UNIVERSITY OF CALIFORNIA,
IRVINE

Prediction-Constrained Latent Variable Models

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

John Gabriel Hope

Dissertation Committee:
Erik Sudderth, Chair
Padhraic Smyth
Alexander Ihler

2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank my advisor Erik Sudderth who has been an invaluable and patient guide throughout my long and not always straightforward journey through graduate school. I would also like to thank my mentor Mike Hughes, who planted the seeds of this work, provided me with continued support and without whom this thesis would not have been possible. My other committee members, Padhraic Smyth and Alex Ihler both contributed insightful comments and suggestions for improving my work. I would also like to thank the other members of Erik's Learning Inference and Vision group, who have helped and supported my personal and academic growth immensely over the years. Finally, I would like to thank my family who gave me the resources to succeed and who have always been my strongest supporters.

This work was supported in part by NSF CAREER Award No. IIS-1349774 and the UCI Machine Learning and Physical Sciences graduate training program (NSF award 1633631).

VITA

John Gabriel Hope

EDUCATION

Doctor of Philosophy in Computer Science **2017-2023**
University of California, Irvine *Irvine, CA*

Master of Science in Computer Science **2015-2017**
Brown University *Providence, RI*

Bachelor of Science in Computer Science **2010-2014**
Washington University in St. Louis *St. Louis, MO*

RESEARCH EXPERIENCE

Graduate Research Assistant **2017–2023**
University of California, Irvine *Irvine, California*

TEACHING EXPERIENCE

Teaching Assistant **2017-2018, 2021-2023**
University of California, Irvine *Irvine, CA*

Teaching Assistant **2017**
Brown University *Providence, RI*

Teaching Assistant **2012-2014**
Washington University in St. Louis *St. Louis, MO*

REFEREED CONFERENCE PUBLICATIONS

A Decoder Suffices for Query-Adaptive Variational Inference **Aug 2023**
Uncertainty in Artificial Intelligence

Prediction-Constrained Markov Models for Medical Time Series with Missing Data and Few Labels **Dec 2022**
Learning from Time Series For Health (TS4H) Workshop at NeurIPS 2022

Learning Consistent Deep Generative Models from Sparsely Labeled Data **Jan 2022**
4th Symposium on Advances in Approximate Bayesian Inference

Prediction-Constrained Hidden Markov Models for Semi-Supervised Classification **July 2021**
Time Series Workshop at ICML 2021

Semi-Supervised Prediction-Constrained Topic Models **Apr 2018.**
Artificial Intelligence & Statistics

Prediction-Constrained Topic Models for Antidepressant Prediction **Dec 2017**
ML for Health Workshop at NIPS 2017

ABSTRACT OF THE DISSERTATION

Prediction-Constrained Latent Variable Models

By

John Gabriel Hope

Doctor of Philosophy in Computer Science

University of California, Irvine, 2023

Erik Sudderth, Chair

Latent variable models provide a robust framework for modeling complex data distributions while accounting for known or desired inductive biases. The compact, low-dimensional representations learned by mixture models, variational autoencoders and other latent variable models also provide useful and interpretable bases for downstream prediction tasks. Our work introduces a novel framework for training latent variable models using *prediction constraints*, which aims to balance two important objectives: high-quality generative modeling of complex data and accurate prediction of semantic labels. Our framework acknowledges the inherent asymmetry of our discriminative objective, which is to learn how to predict labels from data, rather than to predict data from labels. We show that addressing this issue allows us to effectively leverage latent variable models for both supervised and semi-supervised learning while retaining interpretability and generative performance. We further introduce additional *consistency constraints*, derived naturally from the generative model, that requires predictions on reconstructed data to match those on the original data. We show that enforcing consistency is crucial when labels are very sparse. We apply our framework to a variety of latent variable models including: mixture models, topic models, hidden Markov models and variational autoencoders. Our experiments show state-of-the-art semi-supervised learning performance on diverse tasks using each of these models.

Chapter 1

Introduction

Learning to predict an outcome is a central goal in many applied fields. For example, we would like to be able to answer questions like: "what objects are in this image?", "what is the sentiment of this movie review?", and "what are likely outcomes for this ICU patient?". In many cases, answering these questions can be difficult for humans, either because of the number of predictions that need to be made (classifying unlabeled images or reviews on the internet), or because of the complex nature of the domain (for example, a medical outcome that is not well predicted by existing medical theories).

In the era of big data, we can gain access to many instances of observations (e.g. images) with *labels* (i.e. the outcome that we want to predict, e.g. the objects that are present in the image). With *predictive models*, machine learning attempts to leverage this data to *learn* how to make accurate predictions of labels or outcomes, so that the task of making predictions can be delegated to machines, or machines can assist humans at making predictions.

In addition, the large amount of data available can also be used to learn how to generate new data (e.g. new images or text). This is the goal of *generative models*. Latent variable models are a class of generative models that provide a coherent, well-motivated framework

for analyzing the distribution and structure of data. These models assume underlying, but unobserved structures are present within the data. Powerful exact and approximate inference techniques such as MCMC and variational methods allow for tractable inference for latent variables, giving practitioners the ability to answer queries such as the most likely configuration of the latent structure and the values of unseen variables in both the latent and observable spaces. For example, mixture models allow users to visualize and interpret the likely groupings of observations.

Latent variable models further provide a natural approach for feature extraction, as inferred latent variable values can be used as inputs for downstream tasks such as prediction. Unfortunately, this approach can often be sub-optimal, as the goals of data modeling and prediction may not align well. In this thesis we take a deep dive into better integrating prediction into latent variable models. In particular our focus will be on a new technique for training latent variable models that we call *prediction-constrained training*. This approach allow for a better combination of the strengths of latent variable models and prediction models.

Prediction tasks such as classification and regression have also been very well-studied in the field of machine learning. Over the past decade the deep learning revolution has transformed the field of machine learning with unprecedented results on tasks such as image recognition, image and text generation, and natural language interaction. Deep learning models are highly effective, but often lack a number of desirable properties such as: interpretability; it is not easy to understand *how* their predictions or generations are made, structure; it can be difficult to encode domain knowledge directly into the model, and robustness; deep models typically require large volumes of data for effective training, providing poor results when data is sparse. By providing a controllable and interpretable model for the generation of data, latent variable models allow this shortcomings to be addressed. In this work we show that making predictions via inferred latent structures allows for more interpretable predictions and more robust predictions, particularly when output variables are sparsely observed.

1.1 Overview of Contributions

The remainder of this work introduces our prediction-constrained latent variable modeling framework in detail, and provides specific examples of applications to models for different types of data. We address modifications and design choices necessary in order to achieve optimal performance with each model. We also demonstrate the effectiveness of our approach through extensive experiments with each model. Finally we conclude with a discussion of future research directions for prediction-constrained modelling.

We summarize the contributions of each chapter below.

Chapter 2: Prediction-Constrained Training of Latent Variable Models

We begin this section by introducing supervised latent variable models as a framework for joint generative and discriminative modeling. We then discuss the core of our contribution, which is the prediction-constraint framework for training supervised latent variable models. We compare our approach with a variety of related approaches from the relevant literature and illustrate the advantages and disadvantages of prediction-constrained training. Finally we introduce a specific example of a prediction-constrained latent variable model, the prediction-constrained mixture model, in order to illustrate how the approach is applied in practice. We show how to effectively train the prediction constrained mixture and discuss results on toy data.

