now discuss each of the key components of our model in turn.

6.3 A confidence-weighted factorization model

We begin our discussion by addressing why a black-box application of models from collaborative filtering, say, is insufficient. The analysis of the previous section suggests that response prediction can be solved by just feeding P^{MLE} as the input to a collaborative filtering model, such as one based on latent features. However, a standard model of

$$P_{ij}^{\text{MLE}} = u_i^T v_j$$

does *not* meet the requirements (ii) and (iii) we brought up in Section 6.1.4. In particular, it does not output valid probabilities in $[0,1]$, and it completely ignores the issue of confidence for entries with a few views. This necessitates fundamental changes to the factorization framework, which we detail below.

6.3.1 Confidence-weighted factorization

We can think of the problem of estimating $\Pr[\text{Click}|\text{View};i,j]$ as an analogue of the binary classification problem of estimating $\Pr[y=1|x]$. If we consider a click to be a "positive" event, and a non-click a "negative" event, we can think of the dyad (i,j) as comprising C_{ij} *duplicated* positive labels, and $V_{ij}-C_{ij}$ duplicated negative labels. Therefore, we have constructed a two dimensional table, where each cell consists of several binary labels. (Lifshits and Nowotka, 2007) constructs a similar table, but unlike our setting uses it in conjunction with explicit features for pages and ads.

There are now two problems to solve: we need to compute the probability of each individual label in the table being positive or negative, and we need to deal with the fact that there are multiple such labels per cell. For the first problem, suppose for the moment that each cell (i, j) has a single binary label X_{ij}. We now assume that for a fixed set of page latent vectors u, the probability of a positive label arising for ad j may be modelled via logistic regression with a weight vector v_j, giving us

$$P_{ij}^{\text{MF}} = \sigma(u_i^T v_j).$$

This is clearly symmetric to assuming that for a fixed set of ad vectors v, the probability of a positive label for a page i has the given form. If we now optimize over u, v jointly, we get the factorization model

$$\min_{u,v} \frac{1}{|\mathcal{O}|} \sum_{(i,j)\in\mathcal{O}} \left(-X_{ij} \log P_{ij}^{\text{MF}}(u,v) - (1_{ij} - X_{ij}) \log(1 - P_{ij}^{\text{MF}}(u,v))\right) + \Omega(u,v), \quad (6.2)$$

where $\Omega(u,v) = \frac{\lambda_u}{2}||u||_F^2 + \frac{\lambda_v}{2}||v||_F^2$. This objective is of course nothing but an instance of the binary LFL model, as introduced in Section 3.5.2, and consequently is also closely tied to the models discussed in Section 3.6.2.

In the above, each u_i represents a latent feature vector for pages, and each v_j a latent vector for ads. This assumes a low rank structure to P, which induces correlations between the matrix entries. This sharing of parameters allows us to smoothen probability estimates: even if a particular (page, ad) pair (i, j) has a few views, we can still reliably estimate $\hat{P}_{ij}$ by estimating u_i from all other ads shown on the page, and v_j from all other pages the ad is shown on.

The remaining problem is how to deal with the multiple entries in each cell of the table. Noting the connection of the above model to logistic regression, the duplicated

labels in the table are analogous to having the same example x appear several times in a supervised learning task, each time with a different labels. Logistic regression can easily be performed on such a dataset, and it is straightforward to check that the number of duplicated labels merely appear as weights in the likelihood term. The same holds in our problem, and so we get the objective

$$F(u,v;C,V) = \frac{1}{|\mathscr{O}|} \sum_{(i,j)\in\mathscr{O}} \left(-C_{ij} \log P_{ij}^{\mathrm{MF}} - (V_{ij} - C_{ij}) \log(1 - P_{ij}^{\mathrm{MF}})\right) + \Omega(u,v).$$

where P^{MF} is understood to be a function of u, v. This objective clearly takes into account confidence in the entries: if a dyad has small V_{ij}, then very little weight is placed on the likelihood term that measures its probability of click. The form of the objective is somewhat similar to that of (Hu et al., 2008), which targets CF datasets where there are no negative ratings. However, we employ the logistic loss so as to output valid probabilities. Further, our objective is motivated by a principled construction of a table of positive and negative labels from the C, V matrices.

The objective in Equation 6.3 is differentiable, and may be optimized using stochastic gradient descent (SGD), as with a standard factorization model.

6.3.2 Comparison to existing methods

Our confidence-weighted factorization overcomes the limitations of MLE. First, it can make predictions for (page, ad) pairs with no historical views. Second, even when there are historical views, it returns a *smoothened* estimate that is learned from multiple (page, ad) pairs, which has lesser variance than the raw MLE value. One caveat is that our method is *not* guaranteed to converge to the MLE as $V_{ij} \to \infty$, simply because the true probability P_{ij} may not be expressible as $\sigma(u_i^T v_j)$. However, since we noted that our optimization can be thought of as using the logistic loss, which is a *proper loss* function

(Buja et al., 2005), we can expect our estimates to be meaningful approximations of P_{ij}. In Section 6.6 we will demonstrate that empirically, given a large enough number of latent features k, our model converges to the MLE solution when V_{ij} is large.

Recall from Section 6.1.4 that existing methods for response prediction are based on either explicit features or MLE smoothing. We will see what potential advantages the other methods have over the factorization, and use these to guide extensions to our model. Looking at the feature-based methods, the obvious difference to our basic factorization model is that we learn *latent* features from the historical data, and use these to make predictions. In standard collaborative filtering problems, latent features are typically much more expressive than explicit features, because they can capture subtle correlations in the data (Pilászy and Tikk, 2009). However, the features used in response prediction – for example, the spatial placement of the ad on a webpage, the time it was displayed, *et cetera* – are known to be highly influential to CTR by themselves, and so a similar conclusion cannot obviously be made here. Instead, the approaches should be seen as *orthogonal* solutions that should be combined.

Compared to simple statistical models for smoothening the MLE, one advantage of the factorization approach is that we can learn a rich latent structure by choosing a sufficiently large number of latent features k. However, the state-of-the-art LMMH technique (Agarwal et al., 2010a) warrants a closer study. LMMH is based on two ingredients. First, the results of a baseline model only using features (such as logistic regression) are used to compute an expected number of clicks $\hat{C}_{ij}$ for each dyad. Next, one fits a log-linear model on the clicks C and *expected* clicks $\hat{C}$ under some statistical assumptions about the data. This log-linear model is fitted by exploiting *hierarchical* information for the pages and ads. This information is treated separately from the other features (such as placement of an ad) because of its special structure: specifically, as we noted in Section 6.1.4, the hierarchy for an ad directly encodes information about

correlations amongst CTRs. The reason is that this is a high-dimensional categorical variable, whose outcomes appear amongst multiple cells in the data matrix. The use of hierarchies helps LMMH handle the extreme sparsity problem quite effectively.

With this understanding, we now study two important extensions to the factorization model in turn: how to incorporate side-information, and how to incorporate hierarchies. The resulting model will be shown to have superior performance to both LMMH and the feature-based methods.

6.4 Incorporating side-information

As discussed earlier, pages and ads possess explicit features other than just their unique identifiers, such as the content of the ad, its placement on the page, *et cetera*. In the conetxt of collaborative filtering, we have available side-information for the dyad members and their interactions. In response prediction, the explicit features are strongly predictive of the CTR, as demonstrated by the fact that methods based just on explicit features were until recently the state-of-the-art (Richardson et al., 2007). It is thus prudent to incorporate this information into the factorization model for improved accuracy.

6.4.1 A joint factorization and feature model

To combine the latent and explicit features have, we employed the simple linear combination strategy described in Section 3.3.2, which for our problem is:

$$P_{ij}^{\mathrm{MF}} = \sigma(u_i^T v_j + w^T x_{ij}),$$

where x_{ij} consists of the explicit features for the dyad (i, j). To train this model, we found the following alternating strategy to work well: first, note that the model may be

rewritten as

$$P_{ij}^{\mathrm{MF}} = \sigma(\begin{bmatrix}1; w\end{bmatrix}^T \begin{bmatrix}u_i^T v_j; x_{ij}\end{bmatrix}), \tag{6.3}$$

This can be seen as a logistic regression model where the matrix factorization estimates $u_i^T v_j$ are treated as additional *input features* that are augmented with x_{ij}. This suggests a simple learning strategy: first, we train the standard factorization model, yielding estimates P^{MF}. This step is equivalent to assuming that $w \equiv 0$. Now we fix $u_i^T v_j$, and just optimize over w. This can be done by feeding the features $x'_{ij} = \begin{bmatrix}u_i^T v_j; x_{ij}\end{bmatrix}$ into a standard logistic regression model. The resulting solution, P^{SI}, predicts the click probability using both latent structure as well as side-information, thus enjoying the benefits of both approaches.

6.4.2 An iterative refinement procedure

Given the above model, a simple iterative procedure can be applied to further improve performance. Let us rewrite the confidence weighted factorization model of Equation 6.3 as

$$\min_{u,v} \frac{1}{|\mathscr{O}|} \sum_{(i,j)\in\mathscr{O}} -V_{ij}\left(P_{ij}^{\mathrm{MLE}} \log P_{ij}^{\mathrm{MF}} - (1 - P_{ij}^{\mathrm{MLE}}) \log(1 - P_{ij}^{\mathrm{MF}})\right) + \Omega(u,v).$$

Earlier, we motivated the above as a sensible way to incorporate confidence into the factorization model. However, the model is limited by the historical data: recall that if a cell has a small to medium number of views, then P_{ij}^{MLE} will be noisy. This means that our confidence weighting is *itself* noisy. Ideally, we would like to weight each entry by the true probability P_{ij}, which is the optimal measure of confidence to guide our factorization model. But of course, if we knew this quantity, our learning would be complete. Yet this motivates the following EM-style procedure: we take the predictions from our above model, P_{ij}^{SI}, and use these in place of P_{ij}^{MLE} in the above equation. We

now *re-learn* our confidence-weighted factorization model using these new confidences. The idea is that since P_{ij}^{SI} is a more reliable estimate of the true probability that P_{ij}^{MLE}, the factorization model is encouraged to focus its modelling efforts on the "right" dyads. We can now iterate by feeding the results of the newly learned factorization into a logistic regression model, and use the resulting estimates $\hat{P}_{ij}^{SI}$ as a fresh set of confidence weights. This process may be repeated till convergence. This scheme exploits the complementary properties of the two models so that by improving the outputs of one model, we can feed in more reliable inputs to the other.

We note a similarity to the previously discussed approach followed in LMMH, with two subtle but important differences:

(i) LMMH uses a feature-based model to generate the initial refinement of the data, based on which further smoothing is applied. By contrast, we apply the *factorization* as the first step to refine the data, and feed this into the feature-based model; and

(ii) We iterate the process till convergence. Distinction (ii) can naturally be expected to improve performance, a fact we verify empirically.

We argue that distinction (i) is important, because our factorization approach yields much more reliable estimates than a feature-based method if we have even a small amount of historical data, as observed in (Pilászy and Tikk, 2009). The smoothing applied in the first phase fundamentally affects the overall performance, because only if the results of the first model are sufficiently rich does the second stage phase gain a significant advantage over modelling the training data as-is. Put another way, LMMH is made to solve the difficult problem of fitting the residual of an only moderately reliable feature-based model. By contrast, in our case the second phase involves a much simpler task, since most of the structure is already captured by the factorization. These intuitions will be empirically corroborated in Section 6.6.

6.5 Incorporating hierarchies

Recall that the other ingredient behind the success of existing response prediction methods is the use of hierarchical information to overcome the extreme data sparsity. We now look how this can be incorporated into our factorization model. There are three basic ideas: (i) *hierarchical regularization*, where we keep a latent vector for each node in the page and ad hierarchies, and enforce priors based on this to induce correlations; (ii) *agglomerate fitting*, where we refine (i) by constructing more informative priors based on agglomerating subtrees in the hierarchy; and (iii) *residual fitting*, where we use fit the residual of the basic factorization using appropriate latent vectors in the hierarchy. Each of the methods modifies different parts of the objective function in Equation 6.3 (as noted earlier, without altering differentiability). The methods can in fact be used in *conjunction* with each other, and thus represent different facets of a larger approach to incorporating hierarchies. We will return to this matter after first discussing the methods in detail.

6.5.1 Hierarchical regularization

Our first idea is to let every node in the hierarchy possess its *own* latent vector, and use this to construct priors that constrain the latent vectors. For simplicity, we will use the notation u_i to denote the appropriate vector associated with the node i, and similarly for v_j: in Figure 6.2, $v_1, \ldots, v_n$ represent the latent vectors for ads as before, while $v_{n+1}, \ldots, v_{n+c}$ represent the latent vectors for the campaigns, and so on. When we refer to the matrix u, it is now in $\mathbb{R}^{|\mathscr{H}^P| \times k}$, and similarly for v.

Recall that the standard ℓ_2 regularizer on u, v corresponds to placing a zero-mean Gaussian prior on them, i.e. $u_i \sim \mathscr{N}(0, \lambda_u^{-1} I)$. With the above setup, we impose *hierarchical priors* on the latent vectors, such that each latent vector has a prior so that it

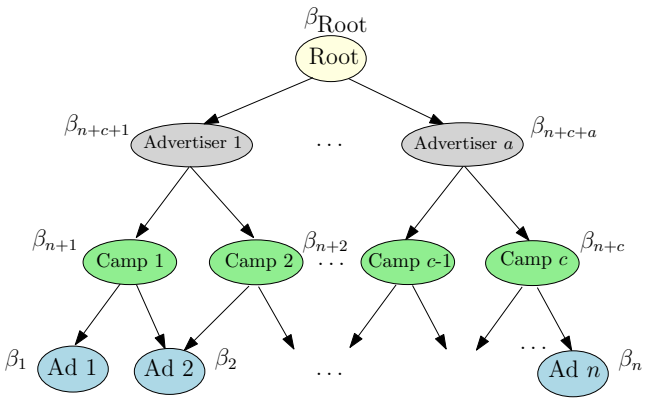


Figure 6.2. Each node in the hierarchy has a latent vector.

behaves like its parent in expectation:

$$u_{\text{Root}} \sim \mathcal{N}(0, \lambda_{\text{Root}}^{-1} I), \forall i \in \mathscr{H}^P - \{\text{Root}\} u_i \sim \mathcal{N}(u_{\text{Par}(i)}, \lambda_u^{-1} I) \tag{6.4}$$

To find the joint prior over u, note that given its parent, a node is conditionally independent of all higher level nodes. Thus, we replace the standard regularizer in Equation 6.3 by

$$\begin{aligned}\Omega(u,v) = &\frac{\lambda_u}{2} \sum_{i \in \mathscr{H}^P} ||u_i - u_{\text{Par}(i)}||_2^2 + \frac{\lambda_v}{2} \sum_{j \in \mathscr{H}^A} ||v_j - v_{\text{Par}(j)}||_2^2 + \\ &\frac{\lambda_R}{2}(||u_{\text{Root}}||_2^2 + ||v_{\text{Root}}||_2^2).\end{aligned} \tag{6.5}$$

The regularizer helps in estimating vectors when there are few corresponding views. Suppose there are two siblings u, v with a common parent, and that node u has only a few views while node v has many views. For v, the dominating term in the objective will be the loss function, so its parameters will be optimized to be predictive for the CTR. For u, the regularizer will dominate and push its latent vector to be similar to the parent node. In turn, the parent is encouraged to be close to its children, and so u, v are indirectly encouraged to be similar to each other. This means that u will "borrow strength" from v.

The above easily handles hierarchies that are not trees: instead of the prior

mean being a parent node's latent vector, we can use the *average* of all parents' vectors. This approach has a total of $(|\mathscr{H}^P| + |\mathscr{H}^A|)k$ parameters. In practice, hierarchies are often bottom-heavy, so that $|\mathscr{H}^X| = O(|X|)$. Thus we learn roughly the same number of parameters as in the standard setting. Of course, the constant in the $O(\cdot)$ makes a difference in two ways: it adds the risk of overfitting, and it potentially increases the number of local optima. We will empirically verify that neither of these risks are seriously manifested in practice.

6.5.2 Agglomerate fitting

In the previous section, the latent vectors for non-leaf nodes only appear in the regularizer, and hence are only indirectly affected by the click and view data. Intuitively, by making them directly depend on the data, they will serve as more informative priors for the leaf nodes. To do this, we use the hierarchy to *agglomerate* the click/view data across many pages and ads, and try to predict this data using the appropriate latent vectors. For example, for a (page, campaign) pair (i,c), we agglomerate the clicks/views for all children of c when shown on page i. We model the resulting data using the vectors u_i and v_c. This will learn a sensible prior for the children's latent vectors.

Formally, we construct the Cartesian product of the hierarchies, $G := \mathscr{H}^P \times \mathscr{H}^A$. Any $(u,v) \in G$ is a tuple of nodes from the page and ad hierarchies, and we associate with it the aggregated clicks of all its children:

$$C^{\text{Agg}}(u,v) = \sum_{(u',v'):(u,v)\in \text{Par}((u',v'))} C^{\text{Agg}}(u',v').$$

The base case is when both u and v are leaf nodes in the respective hierarchies, so that $C^{\text{Agg}}(u,v)$ is the standard click value C_{ij}. We repeat the same process for the views. We now have click and view matrices $C^{\text{Agg}}, V^{\text{Agg}} \in \mathbb{R}^{|\mathscr{H}^P|\times|\mathscr{H}^A|}$. We optimize the objective

of Equation 6.3 using $F(u, v; C^{\text{Agg}}, V^{\text{Agg}})$ along with the regularizer Ω of Equation 6.5. Therefore, every latent vector appears in both the objective and the regularizer. Figure 6.3 illustrates how we can think of the new matrices as being augmentations of the original C, V.

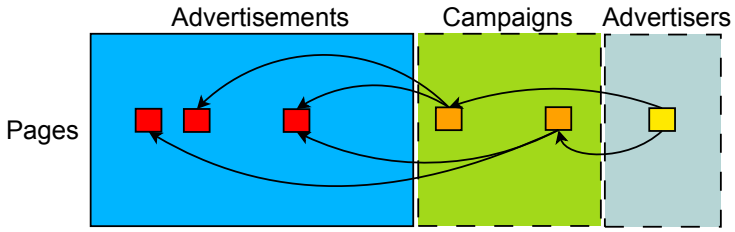


Figure 6.3. Illustration of the agglomeration process. Arrows denote that the clicks/views are added up.

There is a subtle problem with implementing agglomerative fitting as-is: the clicks and views of individual (page, ad) pairs will be "drowned out" by the ones corresponding to parent nodes, and our predictions will no longer be fine-grained for each individual pair. Concretely, consider some page i, an ad j, and the ad's campaign c. Then, the number of clicks/views for the (page, campaign) pair (i, c) will be greater than the clicks/views for the (page, ad) pair (i, j), by construction of the agglomerated click and view matrices. Since the model of Equation 6.3 is confidence weighted by the number of views, we will essentially ignore the contributions of the (page, ad) pairs. One fix to this problem is to train our model in stages. Letting u^{Par} and v^{Par} denote the latent vectors for non-leaf nodes, we perform the following steps:

1. Learn u, v with u^{Par}, v^{Par} fixed. This is the standard factorization model of Equation 6.3.
2. Learn u^{Par}, v^{Par} with u, v fixed. This asks the parent nodes to be predictive for the agglomerated data, but *using* the already learnt vectors for pages and ads e.g. when we predict for a (page, campaign) pair.

3. Relearn u, v with $u^{\text{Par}}, v^{\text{Par}}$ fixed. This asks us to respect the hierarchical prior when reconstructing the data matrix. Since the parent nodes are not optimized at this stage, we do not overfit on the agglomerated entries. We can use the result of step (1) as initialization here.

One can think of this process as using hierarchies to give a principled way of finding good local optima for u, v.

6.5.3 Residual fitting

The previous extensions all enforce similarity among parameters based on the hierarchy, but finally use the prediction $\sigma(u_i^T v_j)$. A sensible idea is to modify this prediction itself based on the hierarchy. Specifically, for the pair (i, j), we use the prediction $\sigma(\tilde{u}_i^T \tilde{v}_j)$, where $\tilde{u}, \tilde{v}$ modify the original vectors u, v based on the hierarchy. A simple choice is the additive model

$$\tilde{u}_{ik} = u_{ik} + \sum_{u \in \text{Path}(i) - \{i\}} u_{uk}$$

and similarly for $\tilde{v}$. Here, the fine-grained latent features for each page are modelled as corrections over coarser latent features of parent nodes, which may be thought of as *bias* terms. This technique easily generalizes to non-tree hierarchies if we use the average latent vector of all parent nodes on a *per-level* basis.