Chapter 3: Prediction-Constrained Training of Hidden Markov Models

In this chapter we introduce the first more complex example of prediction-constrained latent variable model training with real-world examples. Specifically we introduce the prediction constrained hidden Markov model for time-series data. We show how to adapt the prediction-

constrained training framework to this model by using a prediction approach based on the marginals of state assignments. We further show extensions of the prediction-constrained model using non-linear predictors and per-timestep predictors. We demonstrate the effectiveness of our approach on semi-supervised datasets from the human activity recognition and healthcare domains. These experiments further show the interpretability of the prediction-constrained model alongside its natural ability to handle missing data.

Chapter 4: Prediction-Constrained Training of Topic Models

In this section we show a second case study of prediction-constrained training for latent variable models; prediction-constrained topic models. We show how to perform prediction-constrained training for the latent Dirichlet allocation topic models of bag-of-words documents. Our approach makes predictions based on *maximum a posteriori* estimate of the predocument topic distribution, while training the generative model with the same approach. We show that this is an efficient and effective approach to training supervised topic models and demonstrate promising results on a range of realistic semi-supervised text datasets including movie and restaurant reviews and electronic health records.

Chapter 5: Prediction-Constrained Training of Deep Latent Variable Models

As a final case study for prediction-constrained latent variable models, we showcase applications to deep generative models such as variational autoencoders. We show how to integrate prediction-constrained training with amortized variational inference, as well as the pitfalls of this approach. We further introduce a novel addition in consistency constraints and show that this is critical for effective semi-supervised learning with prediction-constrained variational autoencoders. We also discuss a number of other extensions to the prediction-constrained variational autoencoder framework such as: aggregate consistency for more robust semi-

supervised training, prediction-constrained hierarchical VAEs with more expressiveness, and affine transform VAEs that allow explicit modelling of affine-transformed images. We couple these technical introductions with thorough experiments demonstrating the advantages of our approach.

Chapter 2

Prediction-Constrained Training of Latent Variable Models.

In this chapter, we introduce a new framework for training supervised variants of latent variable models with *prediction-constraints.* Our approach is applicable to a broad family of latent variable models. Later sections and chapters provide concrete learning algorithms for supervised mixture models [Everitt and Hand, 1981], hidden Markov models [Rabiner and Juang, 1986a], topic models [Blei, 2012] and factor analysis models. In later chapters we will also discuss extensions to deep latent variable models such as variational autoencoders [Kingma and Welling, 2014] and structured variational autoencoders [Johnson et al., 2016c]. We emphasize that this framework could also be applied more broadly to allow supervised training of other latent variable models like dynamic topic models [Blei and Lafferty, 2006], linear dynamical system models [Shumway and Stoffer, 1982, Ghahramani and Hinton, 1996], stochastic block models for relational data [Wang and Wong, 1987, Kemp et al., 2006], and many more.

2.1 Supervised Latent Variable Models

We consider the broad family of latent variable models illustrated in Fig. 2.1. We now assume an observed dataset of D paired observations $\{x_d, y_d\}_{d=1}^{D}$. We refer to x_d as data and y_d as labels or targets, with the understanding that in intended applications, we can easily access some new data x_d but often need to predict y_d from x_d. For example, the pairs x_d, y_d may be text documents and their accompanying class labels, images and accompanying scene categories, or patient medical histories and their accompanying diagnoses. In this work we specifically consider the case of classification, where y_d is discrete and categorical (non-ordinal), however we note that most of our approaches could be extended to the case of regression or structured prediction.

We assume that each of the data pairs $\{x_d, y_d\}$ is generated independently conditioned on its own unobserved latent variable h_d. For a simple mixture model, h_d is an integer indicating the associated data cluster. For more complex members of our family like topic models, h_d may be a set of several document-specific hidden variables. We now assume the following generic process for generating the 3 random variables:

1. Draw a latent value from the prior: $h_d \mid \xi^h \sim p(h_d \mid \xi^h)$,
2. Draw an observation conditioned on the latent value: $x_d \mid h_d, \xi^x \sim p(x_d \mid h_d, \xi^x)$
3. Draw a label conditioned on the latent value and observation: $y_d \mid h_d, x_d, \xi^y \sim p(y_d \mid h_d, x_d, \xi^y)$

Steps 1 and 2 are inherited from the latent variable model framework, while step 3 introduces the label variable y as a child of the observation and latent value. The form of each conditional as well as the global model parameters $\xi = \{\xi^h, \xi^x, \xi^y\}$ will depend on the specific

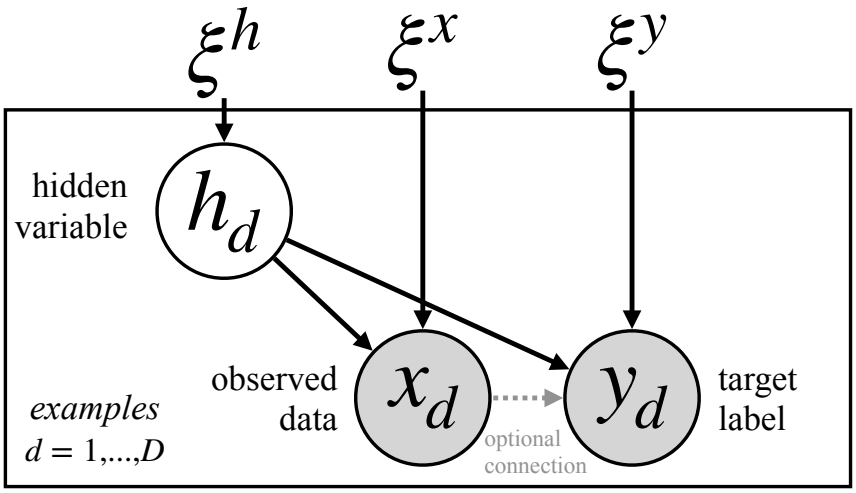


Figure 2.1: Graphical model representation of downstream supervised latent variable models.

model chosen. Under these assumptions, the joint density for datum d then factorizes as

$$p(x_d, y_d, h_d \mid \xi^h, \xi^x, \xi^y) = p(h_d \mid \xi^h) p(x_d \mid h_d, \xi^x) p(y_d \mid x_d, h_d, \xi^y). \tag{2.1}$$

We assume all of the above distributions have parameterized probability density or mass functions which can be easily evaluated and differentiated. The global parameters ξ^h, ξ^x, and ξ^y specify each distribution. When training our model, we may treat the full set of global parameters $\xi = \{\xi^h, \xi^x, \xi^y\}$ as random variables with associated prior density $p(\xi) = p(\xi^h, \xi^x, \xi^y)$.

Our chosen model family is an example of a *downstream* LVM: the core assumption of Eq. (2.1) is that the generative process produces both observed data x_d and targets y_d conditioned on the hidden variable h_d. In contrast, *upstream* or *label-conditioned* models such as Dirichlet-multinomial regression [Mimno and McCallum, 2008], DiscLDA [Lacoste-Julien et al., 2009], and labeled LDA [Ramage et al., 2009] assume that observed labels y_d are generated first, and then combined with hidden variables h_d to produce data x_d. For labeled-conditioned models, inference is challenging when labels are missing. For example, in downstream models $p(h_d \mid x_d)$ may be computed by omitting factors containing y_d, while

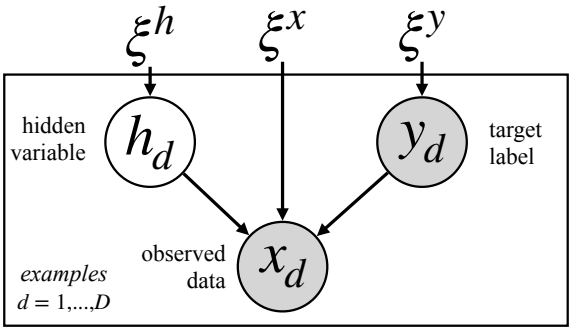


Figure 2.2: Graphical model representation of upstream or label-conditioned supervised latent variable models.

upstream models must explicitly marginalize over all possible y_d. Similarly, upstream prediction of labels y_d from data x_d is more complex than for downstream models. That said, our prediction-constrained LVM framework could also be used to produce novel learning algorithms for upstream LVMs.

Given this general model family, there are two core problems of interest. The first is global parameter learning: estimating values or approximate posteriors for ξ given training data $\{x_d, y_d\}$. The second is local prediction: estimating the target y_d given data x_d and model parameters ξ. While we focus primarily on these tasks, other related problems may also be of interest, such as predicting observations x_d from labels y_d, or imputing missing dimensions of x_d.

2.1.1 Regularized Maximum Likelihood Optimization for Training Global Parameters

A classical approach to estimating $\xi = \{\xi^h, \xi^x, \xi^y\}$ would be to maximize the marginal likelihood of the training data x and targets y, integrating over the hidden variables h. This

is equivalent to minimizing the following objective function:

$$\min_{\xi^h,\xi^x,\xi^y} \quad -\left[\sum_{d=1}^{D} \log p(x_d, y_d \mid \xi^h, \xi^x, \xi^y)\right] + R(\xi^h, \xi^x, \xi^y), \tag{2.2}$$

$$p(x_d, y_d \mid \xi^h, \xi^x, \xi^y) = \int p(h_d \mid \xi^h) p(x_d \mid h_d, \xi^x) p(y_d \mid x_d, h_d, \xi_y)\, dh_d.$$

Here, $R(\xi^h, \xi^x, \xi^y)$ denotes a (possibly uninformative) regularizer for the global parameters. If $R(\xi^h, \xi^x, \xi^y) = -\log p_0(\xi^h, \xi^x, \xi^y)$ for some prior density function $p_0(\xi^h, \xi^x, \xi^y)$, Eq. (2.2) is equivalent to *maximum a posteriori* (MAP) estimation of ξ^h, ξ^x, ξ^y. To simplify optimization we choose to use point estimates of ξ rather than marginalizing over these parameters.

One problem with standard ML or MAP training is that the inputs x_d and targets y_d are given equal weight. However, because practical models are typically misspecified and only approximate the generative process of real-world data, solving this objective can lead to solutions that are not matched to the practitioner's goals. We care much more about predicting patient mortality rates than we do about estimating past incidences of routine checkups. Especially because inputs x_d are usually higher-dimensional than targets y_d, conventionally trained LVMs may have poor predictive performance [Zhang and Kjellström, 2014].

2.2 Prediction-Constrained Optimization for Training Global Parameters

As an alternative to maximizing the joint likelihood, we consider a *prediction-constrained* objective, where we wish to find the best possible generative model for data x that meets some quality threshold for prediction of targets y given x. A natural quality threshold for our probabilistic model is to require that the sum of negative log conditional probabilities $p(y_d \mid x_d, \xi^h, \xi^x, \xi^y)$ must be bounded by some scalar value ϵ. This leads to the following

constrained optimization problem:

$$\begin{aligned} \min_{\xi^h, \xi^x, \xi^y} \quad & -\left[\sum_{d=1}^{D} \log p(x_d \mid \xi^h, \xi^x)\right] + R(\xi^h, \xi^x, \xi^y), \\ \text{subject to} \quad & -\sum_{d=1}^{D} \log p(y_d \mid x_d, \xi^h, \xi^x, \xi^y) \leq \epsilon. \end{aligned} \tag{2.3}$$

We emphasize that the conditional probability $p(y_d \mid x_d, \xi^h, \xi^x, \xi^y)$ *marginalizes* the hidden variable h_d:

$$p(y_d \mid x_d, \xi^h, \xi^x, \xi^y) = \int p(y_d \mid x_d, h_d, \xi_y) p(h_d \mid x_d, \xi^h, \xi^x)\, dh_d. \tag{2.4}$$

This marginalization allows us to make predictions for y_d that correctly account for our uncertainty in h_d given x_d, and importantly, given *only* x_d. At test time y_d may not be observed so we want to be able to accurately infer h_d and therefore y_d given x_d; however x_d is assumed to always be observed, so it is not necessary to do the inverse.

We note that we define our constraint in terms of the *total* prediction loss across the dataset. We make this choice such that our constraint framing agrees with our choices of performance metrics (average accuracy, etc.). A reasonable alternative choice would be to define the constraint *per-observation*.

A natural alternative form of the prediction-constraint framework would be to use a set of constraints specified *per-instance*:

$$\begin{aligned} \min_{\xi^h, \xi^x, \xi^y} \quad & -\left[\sum_{d=1}^{D} \log p(x_d \mid \xi^h, \xi^x)\right] + R(\xi^h, \xi^x, \xi^y), \\ \text{subject to} \quad & -\log p(y_d \mid x_d, \xi^h, \xi^x, \xi^y) \leq \epsilon_d \quad \forall\ d \in \{1...D\}. \end{aligned} \tag{2.5}$$

However this approach is challenging to integrate with stochastic optimization methods and is less aligned with the goals of practitioners; typically model quality is measured in terms of

average performance. Thus in practice we consider only the constrain on the total prediction loss.

The Karush-Kuhn-Tucker (KKT) [Kuhn and Tucker, 2013] conditions induce an equivalent unconstrained objective that maximizes the unsupervised likelihood but penalizes inaccurate predictions:

$$\min_{\xi^h,\xi^x,\xi^y} \quad -\left[\sum_{d=1}^{D} \log p(x_d \mid \xi^h, \xi^x)\right] - \lambda_\epsilon \left[\sum_{d=1}^{D} \log p(y_d \mid x_d, \xi^h, \xi^x, \xi^y)\right] + R(\xi^h, \xi^x, \xi^y), \tag{2.6}$$

Here, $\lambda_\epsilon > 0$ is a Lagrange multiplier chosen to ensure that the prediction constraint is achieved; smaller loss tolerances ϵ require larger λ_ϵ. For each distinct value of λ_ϵ, the solution to Eq. (2.6) also solves the constrained problem in Eq. (2.3) for a particular threshold ϵ. While the mapping between λ_ϵ and ϵ is monotonic, it is not constructive and lacks a simple parametric form.

We define the optimization problem in Eq. (2.6) to be our *prediction-constrained* (PC) training objective. This objective directly encodes the asymmetric relationship between data x_d and labels y_d by prioritizing prediction of y_d from x_d when $\lambda_\epsilon > 1$. This contrasts with the *joint maximum likelihood* objective in Eq. (2.2) which treats these variables symmetrically, and (especially when x_d is high-dimensional) may not accurately model the predictive distribution $p(y_d \mid x_d)$. In the special case where $\lambda_\epsilon = 1$, the PC objective of Eq. (2.6) reduces to the ML objective of Eq. (2.2).