6.5.4 Putting it all together: a hybrid method

We can easily combine the ideas of the previous sections to create the following hybrid approach:

(i) our prediction for dyad (i, j) involves all latent vectors along the respective paths in the hierarchy,

(ii) we impose hierarchical priors on the latent vectors, and

(iii) we force the latent vectors for parent nodes to be predictive for agglomerated data.

We later demonstrate that empirically, this hybrid performs significantly better than any of its individual components. We also note that the hybrid can be easily augmented with additional side-information using the framework detailed in Section 6.4.

6.5.5 Handling cold-start pages and ads

We quickly comment on how the hierarchical extensions let us estimate the latent vectors for a cold-start page or ad. As a simple example, suppose there are several pages that share a common parent node. Suppose one of them, page i, has no past history. Then, if we use hierarchical regularization, u_i will only appear in the regularization term Ω in Equation 6.5. Thus, it optimized by setting it equal to $u_{\text{Par}(i)}$. If this parent node is estimated via agglomerative fitting, it will be a good representative of the common structure between the siblings of i. Therefore, we will be able to make reasonable predictions for page i on test data. This argument holds equally for entities that are "almost cold-start" i.e. which have extremely limited past history. Having inferred the latent vectors in this manner, we can now employ the framework of Section 6.4.

6.6 Experimental design

We now describe the design of our experiments with the model proposed above.

6.6.1 Aims of the experiments

Our experiments aim to address the following questions:

- how well does the basic confidence-weighted factorization model perform compared to existing response prediction methods?,

- do the hierarchical extensions improve performance?, and
- does exploiting latent features and side-information offer noticeable performance gains?

To answer these questions, we conducted experiments on three very large real-world datasets, which shows our model is highly scalable and that it outperforms current state-of-the-art methods.

6.6.2 Datasets used

Our datasets were collected from Yahoo! traffic streams. We used the same three datasets as (Agarwal et al., 2010a). These datasets each involve the interaction between ads and pages, and differ in the nature of the interaction. The first data set, **Click**, records the event of an ad being clicked when shown on a page. Both pages and ads have a two level hierarchy in this dataset. The second dataset, **PVC**, measures how many users performed a pre-determined action after *viewing* an ad. Finally, the post-click conversion dataset (**PCC**) measures how many users performed a pre-determined action after *clicking* an ad. Both **PVC** and **PCC** have four level ad hierarchies, and the same two level page hierarchy as with **Click**. Of the three datasets, **PVC** is the sparsest, since post-view conversions are generally difficult to measure. There are $\sim$(90B, 3B) (train, test) records for **Click**, $\sim$(7B, 250M) for **PVC**, and $\sim$(500M, 20M) for **PCC**. The three datasets also include *interaction features* for the user involved in each interaction (e.g. the age and gender of the user that clicks on an ad, how recently the ad was shown to the user, *et cetera*). Since the datasets are proprietary, we cannot report the corresponding number of pages and ads, clicks and views, nor the number of days used to construct the train-test split. Instead, we just report the number of nonzero records ($\geq$ 1 view) in the train and test sets. For each dataset, following (Agarwal et al., 2010a), we split the test data into 20 components and use these to estimate the variance of our results.

6.6.3 Methods compared

We implemented the confidence-weighted factorization of Section 6.3 (denoted "CWFact" in our figures), two of our hierarchical extensions (denoted "Agglomerative", and "Residual"), and the hybrid hierarchical method of Section 6.5.4 (denoted "Hybrid"). (The hierarchical regularization method in Section 6.5.1 by itself showed similar performance to CWFact, as is essentially subsumed by Agglomerative.) The basic versions of these methods did not include explicit features, and are purely based on matrix factorization. We additionally ran the CWFact and Hybrid methods with side-information using the joint model of Section 6.4, and denote these methods by "CWFact+LogReg" and "Hybrid+LogReg". Finally, the method "Hybrid+LogReg++" used the iterative scheme detailed in Section 6.4.2, where we iteratively use the model's predicted number of clicks as input to a residual model. We ran this model till convergence.

We trained all models using stochastic gradient descent (SGD), and used the MapReduce code from the Apache Mahout project (http://mahout.apache.org/) to scale to the challenging sizes of the datasets. We parallelized the optimization by fixing the page latent features u_i and then optimizing for the advertisement latent features v_j using SGD. This optimization of each individual v_j can be done in parallel. Once the estimates of the v_j's converged, we fixed their values and optimized the u_i's in parallel; this alternating scheme was repeated until convergence. For the hierarchical regularization scheme, we optimized the parameters level-by-level based on the hierarchy. Thus, for the initial optimization of advertisement features v_j, the campaign features were fixed and regularization was done towards these fixed latent features. Once optimization of the advertisements concluded, the campaign features were optimized with regularization towards fixed advertiser features, and so on.

Regarding parameter selection, we picked the strengths of regularization λ_u, λ_v

by cross-validation. We fixed the number of latent features $k = 100$. Qualitatively, we found that the gains of our model were modest with a much smaller value of $k = 10$, indicating that it is important to choose a sufficiently large number of latent features to capture the structure in the data.

We compare to three state-of-the-art methods, discussed earlier in Section 6.1.4. The first is the model of (Agarwal et al., 2010a), denoted "LMMH". The second and third are variants of a logistic regression model that uses explicit features for pages and ads, similar to (Richardson et al., 2007). Prior to LMMH, these models were the state-of-the-art for response prediction (Richardson et al., 2007). We fed these logistic methods input features derived from the page and ad hierarchies. For example, each ad has its corresponding advertiser ID as a categorical feature. The first variant of logistic regression, "LogReg", just used all the raw features as input. The second variant, "LogRegHash", also uses *cross-features* of pages and ads. To mitigate the dramatic increases in the number of features, following (Agarwal et al., 2010a), we applied the hashing trick of (Weinberger et al., 2009) to compress the features into a smaller number of bins.

6.6.4 Evaluation methodology

We use the standard performance metric of Bernoulli log-likelihood, which measures how well the predicted CTRs match the CTR on the test set. For a test set $\mathscr{T}$ of (page, ad) pairs with clicks and views $C^{\mathscr{T}}, V^{\mathscr{T}}$, given the model's predictions $\hat{P}_{ij}$, the log-likelihood is:

$$\mathscr{L} = -\sum_{(i,j)\in\mathscr{T}} \log \hat{P}_{ij}^{C_{ij}^{\mathscr{T}}} (1-\hat{P}_{ij})^{(V_{ij}^{\mathscr{T}} - C_{ij}^{\mathscr{T}})}.$$

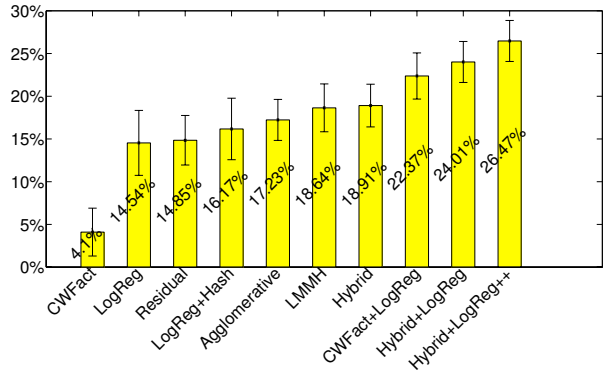
Again, for confidentiality reasons we cannot report raw log-likelihood numbers for any method. Hence, on all datasets we report the % lift in likelihood over the appropriate

baselines.

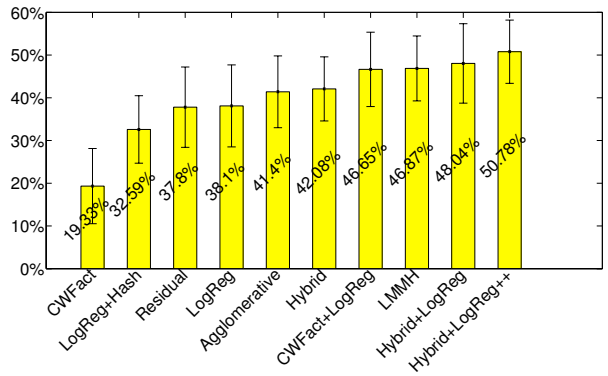
6.7 Experimental results

Our results are summarized in Figure 6.4. We see that for all datasets, the Hybrid+LogReg++ model gives the best log-likelihood lift, in particular outperforming LMMH in terms of average lift, and with a consistently lower standard deviation. The multiple iterations used in Hybrid+LogReg++ are seen to have a useful boost over Hybrid+LogReg, which by itself outperforms LMMH on all but PCC, where LMMH performs marginally better. This shows that the combination of factorization, hierarchies and side-information gives state-of-the-art performance. Note also that the good performance of Hybrid+LogReg shows that it is not just the multiple iterations that give us the advantage over LMMH. A closer study reveals the value of each of these components in our final model. The importance of using hierarchies is demonstrated by the surprisingly poor performance of the basic CWFact model, which is outperformed by even the simple LogReg models. However, these feature-based models are in turn significantly outperformed by the Hybrid method that adds hierarchies to the factorization. Note also that the Hybrid method always manages to significantly improve over the individual Agglomerative and Residual sub-models. Yet, we conclude that it imperative to have a model that *combines* latent features and side-information, because the Hybrid method by itself is merely competitive with LMMH; it is only when we add side-information to the factorization that we start to see improvements.

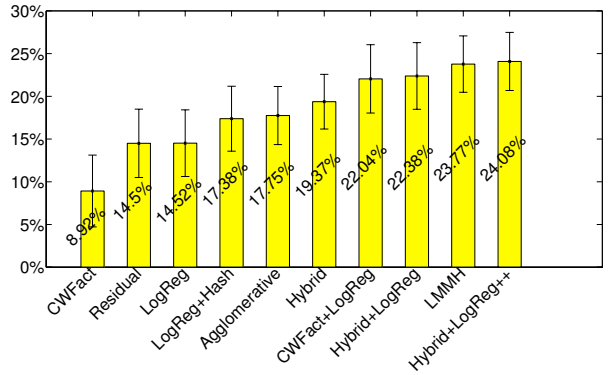
For our models that used both latent features and side-information, we found that the LogReg component invariably put relatively little weight on the standard features, meaning that the factorization "feature" was considered most important. This is expected, since the factorization model by itself is strongly predictive of the CTR. To further analyze the interplay between the factorization and side-information, Figures 6.5a and 6.5b gives



(a) Click.

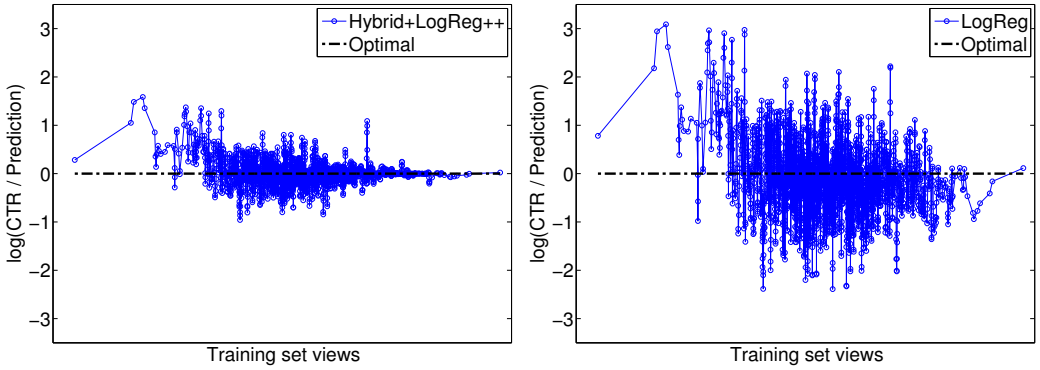


(b) PVC.



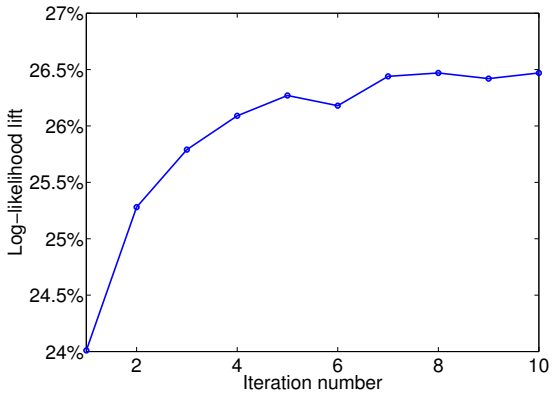
(c) PCC.

Figure 6.4. Log-likelihood lifts on large-scale datasets.



(a) Hybrid+LogReg++ versus test CTR.

(b) LogReg versus test CTR.



(c) Per-iteration lift.

Figure 6.5. Analysis of factorization model's performance on Click. See text regarding the missing axes in (a) and (b).

log-log plots of the ratio of predictions of the Hybrid+LogReg++ model and the LogReg model to the test set CTR, ordered by increasing number of views on the training set. The flat line in black is the optimal solution where the model prediction matches the test set CTR, and both are displayed on the same y-range for ease of comparison. (Recall that we are unable to report exact view numbers for our datasets. Hence, in both plots, we removed identifying information about the number of views on the x-axis.) There are two striking characteristics in the plots: first, the Hybrid+LogReg++ model has significantly less variance than LogReg, which shows that its factorization component helps the LogReg component by capturing most of the structure in the data through latent features. Second, the Hybrid+LogReg++ model converges much quicker to the true CTR than LogReg model in terms of number of views. This shows that our model can successfully smoothen at a much greater degree of sparsity in the training data, corresponding to dyads with a few number of views. As we noted earlier, similar behaviour has been observed for standard collaborative filtering data (Pilászy and Tikk, 2009).

Finally, to see how the performance of Hybrid+LogReg++ varies across each iteration, Figure 6.5c shows the lifts on the **Click** dataset after each iteration. An iteration here refers to a single application of both the Hybrid and LogReg models, using the results of the previous iteration as input. The results are encouraging: we almost always improve the log-likelihood as we run the model for more iterations. (We observed similar results on the other datasets.) This shows that with minimal added cost, the simple iterative scheme of Section 6.4 offers further gains for our framework.

6.8 Conclusion

In this chapter, we showed how we can improve on the state-of-the-art in response prediction using a novel latent feature approach, based on a dyadic interpretation of the problem. Our model exploits both latent and explicit features, the latter being in the

form of descriptions of webpages and ads. We proposed an iterative procedure to further exploit the complementary nature of latent and explicit features. Finally, we showed how to incorporate hierarchical information for pages and ads into our model. Experimental results on web-scale Yahoo! traffic data show that our method outperforms existing response prediction approaches, and that it can handle high amounts of sparsity in the training data.

There are several avenues for future work. First, the matrix factorization model proposed here uses MAP estimates of the weight vectors. A Bayesian approach involving marginalization of these parameters would be useful, though it poses challenges due to the interdependencies between parameters. Second, it is important to address other real-world complexities that arise in response prediction, such as the fact that ad behaviour is time-varying.

6.9 Acknowledgements

Chapter 6, in part, contains material as it appears in "Response prediction using collaborative filtering with hierarchies and side-information", Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '11). Aditya Krishna Menon, Krishna-Prasad Chitrapura, Sachin Garg, Deepak Agarwal, and Nagaraj Kota. 2011. The dissertation author was the primary investigator and author of this paper.

Chapter 7

Predicting Labels for Dyad Members

Thus far, we have focussed on dyadic prediction problems where the label of interest is the interaction between the members of a dyad (i, j). We have seen many practically important applications of this problem, such as collaborative filtering and link prediction. A related problem that has received less attention is that of predicting labels associated with the *individual* dyad members. This problem is also of practical importance: for example, we may have labels associated with some subset of users in a social network dataset, telling us how the user responded to an advertising campaign, and we would like to extend these labels to the entire set of users. In this chapter, we show how we can reduce the problem to the standard dyadic prediction problem studied thus far. We contrast this approach to existing methods in the related literature of within-network classification, and show analytically and empirically the benefits of our approach.

7.1 Problem definition

To define the dyadic label prediction problem, we will first review the well-studied problem of *within-network classification* in graphs.

7.1.1 Within-network classification

Suppose we have as input a graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ that is *partially labelled*, by which we mean that we know the values of a labelling function $f : \mathscr{V} \to \mathscr{Y}$ for some subset of nodes $\mathscr{V}' \subset \mathscr{V}$. The goal of within-network classification (or *relational learning*) is to predict the labels for the remaining nodes $\mathscr{V} \setminus \mathscr{V}'$ (Macskassy and Provost, 2007). Figure 7.1 gives an illustration of the problem. This problem has many practical applications, such as:

- Predicting characteristics of users or movies in collaborative filtering datasets. For example, based on users' preferences for certain movies, we may wish to predict whether they are likely to be interested in a new marketing campaign.

- Scoring suspiciousness or trustworthiness of users in a social network, based on their social relationships and certain observed characteristics or features. An important specific example is uncovering likely terrorists based on their connections (Sen and Getoor, 2006).

- Predicting which strains of bacteria will be observed in various food processing plants, based on a graph with links between facilities with historical precedent of contamination co-ocurrence (Sarkar et al., 2008).

The within-network classification problem can be seen as an extension of the link prediction problem in graphs, where the goal is to predict a label on an edge. Recall from Section 2.4.3 that there is an equivalence between link prediction and dyadic prediction. In the language of the latter, within-network classification can be seen as the problem of predicting the label associated with a single dyad member, rather than the interaction between dyad members. In what follows, we will refer to the dyadic interpretation of this problem as *dyadic label prediction*. In fact, we will subsequently formalize the problem

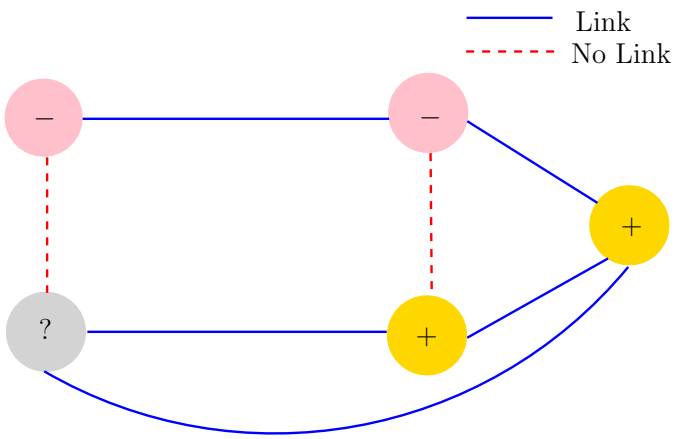


Figure 7.1. Example of within-network classification. Here, nodes may either be labelled as "+" or "-". The goal is to determine the label of the bottom left node, based on its links to other nodes.

in the language of dyadic, rather than link prediction. The main reason is to inherit the generality with which dyadic prediction is often treated, most pertinently the fact that we treat missing labels for dyad interactions as the norm. By contrast, in within-network classification, it is generally assumed that the status of all potential edges in the graph is completely known. This assumption is clearly restrictive.

7.1.2 Formal definition of dyadic label prediction

We now describe the dyadic label prediction problem formally. For convenience, we will ignore the existence of side-information for the dyad members, although these may be incorporated seamlessly. Thus, the only information available for the dyads is their identity. Further, we will describe the problem in terms of collaborative filtering, so that we may use the words "user" and "movie" and "rating". This is merely for concreteness of exposition.

Recall that a standard collaborative filtering problem may be thought of as involving a data matrix $X \in \mathscr{Y}^{m \times n}$, where m is the number of users, n is the number of movies, and $\mathscr{Y}$ is the space of possible ratings, plus a special entry "?" to denote a rating

that is missing. The goal is to fill in the missing entries in X. Now consider the setting where we also have a *user label matrix* $Y \in \{0,1,?\}^{m \times L}$. Here, a single row $y_i \in \{0,1\}^L$ is referred to as the *label* for the user i. Note that the label is a vector, meaning that we address the *multilabel learning* setting. (Of course, this means we can also handle the multiclass setting.) We refer to the individual elements in a label as a *tag*. Our goal now is to fill in the missing entries of Y, or equivalently to predict the complete label vectors for all the users in the data.

We make three observations about this problem. First, it does *not* require that the entries in X are fully observed. This generality is essential in many real-world applications, such as predicting labels for users based on a partially observed social network. Second, the placement of missing entries in Y is allowed to be arbitrary. However, it is common to consider the setting where the label vectors are fully specified for some dyad members, and completely unspecified for the rest. (That is, for each i, $\sum_{l=1}^{L} \mathbf{1}[y_{il} = \text{“?”}] \in \{0, L\}$.) Third, the problem as stated is a form of transductive learning, because the goal is only to make predictions for the users in X. In principle, it is simple to extend this to allow for cold-start settings, where we must label a node for which none of its connections are known.

7.1.3 Existing approaches

To our knowledge, essentially all prior work on the dyadic label prediction problem has been in the context of within-network classification. Thus, we review some existing methods in this literature, which fall into two main schools. The first is to exploit the topological structure of the graph to infer labels for unobserved nodes. The idea is that if the neighbors of a node all have a positive label, that can help us decide with confidence whether to label the node as positive or not (depending on whether there is a positive or negative neighbor-label correlation). A popular method along these lines

is wvRN (Macskassy and Provost, 2003, 2007), which assumes that the neighbors of a node capture all the relevant information for predicting its label, and that the weights of edges correspond to the strength of influence of one node to another. More sophisticated extensions of this idea involve random walks on the input graph (Callut et al., 2008), where intuitively one assigns the label of a node based not only on its neighbors but also on nodes that are easy in some sense to reach from it.

As should be familiar at this stage, a limitation of the above schemes is that they are based on pre-defined rules, and do not adapt to the characteristics of the given data. In this sense, they are similar to the unsupervised dyadic prediction methods discussed in Section 2.5.1. These limitations are overcome by the other school of model, which is to learn *latent features* from the graph, and then feed these into a supervised learning algorithm. The motivation here is clear: if we were given predictive features for the nodes, then the problem would simply reduce to supervised learning. Since we do not necessarily possess such features, we try to learn them from the link structure amongst the nodes. The general strategy is:

1. Denote the adjacency matrix of the graph as $X \in \mathcal{X}^{m \times m}$, and the label matrix as $Y \in \{0, 1, ?\}^{m \times L}$. Then, learn a latent representation $U \in \mathbb{R}^{m \times k}$ of the nodes based on X and Y. As usual, k denotes the number of latent features.

2. Train a multilabel classifier to predict the labels y_i for the labelled nodes, using the feature representation $u_i \in \mathbb{R}^k$.

3. Use the above classifier to predict the labels for the unlabeled nodes.

The hope is that the learned U captures some useful information about the nodes that is predictive of the labels.

Given this general formulation, there are at least two ways to conduct the first step. One can learn U just from the input X, independent of the labels Y; we call these

unsupervised latent features. A popular method using unsupervised latent features is named SocDim (Tang and Liu, 2009). A limitation of this approach is that we might end up learning features that are predictive for the ratings, but uninformative for the labels; see the examples discussed in (Yu et al., 2006), for example. The other way is to use both X and Y to influence the matrix U; we call these *supervised latent features*. Supervised latent features are used in the supervised matrix factorization (SMF) approaches applied to network prediction in (Zhu et al., 2007), and to latent semantic indexing in (Yu et al., 2005). The idea of these methods is to learn U to jointly optimize least-squares reconstruction error of X and a one-versus-rest SVM classifier for the labels. We will discuss both types of method in more detail in Section 7.3.

In the following, we will focus on analyzing and comparing ourselves to the latent feature approach, as it has been shown to give state of the art results (Tang and Liu, 2009).

7.2 Our approach: reduction to dyadic prediction

This section shows how we can predict labels for dyad members by creating dummy entities, and reducing the problem to standard dyadic prediction. This reduction has not been analyzed in previous research. Although the reduction is useful, we explain why a better strategy is to have a tradeoff between predicting missing ratings and predicting labels.

7.2.1 Reduction: labels as special movies

Consider first the setting where one learns unsupervised latent features $U \in \mathbb{R}^{m \times k}$ from the input $X \in \mathscr{Y}^{m \times n}$. We need to learn a classifier to model $Y \in \{0, 1?\}^{m \times L}$ based on U. Arguably the simplest solution is to learn an independent linear model for each tag in Y. Here, we learn a weight $W \in \mathbb{R}^{L \times k}$, where L is the number of tags and k the number

of latent features, so as to minimize

$$\min_{W} \sum_{(i,t)\in\mathscr{T}} \ell(y_{it}, (Wu_i)_t) + \lambda\Omega(W), \tag{7.1}$$

where $\mathscr{T}$ denotes the set of non-missing tags. The choice of loss ℓ and regularizer Ω may be chosen as appropriate for the nature of the data.

The above approach is reasonable, but has two apparent limitations. First, it does not attempt to exploit any correlations amongst the tags[1]. Second, as discussed earlier, learning latent features in an unsupervised manner ignores the label information, which means that the latent features may be uncorrelated with the labels and thus yield poor accuracy. Motivated by this, suppose we decide to do this optimization jointly with U, so that the latent features in U are supervised:

$$\min_{U,V,W} \sum_{(i,t)\in\mathscr{T}} \ell(y_{it}, (Wu_i)_t) + \sum_{(i,j)\in\mathscr{O}} \ell(x_{ij}, u_i^T v_j) + \lambda\Omega(U,V,W), \tag{7.2}$$

where $\mathscr{O}$ denotes the set of observed dyadic interaction labels in X. We note that to make predictions for test data, one can use the learned weights W, or train a classifier on the learned U in a second stage.

One consequence of learning U in this manner is that the predictions for the tags are no longer independent of each other. The reason is that we are learning U jointly with W. Therefore, there are implicit interactions between the weights of the tags, because they need to be predictive for the rest of the data matrix. We also note that this approach exploits the transductive nature of the problem: we are only interested in making predictions for users known during training, and so we use the derived latent

[1] In fact, in the multiclass setting, where each label has exactly one active tag, the above constitutes a one-versus-rest scheme to multiclass classification. This scheme has been shown to work well in practice (Rifkin and Klautau, 2004), although theoretically its statistical consistency is not guaranteed (Zhang, 2004a).

features for each user as predictive covariates for the labels. While the latent features depend explicitly on the users with a label known during training, they also depend implicitly on other users. This is because the latent features for labelled users are used to derive the latent features for movies, which are shared by all users.

The idea of learning supervised latent features is not new. Indeed, in the case of fully observed data in $\mathscr{O}$, and square-loss, the objective of Equation 7.2 is the SMF approach of (Zhu et al., 2007; Yu et al., 2005). However, to our knowledge, it has not been pointed out before that the objective is still sensible in the incomplete case, and may be equivalently interpreted as

$$\min_{U,\tilde{V}} \sum_{(i,j)\in\mathscr{O}'} \ell(\tilde{x}_{ij}, u_i^T \tilde{v}_j) + \lambda\Omega(U,\tilde{V}),$$

where $\mathscr{O}' = \mathscr{O} \cup \mathscr{T}$, $\tilde{X} = \begin{bmatrix} X & Y \end{bmatrix} \in \mathbb{R}^{m\times(n+L)}$, and $\tilde{V} = \begin{bmatrix} V & W \end{bmatrix} \in \mathbb{R}^{k\times(n+L)}$. That is, we are merely performing dyadic prediction on the augmented matrix $\tilde{X}$. This can be interpreted as treating the labels as new "movies" in the dataset, for which we want to make predictions as we do normally with the ratings of other movies. In the context of within-network classification, we can think of adding a dummy node for each tag, connected to every labelled node. The weights on these edges represent the corresponding labels for the nodes; Figure 7.2 provides an illustration.

7.2.2 Adding a tradeoff: reconstruction versus label accuracy

The above reduction of labels to dummy movies suggests a simple extension of existing dyadic prediction methods to predict labels for users. However, this reduction is not necessarily optimal with respect to the end-goal in our problem. Above, the joint optimization assumes that we want to reconstruct the augmented matrix, which means that mispredicting a label in Y is equally as bad as mispredicting a rating in X. However,

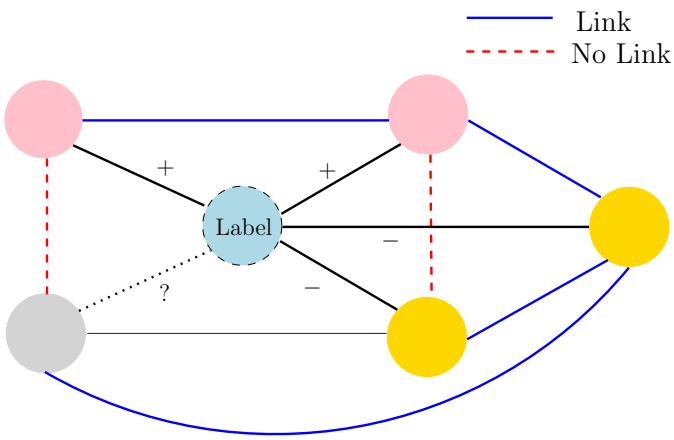


Figure 7.2. Adding a dummy node to represent the node labels.

it may often be the case that our ultimate goal is only the accuracy of predicting the labels. The data matrix in this case is just an incomplete training set for which we are trying to predict some labels. Accurately filling in the missing features in this set is only a means to the ultimate end of providing a classification for the user.

Therefore, in general, we need to have a user-controllable *tradeoff*, μ, between minimizing the reconstruction error and minimizing the label training error:

$$\min_{U,V,W} \sum_{(i,j)\in\mathscr{O}} (1-\mu)\cdot\ell(x_{ij},u_i^T v_j) + \sum_{(i,t)\in\mathscr{T}} \mu\cdot\ell(y_{it},(Wu_i)_t) + \lambda\Omega(U,V,W). \tag{7.3}$$

A similar tradeoff was suggested in the context of supervised latent semantic indexing (Yu et al., 2005). When $\mu = 1$, we attempt to learn U solely based on the label matrix. This approach is of limited value in the aforementioned common setting where the rows of Y are either completely specified or completely missing, as we would have to deal with a cold-start problem to make predictions. When $\mu = 0$, we solely attempt to model the interaction labels X, and completely ignore the label matrix. This ostensibly reduces to the unsupervised latent feature objective of Equation 7.1; note however that strictly, the above would end up learning U based on the regularizer, rather than use a

pre-determined value from an earlier stage. For intermediate values of μ, we attempt to leverage information from both the interaction labels and the individual dyad member labels.

At this stage, we take the opportunity to step back and revisit our implicit claim that it is desirable to learn supervised latent features from the data. This claim is intuitively reasonable, but there are some issues to be mindful of. First, compared to the unsupervised approach, one has to choose an additional tradeoff parameter μ by cross-validation. Second, introducing a dependence of U on the labels also introduces a risk of overfitting on the training labels. Clearly when $\mu = 1$ the solution is essentially powerless in the cold-start case, because it just tries to learn latent features that explain the labels of the known training examples. But even for nonzero μ, overfitting is possible, especially when the tags are sparsely populated, which is common for many multilabel problems. The similar issue is mitigated in standard collaborative filtering using ℓ_2 regularization of the weights, but in the label prediction problem there is an interplay between the tradeoff μ and the regularizer λ_W. These issues are explored in our experimental results.

7.2.3 Applying the LFL model to the label prediction task

In Equations 7.2 and 7.3, we deliberately presented the supervised latent feature approach in some generality. In principle, any suitable dyadic prediction method may be used to attack the dyadic label prediction task. We will focus on the use of the LFL method introduced in Chapter 3. The strengths of the method that we discussed in that chapter apply equally here; in brief, it allows us to exploit side-information for the dyad members, handle a range of different label types, and estimate probabilities for the labels. Each of these is a desirable property for a general dyadic label prediction model.

For the case of modelling the link structure of a graph, as discussed in Section 5.2.2, we can derive different models depending on the nature of the graph. Following

the literature on within-network classification, we will focus on the case of unweighted input graphs, so that

$$\Pr[x_{ij} = 1|\theta] = \sigma(u_i^T \Lambda u_j),$$

with $\Lambda^{(r)}$ being diagonal if the input graph is symmetric, and an arbitrary matrix otherwise. Further, we will focus on the case where the node labels constitute a sequence of L binary tags. For the model for the tth tag for node i, we use

$$\Pr[y_{it} = 1|\theta] = \sigma(u_i^T w_t),$$

which can be seen as a type of logistic regression for the tag r using the latent feature u_i as input. When optimizing log-likelihood, the final model is

$$\min_{U,\Lambda,W} \mu \sum_{(i,j)\in\mathscr{O}} -\log \sigma(\tilde{x}_{ij} \cdot u_i^T \Lambda u_j) + (1-\mu) \sum_{(i,t)\in\mathscr{T}} -\log(y_{it} \cdot u_i^T w_t) + \lambda \Omega(U,V,W),$$

where $\tilde{z} = 2z - 1$.

7.3 Comparing the latent feature methods

We now look more closely at three latent feature methods: the SocDim (Tang and Liu, 2009) and SMF (Zhu et al., 2007) methods in the within-network classification literature, and the LFL method described above. Table 7.1 summarizes the differences between the methods.

To begin, we can immediately identify three dimensions along which the methods differ. First, as discussed in Section 2, a key difference between SocDim and the SMF and LFL methods is that the former uses unsupervised latent features. Second, SocDim relies on the input graph being symmetric, as will be described shortly. Third, both SocDim and SMF assume that the adjacency matrix of the input graph has no missing

Table 7.1. Comparison of latent feature based methods for label prediction.

Item	LFL	SMF	SocDim
Supervised latent features?	Yes	Yes	No
Asymmetric graphs?	Yes	Yes	No
Missing edges?	Yes	No	No
Finds latent features of?	Data	Data	Modularity
Single minimum?	No	No	Yes

data, while LFL does not.

We now attempt to study the differences between the methods in settings where the above differences are not manifested. In particular, assume that $X \in \{0,1\}^{n \times n}$ is the adjacency matrix of an unweighted graph, with no missing entries. Further, let $X = X^T$ so that the graph is undirected. Finally, let us make SMF and LFL disregard the labels during the training phase, so that both operate in the *unsupervised* setting. We look at the objective functions for all three methods in turn:

- **SocDim**. The first step of the SocDim method is an eigendecomposition of the *modularity matrix* of X, defined as $\mathscr{Q}(X) = X - \frac{1}{2|E|} dd^T$, where d is a vector of the node degrees and $|E|$ is the number of edges in the graph. These eigenvectors are then used as a latent feature representation of the nodes in the graph. (The assumption that X is symmetric is needed for the eigenvalues of X to be real.) Since the eigenvectors of a symmetric matrix equal its singular vectors, up to a sign flip, we can reformulate the first step as

$$\min_{U,\Lambda} ||\mathscr{Q}(X) - U\Lambda U^T||_F^2.$$

  When Λ is constrained to be diagonal, the optimal solution U^* of this equation corresponds to the eigenvectors of $\mathscr{Q}(X)$ up to rotation.

- **SMF**. In SMF, the objective function is that of Equation 7.3, except that we assume

that there is no missing data. When we set $\mu = 1$, corresponding to no influence of the labels on the training process, the objective is

$$\min_{U,\Lambda} ||X - U\Lambda U^T||_F^2 + \frac{\lambda_U}{2}||U||_F^2.$$

- **LFL**. For the case of binary ratings, the LFL model reduces to

$$\Pr[y = 1|x = (i, j); U, V] = \sigma(u_i^T \Lambda u_j)$$

where $\sigma(\cdot)$ is the sigmoid function. For log-likelihood, the objective is quite different from the other methods. But optimizing for squared error yields

$$\min_{U,\Lambda} ||X - \sigma(U^T \Lambda U)||_F^2 + \frac{\lambda_U}{2}||U||_F^2.$$

We see that the three methods are all instantiations of the following general problem:

$$\min_{U,\Lambda} ||f(X) - g(U, \Lambda)||_F^2 + \frac{\lambda_U}{2}||U||_F^2.$$

For SocDim, $f(X) = \mathscr{Q}(X)$ and $g(U, \Lambda) = U\Lambda U^T$; for SMF, $f(X) = X$ and $g(U, \Lambda) = U\Lambda U^T$; and for LFL, $f(X) = X$ and $g(U, \Lambda) = \sigma(U\Lambda U^T)$. In this general scheme, we consider a transformed version of the data matrix, and then consider a low-rank approximation that is itself passed through a transformation.

SocDim and LFL differ in terms of which of the components, the data matrix or the low-rank approximation, they choose to transform. Both approaches can be seen to induce nonlinearity in the modelling of the input graph, but the transforms have different motivations. The aim of the sigmoidal transform is simply to make the entries lie in $[0, 1]$. The aim of the modularity transform is to normalize the degree distributions of the

nodes in the network, so that high degree nodes do not overshadow the rest of the graph. Interestingly, in collaborative filtering one can do something similar to the regularizer (Weimer et al., 2008):

$$\min_U ||X - \sigma(UU^T)||^2_{\mathscr{O}} + \frac{\lambda_U}{2}\text{tr}[U^T DU].$$

Here, $D = \text{diag}(1/\sqrt{d_1}, \ldots, 1/\sqrt{d_n})$, where $d_i = \sum_j X_{ij}$.

Another issue regarding the data transformation is the impact of the precise choice of $f(X)$. If we use $f(X) = X$, and thus operate on the raw adjacency matrix, do the results for SocDim significantly worsen? Further, is the use of the modularity $\mathscr{Q}$ essential, or can one use other popular transformations, such the normalized Laplacian from the spectral clustering literature (von Luxburg, 2007), $\mathscr{L} = I - D^{-1/2}XD^{-1/2}$, where D is a diagonal matrix of node degrees? It is also interesting to see what impact these transforms have on the SMF approach.

In the unsupervised setting under consideration, SocDim and SMF perform essentially the same optimization, except that SocDim works on the modularity matrix. Of course, one can consider a variant of SMF which operates on the modularity matrix; does that imply that the solutions of the two methods will be similar when $\mu = 1 - \varepsilon$ for ε small? This is not necessarily true, because of the nature of the optimization process. In SocDim, we optimize the objective function using an analytic solution, namely the eigenvectors of the modularity matrix. In SMF for $\mu < 1$, we have to resort to gradient descent to optimize the objective function, since there is no closed form solution.[2] But the objective is not jointly convex in U and Λ, so one can only find a local minimum. This means that even for μ close to 1, one may not exactly recover the eigenvectors of the data matrix. So, an advantage of SocDim is that it is immune to issues of local

[2]When we have labels for all the data, a closed form solution in terms of a generalized eigenvector is possible (Yu et al., 2005).

minima, albeit at the cost of being more expensive for large datasets, since it involves an eigendecomposition, whose runtime is superlinear in the size of the data matrix.

7.4 Experimental design

Having described a general approach to the dyadic label prediction problem, we now wish to test empirically how well it performs. This section explains the design of our experiments, covering first the issues to be explored, next the datasets to be used, and last the methods to be compared.

7.4.1 Aims of the experiments

The following questions are addressed in the experiments.

- **Does supervised learning of latent features help?**. As discussed in the previous section, while supervised latent features have an intuitive advantage over their unsupervised counterparts, it is important to compare the two empirically. We do so on some benchmark datasets for within-network classification, described below.

- **Does missing data harm existing methods?**. For general dyadic prediction problems, there are often many missing dyadic observations. Existing methods for predicting labels for network data, and in particular the ones based on learning latent features, do not deal with this issue. But can we just treat the missing observations as being edges with weight 0? Does this have a noticeable impact on the accuracy of label prediction? How does this simple approach compare with the LFL approach, which is designed to handle missing dyadic observations?

- **The value of data transformation**. The SocDim method involves learning latent features from the modularity matrix of the data. The idea of learning latent features from a transformation of the original data matrix has been studied in the spectral

clustering literature. An interesting question is whether performance is worse when the raw data matrix is used to learn latent features. If so, how do we choose between different transformations, such as the modularity and the Laplacian?

7.4.2 Datasets used

The experiments use the following datasets, which possess very different characteristics.

- Blogcatalog. This dataset comprises links between bloggers in the BlogCatalog directory (Tang, 2010). The input graph comprises 333983 friendship links between 10312 users. (These are assumed to exhaust all friendship relations, and so the graph is fully obseved.) The labels provided are the stated interests of users, which are divided into 39 possible categories. A user may have several interests, so this is a multilabel problem. Following (Tang and Liu, 2009), we report results over 10 random 10–90 train-test splits of the data, where in each split we randomly select 10% of the nodes and preserve their label vectors, while for the labels for the rest of the nodes are made missing.