Extension: Prediction constraints for individual data items

In Eq. (2.3), we defined our prediction quality constraint using the sum (or equivalently, the average) of the observation-specific losses $\log p(y_d \mid x_d, \xi^h, \xi^x, \xi^y)$. An alternative, more strin-

gent training objective would enforce separate prediction constraints for each observation:

$$\begin{aligned} \min_{\xi^h, \xi^x, \xi^y} \quad & -\left[\sum_{d=1}^{D} \log p(x_d \mid \xi^h, \xi^x)\right] + R(\xi^h, \xi^x, \xi^y), \\ \text{subject to} \quad & -\log p(y_d \mid x_d, \xi^h, \xi^x, \xi^y) \leq \epsilon_d \quad \text{for all } d. \end{aligned} \tag{2.7}$$

This modified optimization problem would generalize Eq. (2.6) by allocating a distinct Lagrange multiplier weight λ_{ϵ_d} for each observation d.

2.2.1 Semi-supervised prediction constraints for data with missing labels

In many applications, we have a dataset of D observations $\{x_d\}_{d=1}^{D}$ for which only a subset $\mathcal{D}^S \subset \{1, 2, \ldots D\}$ have observed labels y_d; the remaining labels are unobserved, we call the unlabeled portion of the dataset $D^U = D \setminus D^S$. For semi-supervised learning problems like this, we generalize Eq. (2.3) to only enforce the label prediction constraint for the observations in $\mathcal{D}^S$, so that the PC objective of Eq. (2.3) becomes:

$$\begin{aligned} \min_{\xi^h, \xi^x, \xi^y} \quad & -\left[\sum_{d=1}^{D} \log p(x_d \mid \xi^h, \xi^x)\right] + R(\xi^h, \xi^x, \xi^y), \\ \text{subject to} \quad & -\sum_{d:\mathcal{D}^S} \log p(y_d \mid x_d, \xi^h, \xi^x, \xi^y) \leq \epsilon. \end{aligned} \tag{2.8}$$

In general, the value of ϵ will need to be adapted based on the amount of labeled data. In the unconstrained form

$$\min_{\xi^h, \xi^x, \xi^y} \quad -\left[\sum_{d=1}^{D} \log p(x_d \mid \xi^h, \xi^x)\right] - \lambda_\epsilon \left[\sum_{d:\mathcal{D}^y} \log p(y_d \mid x_d, \xi^h, \xi^x, \xi^y)\right] + R(\xi^h, \xi^x, \xi^y), \tag{2.9}$$

as the fraction of labeled data $b = \frac{|\mathcal{D}^S|}{|D|}$ gets smaller, we will need a much larger Lagrange multiplier λ_ϵ to uphold the *same* average quality in predictive performance. This occurs simply because as b gets smaller, the data likelihood term $\log p(x_d)$ will continue to get larger in relative magnitude compared to the label prediction term $\log p(y_d \mid x_d)$.

2.2.2 Constraints on a general expected loss

Penalizing aggregate log predictive probability is sensible for many problems, but for some applications other loss functions are more appropriate. More generally, we can penalize the *expected loss* between the true labels y_d and predicted label distribution $\hat{y}(x_d, h_d, \xi^y)$ under the LVM posterior $p(h_d \mid x_d, \xi^x, \xi^h)$:

$$\begin{aligned} \min_{\xi} \quad & -\left[\sum_{d=1}^{D} \log p(x_d \mid \xi^x, \xi^h)\right] + R(\xi), \\ \text{subject to} \quad & \sum_{d=1}^{D} \mathbb{E}_{p(h_d \mid x_d, \xi^x, \xi^y)}\left[\ell(y_d, \hat{y}(h_d, \xi^y))\right] \leq \epsilon. \end{aligned} \tag{2.10}$$

Here $\ell(\cdot, \cdot)$ denotes the label loss. This more general approach allows us to incorporate classic non-probabilistic loss functions like the hinge loss or epsilon-insensitive loss, or to penalize errors asymmetrically in classification problems, when measuring the quality of predictions. We note that under this framework the maximum likelihood loss with $\hat{y}(z_d, \xi^y)$, specifying the parameters of a distribution on y such that:

$$\ell(y_d,\ \hat{y}(h_d,\ \xi^y)) \triangleq -\log p(y_d \mid \hat{y}(h_d, \xi^y)), \tag{2.11}$$

is an upper bound on the original prediction constraint:

$$
\begin{aligned}
-\log\left[p(y_d \mid x_d, \xi^h, \xi^x, \xi^y)\right] &= -\log \mathbb{E}_{p(h_d|x_d,\xi^x,\xi^y)}\left[p(y_d \mid \hat{y}(h_d, \xi^y))\right] \\
&\leq -\mathbb{E}_{p(h_d|x_d,\xi^x,\xi^y)}\left[\log p(y_d \mid \hat{y}(h_d, \xi^y))\right].
\end{aligned} \tag{2.12}
$$

In order to develop simple and tractable learning algorithms, we will use this form in most applications.

Example: balanced cross-entropy

For unbalanced binary tasks where $y \in \{0, 1\}$ and $\delta_+ \in (0, 1)$ denotes the fraction of positive examples, we find that a re-weighted a cross-entropy loss is useful in practice. We re-weight such that positive examples have weight $\frac{1-\delta_+}{\delta_+}$, while negative examples have weight 1; this equalizes the loss contribution of the two classes. Under the assumption that $\hat{y}(h_d, \xi^y) \triangleq p(y_d = 1|x_d, h_d, \xi^y)$ outputs the conditional probability of a positive label, this loss can be written as:

$$
\ell(y_d,\ \hat{y}(h_d,\ \xi^y)) \triangleq -y_d \left(\frac{1-\delta_+}{\delta_+}\right) \log \hat{y}(h_d,\ \xi^y) - (1 - y_d) \log \hat{y}(h_d,\ \xi^y) \tag{2.13}
$$

Balancing defines a smooth upper bound on the *balanced accuracy* [Brodersen et al., 2010], a common evaluation metric for image classification [Griffin et al., 2007]. While balancing does not arise from generative models of labels, it can naturally be incorporated into our PC framework, and can be directly generalized to unbalanced multi-class classification problems.

2.3 Relationship to Other Supervised Learning Frameworks

While the definition of the PC training objective in Eq. (2.6) is straightforward, it has desirable features that are not shared by other supervised training objectives for downstream LVMs. In this section we contrast the PC objective with alternative approaches from the latent variable model literature that incorporate labels.

2.3.1 Advantages over standard joint likelihood training

For our chosen family of supervised downstream LVMs, the most standard training method is to find a point estimate of global parameters ξ that maximizes the (regularized) joint log-likelihood $\log p(x_d, y_d \mid \xi)$ as in Eq. (2.2). Related Bayesian methods that approximate the posterior distribution $p(\xi \mid x_d, y_d)$, such as variational methods [Wainwright and Jordan, 2008] and Markov chain Monte Carlo methods [Andrieu et al., 2003], estimate moments of the same *joint* likelihood (see Eq. (2.1)) relating hidden variables h_d to data x_d and labels y_d.

For example, supervised LDA [McAuliffe and Blei, 2008, Wang et al., 2009] learns latent topic assignments h_d by optimizing the joint probability of bag-of-words document representations x_d and document labels y_d. One of several problems with this joint likelihood objective is *cardinality mismatch*: the relative sizes of the random variables x_d and y_d can reduce predictive performance. In particular, if y_d is a one-dimensional binary label but x_d is a high-dimensional word count vector, the optimal solution to Eq. (2.2) will often be indistinguishable from the solution to the *unsupervised* problem of modeling the data x alone. Low-dimensional labels can have negligible impact on the joint density compared to the high-dimensional words x_d, causing learning to ignore subtle features that are critical for

the prediction of y_d from x_d. Despite this issue, recent work continues to use this training objective [Wang and Zhu, 2014, Ren et al., 2017].