- WebKB. This dataset comprises hyperlinks between webpages from 4 different universities (Cornell, Texas, Washington, Wisconsin). We used the processed version of the dataset from http://www.cs.umd.edu/~sen/lbc-proj/LBC.html. The input graph comprises 1608 hyperlinks between 877 webpages. Each webpage is labelled according to the nature of the webpage, which is one of $\{$ course, faculty, student, project, staff $\}$. A webpage has one and only one label, and so this is a multiclass classification problem. We report results over 10 random 10–90 splits of the data, as with the Blogcatalog dataset.

- Senator. This dataset comprises roll call data from the 109th session of the United States senate. It consists of the votes of 101 senators concerning 315 bills, with possible votes being "Yea" or "Nay." The dataset was studied previously in (Blei and McAuliffe, 2010). The data can be thought of as a binary ratings matrix for senators by bills. The goal is to predict whether or not a senator is a Republican or Democrat. This task is not particularly difficult, because the voting patterns of senators are highly predictive of their political affiliation. However, the task is representative of an important problem in political science, where one is interested in using behavioral records to test hypotheses about the nature of political dynamics.[3]

7.4.3 Methods compared and evaluation scheme

The experiments compare the three latent feature methods for within-network classification we have discussed thus far, namely SocDim, SMF, and the LFL model. We implement SMF and LFL ourselves, and use the code for SocDim provided by (Tang, 2010).

For all methods, the learned latent features are passed through a linear L2-SVM[4] to get final label predictions. While not necessary for the supervised latent feature methods, we did this for consistency in the results. We use LibLinear for the SVM implementation (Fan et al., 2008). For multilabel datasets, following (Tang and Liu, 2009), we make a prediction for each tag independently (i.e. we use the binary relevance multilabel learning method). For each method, for computational tractability, we used a *single* strength of regularization for each tag, which was picked to minimize the test error

[3] A few entries in the senator dataset are missing, which raises the point that even well-curated datasets often do have missing entries. In general it is important to have a principled way to handle these. For this dataset missing votes are treated as "Nay" for methods that cannot handle missingness directly. The impact of this choice is small for all methods.

[4] By this mean, we mean an SVM with squared hinge loss. We found this to give better results than the standard hinge loss.

over 10 random internal 80–20 splits of the training data.

Following (Tang and Liu, 2009), for multilabel data we assume that the number of labels are known, and we measure how well the predicted score for each tag agrees with the true label. Agreement is measured using the F1 micro and macro scores, which for true tags y_{il} and predictions $\hat{y}_{il}$ are defined as

$$\text{F1-Micro} = 2\frac{\sum_{l=1}^{L}\sum_i y_{il}\hat{y}_{il}}{\sum_{l=1}^{L}\sum_i (y_{il}+\hat{y}_{il})}$$

and

$$\text{F1-Macro} = \frac{2}{L}\sum_{l=1}^{L}\frac{\sum_i y_{il}\hat{y}_{il}}{\sum_i (y_{il}+\hat{y}_{il})}.$$

For the multiclass datasets, 0-1 error is the performance measure. In all experiments, we perform cross-validation to choose the regularization and μ parameters.

7.5 Experimental results

We now present results that aim to answer the questions we posed in the previous section.

7.5.1 Does supervised learning of latent features help?

We present results on each dataset introduced in the previous section in turn.

Results on Blogcatalog dataset

We report the results of several baselines from (Tang and Liu, 2009), such as a neighbourhood method (wvRN), a model based on network-derived features (LBC), and a clustering approach (LGC). Two final baselines are "Majority", which makes a constant prediction of the base rate for each tag, and a random guesser ("Random"). The results of these baselines are from (Tang and Liu, 2009), which only reports the mean performance

metrics across the random train-test splits. (However, as we will see, the variances across splits does not appear to be significant for other methods.) Finally, we report the results of the three latent feature methods we have discussed earlier, namely, SocDim, SMF, and LFL. Recall that there are no missing entries in the Blogcatalog dataset; consequently, the difference between the SMF and LFL methods is only in the choice of loss function, in this case, squared versus logistic loss. Note also that we applied SocDim to the raw data matrix, rather than the modularity, as suggested in (Tang and Liu, 2009); we will separately study the impact of the data transformation in Section 7.5.3.

Table 7.2 summarizes the results on the Blogcatalog dataset. We find that, consistent with the results reported in (Tang and Liu, 2009), the SocDim method performs well for a range of latent dimensionalities k, and in particular manages to clearly outperform the baseline methods with sufficiently large k. Further, we note that the SMF and LFL methods generally manage to outperform SocDim in terms of both the Micro- and Macro-F1 scores. As expected, there is generally no clear winner between the SMF and LFL methods, as they only differ in terms of the loss function used.

Results on WebKB dataset

On the WebKB dataset, we only report results for $k = 1$ and $k = 10$ latent features, because we did not find significant improvements for larger k. Table 7.3 summarizes the results. We see that for $k = 1$, supervision of latent features manages gives a reasonable boost in performance over the SocDim method. Interestingly, for $k = 10$, SocDim performs significantly better than the supervised approaches. We found this to be true even with $\mu = 0$, i.e. even when training SMF and LFL in a purely unsupervised manner. This may be a result of the non-convexity of the objective function. Further, as the dataset is quite a bit smaller than Blogcatalog, it may be that stochastic gradient training finds it difficult to effectively optimize the data (as on larger datasets, we expect there to be

Table 7.2. Results on Blogcatalog dataset. Entries marked with a * indicate results reported in the published literature, as opposed to the result of code we ran ourselves.

Method	**Micro-F1**	**Macro-F1**
Random*	0.0484	0.0414
Majority*	0.1651	0.0252
LGC*	0.1661	0.0857
LBC*	0.1352	0.0334
wvRN*	0.1951	0.0625
SocDim $k = 1$	0.1628 ± 0.0046	0.0266 ± 0.0012
SocDim $k = 10$	0.1732 ± 0.0038	0.0413 ± 0.0051
SocDim $k = 25$	0.2539 ± 0.0051	0.0903 ± 0.0041
SocDim $k = 50$	0.2804 ± 0.0052	0.1149 ± 0.0033
SocDim $k = 100$	0.2797 ± 0.0051	0.1290 ± 0.0036
SMF $k = 1$	0.1651 ± 0.0042	0.0276 ± 0.0018
SMF $k = 10$	0.2741 ± 0.0044	0.1036 ± 0.0051
SMF $k = 25$	0.3127 ± 0.0027	0.1396 ± 0.0046
SMF $k = 50$	0.3291 ± 0.0022	0.1556 ± 0.0041
SMF $k = 100$	0.3230 ± 0.0050	0.1543 ± 0.0057
LFL $k = 1$	0.1658 ± 0.0034	0.0275 ± 0.0010
LFL $k = 10$	0.2866 ± 0.0032	0.1163 ± 0.0033
LFL $k = 25$	0.2962 ± 0.0055	0.1362 ± 0.0036
LFL $k = 50$	0.2911 ± 0.0051	0.1126 ± 0.0057
LFL $k = 100$	0.3206 ± 0.0060	0.1497 ± 0.0047

greater redundancy in the individual gradients).

Results on Senator dataset

Results on the Senator dataset are presented in Table 7.4, where we study the 0-1 accuracy of the various methods, this time choosing $k = 1$ and $k = 5$ latent features. Here, we again find that there is an advantage to learning supervised latent features over SocDim, but it is not a significant one; this is likely due to the relative easiness of the prediction task, as evidenced by the high accuracy of the methods when $k = 1$. (This roughly says that the single dimension that best explains Senators' voting patterns is strongly tied to their party affiliation.) Interestingly, we found that SMF does slightly worse than SocDim, which is a type of overfitting, perhaps manifest due to the relatively

Table 7.3. Results on WebKB dataset.

Method	Micro-F1	Macro-F1
SocDim $k = 1$	0.5498 ± 0.0049	0.1774 ± 0.0010
SocDim $k = 10$	0.6031 ± 0.0310	0.3171 ± 0.0758
SMF $k = 1$	0.5654 ± 0.0177	0.2406 ± 0.0543
SMF $k = 10$	0.5374 ± 0.0203	0.2549 ± 0.0265
LFL $k = 1$	0.5477 ± 0.0211	0.1934 ± 0.0163
LFL $k = 10$	0.5506 ± 0.0329	0.2905 ± 0.0470

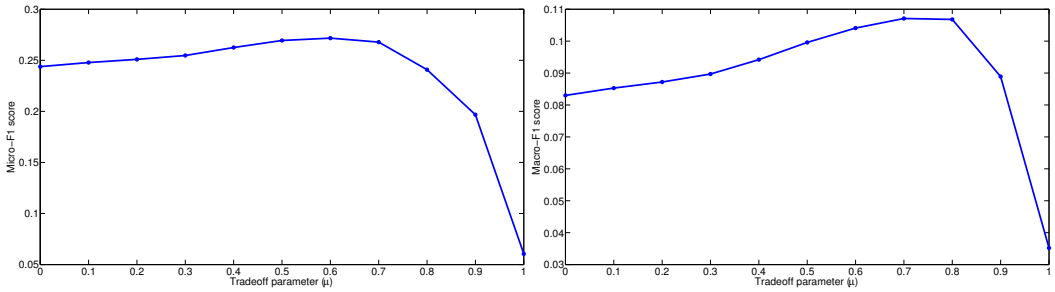
Table 7.4. Results on Senator dataset.

Method	0-1 Accuracy
SocDim $k = 1$	0.0284
SocDim $k = 5$	0.0284
SMF $k = 1$	0.0321
SMF $k = 5$	0.0494
LFL $k = 1$	0.0199
LFL $k = 5$	**0.0132**

small size of the dataset.

How does performance vary with the supervision tradeoff?

The above results indicate that supervision in learning the latent features can be useful in obtaining higher accuracy. In Figure 7.3, we study the impact of the tradeoff parameter μ on the performance of SMF model on the Blogcatalog dataset. We vary the value of μ, and study the test set performance in the Micro- and Macro-F1 scores. (This figure is to gain insight into the role of μ; we of course do not pick μ by tuning to the test set, but rather by cross-validation.) Recall that $\mu = 0$ corresponds to unsupervised model, while $\mu = 1$ is a model where the factorization is performed solely on the partially observed label matrix. We see that supervision does manage to improve performance in both micro- and macro-F1 scores, and that as expected when μ is close to 1 performance sharply degrades, as the model is incapable of learning any useful information.



(a) Micro-F1 score. **(b)** Macro-F1 score.

Figure 7.3. Impact of tradeoff parameter μ on supervised matrix factorization with $k = 10$, Blogcatalog dataset.

7.5.2 Does missing data harm existing methods?

We look to verify that the presence of missing data can potentially be harmful to both SocDim and SMF. We artificially censored a fixed random fraction p of the positive links in the Blogcatalog data, where p was varied from 0% to 90%. We then trained SocDim and SMF on this data, treating these censored entries as negative links. (That is, we treat the unknown status links as being known absent, which is a naïve mechanism of coping with missing data.) We also trained the LFL method on this data, which only models the known status links. For consistency, all methods operated on the raw data matrix (in particular, SocDim does not operate on the modularity), and were trained with $k = 50$. (For lower k, the performance of SocDim in the full data case is not significantly better than the majority classifier, and so the results are not as illustrative.) The experiment was repeated 10 times, and the results averaged. We show the results in Figure 7.4. As expected, as more positive links are censored, the performance of all methods worsens. The ranking of methods is also as expected: SMF does better than SocDim, indicating that supervision can partially compensate for the missing data, and LFL does better than SMF, indicating that it is important to have a principled manner of dealing with missing data.

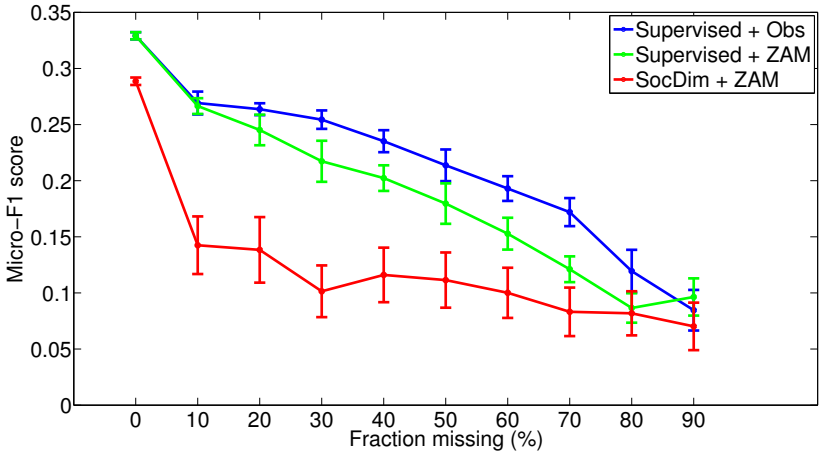


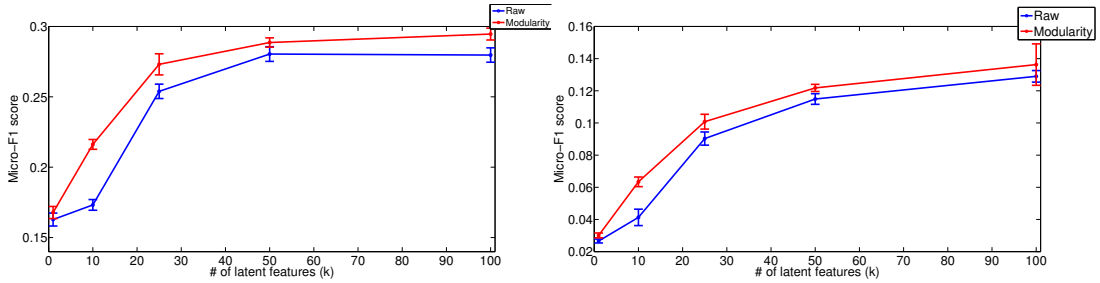
Figure 7.4. Impact of missing data on performance of methods, $k = 50$, Blogcatalog dataset. "ZAM" means missing entries are treated as zero, and "Obs" means only observed entries are modelled, and the missing ones ignored.

7.5.3 Does the choice of data transformation affect results?

As discussed earlier, the original formulation of SocDim in (Tang and Liu, 2009) proposed to compute the eigendecomposition on the modularity matrix, rather than the raw data matrix. It is of interest as to how much, if at all, this influences the performance of the method, and further whether we see similar gains when learning supervised latent features on the modularity matrix. Figure 7.5 compares the performance of SocDim on the raw data matrix and modularity matrix. We see that the modularity representation has a consistent, but somewhat slight, improvement over the raw data representation. (Even when based on the modularity, SocDim is outperformed by supervised learning of latent features.) The differences between the approaches are fairly consistent as the number of latent features, k, varies.

7.6 Conclusion

In this chapter, we shifted attention from dyadic prediction, the focus of the thesis thus far, to the problem of predicting labels associated with the *individual* dyad members.



(a) Micro-F1 score. **(b)** Macro-F1 score.

Figure 7.5. Impact of data transformation on performance of SocDim.

We showed a simple reduction of this problem to dyadic prediction problem, but argued for a slight modification of this reduction to improve performance. We discussed why we can think of this method as being a generalization of popular latent feature methods in the within-network classification literature. Experimental results show that our method performs favourably compared to these methods.

7.7 Acknowledgements

Chapter 7, in part, contains material as it appears in "Predicting labels for dyadic data", Data Mining and Knowledge Discovery, Volume 21, Number 2, pages 1384–5810, 2010. Aditya Krishna Menon and Charles Elkan. The dissertation author was the primary investigator and author of this paper.

We thank Lei Tang for gracious help with running the code for SocDim and for answering several queries regarding the same. We also thank David Blei for providing the Senator dataset.

Chapter 8

Conclusion

In this dissertation, we have studied how several seemingly disparate problems, such as collaborative filtering, link prediction, and clickthrough rate estimation, may be seen as instances of the generic dyadic prediction framework (Chapter 2). We then proposed a log-linear model, LFL, capable of addressing these problems (Chapter 3). The LFL model learns latent features from dyadic data, and its salient features include its ability to estimate a probability distribution over labels, and its amenability to highly scalable stochastic gradient training. We demonstrated the predictive power of this model on some simple dyadic problems, before proceeding to a detailed analysis of its application to collaborative filtering (Chapter 4), link prediction (Chapter 5), and clickthrough rate prediction (Chapter 6). While each application required some domain-specific modification - for example, the use of hierarchical information for clickthrough rate estimation (Section 6.5) - it is a virtue of the approach that such extensions were easily amenable, without requiring a fundamentally different solution strategy. Finally, we also showed how the problem of labelling nodes in a graph may be addressed in the dyadic framework (Chapter 7).

We now discuss some avenues for future work in the area of dyadic prediction.

- Since the Netflix prize, there has been an impressive body of literature on the topic

of collaborative filtering and recommender systems, which is an important instance of dyadic prediction. Nonetheless, there are still several challenges that remain to be explored, and can potentially improve the quality of recommendation engines that power many industrial systems.

- First, recent work (Salakhutdinov and Srebro, 2010) has shown that a naïve use of matrix factorization techniques may yield sub-optimal solutions, and proposes a counter-intuitive regularization scheme wherein users with fewer ratings are penalized less than those with more ratings. This may be related to a subtle bias of standard models towards prolific users, and that our understanding of this problem is limited. Understanding of this issue would allow for interesting extensions to recommender systems, such as providing better recommendations for a subset of users (e.g. those with memberships).
- Second, in some settings, we can expect there to be several unlabelled dyads with features, e.g. in privacy breach detection, we have a log of all accesses, but only a few of them are labelled. There may be an opportunity to exploit this unlabelled data to improve predictions; a challenge will be to integrate this with latent features extracted for users and movies.
- Third, much of the research in the field has ignored qualitative issues, such as determining subsets of users and/or movies that are most crucial to the good performance of the system (the "tastemakers"). Identifying these sets can not only help in designing improved models, but also reveal business insights that may be practically beneficial. Finally, study of the problem of generating diverse recommendations still remains nascent, and largely in the realm of heuristics. Sharpening our theoretical understanding of this desideratum may lead to effective, principled solutions to this problem.

- Cross-network link prediction is the problem of using characteristics from one network and using them to predict for another. For example, based on data from one high school, predict something about another high school. The techniques of this dissertation are largely not applicable, because there is in some sense an extreme cold-start: *none* of the objects in the test set were observed during training. Of course, in principle methods based on bilinear regression will still be appropriate, but an interesting question is whether it is possible to infer meaningful information from the topological structure of the network, and use this as an input to such a method.

- We believe the use of hierarchical information in Chapter 6 is best thought of as a means of overcoming extreme label sparsity. The poor performance of latent feature models without this modification suggests that more study of this problem is essential, especially in other domains where a similar problem arises. A generic solution to the problem of overcoming sparsity based on side-information would be valuable.

- In Section 2.4, we saw that dyadic prediction is closely related to several other frameworks in machine learning, such as matrix completion and multilabel learning. A systematic unification of these frameworks would allow for sharing of advances in each. For example, we did not explore many ideas from multilabel learning in our models. Another interesting possibility is the reduction of some of these problems to the dyadic framework. In particular, it would be interesting to attack problems like semi-supervised learning using a similare reduction approach to the one used in Chapter 7.

Appendix A

PCA, SVD, and the Eigendecomposition

We discuss three classical matrix factorization techniques: the singular value and eigen decompositions, and principal component analysis. We point out the connections amongst these decompositions.

A.1 Singular value decomposition (SVD)

Given a rectangular matrix $X \in \mathbb{R}^{m\times n}$, it is a remarkable fact that we may always decompose it as

$$X = USV^T,$$

where $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$, and $S \in \mathbb{R}^{m\times n}$. This is known as the *singular value decomposition* (SVD). The matrices U,V are orthogonal, so that $UU^T = U^TU = I_m$ and $VV^T = V^TV = I_n$, and the matrix $S \succeq 0$ is "rectangular diagonal" matrix with non-negative entries, meaning it is of the form

$$S = \begin{bmatrix} S' & 0_{\min(m,n)\times(n-\min(m,n))} \\ 0_{m-\min(m,n),\min(m,n)} & 0_{m-\min(m,n),n-\min(m,n)} \end{bmatrix},$$

where $S' \succeq 0$ is a diagonal matrix of size $\min(m,n) \times \min(m,n)$. The columns of U are known as the *left singular vectors*, the columns of V the *right singular vectors*, and the entries along the diagonal of S the *singular values* of X. The number of strictly positive entries $r \leq \min(m,n)$ in S is exactly the rank of X.

One of the most common operations one performs using the SVD is to form the *k-truncated* or *rank k* SVD

$$\hat{X}_k = U_k S_k V_k^T,$$

where we take the first k columns of U, V and consider the $k \times k$ submatrix S_k. Due to the orthogonality of U, V, one can equivalently write $\hat{X}_k = U_k U_k^T X = X V_k V_k^T$. This matrix has rank at most k, and a classic theorem of Eckart-Young-Mirsky says that it is a good approximation to the matrix X in the following specific sense.

Theorem 2 ((Eckart and Young, 1936; Mirsky, 1960))**.** *For a given matrix $X \in \mathbb{R}^{m \times n}$, let $\hat{X}_k$ be its k-truncated SVD. Let $||\cdot||_M$ be a unitarily invariant matrix norm. Then, for all $k \leq rank(X)$,*

$$||X - \hat{X}_k||_M = \min_{rank(B)=k} ||X - B||_M. \tag{A.1}$$

Observe that it must be the case that $X = \hat{X}_{\text{rank}(X)}$, as expected. The above theorem applies to the Frobenius norm, $||\cdot||_F$. Note that it says that $\hat{X}_k$ is a feasible solution to the optimization

$$\min_B ||X - B||_F^2 : \text{rank}(B) = k,$$

which is non-convex due to the rank constraint. As the SVD of X can be computed using an eigendecomposition, which in turn may be computed upto arithmetic precision using a technique like the power method (Trefethen and Bau, 1997), this means that there is nonetheless a tractable procedure to minimize this objective.

The singular vectors are only unique upto sign-flips: in particular, multiplying

both U and V by -1 will also recover B. More generally, multiplying the kth column of U and V by -1 will also recover B. Further, if for some k, k', $S_{kk} = S_{k'k'}$, the corresponding k and k'th singular vectors are not unique.

A.2 Eigendecomposition of square matrices

Let $A \in \mathbb{R}^{m \times m}$ be any square matrix. Then, A may be expressed in the form

$$A = PDP^{-1},$$

where $P \in \mathbb{C}^{m \times m}$ and $D = \text{diag}(d) \in \mathbb{R}^{m \times m}$ for some $d \in \mathbb{C}^m$. This is known as the *eigendecomposition* of A; the columns of P are known as the *eigenvectors* of A, and the entries along the diagonal in D are the *eigenvalues*. In general, the entries along the diagonal of D will include complex numbers (and their conjugate pairs, since the matrix A is real valued).

In the case where A is also symmetric, then one can show that $P^T = P^{-1} \in \mathbb{R}^{m \times m}$, and that the entries along the diagonal of D are real numbers. So, the above decomposition simplifies to

$$A = PDP^T,$$

where $D = \text{diag}(d)$ for some $d \in \mathbb{R}^m$.

If we further have that A is positive semi-definite, then the above decomposition is still true, and we can further guarantee that the entries along the diagonal of D are non-negative.

The eigenvectors are only unique upto sign-flips: in particular, multiplying any column of P by -1 will also recover A. Further, if for some k, k', $D_{kk} = D_{k'k'}$, the corresponding k and k'th eigenvectors are not unique.

Table A.1 summarizes the decompositions.

Table A.1. Summary of singular value and eigen decompositions in different settings.

Matrix type	**Decomposition**	**Constraints**
$B \in \mathbb{R}^{m\times n}$ rectangular	$B = USV^T$, the SVD	$UU^T = U^TU = I$ $VV^T = V^TV = I$ $S = \text{diag}(s) \geq 0$
$A \in \mathbb{R}^{m\times m}$ square	$A = PDP^{-1}$, the eigendecomposition	$P \in \mathbb{C}^{m\times m}$ $D = \text{diag}(d) \in \mathbb{C}^m$
$A \in \mathbb{R}^{m\times m}$ square symmetric	$A = PDP^T$	$P \in \mathbb{R}^{m\times m}$ $D = \text{diag}(d) \in \mathbb{R}^m$
$A \in \mathbb{R}^{m\times m}$ square positive semidefinite	$A = PDP^T$	$P \in \mathbb{R}^{m\times m}$ $D = \text{diag}(d) \in \mathbb{R}^m_+$

A.3 Principal component analysis (PCA)

Suppose that we have n data points $\{y_i\}_{i=1}^n$ where each $y_i \in \mathbb{R}^D$ for $D \gg n$. While each data point is ostensibly very high-dimensional, we may believe that the data together live on some low-dimensional subspace of $\mathbb{R}^D$. In principal component analysis (PCA), we assume that there is some *rotation* matrix $U \in \mathbb{R}^{D\times K}$ such that $U^TU = I_K$, some set of *latent vectors* $\{v_i \in \mathbb{R}^K\}_{i=1}^n$, $K \ll D$, and some *translation* vector $\mu \in \mathbb{R}^D$ so that

$$y_i = Uv_i + \mu. \tag{A.2}$$

The intuition for this model is as follows. Each data point is believed to natively reside in some low dimensional space $\mathbb{R}^K$. However, the observed data is the result of artificially padding this data with zeros to make it lie in $\mathbb{R}^D$, and then rotating and translating it. The first two steps correspond to the multiplication by U, and the last step the addition of μ.

In matrix notation, if $Y \in \mathbb{R}^{D\times n}$ is the matrix of all the y_i's, then $Y = UV + M$, where $M = \mu\mathbf{1}^T$. It can be shown that the optimal solution to this problem is to set the

columns of (see e.g. (Bishop, 2006)) U to be the top K eigenvectors of the covariance matrix $(Y-M)(Y-M)^T$, and $V = U^T Y$.

A.4 Relationship between the transformations

We discuss the relationship between SVD, the eigendecomposition, and PCA.

A.4.1 SVD as an eigendecomposition

Let $B \in \mathbb{R}^{m\times n}$ be any rectangular matrix. Consider now the matrix

$$G = BB^T \in \mathbb{R}^{m\times m}.$$

By the singular value decomposition, and using the orthonormality of the singular vectors,

$$G = USV^T VS^T U^T = USS^T U^T.$$

Certainly G is a symmetric square matrix, so it also admits an eigedecomposition

$$G = PDP^T.$$

If the eigendecomposition of G is unique – that is, if G is full rank – then, we can say that $U = P$, and $D = SS^T$. Now note that $SS^T \in \mathbb{R}^{m\times m}$ is a diagonal matrix whose entries are the squares of the diagonal entries of S, suitably padded by zeros. That is, the left singular vectors of B equal the eigenvectors of BB^T, while the singular values of B equal the square roots of the eigenvalues of BB^T. (Note that BB^T is a positive semidefinite matrix, so its eigenvalues are non-negative.) If the eigendecomposition is not unique, we can nonetheless say that the *space* spanned by left singular vectors equals that spanned by the eigenvectors of BB^T.

By a similar argument, constructing

$$C = B^T B \in \mathbb{R}^{n \times n}$$

tells us that the right singular vectors of B equal the eigenvectors of $B^T B$, and the singular values of B equal the square roots of the eigenvalues of BB^T.

Note that by orthonormality of V, $BV = US$. If no element of S is zero, then $U = BVS^{-1}$. That is, if we can compute the right singular vector and the singular values, then the left singular vector may be computed by a single matrix multiplication.

A.4.2 SVD versus eigendecomposition of a square matrix

A natural question is what relationship there is between the eigen and singular value decompositions of a square matrix $A \in \mathbb{R}^{m \times m}$. The eigendecomposition is

$$A = PDP^{-1},$$

and the singular value decomposition is

$$A = USV^T.$$

Note that the entries of D may be complex valued, while those in S are guaranteed to be non-negative. Therefore, there is no direct relationship between the components in the two decompositions.

Suppose further that A is symmetric. Then the elements of D are real valued, and the eigendecomposition is

$$A = PDP^T.$$

Now observe that $AA^T = PDP^T PDP^T = PD^2P^T$ by orthonormality of P. But we know

that the singular values S of A are the square roots of the eigenvalues of AA^T. That is, we have that $S = |D|$, or in other words the singular values are the absolute values of the eigenvalues. The singular vectors are then the eigenvectors, with the sign of the corresponding eigenvalue appropriately absorbed into *either* the left or right singular vector. Note that as a result of this sign absorption, it is *not* true that $U = V$; they are only equal in absolute value.

Finally, if A is positive semidefinite, the above still holds, and tells us that the singular values equal to the eigenvalues, and that the singular vectors equal to the eigenvectors.

Table A.2 summarizes the various relationships between the two decompositions.

Table A.2. Relationships between singular value and eigen decompositions in different settings.

Matrix type	**Decompositions**	**Relationship**
$B \in \mathbb{R}^{m \times n}$ rectangular	$B = USV^T$	$U = P$
	$BB^T = PDP^T$	$V = Q$
	$B^T B = QEQ^T$	$S = D^{1/2} = E^{1/2}$ with appropriate zero padding
$A \in \mathbb{R}^{m \times m}$ square	$A = PDP^{-1}$	No general relationship
	$A = USV^T$	
$A \in \mathbb{R}^{m \times m}$ square symmetric	$A = PDP^T$	$S = \|D\|$
	$A = USV^T$	$U = P \cdot \text{sign}(D), V = P$ or vice-versa
$A \in \mathbb{R}^{m \times m}$ square positive semidefinite	$A = PDP^T$	$S = D$
	$A = USV^T$	$U = V = P$

A.4.3 SVD and PCA

The PCA solution is related to the SVD of the matrix Y. Recall that the eigenvectors of YY^T are the same as the left singular vectors of Y. Therefore, the PCA solution is

equivalently the SVD of the matrix $Y - M$. PCA is also intimately related to several classical statistical tools such as linear discriminant analysis, canonical correlation analysis, and so on (De la Torre, 2012).

A.5 Conclusion

Appendix A, in part, contains material as it appears in "Fast Algorithms for Approximating the Singular Value Decomposition", ACM Transactions on Knowledge Discovery from Data, Volume 5, Number 2, Article 13. 2011. Aditya Krishna Menon and Charles Elkan. The dissertation author was the primary investigator and author of this paper.

Appendix B

Approaches to Matrix Completion

We discuss two approaches to dealing with (and learning to predict) missing data in a martix: the probabilistic version of PCA, and trace norm regularization. We point out the relationship between the two approaches.

B.1 Probabilistic PCA

Let us first rewrite the basic PCA model of Equation A.2 as

$$y_{ij} = u_i^T v_j + \mu_j.$$

We will now assume that each of y_{ij}, u_i, v_j are generated by some underlying probability distribution. Specifically, suppose we add isotropic Gaussian noise to the model of Equation A.2, and assume Gaussian priors for the variables u_i, v_j. For simplicity, in what follows we will drop the bias term. We get the model

$$\begin{aligned} y_j|U, v_j, \sigma_\varepsilon &\sim \mathcal{N}(U^T v_j, \sigma_\varepsilon^2 I) \\ u_i &\sim \mathcal{N}(0, \sigma_U^2 I) \\ v_j &\sim \mathcal{N}(0, \sigma_V^2 I). \end{aligned}$$

This model is known as probabilistic PCA Tipping and Bishop (1999). It is closely related to factor analysis Lawley (1943), where the noise model is not isotropic Whittle (1952). It may be shown that as $\sigma_\varepsilon \to 0$, the maximum likelihood estimate (MLE) of U approaches the solution of classical PCA, as expected.

An advantage of the probabilistic framework is that it provides a mechanism for dealing with missing data. Suppose that only some entries $\mathscr{O}$ in Y are observed, and the rest are censored. Let y_j^{obs} denote the set of observed values for the jth input, and similarly let U^{obs} correspond to those rows of U. We can modify our above model to only involve the observed elements in the data:

$$\begin{aligned} y_j^{\text{obs}} | U, v_j, \sigma_\varepsilon &\sim \mathscr{N}((U^{\text{obs}})^T v_j, \sigma_\varepsilon^2 I) \\ u_i &\sim \mathscr{N}(0, \sigma_U^2 I) \\ v_j &\sim \mathscr{N}(0, \sigma_V^2 I). \end{aligned}$$

When we have missing data, one goal is to predict the values at unobserved cells. Mathematically, this can be cast as the problem of evaluating $\Pr[y_{i^* j^*} | Y^{\text{obs}}]$ where $(i^*, j^*) \notin \mathscr{O}$. We now detail several ways in which this distribution may be evaluated (or approximated), beginning with the Bayesian approach.

B.1.1 Fully Bayesian treatment

To compute the predictive distribution $\Pr[y_{i^* j^*} | Y^{\text{obs}}]$ in a Bayesian fashion, we must integrate out all model parameters and latent variables. In our setting, this involves integrating out U, V, and appealing to conditional independences amongst variables:

$$\Pr[y_{i^* j^*} | Y^{\text{obs}}] = \int_U \int_V \Pr[y_{i^* j^*} | U, V] \cdot \frac{\Pr[Y^{\text{obs}} | U, V]}{\Pr[Y^{\text{obs}}]} \cdot \Pr[U] \cdot \Pr[V].$$

Unfortunately, the resulting distribution does not have a closed form solution, so one has to resort to approximation schemes such as variational methods or Monte-Carlo sampling Bishop (1999). While such approaches have been successful, computationally cheaper alternatives are available if one is willing to rely on point estimates for either U, V (or both). We begin with the approximation where we only marginalize out one variable.

B.1.2 Partial marginalization

While a fully Bayesian treatment of the predictive distribution requires approximation schemes, it turns out that if we marginalize out just one of U or V, there is a closed form solution for the resulting distribution. In particular, it can be checked that (see e.g. Lawrence and Urtasun (2009))

$$y_i^{\text{obs}}|U;\sigma_V,\sigma_\varepsilon \sim \mathcal{N}(0,\sigma_V^{-1}UU^T+\sigma_\varepsilon^2 I)$$
$$u_i \sim \mathcal{N}(0,\sigma_U^2 I).$$

We could have equivalently marginalized out U in the above and written an equation in terms of V. Indeed, this is called dual probabilistic PCA Lawrence (2005). The choice of which variable to marginalize out depends on whether $N \ll D$ or $D \ll N$. Attempting to further integrate out U from the above leads us to the same situation as the fully Bayesian approach. Instead, we may look to just find a MAP estimate of U, by maximizing the log-posterior

$$\begin{aligned}\log \Pr[U|Y^{\text{obs}}] &= \log \Pr[Y^{\text{obs}}|U] + \log \Pr[U] + \text{constant} \\ &= -\sum_{i=1}^{m}(y_i^{\text{obs}})^T(\sigma_V^{-1}UU^T+\sigma_\varepsilon^2 I)^{-1}y_i^{\text{obs}} - \log\det(\sigma_V^{-1}UU^T+\sigma_\varepsilon^2 I) - \\ &\frac{1}{2\sigma_U^2}||U||_F^2 \end{aligned} \tag{B.1}$$

Further, we can use formulae for convolution of Gaussians to evaluate the predictive distribution given a point-estimate of U:

$$\mathbb{E}[y_{i_* j_*}|U] = u_{i_*}^T U^T (UU^T + \sigma_\varepsilon^2)^{-1} y_{j_*}^{\text{obs}}.$$

This is identical to the prediction under a Gaussian process for regression problems. Indeed, for a fixed U, the above model is equivalent to *Bayesian linear regression* Bishop (2006). The supervised learning analog is: each column y_j comprises the labels, each row u_i comprises the feature representations of the inputs, and each v_j comprises the weights. What we have done above is marginalize the weights out and obtained a distribution for $\Pr[\mathbf{y}|\mathbf{x}]$. The connection between probabilistic PCA and Gaussian processes has motivated nonlinear extensions of the former Lawrence (2005).

Another classical solution to the problem of inference in latent variable models is the EM algorithm Dempster et al. (1977). In our problem, we can treat V as being the latent variable, and U as being the model parameter Ilin and Raiko (2010). We would the alternately estimate $p(V|Y,U)$ and U. Given that the marginalization can be done in closed form, however, it is not clear that there is a compelling reason to use EM in this case.

B.1.3 MAP estimation

We can get point estimates of both U,V by finding their MAP values. It is not hard to check that

$$\begin{aligned}\Pr[U,V|Y^{\text{obs}}] &\propto \Pr[Y^{\text{obs}}|U,V] \cdot \Pr[U,V] \\ &= \left(\prod_{(i,j)\in\mathcal{O}} \Pr[y_{ij}|u_i,v_j] \right) \cdot \prod_{i=1}^{m} \Pr[u_i] \cdot \prod_{j=1}^{n} \Pr[v_j]. \qquad \text{(B.2)}\end{aligned}$$

Taking logs, the MAP estimates are seen to correspond to an ℓ_2 regularized alternating least-squares problem in U and V:

$$(U_{\text{MAP}}, V_{\text{MAP}}) = \underset{U,V}{\operatorname{argmin}} ||Y - U^T V||_{\mathcal{O}}^2 + \frac{\sigma_\varepsilon^2}{2\sigma_U^2}||U||_F^2 + \frac{\sigma_\varepsilon^2}{2\sigma_V^2}||V||_F^2.$$

Contrasting this to the standard SVD decomposition of Equation A.1, we see that the above may be seen as an instance of weighted SVD,

$$\min_{U,V} ||W \odot (Y - U^T V)||_F^2$$

where $\odot$ denotes Hadamard product. This type of weighted matrix problem and variants thereof have been studied in many guises in statistics Gabriel and Zamir (1979); Ruhe and Wedin (1980); Golub and Pereyra (1973); Wold (2004); Geladi (1988). This objective is non-convex, but we may hope that as with the unweighted case, there are closed form solutions. Unfortunately, this is not the case Srebro and Jaakkola (2003), although it may be shown that every local optimum is a global optimum. (This does not discount saddle points, however.)

In the above scheme, the missing data is ignored. It is possible to take into account missing data using a different variant of the EM algorithm to the one discussed earlier. Suppose we treat U, V as model parameters, and the missing data Y^{obs^c} as the latent variables. Following the standard procedure for EM, we note that

$$\log \Pr[Y^{\text{obs}}|U,V] = \log \int_{Y^{\text{obs}^c}} \Pr[Y|U,V] \geq \mathbb{E}_{Y^{\text{obs}^c} \sim p(Y^{\text{obs}^c}|U',V')} \log \Pr[Y|U,V].$$

We now alternate between estimating $p(Y^{\text{obs}^c}|U',V')$ and maximizing the above for U, V.

It can be checked that the maximization problem is exactly

$$\begin{aligned}\mathbb{E}_{Y^{\text{obs}^c}} \log \Pr[Y|U,V] &= \mathbb{E}_{Y^{\text{obs}^c}} \left[\sum_{(i,j)\in\mathcal{O}} \log \Pr[y_{ij}|U,V] + \sum_{(i,j)\notin\mathcal{O}} \log \Pr[y_{ij}|U,V] \right] \\ &= \sum_{(i,j)\in\mathcal{O}} (Y_{ij} - u_i^T v_j)^2 + \sum_{(i,j)\notin\mathcal{O}} ((u_i')^T v_j' - u_i^T v_j)^2.\end{aligned}$$

Therefore, training can be performed in the following iterative manner:

- Initialize random values for Y^{obs^c}
- Let $Y = Y^{\text{obs}} \oplus Y^{\text{obs}^c}$
- Compute the SVD of Y, call it USV^T
- Compute the K-truncated SVD $U_K S_K V_K^T$, and use these as the new guess for Y^{obs^c}
- Repeat steps (2)–(4) till convergence.

Variants of this simple imputation scheme have been studied Josse et al. (2011). There is also a majorization-minimization perspective on the scheme Kiers (1997). Imputation was popular in early collaborative filtering applications of latent feature models Gu et al. (2010); Kurucz et al. (2007). However, it has the disadvantage of requiring the storage of the entire $m \times n$ matrix, which is generally prohibitive. Further, it has been empirically observed that this repeated imputation leads to worse local optima than alternating least squares Kiers (1997). Indeed, conceptually, if our interest is in modelling $\log \Pr[Y^{\text{obs}}|U,V]$, there is no need to perform the imputation, because we already have a tractable expression for this likelihood in Equation B.2. (One reason this may not be true if we want to model the mechanism by which the data is missing; see Section B.4.)