2.3.2 Advantages over maximum conditional likelihood training

Motivated by similar concerns about joint likelihood training, Jebara and Pentland [1999] introduce a method to explicitly optimize the conditional likelihood $\log p(y_d \mid x_d, \xi)$ for a particular LVM, the Gaussian mixture model. They replace the conditional likelihood with a more tractable lower bound, and then monotonically increase this bound via a coordinate ascent algorithm they call *conditional expectation maximization* (CEM). Chen et al. [2015] instead use a variant of backpropagation to optimize the conditional likelihood of a supervised topic model.

One concern about the conditional likelihood objective is that it *exclusively* focuses on the prediction task; it need not lead to good models of the data x, and cannot incorporate unlabeled data. In contrast, our prediction-constrained approach allows a principled trade-off between optimizing the marginal likelihood of data and the conditional likelihood of labels given data.

2.3.3 Advantages over label replication

We are not the first to notice that high-dimensional data x_d can swamp the influence of low-dimensional labels y_d. Among practitioners, one common workaround to this imbalance is to retain the symmetric maximum likelihood objective of Eq. (2.2), but to *replicate* each label y_d as if it were observed r times per document: $\{y_d, y_d, \ldots, y_d\}$. Applied to supervised LDA, label replication leads to an alternative *power sLDA* topic model [Zhang and Kjellström, 2014].

Label replication still leads to nearly the same per-document joint density as in Eq. (2.1), except that the likelihood density is raised to the r-th power: $p(y_d \mid x_h, h_d, \xi^y)^r$. While label replication can better "balance" the relative sizes of x_d and y_d when $r \gg 1$, performance gains over standard supervised LDA are often negligible [Zhang and Kjellström, 2014], because this approach does not address the *asymmetry issue*. To see why, we examine the label-replicated training objective:

$$\min_{\xi} \quad -\sum_{d=1}^{D} \log \left[\int p(h_d \mid \xi^h) p(x_d \mid h_d, \xi^x) p(y_d \mid x_d, h_d, \xi^y)^r \, dh_d \right] + R(\xi). \tag{2.14}$$

This objective does not contain any direct penalty on the predictive density $p(y_d \mid x_d)$, which is the fundamental idea of our prediction-constrained approach and a core term in the objective of Eq. (2.6). Instead, only the symmetric joint density $p(x, y)$ is maximized, with training assuming both data x and replicated labels y are present. It is easy to find examples where the optimal solution to this objective performs poorly on the target task of predicting y given only x, because the training has not directly prioritized this asymmetric prediction. In later sections such as the case study in Fig. 2.4, we provide intuition-building examples where maximum likelihood joint training with label replication fails to give good prediction performance for *any* value of the replication weight, while our PC approach can do better when λ_ϵ is sufficiently large.

Example: Label replication may lead to poor predictions.

Even when the number of replicated labels $r \to \infty$, the optimal solution to the label-replicated training objective of Eq. (2.14) may be sub-optimal for the prediction of y_d given x_d. To demonstrate this, we consider a toy example involving two-component Gaussian mixture models.

Consider a one-dimensional data set consisting of six evenly spaced points, $x = \{1, 2, 3, 4, 5, 6\}$.

The three points where $x \in \{2, 4, 5\}$ have positive labels $y = 1$, while the rest have negative labels $y = 0$. Suppose our goal is to fit a mixture model with two Gaussian components to these data, assuming minimal regularization (that is, sufficient only to prevent the probabilities of clusters and targets from being exactly 0 or 1). Let $h_d \in \{0, 1\}$ indicate the (hidden) mixture component for x_d.

If $r \gg 1$, the $p(y_d \mid x_d, h_d, \xi^y)^r$ term will dominate in Eq. (2.14). This term can be optimized by setting $h_d = y_d$, and the probability of $y_d = 1$ to close to 0 or 1 depending on the cluster. In particular, we choose $p(y_d = 1 \mid h_d = 0) = 0.0001$ and $p(y_d = 1 \mid h_d = 1) = 0.9999$. If one computes the maximum likelihood solution to the remaining parameters given these assignments of h_d, the resulting labels-from-data likelihood equals $\sum_{d=1}^{D} \log p(y_d \mid x_d) = -3.51$, and two points are misclassified. Misclassification occurs because the two clusters have significant overlap.

However, there exists an alternative two-component mixture model that yields better labels-given-data likelihood and makes fewer mistakes. We set the cluster centers to $\mu_0 = 2.0$ and $\mu_1 = 4.5$, and the cluster variances to $\sigma_0 = 5.0$ and $\sigma_1 = 0.25$. Under this model, we get a labels-given-data likelihood of $\sum_{d=1}^{D} \log p(y_d \mid x_d) = -2.66$, and only one point is misclassified. This solution achieves a lower misclassification rate by choosing one narrow Gaussian cluster to model the adjacent positive points $x \in \{4, 5\}$ correctly, while making no attempt to capture the positive point at $x = 2$. Therefore, the solution to Eq. (2.14) is sub-optimal for making predictions about y_d given x_d.

This counter-example also illustrates the intuition behind why the replicated objective fails: increasing the replicates of y_d forces h_d to take on a value that is predictive of y_d during training, that is, to get $p(y_d \mid h_d)$ as close to 1 as possible. However, there are no guarantees on $p(h_d \mid x_d)$ which is necessary for predicting y_d given x_d. See Fig. 2.4 for an additional in-depth example.

2.3.4 Advantages over posterior regularization

The *posterior regularization* (PR) framework introduced by Graça et al. [2008], and later refined in Ganchev et al. [2010], is notable early work which applied explicit performance constraints to latent variable model objective functions. Most of this work focused on models for only two local random variables: data x_d and hidden variables h_d, without any explicit labels y_d. Mindful of this, we can naturally express the PR objective in our notation, explaining data x explicitly via an objective function and incorporating labels y only later in the performance constraints.

The PR approach begins with the same overall goals of the expectation-maximization treatment of maximum likelihood inference: frame the problem as estimating an approximate posterior $q(h_d \mid \hat{v}_d)$ for each latent variable set h_d, such that this approximation is as close as possible in KL divergence to the real (perhaps intractable) posterior $p(h_d \mid x_d, y_d, \xi)$. Generally, we select the density q to be from a tractable parametric family with free parameters $\hat{v}_d$ restricted to some parameter space $\hat{v}_d \in \mathcal{V}$ which makes q a valid density. This leads to the objective

$$\min_{\xi, \{\hat{v}_d\}_{d=1}^D} \quad R(\xi) - \sum_{d=1}^{D} \mathcal{L}(x_d, \hat{v}_d, \xi), \tag{2.15}$$

$$\mathcal{L}(x_d, \hat{v}_d, \xi) \triangleq \mathbb{E}_{q(h_d|\hat{v}_d)}\Big[\log p(x_d, h_d \mid \xi) - \log q(h_d \mid \hat{v}_d)\Big] \leq \log p(x_d \mid \xi). \tag{2.16}$$

Here, the function $\mathcal{L}$ is a strict *lower bound* on the data likelihood $\log p(x_d \mid \xi)$ of Eq. (2.2). The popular EM algorithm optimizes this objective via coordinate descent steps that alternately update variational parameters $\hat{v}_d$ and model parameters ξ. The PR framework of Graça et al. [2008] adds additional constraints to the approximate posterior $q(h_d \mid \hat{v}_d)$ so that some additional loss function of interest, over both observed and latent variables, has

bounded value under the distribution $q(h_d \mid \hat{v}_d)$:

$$\text{Posterior Regularization (PR):} \quad \mathbb{E}_{q(h_d|\hat{v}_d)}\Big[\ell(y_d, \hat{y}(h_d, \xi^y))\Big] \leq \epsilon. \tag{2.17}$$