B.2 Trace norm regularization

In matrix completion, we are given an input matrix $Y \in \mathbb{R}^{m \times n}$ where only the entries in $\mathcal{O}$ are observed. We would like to reconstruct the missing entries. This of course requires some assumptions, and a common one is that Y is low-rank. Thus, our goal is to solve

$$\min_X ||Y - X||_{\mathcal{O}}^2 : \text{rank}(X) \leq r.$$

This problem is NP-hard is general Vandenberghe and Boyd (1996). There are at least two ways to proceed. One way is to look at approximation algorithms. Greedy selection approaches with provable guarantees were provided in Lee and Bresler (2009); Shalev-Shwartz et al. (2011), among others. The other way, which is more popular, is using convex surrogates. Consider the trace-norm $||\cdot||_*$ of a matrix X, defined as

$$||X||_* = \text{tr}[(X^T X)^{1/2}].$$

This is exactly the sum of the singular values of X, and is a convex relaxation to the rank of X Fazel et al. (2001). To see this intuitively, note that $\text{rank}(X)$ is the number of nonzero singular values of X, or equivalently the ℓ_0 norm of the singular spectrum of X, while the trace norm of X is the ℓ_1 norm of this spectrum. Using the trace norm in place of the rank gives the convex objective

$$\min_X ||Y - X||_{\mathcal{O}}^2 + \mu ||X||_*. \tag{B.3}$$

This objective was first used as a heuristic surrogate for the rank in Fazel et al. (2001). Subsequently, a remarkable line of work Candès and Recht (2009); Recht et al. (2010) has shown that if Y is low-rank, then the above objective will exactly recover Y with

high probability over the sampling. However, there are two limitations to this style of result. First, it makes a possibly strong assumption about the singular values of Y, known as "incoherence". For "coherent" matrices, there is no guarantee on how good the reconstruction will be. Second, it requires that the sampling process be uniform, which is not realistic. In learning theory, recent work has looked to remove the incoherence assumption and instead provide bounds with error rate proportional to the inherent uncertainty Foygel and Srebro (2011). Roughly, this is like the agnostic setting in PAC learning. There has also been work on removing the uniform sampling assumption Shamir and Shalev-Shwartz (2011), though this operates in a slightly different transductive setting and does not fully reflect the real-world problem.

Existing approaches to trace norm minimization include singular value thresholding Cai et al. (2010), approximate SDP solving Jaggi and Sulovský (2010), and fixed point continuation Ma et al. (2011). Recently, Avron et al. (2012) showed how to apply stochastic subgradient descent on the objective.

B.3 Comparing the trace norm and PCA

It can be shown that Srebro et al. (2004)

$$||X||_* = \min_{U,V:X=U^TV} \frac{1}{2}(||U||_F^2 + ||V||_F^2).$$

This says that the objective of Equation B.3 is equivalent to

$$\min_X ||Y - U^TV||_{\mathcal{O}}^2 + \frac{\mu}{2}(||U||_F^2 + ||V||_F^2),$$

which is exactly the same as Equation B.2. (In fact, there is an analogue to the marginalized likelihood of Equation B.1, which uses a penalty of $\log\det[VV^T]$ instead of the trace

norm penalty Yu et al. (2009).)

Given this equivalence, we can ask what the advantages of the two approaches are. The trace norm form of the objective has a definite advantage in that it is convex. However, despite the non-convexity of the bilinear form, it is arguably more popular in the literature. While one reason for this is possibly cultural, there are at least a couple of advantages to the bilinear approach:

- The bilinear objective is amenable to faster training procedures, in particular SGD, which is known to generalize faster than batch optimizers Bottou and Bousquet (2007). The other training strategy of alternately optimizing for U and V is also appealing, as it relies on mature supervised learning solvers, and is embarassingly parallel. (While SGD is inherently sequential, recently work has attempted to parallelize it. In the context of collaborative filtering, see for example Recht and Ré (2011).)

- The bilinear objective is easier to modify than the trace norm one, in the sense of not dramatically affecting the training procedure. Practically important modifications include:

  - Optimizing loss functions other than squard error. For example, in some collaborative filtering applications, we saw that optimizing absolute error was desirable (Section 4.8.2), and in link prediction we may wish to optimize a ranking loss (Section 5.3). Again, this is simple to do with the bilinear objective, but it is unclear how to do this in terms of the trace norm.

  - Applying different strengths of regularization on the U and V matrices. Above, we saw that the trace norm is an unweighted average of the two matrices' Frobenius norms. However, it has been argued that weighting the regularization strengths by the number of observations per row and column

can improve performance in settings where the train and test distributions do not match Salakhutdinov and Srebro (2010). It is not clear whether optimization strategies for the standard trace norm apply here, and indeed Salakhutdinov and Srebro (2010) optimize the bilinear form for their experiments.

- Related to the above, we may wish to impose more sophisticated priors on U, V rather than zero-mean Gaussians. For example, Agarwal et al. (2009) imposes a Markov Random Field prior based on an external similarity matrix. It is not clear how, if at all, such a model may be expressed in terms of the trace norm.
- When the data comprises multiple labels or observations per dyad, or indeed involves more than two interacting entities (e.g. if there is a temporal component), the basic assumption that the input X is a matrix is violated, and so it may be difficult to apply the trace norm scheme.

We also note that in practice, it has been observed that local optima due to non-convexity of the bilinear objective appears to not be a major issue Rennie and Srebro (2005).

B.4 Comment on data censoring mechanism

Thus far, we have not paid attention to the scheme by which data is censored so as to cause missing entries. However, the details of this scheme can affect the viability of estimation procedures. In statistics, three types of censoring scheme are considered Little and Rubin (2002). (These have correspondences in the relationship betwen train and test distributions in classification Zadrozny (2004).) In the *missing completely at random* (MCAR) scheme, we assume that the censoring is independent of example and label. In the *missing at random* (MAR) scheme, we assume the censoring is independent

of label, but dependent on example. In the *not missing at random* (NMAR) scheme, we assume that censoring depends on both example and label. The statistical consistency of maximum likelihood estimation may be established under the MCAR and MAR regimes Little and Rubin (2002), but not under MCAR. Indeed, even intuitively, the NMAR setting is difficult to handle unless we incorporate the censoring mechanism as part of the training or estimation procedure. In practical applications of matrix completion, such as collaborative filtering, it is plausible that the censoring mechanism is indeed NMAR (see Section 2.3.1). Nearly all research in the area ignores this issue (although a prominent exception is Marlin (2004)), however, likely because even though theoretical consistency is not guaranteed, commonly employed methods appear to perform well.

Appendix C

Random Effects Models and ANOVA

We first review the setup of grouped data. We then explain the meaning of the term random effects, show how they apply to the classical ANOVA technique, and finally discuss the connection to dyadic prediction problems.

C.1 Setup: modelling grouped data

Suppose we wish to analyze the relationship between some variable y and data x, where x comprises a number of groups (sometimes called treatments or sub-populations), each of which has related (but distinct) properties. Concretely, suppose each datum x consists of a series of categorical variables, so that the possible values for each variable defines a group. (In statistics, each such categorical variable is sometimes referred to as a *factor*, and the possible values of a factor referred to as *levels*.) A typical analysis we may wish to perform with such data is determining whether some quantity of interest varies across groups. We give a simple example of such data from (Snedecor and Cochran, 1989, pp. 237–238). Suppose we want to analyze the relationship between the content y of some chemical and the type x of a tea leaf (e.g. Assam, Ceylon, *et cetera*). In particular, we may wish to know whether the chemical contents of tea vary significantly across different tea leaves. To do this, we decide on some M number of different leaf types, $x_1, x_2, \ldots, x_M$, and for each type collect N samples of leaves of that type. For each sample,

we measure the content of the chemical, giving us the set of observations $\{y_{ij}\}_{i=1}^{M}{}_{j=1}^{N}$, where i is the ith group and j the jth observation within that group. Letting $\bar{y}_{i:}$ denote the average y value for the tea leaf type i, we wish to know if we can say with confidence that e.g. $\bar{y}_{i:} \neq \bar{y}_{i':}$, for $i \neq i'$.

C.2 What do fixed and random effects mean?

There are three basic types of models for grouped data, namely, fixed, random, and mixed effects. The term "effect" refers to the relationship between the identity of a group and the variable of interest, which is generally modelled by a single real weight α_i for group i. Effects are characterized as either fixed and random, so that a model possessing solely fixed (resp. random) effects is called a fixed (resp. random) effects model, while a model possessing both fixed and random effects is called a mixed effects model. There are at several definitions of fixed and random effects that are prevelant; see (Gelman, 2005) for further commentary on this issue. Here, we focus on two definitions, which we call "classical" and "modern". The classical definition (McCulloch et al., 2008) of a fixed effect is one where we assume that the observed groups are an exhaustive representation of all groups one wishes to make statements about. If on the other hand our observed groups are a random sample from some population, then the effect for a group is random. For example, if we have samples of every type of tea leaf in the world, our groups are an exhaustive representation of the underlying population, and the effect is fixed. But if we have observations for only a few tea leaves, our findings may be different than if we had chosen a different set of leaves. Thus, we think of the effect of the tea leaf type as being random.

A modern definition of a random effects model, one which is based on the Bayesian framework, is that it is simply an instance of a hierarchical model (Hoff, 2009; Gelman and Hill, 2007) where parameters are estimated at multiple levels of the hierarchy.

This is akin to the empirical Bayes method, where priors are estimated from the data (Robbins, 1964). It is likely the the case that a hierarchical model is a natural fit for a problem where we believe there to be a random effect as per the previous definition.

Concretely, consider the following model for the tea leaves example:

$$y_{ij} = \alpha_i + \varepsilon_{ij} : \varepsilon_{ij} \sim \mathcal{N}(0, \sigma_\varepsilon^2). \tag{C.1}$$

This would be considered a fixed effect model by most definitions of the term[1], and in particular by the above two definitions. Note that α_i is a group-specific parameter, whose aim is to measure the average observation within group i. As there is no accounting for the uncertainty in the sampling of the groups – that is, the α's are considered fixed parameters to be estimated – this model is appropriate for the case where the groups are not assumed to be random samples from a population.

Consider now the model

$$y_{ij} = \alpha_i + \varepsilon_{ij} : \alpha_i \sim \mathcal{N}(\alpha, \sigma_\alpha^2), \varepsilon_{ij} \sim \mathcal{N}(0, \sigma_\varepsilon^2). \tag{C.2}$$

The only difference to Equation C.1 is the introduction of what appears to be a prior on the α_i's. But while a prior encodes firm knowledge into a problem, here the parameter α is unknown, and is in fact *learned* along with the other parameters. This can be thought of as an "empirical prior", or more simply as a way to enforce sharing amongst parameters (Hoff, 2009). Intuitively, this constraint means that if α_1 is large, it increases our belief that α_2 should also be large, as they are both samples from the sample population. The precise nature of the tradeoff between this sharing of information and the fixed effects estimates is controlled by σ_α^2.

[1] In the terminology of (Kreft and de Leeuw, 1998), this would be considered a random *coefficient* model.

The above generalizes to regression: a fixed effects linear regression model for grouped data, where the observation j in group i has covariates x_{ij}, may look like

$$y_{ij} = w_i^T x_{ij} + \varepsilon_{ij} : \varepsilon_{ij} \sim \mathcal{N}(0, \sigma_\varepsilon^2),$$

while the random effects analogue is

$$y_{ij} = w_i^T x_{ij} + \varepsilon_{ij} : w_i \sim \mathcal{N}(w_0, \sigma_w^2), \varepsilon_{ij} \sim \mathcal{N}(0, \sigma_\varepsilon^2).$$

Note that in each case, there is a separate set of weights w_i for each group i. In the random effects model, the weights are encouraged to be similar to each other, all else being equal.

C.3 Removing the "empirical prior"

Note that the model in Equation C.2 can also be written as

$$y_{ij} = \mu + \alpha_i + \varepsilon_{ij} : \alpha_i \sim \mathcal{N}(0, \sigma_\alpha^2), \varepsilon_{ij} \sim \mathcal{N}(0, \sigma_\varepsilon^2),$$

with appropriate redefinition of variables. Here, we no longer "learn the prior", and so the appropriateness of the term random effect comes from the fact that there is a random parameter associated with each group[2]. The parameter μ here is treated as a fixed constant that represents the overall mean, while α_i represents the per-group offset from the global mean. In the linear regression case, a similar rewriting gives

$$y_{ij} = (w_0 + w_i)^T x_{ij} + \varepsilon_{ij} : w_i \sim \mathcal{N}(0, \sigma_w^2), \varepsilon_{ij} \sim \mathcal{N}(0, \sigma_\varepsilon^2),$$

[2]We may additionally wish to assume α is a sample from some distribution, in which case we would need to resort to the machinery of a hierarchical model

which can be seen as a regularized linear regression for each group, with an additional global set of weights w_0.

C.4 Relationship to ANOVA models

Fixed and random effects arise in models for analysis of variance (ANOVA) of grouped data. Given observations $\{y_{ij}\}_{i=1}^{M}{}_{j=1}^{N}$, a fixed effects two-way ANOVA model is

$$y_{ij} = \mu + \alpha_i + \beta_j + \varepsilon_{ij} : \varepsilon_{ij} \sim \mathcal{N}(0, \sigma_\varepsilon^2).$$

A random effects two-way ANOVA is

$$y_{ij} = \mu + \alpha_i + \beta_j + \varepsilon_{ij} : \alpha_i \sim \mathcal{N}(0, \sigma_\alpha^2), \beta_j \sim \mathcal{N}(0, \sigma_\beta^2), \varepsilon_{ij} \sim \mathcal{N}(0, \sigma_\varepsilon^2),$$

where, as before, we chose to not explicitly represent the "prior" means for α_i and β_j. In both cases, one keeps a parameter α_i for each group and β_j for the observation within a group.

Appendix D

Stochastic Gradient Descent

In supervised learning problems, we have training samples $\{(x_i, y_i)\}_{i=1}^n$ where each (x_i, y_i) is an iid draw from some unknown distribution $\mathscr{D}$ over $\mathscr{X} \times \mathscr{Y}$. Our goal is to learn some mapping $f : \mathscr{X} \to \mathscr{Y}$ that achieves good generalization performance with respect to some loss function $\ell : \mathscr{Y} \to \mathscr{Y} \to \mathbb{R}_+$:

$$\mathscr{E}[f] = \mathbb{E}_{(x,y)\sim\mathscr{D}}\left[\ell(f(x), y)\right].$$

We may attempt to minimize the quantity $\mathscr{E}[f]$ by instead minimizing its empirical counterpart,

$$\hat{\mathscr{E}}[f] = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i).$$

When the mapping f is parameterized by some $\theta \in \Theta$, we may equivalently consider the optimization

$$\begin{aligned} &\min_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i;\theta), y_i) \\ :=&\min_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^{n} \ell_i(\theta). \end{aligned}$$

How does one go about solving this optimization problem in general? One

classical mathematical technique is *gradient descent* (Boyd and Vandenberghe, 2004, pp. 466 – 475). The basic fact this exploits is that the gradient $\nabla g(\theta)$ of a function $g(\theta)$ points in the direction of greatest increase. For our objective, this suggests the iterative update scheme

$$\theta_{t+1} = \theta_t - \eta_t \cdot \frac{1}{n} \sum_{i=1}^{n} \nabla \ell_i(\theta_t).$$

Here, η_t is the *learning rate*, which determines how far we move in the direction of the gradient.

Above, our gradient is the average of the gradient of the loss on each training example. A simple observation is that an unbiased estimator for this quantity is the gradient of the loss for a single randomly chosen training example. In particular, consider the update rule

$$\theta_{t+1} = \theta_t - \eta_t \cdot \nabla \ell_{i(t)}(\theta_t),$$

where $i(t)$ is drawn uniformly at random from $\{1, 2, \ldots, n\}$. This optimization strategy is called *stochastic gradient descent*. Clearly, this stochastic approximation to the true gradient introduces noise, and reduces the rate of convergence to the minimizer of the training objective. Nonetheless, there are at least two important reasons why stochastic gradient descent is useful for learning problems. First, unlike gradient descent, each individual update is independent of the number of training examples, which makes it suitable for large scale learning tasks. Second, stochastic gradient descent minimizes the generalization error quicker than gradient descent (Bottou and Bousquet, 2007), even though it minimizes the training error slower.

Appendix E

Gradients of the LFL Model

Consider the probability model

$$\Pr[y|x=(i,j)] = \frac{\exp(W_{ij}^{(y)})}{\sum_{y'}\exp(W_{ij}^{(y')})}.$$

Clearly,

$$\log\Pr[y|x=(i,j)] = W_{ij}^{(y)} - \log\sum_{y'}\exp(W_{ij}^{(y')}).$$

For any parameter θ,

$$\begin{aligned}\frac{\partial}{\partial\theta}\left(\log\Pr[y|x=(i,j)]\right) &= \frac{\partial}{\partial\theta}(W_{ij}^{(y)}) - \frac{\sum_{y'}\exp(W_{ij}^{(y')})\partial_\theta W_{ij}^{(y')}}{\sum_{y'}\exp(W_{ij}^{(y')})} \\ &= \frac{\partial}{\partial\theta}(W_{ij}^{(y)}) - \sum_{y'}\Pr[y'|x]\partial_\theta W_{ij}^{(y')} \\ &= \sum_{y'}(\mathbf{1}[y=y'] - \Pr[y'|x])\partial_\theta W_{ij}^{(y')}.\end{aligned}$$

Hence,

$$\frac{\partial}{\partial\theta}\left(\Pr[y|x=(i,j)]\right) = \Pr[y|x=(i,j)]\cdot\sum_{y'}(\mathbf{1}[y=y'] - \Pr[y'|x])\partial_\theta W_{ij}^{(y')}.$$

First, we consider the derivatives for various scoring functions, W.

- **Complete**. When $W_{ij}^{(y)} = (u_i^{(y)})^T v_j^{(y)}$, we have

$$\frac{\partial}{\partial u_i^{(Y)}} W_{ij}^{(y)} = \mathbf{1}[y = Y] \cdot v_j^{(y)}$$
$$\frac{\partial}{\partial v_j^{(Y)}} W_{ij}^{(y)} = \mathbf{1}[y = Y] \cdot u_i^{(y)}$$

  and so

$$\frac{\partial}{\partial u_i^{(Y)}} \log \Pr[y|x = (i,j)] = (\mathbf{1}[y = Y] - \Pr[Y|x]) \cdot v_j^{(Y)}$$
$$\frac{\partial}{\partial v_j^{(Y)}} \log \Pr[y|x = (i,j)] = (\mathbf{1}[y = Y] - \Pr[Y|x]) \cdot u_i^{(Y)}.$$

- **Item-only**. When $W_{ij}^{(y)} = u_i^T v_j^{(y)}$, we have

$$\frac{\partial}{\partial u_i} W_{ij}^{(y)} = v_j^{(y)}$$
$$\frac{\partial}{\partial v_j^{(Y)}} W_{ij}^{(y)} = \mathbf{1}[y = Y] \cdot u_i$$

  and so

$$\frac{\partial}{\partial u_i} \log \Pr[y|x = (i,j)] = v_j^{(y)} - \sum_Y \Pr[Y|x] \cdot v_j^{(Y)}$$
$$\frac{\partial}{\partial v_j^{(Y)}} \log \Pr[y|x = (i,j)] = (\mathbf{1}[y = Y] - \Pr[Y|x]) \cdot u_i.$$

- **Diagonal scaler**. When $W_{ij}^{(y)} = u_i^T \Lambda^{(y)} v_j$, we have

$$\frac{\partial}{\partial u_i} W_{ij}^{(y)} = \Lambda^{(y)} v_j$$
$$\frac{\partial}{\partial v_j} W_{ij}^{(y)} = (\Lambda^{(y)})^T u_i$$
$$\frac{\partial}{\partial \Lambda^{(Y)}} W_{ij}^{(y)} = \mathbf{1}[y = Y] u_i v_j^T$$

and so

$$\frac{\partial}{\partial u_i} \log \Pr[y|x = (i,j)] = (\Lambda^{(y)} - \sum_Y \Pr[Y|x] \Lambda^{(Y)}) \cdot v_j$$
$$\frac{\partial}{\partial v_j} \log \Pr[y|x = (i,j)] = (\Lambda^{(y)} - \sum_Y \Pr[Y|x] \Lambda^{(Y)})^T \cdot u_i$$
$$\frac{\partial}{\partial \Lambda^{(Y)}} \log \Pr[y|x = (i,j)] = (\mathbf{1}[y = Y] - \Pr[Y|x]) \cdot u_i v_j^T$$

- **Ordered Diagonal scaler**. When $W_{ij}^{(y)} = u_i^T \tilde{\Lambda}^{(y)} v_j$, and $\tilde{\Lambda}^{(y)} = \sum_{y' \le y} e^{\Lambda^{(y')}}$ we have

$$\frac{\partial}{\partial \Lambda^{(Y)}} W_{ij}^{(y)} = \mathbf{1}[y \ge Y] e^{\Lambda^{(Y)}} \odot u_i v_j^T$$

and so

$$\frac{\partial}{\partial \Lambda^{(Y)}} \log \Pr[y|x = (i,j)] = \left(\mathbf{1}[y \ge Y] e^{\Lambda^{(Y)}} - \sum_{y' \ge Y} \Pr[y'|x] e^{\Lambda^{(Y')}} \right) \odot u_i v_j^T$$

We now consider gradients for log-likelihood and squared loss objectives.

E.1 Gradients for log-likelihood

For the objective

$$F(\theta;(x,y)) = -\log \Pr[y|x;\theta],$$

the gradients are as computed above.

E.2 Gradients for squared loss

For the objective

$$F(\theta;(x,y)) = (\sum_{y'} y' \Pr[y'|x] - y)^2,$$

we get

$$\begin{aligned}\frac{\partial}{\partial \theta} F(\theta) &= 2(\sum_{y'} y' \Pr[y'|x] - y)(\sum_{y''} y'' \partial_\theta \Pr[y''|x]) \\ &= 2(s-y) \cdot \sum_{y'} y' \Pr[y'|x] \cdot \partial_\theta \log \Pr[y'|x],\end{aligned}$$

where $s := \mathbb{E}[y], \Delta_y := \partial_\theta W^{(y)}$.

Bibliography

Abernethy, J., Bach, F., Evgeniou, T., and Vert, J.-P. (2009). A new approach to collaborative filtering: Operator estimation with spectral regularization. *JMLR*, 10:803–826.

Adamic, L. A. and Adar, E. (2003). Friends and neighbors on the web. *Social Networks*, 25(3):211–230.

Agarwal, D. (2008a). *Statistical Challenges in Internet Advertising*, pages 1–17. John Wiley & Sons, Inc.

Agarwal, D. (2008b). *Statistical Methods in E-Commerce Research*, chapter 1. Wiley-Interscience.

Agarwal, D., Agrawal, R., Khanna, R., and Kota, N. (2010a). Estimating rates of rare events with multiple hierarchies through scalable log-linear models. In *KDD '10*, pages 213–222, New York, NY, USA. ACM.

Agarwal, D., Broder, A. Z., Chakrabarti, D., Diklic, D., Josifovski, V., and Sayyadian, M. (2007). Estimating rates of rare events at multiple resolutions. In *KDD '07*, pages 16–25, New York, NY, USA. ACM.

Agarwal, D. and Chen, B.-C. (2009). Regression-based latent factor models. In *KDD '09*, pages 19–28, New York, NY, USA. ACM.

Agarwal, D., Chen, B.-C., and Elango, P. (2009). Spatio-temporal models for estimating click-through rate. In *WWW '09*, pages 21–30, New York, NY, USA. ACM.

Agarwal, D., Chen, B.-C., and Elango, P. (2010b). Fast online learning through offline initialization for time-sensitive recommendation. In *KDD*, pages 703–712.

Agarwal, D. and Merugu, S. (2007). Predictive discrete latent factor models for large scale dyadic data. In *KDD '07*, pages 26–35, New York, NY, USA. ACM.

Agarwal, S. (2012). Surrogate regret bounds for bipartite ranking via strongly proper

losses. *CoRR*, abs/1207.0268.

Agresti, A. (2010). *Analysis of ordinal categorical data*. Wiley Series in Probability and Statistics, 2 edition.

Airoldi, E., Blei, D. M., Fienberg, S. E., and Xing, E. P. (2008). Mixed membership stochastic blockmodels. In *NIPS*, pages 33–40.

Aldous, D. (1985). Exchangeability and related topics. In *Ecole d'Ete de Probabilities de Saint-Flour XIII 1983*, pages 1–198. Springer.

Anderson, J. (1984). Regression and ordered categorical variables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 46:1–30.

Anil, R. (2012). "what do you know?" - latent feature approach for the kaggle's grockit challenge. UCSD Master's Project Report.

Avron, H., Kale, S., Kasiviswanathan, S., and Sindhwani, V. (2012). Efficient and practical stochastic subgradient descent for nuclear norm regularization. *CoRR*. arXiv/1206.6384.

Banerjee, S. and Ramanathan, K. (2008). Collaborative filtering on skewed datasets. In *WWW '08*, pages 1135–1136, New York, NY, USA. ACM.

Batagelj, V., Ferligoj, A., and Doreian, P. (1999). Generalized blockmodeling. *Informatica (Slovenia)*, 23(4).

Bates, D. M. (2010). *lme4: Mixed-effects modeling with R*. http://lme4.r-forge.r-project.org/book/.

Beck, N., King, G., and Zeng, L. (2000). Improving quantitative studies of international conflict: A conjecture. *American Political Science Review*, 94(1):21–36.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305.

Bertsekas, D. P. (1999). *Nonlinear Programming*. Athena Scientific, Belmont, MA.

Bishop, C. M. (1999). Bayesian pca. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 382–388, Cambridge, MA, USA. MIT Press.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Blei, D. M. and McAuliffe, J. D. (2010). Supervised topic models. Revised version at http://arxiv.org/PS_cache/arxiv/pdf/1003/1003.0783v1.pdf.

Bock, R. D. (1972). Estimating item parameters and latent ability when responses are scored in two or more nominal categories. *Psychometrika*, 37(1):29–51.

Bolt, D. M. and Johnson, T. R. (2009). Addressing score bias and differential item functioning due to individual differences in response style. *Applied Psychological Measurement*, 33(5):335–352.

Bottou, L. and Bousquet, O. (2007). The tradeoffs of large scale learning. In *NIPS*.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, New York, NY, USA.

Breiman, L. (2001). Statistical modeling: The two cultures. *Statistical Science*, 16(3):199–215.

Buja, A., Stuetzle, W., and Shen, Y. (2005). Loss functions for binary class probability estimation and classification: Structure and applications. Technical report, University of Pennsylvania.

Buntine, W. L. (2002). Variational extensions to em and multinomial pca. In *Proceedings of the 13th European Conference on Machine Learning*, ECML '02, pages 23–34, London, UK, UK. Springer-Verlag.

Cai, J.-F., Candès, E. J., and Shen, Z. (2010). A singular value thresholding algorithm for matrix completion. *SIAM J. on Optimization*, 20(4):1956–1982.

Callut, J., Françoisse, K., Saerens, M., and Dupont, P. (2008). Semi-supervised classification from discriminative random walks. In *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*, ECML PKDD '08, pages 162–177, Berlin, Heidelberg. Springer-Verlag.

Candès, E. J. and Recht, B. (2009). Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772.

Carroll, J. and Chang, J.-J. (1970). Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition. *Psychometrika*,

35:283–319.

Chawla, N. V., Japkowicz, N., and Kotcz, A. (2004). Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl.*, 6:1–6.

Chiluka, N., Andrade, N., and Pouwelse, J. (2011). A link prediction approach to recommendations in large-scale user-generated content systems. In *Proceedings of the 33rd European conference on Advances in information retrieval*, ECIR'11, pages 189–200, Berlin, Heidelberg. Springer-Verlag.

Clauset, A., Moore, C., and Newman, M. E. J. (2008). Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101.

Clinton, J., Jackman, S., and Rivers, D. (2004). The statistical analysis of roll call data. *American Political Science Review*, 98(02):355–370.

Collins, M., Dasgupta, S., and Schapire, R. E. (2001). A generalization of principal components analysis to the exponential family. In *NIPS*, pages 617–624.

Coull, B. A. and Agresti, A. (2003). Generalized log-linear models with random effects, with application to smoothing contingency tables. *Statistical Modelling*, 3:251–271.

Crammer, K. and Singer, Y. (2001). Pranking with ranking. In *NIPS*, pages 641–647.

Dagan, I., Pereira, F., and Lee, L. (1994). Similarity-based estimation of word cooccurrence probabilities. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, ACL '94, pages 272–278, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dawid, A. P. and Skene, A. M. (1979). Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):pp. 20–28.

De la Torre, F. (2012). A least-squares framework for component analysis. *IEEE Transactions Pattern Analysis and Machine Intelligence (PAMI)*, 34(6):1041–1055.

Defazio, A. and Caetano, T. (2012). A graphical model formulation of collaborative filtering neighbourhood methods with fast maximum entropy training. *CoRR*, abs/1206.4622.

Delannay, N. and Verleysen, M. (2008). Collaborative filtering with interlaced generalized linear models. *Neurocomput.*, 71(7-9):1300–1310.

Dembczynski, K., Cheng, W., and Hüllermeier, E. (2010). Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML*, pages 279–286.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society Ser. B*, 39.

Denham, W. W. (1971). Alyawarra dataset. http://www1.aiatsis.gov.au/exhibitions/AlyaWeb/public/about.html.

Doppa, J. R., Yu, J., Tadepalli, P., and Getoor, L. (2010). Learning algorithms for link prediction based on chance constraints. In *ECML/PKDD (1)*, pages 344–360.

Dunlavy, D. M., Kolda, T. G., and Acar, E. (2011). Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data*. In Press.

Eckart, C. and Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9.

Fazel, M., Hindi, H., and Boyd, S. (2001). A rank minimization heuristic with application to minimum order system approximation. In *American Control Conference, 2001. Proceedings of the 2001*, volume 6, pages 4734–4739.

Feuerverger, A., He, Y., and Khatri, S. (2012). Statistical significance of the netflix challenge. *Statistical Science*, pages 202–231.

Fienberg, S. and Wasserman, S. (1981). Categorical data analysis of single sociometric relations. *Sociological Methodology*, 12:156–192.

Finch, H. (2008). Estimation of item response theory parameters in the presence of missing data. *Journal of Educational Measurement*, 45(3):225–245.

Fisher, R. A. and Mackenzie, W. A. (1923). Studies in crop variation. ii. the manurial response of different potato varieties. *The Journal of Agricultural Science*, 13(03):311–320.

Foster, D. P. and Stine, R. A. (2004). Variable selection in data mining: Building a predictive model for bankruptcy. *Journal of the American Statistical Association*,

pages 303–313.

Foygel, R. and Srebro, N. (2011). Concentration-based guarantees for low-rank matrix reconstruction. *Journal of Machine Learning Research - Proceedings Track*, 19:315–340.

Funk, S. (2006). Netflix update: try this at home. http://sifter.org/~simon/journal/20061211.html.

Gabriel, K. R. (1998). Generalized bilinear regression. *Biometrika*, 85:689–700.

Gabriel, K. R. and Zamir, S. (1979). Lower rank approximation of matrices by least squares with any choice of weights. *Technometrics*, 21(4):pp. 489–498.

Gammerman, A., Vovk, V., and Vapnik, V. (1998). Learning by transduction. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, UAI'98, pages 148–155, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Gantner, Z., Drumond, L., Freudenthaler, C., Rendle, S., and Schmidt-Thieme, L. (2010a). Learning attribute-to-feature mappings for cold-start recommendations. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 176–185, Washington, DC, USA. IEEE Computer Society.

Gantner, Z., Drumond, L., Freudenthaler, C., Rendle, S., and Schmidt-Thieme, L. (2010b). Learning attribute-to-feature mappings for cold-start recommendations. In *ICDM '10*, pages 176–185. IEEE.

Geladi, P. (1988). Notes on the history and nature of partial least squares (pls) modelling. *Journal of Chemometrics*, 2(4):231–246.

Gelman, A. (2005). Analysis of variancewhy it is more important than ever. *Annals of Statistics*, 33(1):1–53.

Gelman, A. and Hill, J. (2007). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press.

Gemulla, R., Nijkamp, E., Haas, P. J., and Sismanis, Y. (2011). Large-scale matrix factorization with distributed stochastic gradient descent. In *KDD*, pages 69–77.

Ghosn, F., Palmer, G., and Bremer, S. (2004). The MID3 data set, 1993-2001: Procedures, coding rules, and description. *Conflict Management and Peace Science*, 21:133–154.

Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70.

Goldenberg, A., Zheng, A. X., Fienberg, S. E., and Airoldi, E. M. (2010). A survey of statistical network models. *Found. Trends Mach. Learn.*, 2(2):129–233.

Golub, G. H. and Pereyra, V. (1973). The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM Journal on Numerical Analysis*, 10(2):pp. 413–432.

Gorrell, G. (2006). Generalized hebbian algorithm for incremental singular value decomposition in natural language processing. In McCarthy, D. and Wintner, S., editors, *EACL*. The Association for Computer Linguistics.

Görür, D., Jäkel, F., and Rasmussen, C. E. (2006). A choice model with infinitely many latent features. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 361–368, New York, NY, USA. ACM.

Graepel, T., Candela, J. Q., Borchert, T., and Herbrich, R. (2010). Web-scale Bayesian click-through rate prediction for sponsored search advertising in Microsoft's Bing search engine. In *ICML '10*, pages 13–20.

Gu, Q., Zhou, J., and Ding, C. H. Q. (2010). Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs. In *SDM*, pages 199–210. SIAM.

Gunawardana, A. and Meek, C. (2009). A unified approach to building hybrid recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 117–124, New York, NY, USA. ACM.

Hambleton, R. K., Swaminathan, H., and Rogers, H. J. (1991). *Fundamentals of Item Response Theory*. Sage Press.

Handcock, M. S., Raftery, A. E., and Tantrum, J. M. (2007). Model-based clustering for social networks. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 170(2):301–354.

Hartzel, J., Agresti, A., and Caffo, B. (2001). Multinomial logit random effects models. *Statistical Modelling*, 1:81–102.

Hasan, M. A., Chaoji, V., Salem, S., and Zaki, M. (2006). Link prediction using supervised learning. In *SDM Workshop on Link Analysis, Counterterrorism and*

Security.

Heckman, J. J. (1979). Sample selection bias as a specification error. *Econometrica*, 47(1):pp. 153–161.

Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 230–237, New York, NY, USA. ACM.

Hoff, P. (2007). Modeling homophily and stochastic equivalence in symmetric relational data. In *NIPS*.

Hoff, P. D. (2003). Bilinear mixed effects models for dyadic data. *Journal of the American Statistical Association*, 32:100–286.

Hoff, P. D. (2005). Bilinear mixed-effects models for dyadic data. *Journal of the American Statistical Association*, 100:286–295.

Hoff, P. D. (2006). Multiplicative latent factor models for description and prediction of social networks. Working paper no. 54, Center for Statistics and the Social Sciences, University of Washington-Seattle.

Hoff, P. D. (2009). *A First Course in Bayesian Statistical Methods*. Springer Texts in Statistics. Springer New York, New York, NY.

Hoff, P. D., Raftery, A. E., and Handcock, M. S. (2001). Latent space approaches to social network analysis. *Journal Of The American Statistical Association*, 97:1090–1098.

Hofmann, T., Puzicha, J., and Jordan, M. I. (1999). Learning from dyadic data. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 466–472, Cambridge, MA, USA. MIT Press.

Hripcsak, G. and Heitjan, D. F. (2002). Measuring agreement in medical informatics reliability studies. *Journal of Biomedical Informatics*, 35(2):99 – 110.

Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *ICDM '08*, pages 263–272, Washington, DC, USA. IEEE Computer Society.

Ilin, A. and Raiko, T. (2010). Practical approaches to principal component analysis in the presence of missing values. *J. Mach. Learn. Res.*, 99:1957–2000.

Jaggi, M. and Sulovský, M. (2010). A simple algorithm for nuclear norm regularized problems. In *ICML*, pages 471–478.

Jamieson, S. (2004). Likert scales: how to (ab)use them. *Medical Education*, 38(12):1217–1218.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *KDD '02*, pages 133–142, New York, NY, USA. ACM.

Johnson, C. (1990). Matrix completion problems: A survey. In Johnson, C. R., editor, *Matrix Theory and Applications 40, Proc. Sympos. Appl. Math.*, pages 171–198. AMS, Providence, RI.

Joseph, L., Gyorkos, T. W., and Coupal, L. (1995). Bayesian estimation of disease prevalence and the parameters of diagnostic tests in the absence of a gold standard. *American Journal of Epidemiology*, 141(3):263–272.

Josse, J., Pagès, J., and Husson, F. (2011). Multiple imputation in principal component analysis. *Adv. Data Anal. Classif.*, 5(3):231–246.

J.S. Kong, K. T. . J. K. (2012). The love-hate square counting method for recommender systems. *JMLR*, 18:249–261.

Kaggle Inc. (2012). What do you know? contest. http://www.kaggle.com/c/WhatDoYouKnow.

Karatzoglou, A., Smola, A. J., and Weimer, M. (2010). Collaborative filtering on a budget. *Journal of Machine Learning Research - Proceedings Track*, 9:389–396.

Kashima, H., Kato, T., Yamanishi, Y., Sugiyama, M., and Tsuda, K. (2009). Link propagation: A fast semi-supervised learning algorithm for link prediction. In *SDM*, pages 1099–1110.

Kemp, C., Tenenbaum, J. B., Griffiths, T. L., Yamada, T., and Ueda, N. (2006). Learning systems of concepts with an infinite relational model. In *Proceedings of the 21st national conference on Artificial intelligence - Volume 1*, AAAI'06, pages 381–388. AAAI Press.

Kenny, D. A. and Voie, L. L. (1984). The social relations model. In Berkowitz, L., editor, *Advances in Experimental Social Psychology*, volume 18, pages 141 – 182. Academic Press.

Keshavan, R. H., Montanari, A., and Oh, S. (2009). Matrix completion from noisy entries. *CoRR*, abs/0906.2027.

Khanna, R., Zhang, L., Agarwal, D., and Chen, B.-C. (2012). Parallel matrix factorization for binary response. Technical report, Unpublished manuscript.

Kiers, H. (1997). Weighted least squares fitting using ordinary least squares algorithms. *Psychometrika*, 62(2):251–266.

Kingman, J. F. C. (1978). Uses of exchangeability. *The Annals of Probability*, 6(2):pp. 183–197.

Kolda, T., Lewis, R., and Torczon, V. (2003). Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45(3):385–482.

Koren, Y. (2008). Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *KDD '08*, pages 426–434, New York, NY, USA. ACM.

Koren, Y. (2010). Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Trans. Knowl. Discov. Data*, 4(1):1:1–1:24.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Koren, Y. and Bell, R. M. (2011). Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 145–186.

Koren, Y. and Sill, J. (2011). Ordrec: an ordinal model for predicting personalized item rating distributions. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 117–124, New York, NY, USA. ACM.

Kotlowski, W., Dembczynski, K., and Hüllermeier, E. (2011). Bipartite ranking through minimization of univariate loss. In *Proc. 28th International Conference on Machine Learning (ICML)*.

Kozma, L., Ilin, A., and Raiko, T. (2009). Binary principal component analysis in the netflix collaborative filtering task. In *Machine Learning for Signal Processing, 2009. MLSP 2009. IEEE International Workshop on*, pages 1–6.

Kreft, I. and de Leeuw, J. (1998). *Introducing Multilevel Modeling*. Sage, London.

Kurucz, M., Benczúr, A. A., and Torma, B. (2007). Methods for large scale svd with

missing values. In *KDDCup 2007*.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

Larochelle, H. and Bengio, Y. (2008). Classification using discriminative restricted Boltzmann machines. In *ICML '08*, pages 536–543.

Lawley, D. N. (1943). The application of the maximum likelihood method to factor analysis. *British Journal of Psychology. General Section*, 33(3):172–175.

Lawrence, N. (2005). Probabilistic non-linear principal component analysis with gaussian process latent variable models. *J. Mach. Learn. Res.*, 6:1783–1816.

Lawrence, N. D. and Urtasun, R. (2009). Non-linear matrix factorization with gaussian processes. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 601–608, New York, NY, USA. ACM.

Lawrence, R. D., Hong, S. J., and Cherrier, J. (2003). Passenger-based predictive modeling of airline no-show rates. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 397–406, New York, NY, USA. ACM.

Lee, K. and Bresler, Y. (2009). Admira: Atomic decomposition for minimum rank approximation. *CoRR*, abs/0905.0044.

Lee, M. S., Moore, A. W., Lee, C. M. S., and Moore, A. W. (1995). Learning automated product recommendations without observable features: An initial investigation. Technical Report 95-17.

Lemire, D. and Maclachlan, A. (2005). Slope one predictors for online rating-based collaborative filtering. In *Proceedings of SIAM Data Mining (SDM'05)*.