For our purposes, one possible loss function could be the negative log likelihood for the label y: $\ell(y_d, \hat{y}(h_d, \xi^y)) = -\log p(y_d \mid x_d, h_d, \xi^y)$. It is informative to directly compare the PR constraint above with the PC objective of Eq. (2.10). Our approach directly constrains the expected loss under the *true* hidden-variable-from-data posterior $p(h_d|x_d, \xi)$:

$$\text{Prediction Constrained (PC):} \quad \mathbb{E}_{p(h_d|x_d,\xi)}\Big[\ell(y_d, \hat{y}(h_d, \xi^y))\Big] \leq \epsilon. \tag{2.18}$$

In contrast, the PR approach in Eq. (2.17) constrains the expectation under the *approximate posterior* $q(h_d \mid \hat{v}_d)$. This posterior does not have to stay close to *true* hidden-variable-from-data posterior $p(h_d \mid x_d, \xi)$. Indeed, when we write the PR objective in unconstrained form with Lagrange multiplier λ_ϵ, and assume the loss is the negative label log-likelihood, we have:

$$\min_{\xi, \{\hat{v}_d\}_{d=1}^D} -\mathbb{E}_{q(h_d|\hat{v}_d)}\left[\sum_{d=1}^{D} \log p(x_d, h_d \mid \xi) + \lambda_\epsilon \log p(y_d \mid x_d, h_d, \xi^y) - \log q(h_d \mid \hat{v}_d)\right] + R(\xi) \tag{2.19}$$

Shown this way, we reach a surprising conclusion: the PR objective reduces to a lower bound on the symmetric joint likelihood with labels replicated λ_ϵ times. Thus, it will inherit all the problems of label replication discussed above, as the optimal training update for $q(h_d \mid \hat{v}_d)$ incorporates information from *both* data x_d and labels y_d. However, this does *not* train the model to find a good approximation of $p(h_d \mid x_d, \epsilon)$, which we will show is critical for good predictive performance.

2.3.5 Advantages over maximum entropy discrimination and regularized Bayes

Another key thread of related work putting constraints on approximate posteriors is known as *maximum entropy discrimination* (MED), first published in Jaakkola et al. [1999b] with further details in followup work [Jaakkola et al., 1999a, Jebara, 2001]. This approach was developed for training discriminative models without hidden variables, where the primary innovation was showing how to manage uncertainty about parameter estimation under max-margin-like objectives. In the context of LVMs, this MED work differs from standard EM optimization in two important and separable ways. First, it estimates a posterior for global parameters $q(\xi)$ instead of a simple point estimate. Second, it enforces a margin constraint on label prediction, rather than just maximizing log probability of labels. We note briefly that Jaakkola et al. [1999a] did consider a MED objective for *unsupervised* latent variable models (see their Eq. 48), where the constraint is directly on the expectation of the lower-bound of the log data likelihood. The choice to constrain the data likelihood is fundamentally different from constraining the labels-given-data loss, which was not done for LVMs by the original MED work yet is more aligned with our focus with high-quality predictions.

The key application MED to supervised LVMs has been Zhu et al. [2012]'s MED-LDA, an extension of the LDA topic model based on a MED-inspired training objective. Later work developed similar objectives for other LVMs under the broad name of *regularized Bayesian inference* [Zhu et al., 2014]. To understand these objectives, we focus on Zhu et al. [2012]'s original unconstrained training objectives for MED-LDA for both regression (Problem 2, Eq. 8 on p. 2246) and classification (Problem 3, Eq. 19 on p. 2252), which can be fit into

our notation[1] as follows:

$$\min_{q(\xi),\{\hat{v}_d\}_{d=1}^D} \quad \mathrm{KL}\big(q(\xi) \;||\; p_0(\xi)\big) \\ - \mathbb{E}_{q(\xi)}\Big[\sum_{d=1}^{D} \mathcal{L}(x_d, \hat{v}_d, \xi)\Big] + C\sum_{d=1}^{D} \mathrm{loss}\big(y_d, \mathbb{E}_{q(\xi,h_d)}\big[\hat{y}_d(x_d, h_d, \xi)\big]\big) \tag{2.20}$$

Here $C > 0$ is a scalar emphasizing how important the loss function is relative to the unsupervised problem, $p_0(\xi)$ is some prior distribution on global parameters, and $\mathcal{L}(x_d, \hat{v}_d, \xi)$ is the same lower bound as in Eq. (2.15). We can make this objective more comparable to our earlier objectives by performing point estimation of ξ instead of posterior approximation, which is reasonable in moderate to large data regimes, as the posterior for the global parameters ξ will concentrate. This choice allows us to focus on our core question of how to define an objective that balances data x and labels y, rather than the separate question of managing uncertainty during this training. Making this simplification by substituting point estimates for expectations, with the KL divergence regularization term reducing to $R(\xi) = -\log p_0(\xi)$, and the MED-LDA objective becomes:

$$\min_{\xi,\{\hat{v}_d\}_{d=1}^D} \quad R(\xi) - \sum_{d=1}^{D} \mathcal{L}(x_d, \hat{v}_d, \xi) + C\sum_{d=1}^{D} \mathrm{loss}\big(y_d, \mathbb{E}_{q(h_d)}\big[\hat{y}_d(x_d, h_d, \xi)\big]\big). \tag{2.21}$$

Both this objective and Graça et al. [2008]'s PR framework consider expectations over the approximate posterior $q(h_d)$, rather than our choice of the data-only posterior $p(h_d|x_d, \xi)$. However, the key difference between MED-LDA and the PR objectives is that the MED-LDA objective computes the loss of an expected prediction ($\mathrm{loss}(y_d, \mathbb{E}_q[\hat{y}_d])$), while the earlier PR objective in Eq. (2.17) penalizes the full expectation of the loss ($\mathbb{E}_{q(h_d)}[\mathrm{loss}(y_d, \hat{y}_d)]$). Earlier MED work [Jaakkola et al., 1999a] also suggests using an expectation of the loss,

[1] We note an irregularity between the classification and regression formulation of MED-LDA published by Zhu et al. [2012]: while classification-MED-LDA included labels y only the loss term, the regression-MED-LDA included *two* terms in the objective that penalize reconstruction of y: one inside the likelihood bound term $\mathcal{L}$ using a Gaussian likelihood G as well as inside a separate epsilon-insensitive loss term. Here, we assume that only the loss term is used for simplicity.

$\mathbb{E}_{q(\xi,h_d)}[\text{loss}(y_d, \hat{y}_d(x_d, h_d, \xi))]$. Decision theory argues that the latter choice is preferable when possible, since it should lead to decisions that better minimize loss under uncertainty. We suspect that MED-LDA chooses the former only because it leads to more tractable algorithms for their chosen loss functions.

Motivated by this decision-theoretic view, we consider modifying the MED-LDA objective of Eq. (2.21) so that we take the full expectation of the loss. This swap can also be justified by assuming the loss function is *convex*, as are both the epsilon-insensitive loss and the hinge loss used by MED-LDA, so that Jensen's inequality may be used to bound the objective in Eq. (2.21) from above. The resulting training objective is:

$$\min_{\xi, \{\hat{v}_d\}_{d=1}^D} \quad R(\xi) - \sum_{d=1}^{D} \mathcal{L}(x_d, \hat{v}_d, \xi) + C \sum_{d=1}^{D} \mathbb{E}_{q(h_d)} \Big[\text{loss}(y_d, \hat{y}_d(x_d, h_d, \xi)) \Big]. \tag{2.22}$$

In this form, we see that we have recovered the symmetric maximum likelihood objective with label replication from Eq. (2.14), with y replicated C times. Thus, even this MED effort fails to properly handle the asymmetry issue we have raised, possibly leading to poor generalization performance.

2.3.6 Relationship to Semi-supervised Learning Frameworks

Often, semi-supervised training is performed via optimization of the joint likelihood $\log p(x, y \mid \xi)$, using the EM algorithm to impute missing data [Nigam et al., 1998]. Other work falls under the thread of "self-training", where a model trained on labeled data only is used to label additional data and then retrained accordingly. Chang et al. [2007] incorporated constraints into semi-supervised self-training of an upstream hidden Markov model (HMM). Starting with just a small labeled dataset, they iterate between two steps: (1) train model parameters ξ via maximum likelihood estimation on the fully labeled set, and (2) expand and revise the fully labeled set via a constraint-driven approach. Given several candidate labelings y_d

for some example, their step 2 re-ranks these to prefer those that obey some soft constraints (for example, in a bibliographic labeling task, they require the "title" field to always appear once). Importantly, however, this work's sub-procedure for training from fully labeled data is a symmetric maximum likelihood objective, while our PC approach more directly encodes the asymmetric structure of prediction tasks.

Other work deliberately specifies prior domain knowledge about label distributions, and penalizes models that deviate from this prior when predicting on unlabeled data. Mann and McCallum [2010] propose *generalized expectation* (GE) constraints, which extend their earlier *expectation regularization* (XR) approach [Mann and McCallum, 2007]. This objective has two terms: a conditional likelihood objective, and a new regularization term comparing model predictions to some weak domain knowledge:

$$\log p(y|x,\xi) - \lambda\Delta(\hat{Y}(x,\xi), Y_H). \tag{2.23}$$

Here, Y_H indicates some expected domain knowledge about the overall labels-given-data distribution, while $\hat{Y}(x,\xi)$ is the predicted labels-given-data distribution under the current model. The distance function Δ, weighted by $\lambda > 0$, penalizes predictions that deviate from the domain knowledge. Unlike our PC approach, this objective focuses exclusively on the label prediction task and does not at all incorporate the notion of generative modeling.

2.4 Generalized Prediction-Constraints

The prediction-constrained training framework has a number of compelling advantages, but it is limited to models for which h_d can be explicitly marginalized and the prediction model can be effectively trained via maximum likelihood. In many practical applications these assumptions may not hold, thus we consider generalizations to the prediction-constrained

framework.

2.4.1 Approximating constraints

For many latent variable models it is not tractable to compute the marginalization over the hidden variable shown in Eq. (2.4). In such cases we can approximate the prediction-constrained training objective via samples from the posterior over the hidden variable,

$$\min_{\xi} -\left[\sum_{d=1}^{D} \log p(x_d \mid \xi^h, \xi^x)\right] - \lambda_\epsilon \left[\sum_{d=1}^{D} \log \frac{1}{S} \sum_{s=1}^{S} p(y_d \mid h_{ds}, \xi^y)\right] + R(\xi), \tag{2.24}$$
$$z_{d1} \ldots z_{dS} \sim p(h_d | x_d, \xi^h, \xi^x).$$

In cases where our hidden variable of interest is discrete, this sampling procedure is non-differentiable. In such cases, we consider an alternative prediction scheme using the expectation of the hidden state as the covariate for prediction.

$$\min_{\xi} -\left[\sum_{d=1}^{D} \log p(x_d \mid \xi^h, \xi^x)\right] - \lambda_\epsilon \left[\sum_{d=1}^{D} \log p\big(y_d \mid \mathbb{E}\big[h_d | x_d, \xi^h, \xi^x\big], \xi^y\big)\right] + R(\xi), \tag{2.25}$$

2.4.2 Constraints on posterior approximations

In many cases is impractical to compute or even sample from the true posterior, as in topic models or deep latent variable models, we can instead enforce a prediction constraint using an appropriate posterior approximation $q(h_d \mid x_d, \xi^h, \xi^x) \approx p(h_d \mid x_d, \xi^h, \xi^x)$, making the

approximation,

$$p(y_d \mid x_d, \xi^h, \xi^x, \xi^y) \approx \int p(y_d \mid x_d, h_d, \xi^y) q(z_d \mid x_d, \xi^h, \xi^x)\, dh_d \tag{2.26}$$

In this case our constrained objective becomes:

$$\begin{aligned} \min_{\xi} \quad & -\left[\sum_{d=1}^{D} \log p(x_d \mid \xi^h, \xi^x, \xi^y)\right] + R(\xi), \\ \text{subject to} \quad & -\sum_{d=1}^{D} \mathbb{E}_{q(h_d|x_d,\xi)}\big[\log p(y_d \mid h_d, \xi^y)\big] \leq \epsilon. \end{aligned} \tag{2.27}$$

We stress that our approximate posterior only approximates the posterior given the observation $p(z_d \mid x_d, \alpha, \theta)$ rather than the full posterior $p(z_d \mid x_d, y_d, \alpha, \theta, w)$, which would require knowledge of the true label y_d. We can similarly write our unconstrained PC objective as:

$$\min_{\xi} -\left[\sum_{d=1}^{D} \log p(x_d \mid \xi^h, \xi^x)\right] - \lambda_\epsilon \sum_{d=1}^{D} \mathbb{E}_{q(h_d|x_d,\xi)}\left[\log p(y_d \mid h_d, \xi^y)\right] + R(\xi). \tag{2.28}$$

Note that in cases where the marginalization over h_d is intractable for the prediction constraint, the equivalent marginalization in the maximum likelihood objective, $p(x_d \mid \xi^h, \xi^x) = \int_{h_d} p(x_d, h_d \mid \xi^h, \xi^x) dh_d$ may also be intractable. In such cases we again replace the exact marginal likelihood with a tractable lower bound, using the approximate posterior. The *prediction-constrained evidence lower bound objective* then becomes:

$$\min_{\xi} -\mathbb{E}_{q(h_d|x_d,\xi)}\left[\sum_{d=1}^{D} \log p(x_d, h_d \mid \xi) + \lambda_\epsilon \log p(y_d \mid x_d, h_d, \xi^y) - \log q(h_d|x_d, \xi)\right] + R(\xi) \tag{2.29}$$

Distinction from posterior regularization

This framing raises an important subtlety: what is the difference between the posterior regularized objective and the approximate prediction-constrained ELBO? The key to the distinction is in the treatment of the posterior approximation. We see that the posterior regularization objective jointly optimizes over both the global parameters ξ and the posterior approximation parameters $\hat{v}_d$ as in Eq. (2.19). In the prediction-constrained framework the approximate posterior (or equivalently its parameters $\hat{v}_d$) are *always* chosen to best approximate $p(h_d \mid x_d, \xi)$ given the global parameters ξ and approximating family q. For example, the equivalent to Eq. (2.19) under the prediction-constrained framework would become

$$\min_{\xi} -\mathbb{E}_{q(h_d|\hat{v}_d)}\left[\sum_{d=1}^{D} \log p(x_d, h_d \mid \xi) + \lambda_\epsilon \log p(y_d \mid x_d, h_d, \xi^y)\right] + R(\xi) \tag{2.30}$$
$$\hat{v} = \{\hat{v}_d\}_{d=1}^{D} = \arg\min_{\{\hat{v}_d\}_{d=1}^{D}} \sum_{d=1}^{D} \mathbb{E}_{q(h_d|\hat{v}_d)}\left[\log p(x_d, h_d \mid \xi) - \log q(h_d \mid \hat{v}_d)\right]$$