Li, L. and Lin, H.-T. (2007). Ordinal regression by extended binary classification. In Schölkopf, B., Platt, J. C., and Hofmann, T., editors, *Advances in Neural Information Processing Systems 19*, pages 865–872. MIT Press.

Li, W.-J. and Yeung, D.-Y. (2011). Social relations model for collaborative filtering. In Burgard, W. and Roth, D., editors, *AAAI*. AAAI Press.

Li, W.-J., Yeung, D.-Y., and Zhang, Z. (2011). Generalized latent factor models for social

network analysis. In Walsh, T., editor, *IJCAI*, pages 1705–1710. IJCAI/AAAI.

Liben-Nowell, D. and Kleinberg, J. (2003). The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, CIKM '03, pages 556–559, New York, NY, USA. ACM.

Lichtenwalter, R., Lussier, J. T., and Chawla, N. V. (2010). New perspectives and methods in link prediction. In *KDD*, pages 243–252.

Lifshits, Y. and Nowotka, D. (2007). Estimation of the click volume by large scale regression analysis. In *CSR*, pages 216–226.

Likert, R. (1932). A technique for the measurement of attitudes. *Archives of Psychology*, 22(140):1–55.

Little, R. J. A. and Rubin, D. B. (2002). *Statistical Analysis with Missing Data, Second Edition*. Wiley-Interscience, 2 edition.

Liu, C. and Wang, Y.-M. (2012). Truelabel + confusions: A spectrum of probabilistic models in analyzing multiple ratings. *CoRR*, abs/1206.4606.

Lord, F. (1974). Estimation of latent ability and item parameters when there are omitted responses. *Psychometrika*, 39:247–264. 10.1007/BF02291471.

Lorrain, F. and White, H. C. (1971). Structural equivalence of individuals in social networks. *The Journal of Mathematical Sociology*, 1(1):49–80.

Lu, L. and Zhou, T. (2010). Link prediction in complex networks: A survey. http://arxiv.org/abs/1010.0725.

Ma, S., Goldfarb, D., and Chen, L. (2011). Fixed point and bregman iterative methods for matrix rank minimization. *Math. Program.*, 128(1-2):321–353.

Mackey, L. W., Weiss, D., and Jordan, M. I. (2010). Mixed membership matrix factorization. In Fürnkranz, J. and Joachims, T., editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 711–718, Haifa, Israel. Omnipress.

Macskassy, S. A. and Provost, F. (2003). A simple relational classifier. In *Proceedings of the Second Workshop on Multi-Relational Data Mining (MRDM-2003) at KDD-2003*, pages 64–76.

Macskassy, S. A. and Provost, F. (2007). Classification in networked data: A toolkit and a univariate case study. *J. Mach. Learn. Res.*, 8:935–983.

Mao, J. (2010). Scientific challenges in contextual advertising. In *Proceedings of the 5th international conference on Rough set and knowledge technology*, RSKT'10, pages 2–2, Berlin, Heidelberg. Springer-Verlag.

Mao, Y. and Saul, L. K. (2004). Modeling distances in large-scale networks by matrix factorization. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, IMC '04, pages 278–287, New York, NY, USA. ACM.

Marlin, B. (2004). Collaborative filtering: A machine learning perspective. Masters thesis, University of Toronto.

Marlin, B. M., Zemel, R. S., Roweis, S. T., and Slaney, M. (2007). Collaborative filtering and the missing at random assumption. In *UAI*, pages 267–275.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, 47:149–174.

Maxwell, S. E. and Delaney, H. D. (2003). *Designing Experiments and Analyzing Data: A Model Comparison Perspective*. Routledge Academic, 2 edition.

McCullagh, P. (1980). Regression models for ordinal data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 42:109–142.

McCulloch, C. E., Searle, S. R., and Neuhaus, J. M. (2008). *Generalized, Linear, and Mixed Models*. Wiley, Hoboken, NJ, second edition.

McFadden, D. (2001). Economic choices. *American Economic Review*, 91:351–378.

Meeds, E., Ghahramani, Z., Neal, R., and Roweis, S. (2006). Modeling dyadic data with binary latent factors. In *NIPS*, pages 977–984.

Menon, A. K., Chitrapura, K.-P., Garg, S., Agarwal, D., and Kota, N. (2011). Response prediction using collaborative filtering with hierarchies and side-information. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 141–149, New York, NY, USA. ACM.

Menon, A. K. and Elkan, C. (2010a). Fast algorithms for approximating the singular value decomposition. *ACM Transactions on Knowledge Discovery from Data*. Special issue on Large-scale Data Mining: Theory and Application.

Menon, A. K. and Elkan, C. (2010b). A log-linear model with latent features for dyadic prediction. *IEEE International Conference on Data Mining (ICDM)*, 0:364–373.

Menon, A. K. and Elkan, C. (2010c). Predicting labels for dyadic data. *Data Min. Knowl. Discov.*, 21(2):327–343.

Menon, A. K. and Elkan, C. (2011). Link prediction via matrix factorization. In *ECML/PKDD (2)*, pages 437–452.

Miller, K., Griffiths, T., and Jordan, M. (2009). Nonparametric latent feature models for link prediction. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. K. I., and Culotta, A., editors, *Advances in Neural Information Processing Systems 22*, pages 1276–1284.

Mirsky, L. (1960). Symmetric gauge functions and unitarily invariant norms. *Quarterly Journal of Mathematics, Oxford, Series (2)*, 11:50–59.

Montgomery, D. (2000). *Design and Analysis of Experiments*. John Wiley & Sons, 5th edition.

Mørup, M., Schmidt, M. N., and Hansen, L. K. (2010). Infinite multiple membership relational modeling for complex networks. In *NIPS Workshop on Networks Across Disciplines in Theory and Application*.

Murata, T. and Moriyasu, S. (2007). Link prediction of social networks based on weighted proximity measures. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, WI '07, pages 85–88, Washington, DC, USA. IEEE Computer Society.

Netflix (2006). Netflix prize. http://www.netflixprize.com/.

Ng, A. Y. and Jordan, M. I. (2001). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *NIPS*, pages 841–848.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, New York, 2nd edition.

Nowicki, K. and Snijders, T. A. B. (2001). Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 96(455):1077–1087.

P. Welinder, P. P. (2010). Online crowdsourcing: rating annotators and obtaining cost-effective labels. In *CVPR*.

Paquet, U., Thomson, B., and Winther, O. (2012). A hierarchical model for ordinal matrix factorization. *Statistics and Computing*, 22:945–957. 10.1007/s11222-011-9264-x.

Park, J. and Newman, M. E. J. (2005). A network-based ranking system for us college football. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(10):P10014.

Patz, R. J. and Junker, B. W. (1999). Applications and extensions of mcmc in irt: Multiple item types, missing data, and rated responses. *Journal of Educational and Behavioral Statistics*, 24(4):pp. 342–366.

Pilászy, I. and Tikk, D. (2009). Recommending new movies: even a few ratings are more valuable than metadata. In *RecSys '09*, pages 93–100, New York, NY, USA. ACM.

Porteous, I., Asuncion, A. U., and Welling, M. (2010). Bayesian matrix factorization with side information and dirichlet process mixtures. In *AAAI*.

Porteous, I., Bart, E., and Welling, M. (2008). Multi-hdp: a non parametric bayesian model for tensor factorization. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 3*, AAAI'08, pages 1487–1490. AAAI Press.

Pryor, M. H. (1998). The Effects of Singular Value Decomposition on Collaborative Filtering. Technical Report PCS-TR98-338, Dartmouth College, Computer Science, Hanover, NH.

Rasch, G. (1961). On general laws and the meaning of measurement in psychology. In Neyman, J., editor, *Proceedings of the IV Berkeley Symposium on Mathematical Statistics and Probability*, pages 321–333, Berkeley, CA. University of California Press.

Raykar, V. C., Yu, S., Zhao, L. H., Jerebko, A., Florin, C., Valadez, G. H., Bogoni, L., and Moy, L. (2009). Supervised learning from multiple experts: whom to trust when everyone lies a bit. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 889–896, New York, NY, USA. ACM.

Raymond, R. and Kashima, H. (2010). Fast and scalable algorithms for semi-supervised link prediction on static and dynamic graphs. In *ECML/PKDD (3)*, pages 131–147.

Recht, B., Fazel, M., and Parrilo, P. A. (2010). Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Rev.*, 52(3):471–501.

Recht, B. and Ré, C. (2011). Parallel Stochastic Gradient Algorithms for Large-Scale Matrix Completion. http://pages.cs.wisc.edu/~{}brecht/papers/11.Rec.Re.IPGM.pdf.

Unpublished manuscript.

Reckase, M. (2009). *Multidimensional Item Response Theory*. Springer.

Rendle, S. (2010). Factorization machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 995–1000, Washington, DC, USA. IEEE Computer Society.

Rendle, S., Freudenthaler, C., Gantner, Z., and Lars, S.-T. (2009). BPR: Bayesian personalized ranking from implicit feedback. In *UAI '09*, pages 452–461, Arlington, Virginia, United States. AUAI Press.

Rennie, J. D. M. (2005). Loss functions for preference levels: Regression with discrete ordered labels. In *Proceedings of the IJCAI Multidisciplinary Workshop on Advances in Preference Handling*, pages 180–186.

Rennie, J. D. M. and Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. In *ICML*, pages 713–719.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, CSCW '94, pages 175–186, New York, NY, USA. ACM.

Richardson, M., Dominowska, E., and Ragno, R. (2007). Predicting clicks: Estimating the click-through rate for new ads. In *WWW '07*, pages 521–530.

Rifkin, R. and Klautau, A. (2004). In defense of one-vs-all classification. *J. Mach. Learn. Res.*, 5:101–141.

Robbins, H. (1964). The empirical bayes approach to statistical decision problems. *The Annals of Mathematical Statistics*, 35(1):pp. 1–20.

Roweis, S. (2002). NIPS dataset. http://www.cs.nyu.edu/~roweis/data.html.

Ruhe, A. and Wedin, P. A. (1980). Algorithms for separable nonlinear least squares problems. *SIAM Review*, 22(3):pp. 318–337.

Rzhetsky, A., Shatkay, H., and Wilbur, W. J. (2009). How to get the most out of your curation effort. *PLoS Computational Biology*, 5(5):e1000391.

Salakhutdinov, R. and Mnih, A. (2007). Probabilistic matrix factorization. In *NIPS '07*.

Salakhutdinov, R. and Mnih, A. (2008). Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *ICML '08*, pages 880–887, New York, NY, USA. ACM.

Salakhutdinov, R., Mnih, A., and Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. In *ICML '07*, pages 791–798, New York, NY, USA. ACM.

Salakhutdinov, R. and Srebro, N. (2010). Collaborative filtering in a non-uniform world: Learning with the weighted trace norm. In Lafferty, J. D., Williams, C. K. I., Shawe-Taylor, J., Zemel, R. S., and Culotta, A., editors, *NIPS*, pages 2056–2064. Curran Associates, Inc.

Samejima, F. (1970). Estimation of latent ability using a response pattern of graded scores. *Psychometrika*, 35(1):139–139.

Sarkar, P., Chen, L., and Dubrawski, A. (2008). Dynamic network model for predicting occurrences of salmonella at food facilities. In *Proceedings of the BioSecure International Workshop*, pages 56–63. Springer.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA. ACM.

Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. T. (2000). Application of dimensionality reduction in recommender system – a case study. In *ACM WebKDD Workshop*.

Schein, A. I., Popescul, A., Ungar, L. H., and Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. In *SIGIR '02*, pages 253–260, New York, NY, USA. ACM.

Schein, A. I., Saul, L. K., and Ungar, L. H. (2003). A generalized linear model for principal component analysis of binary data. In *In Proceedings of the 9 th International Workshop on Artificial Intelligence and Statistics*, page 546431.

Schervish, M. J. (1995). *Theory of Statistics*. Springer-Verlag.

Sculley, D. (2009). Large scale learning to rank. In *NIPS Workshop on Advances in Ranking*.

Sculley, D. (2010). Combined regression and ranking. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 979–988, New York, NY, USA. ACM.

Sen, P. and Getoor, L. (2006). Cost-sensitive learning with conditional markov networks. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 801–808, New York, NY, USA. ACM.

Shalev-Shwartz, S., Gonen, A., and Shamir, O. (2011). Large-scale convex minimization with a low-rank constraint. In *ICML'11*, pages 329–336.

Shamir, O. and Shalev-Shwartz, S. (2011). Collaborative filtering with the trace norm: Learning, bounding, and transducing. *Journal of Machine Learning Research - Proceedings Track*, 19:661–678.

Shan, H. and Banerjee, A. (2010). Residual bayesian co-clustering for matrix approximation. In *SDM*, pages 223–234.

Shashua, A. and Levin, A. (2002). Ranking with large margin principle: Two approaches. In *NIPS*, pages 937–944.

Silver, N. (2012). *The Signal and the Noise: The Art and Science of Prediction*. Penguin Books Limited.

Smith, A. F. M. (1984). Present position and potential developments: Some personal views: Bayesian statistics. *Journal of the Royal Statistical Society. Series A (General)*, 147(2):pp. 245–259.

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. pages 194–281.

Smyth, P., Fayyad, U. M., Burl, M. C., Perona, P., and Baldi, P. (1994). Inferring ground truth from subjective labelling of venus images. In *NIPS*, pages 1085–1092.

Snedecor, G. W. and Cochran, W. G. (1989). *Statistical methods*. Iowa State University Press.

Snijders, T. A. and Nowicki, K. (1997). Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of Classification*, 14:75–100. 10.1007/s003579900004.

Srebro, N. and Jaakkola, T. (2003). Weighted low-rank approximations. In *ICML*, pages 720–727.

Srebro, N., Rennie, J. D. M., and Jaakkola, T. (2004). Maximum-margin matrix factorization. In *NIPS*.

Stevens, S. S. (1946). On the Theory of Scales of Measurement. *Science*, 103(2684):677–680.

Suits, D. B. (1957). Use of dummy variables in regression equations. *Journal of the American Statistical Association*, 52(280):pp. 548–551.

Szilágyi, A., Grimm, V., Arakaki, A. K., and Skolnick, J. (2005). Prediction of physical protein-protein interactions. *Physical Biology*, 2(2):S1.

Tang, L. (2010). Social dimension approach to classification in large-scale networks. Website at http://www.public.asu.edu/~ltang9/social_dimension.html.

Tang, L. and Liu, H. (2009). Relational learning via latent social dimensions. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 817–826. ACM.

Teh, Y. W. and Roy, D. (2009). The Mondrian process. *Technology*, 21(1979):1–8.

Tipping, M. E. and Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 61:611–622.

Trefethen, L. N. and Bau, III, D. (1997). *Numerical linear algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

Truyen, T., Phung, D., and Venkatesh, S. (2009). Ordinal Boltzmann machines for collaborative filtering. In *UAI*, pages 548–556, Corvallis, Oregon. AUAI Press.

Tsoumakas, G. and Katakis, I. (2007). Multi-label classification: An overview. *IJDWM*, 3(3):1–13.

Tsuda, K. and Noble, W. S. (2004). Learning kernels from biological networks by maximizing entropy. *Bioinformatics*, 20:326–333.

Uebersax, J. S. and Grove, W. M. (1993). A Latent Trait Finite Mixture Model for the Analysis of Rating Agreement. *Biometrics*, 49(3).

USPS (2010). USPS dataset. Obtained from http://www-i6.informatik.rwth-aachen.de/~keysers/usps.html.

Vandenberghe, L. and Boyd, S. (1996). Semidefinite programming. *SIAM Review*, 38:49–95.

Vert, J.-P. and Jacob, L. (2008). Machine learning for in silico virtual screening and chemical genomics: New strategies. *Combinatorial Chemistry & High Throughput Screening*, 11(8):677–685.

von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416.

Wang, C., Satuluri, V., and Parthasarathy, S. (2007). Local probabilistic models for link prediction. In *ICDM*, pages 322–331.

Ward, M. D., Siverson, R. M., and Cao, X. (2007). Disputes, democracies, and dependencies: A reexamination of the Kantian peace. *American Journal of Political Science*, 51(3):583–601.

Wasserman, S. and Faust, K. (1994). *Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences)*. Structural analysis in the social sciences, 8. Cambridge University Press, 1 edition.

Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of "small-world" networks. *Nature*, 393(6684):440–442.

Weimer, M., Karatzoglou, A., and Smola, A. J. (2008). Improving maximum margin matrix factorization. In *ECML/PKDD (1)*, page 14.

Weinberger, K., Dasgupta, A., Langford, J., Smola, A., and Attenberg, J. (2009). Feature hashing for large scale multitask learning. In *ICML '09*, pages 1113–1120.

Welinder, P., Branson, S., Belongie, S., and Perona, P. (2010). The multidimensional wisdom of crowds. In *NIPS*, pages 2424–2432.

Whitehill, J., Ruvolo, P., Wu, T., Bergsma, J., and Movellan, J. R. (2009). Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*, pages 2035–2043.

Whittle, P. (1952). On principal components and least square methods of factor analysis. *Skand Aktuarietidskr*, 36:223–239.

Wind, D. K. and Mørup, M. (2012). Link prediction in weighted networks. In *Machine Learning for Signal Processing (MLSP), 2012 IEEE International Workshop on*.

Winters, T., Shelton, C., Payne, T., and Mei, G. (2005). Topic extraction from item-level grades. In *American Association for Artificial Intelligence 2005 Workshop on*

Educational Datamining.

Wold, H. (2004). *Partial Least Squares*, pages 581–591. John Wiley & Sons, Inc.

Yamanishi, Y., Vert, J.-P., and Kanehisa, M. (2005). Supervised enzyme network inference from the integration of genomic data and chemical information. In *ISMB (Supplement of Bioinformatics)*, pages 468–477.

Yang, S.-H., Long, B., Smola, A., Sadagopan, N., Zheng, Z., and Zha, H. (2011). Like like alike: joint friendship and interest propagation in social networks. In *WWW '11*, WWW '11, pages 537–546.

Yu, K., Yu, S., and Tresp, V. (2005). Multi-label informed latent semantic indexing. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 258–265. ACM.

Yu, K., Zhu, S., Lafferty, J., and Gong, Y. (2009). Fast nonparametric matrix factorization for large-scale collaborative filtering. In *SIGIR '09*, pages 211–218, New York, NY, USA. ACM.

Yu, S., Yu, K., Tresp, V., Kriegel, H.-P., and Wu, M. (2006). Supervised probabilistic principal component analysis. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 464–473. ACM.

Zadrozny, B. (2004). Learning and evaluating classifiers under sample selection bias. In *ICML*.

Zadrozny, B. and Elkan, C. (2001). Learning and making decisions when costs and probabilities are both unknown. In *KDD*, pages 204–213.

Zhang, L., Agarwal, D., and Chen, B.-C. (2011). Generalizing matrix factorization through flexible regression priors. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 13–20, New York, NY, USA. ACM.

Zhang, T. (2004a). Statistical analysis of some multi-category large margin classification methods. *J. Mach. Learn. Res.*, 5:1225–1251.

Zhang, T. (2004b). Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, 32.

Zheleva, E., Getoor, L., Golbeck, J., and Kuter, U. (2010). Using friendship ties and family circles for link prediction. In Giles, L., Smith, M., Yen, J., and Zhang, H.,

editors, *Advances in Social Network Mining and Analysis*, volume 5498 of *Lecture Notes in Computer Science*, pages 97–113. Springer Berlin Heidelberg.

Zhou, T., Shan, H., Banerjee, A., and Sapiro, G. (2012). Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In *SDM'12*, pages 403–414.

Zhou, Y., Wilkinson, D., Schreiber, R., and Pan, R. (2008). Large-scale parallel collaborative filtering for the netflix prize. In *Proceedings of the 4th international conference on Algorithmic Aspects in Information and Management*, AAIM '08, pages 337–348, Berlin, Heidelberg. Springer-Verlag.

Zhu, S., Yu, K., Chi, Y., and Gong, Y. (2007). Combining content and link for classification using matrix factorization. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 487–494. ACM.

Zhu, X. and Ghahramani, Z. (2002). Learning from labeled and unlabeled data with label propagation. Technical report, CMU.

Ziegler, C.-N., Lausen, G., and Konstan, J. A. (2008). On exploiting classification taxonomies in recommender systems. *AI Communications*, 21(2-3):97–125.

Ziegler, C.-N., Lausen, G., and Lars, S.-T. (2004). Taxonomy-driven computation of product recommendations. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, CIKM '04, pages 406–415, New York, NY, USA. ACM.