This is equivalent to minimizing the KL-divergence between $q(h_d \mid \hat{v}_d)$ and $p(h_d \mid x_d, \xi)$ for all $d \in \{1...D\}$. We refer to the minimization over $\hat{v}_d$ as the *inner optimization* and the minimization over ξ as the *outer optimization*. In this case we view the parameters of the approximate posterior as a function of the observation and global parameters $\hat{v}(x_d, \xi)$ and thus more compactly write our objective as:

$$\min_{\xi} -\sum_{d=1}^{D} \mathbb{E}_{q(h_d|\hat{v}(x_d,\xi))}\left[\log p(x_d, h_d \mid \xi) + \lambda_\epsilon \log p(y_d \mid x_d, h_d, \xi^y)\right] + R(\xi) \tag{2.31}$$

We note that in some cases, such as our prediction-constrained variational autoencoders, this distinction becomes ambiguous and the prediction-constrained model can be viewed as a special case of posterior regularization.

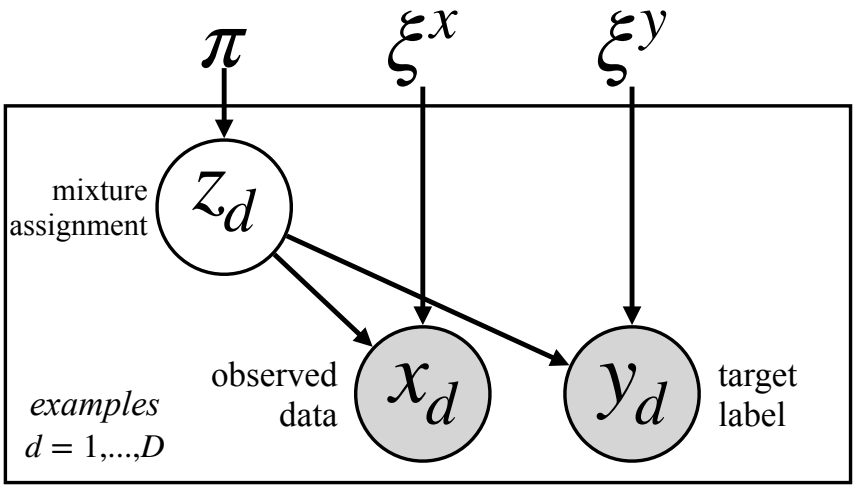


Figure 2.3: Graphical model representation of a supervised mixture model.

2.5 Prediction-constrained Mixture Models

We first present the application of prediction-constrained training to supervised mixture models. Our goal is to illustrate the benefits of our prediction-constrained approach in a situation where the marginalization over h_d in Eq. (2.6) can be computed exactly in closed form. This allows direct comparison of our proposed PC training objective to alternatives like maximum likelihood, without worry about how approximations needed to make inference tractable affect either objective.

Consider a simple supervised mixture model which generates data pairs x_d, y_d, as illustrated in Fig. 2.3. This mixture model assumes there are K possible discrete hidden states, and that the only hidden variable at each data point d is an indicator variable: $h_d = \{z_d\}$, where $z_d \in \{1, 2, \dots K\}$ indicates which of the K clusters point d is assigned to. For the mixture

model, we parameterize the densities in Eq. (2.1) as follows:

$$\log p(z_d = k \mid \xi^h) = \log \pi_k, \tag{2.32}$$

$$\log p(x_d \mid z_d = k, \xi^x) = \log p(x_d \mid \xi^x_k), \tag{2.33}$$

$$\log p(y_d \mid x_d, z_d = k, \xi^y) = \log p(y_d \mid x_d, \xi^y_k). \tag{2.34}$$

The parameter set of the latent variable prior P is simple: $\xi^h = \{\pi\}$, where π is a vector of K positive numbers that sum to one, representing the prior probability of each cluster.

We emphasize that the data likelihood $p(x_d \mid \xi^x_k)$ and label likelihood $p(y_d \mid x_d, \xi^y_k)$ are left in generic form since these are relatively modular: one could apply the mixture model objectives below with many different data and label distributions, so long as they have valid densities that are easy to evaluate and optimize for parameters ξ^x, ξ^y. Fig. 2.3 assumes the particular likelihood choices we used in our toy data experiments (Gaussian distribution for $p(x_d \mid \xi^x_k)$, Bernoulli distribution for $p(y_d \mid x_d, \xi^y_k)$), but we will develop our PC training for more general cases; the mixture component distributions could be other data appropriate distributions and alternative prediction models could include multiclass categorical prediction or regression. The only assumption we make is that each of the K clusters has a separate parameter set: $\xi^x = \{\xi^x_k\}_{k=1}^K$ and $\xi^y = \{\xi^y_k\}_{k=1}^K$.

Related work on supervised mixtures. While to our knowledge, our prediction-constrained optimization objective is novel, there exists prior literature on applying mixtures to supervised problems where the practitioner observes pairs of data covariates x and targets y. One line of work uses generative models with factorization structure like Fig. 2.3, where each cluster k has parameters for generating data ξ^x_k and targets ξ^y_k. For example, Ghahramani and Jordan [1993, Sec. 4.2] consider nearly the same model as in our toy experiments (except for using a categorical over labels y instead of a Bernoulli). They derive an Expectation Maximization (EM) algorithm to maximize a lower bound on the symmetric joint log likelihood

$\log p(x, y \mid \xi)$. Later applied work has sometimes called such models Bayesian profile regression when the targets y are real-valued [Molitor et al., 2010]. These efforts have seen broad extensions to generalized linear models especially in the context of Bayesian nonparametric priors like the Dirichlet process fit with MCMC sampling procedures [Shahbaba and Neal, 2009, Hannah et al., 2011, Liverani et al., 2015]. However, none of these efforts correct for the asymmetry issues we have raised, instead simply using the symmetric joint likelihood.

Other work takes a more discriminative view of the clustering task. Krause et al. [2010] develop an objective called Regularized Information maximization which learns a conditional distribution for y that preserves information from the data x. Other efforts do not estimate probability densities at all, such as "supervised clustering" [Eick et al., 2004]. Many applications of this paradigm exist [Finley and Joachims, 2005, Al-Harbi and Rayward-Smith, 2006, DiCicco and Patel, 2010, Peralta et al., 2013, Ramani and Jacob, 2013, Grbovic et al., 2013, Peralta et al., 2016, Flammarion et al., 2016, Ismaili et al., 2016, Yoon et al., 2016, Dhurandhar et al., 2017].

2.5.1 Objective function evaluation and parameter estimation.

Computing the data log likelihood. The marginal likelihood of a single data example x_d, marginalizing over the latent variable z_d, can be computed in closed form via the function:

$$\log p(x_d \mid \pi, \xi^x) = \log \sum_{k=1}^{K} \exp\Big(\log p(x_d \mid \xi_k^x) + \log \pi_k \Big). \tag{2.35}$$

Computing the label given data log likelihood. Similarly, the likelihood $p(y_d \mid x_d)$ of labels given data, marginalizing away the latent variable z_d, can be computed in closed

form:

$$\log p(y_d \mid x_d, \pi, \xi^x, \xi^y) = \\ \log \left[\sum_{k=1}^{K} \exp\Big(\log p(y_d \mid x_d, \xi_k^y) + \log p(x_d \mid \xi_k^x) + \log \pi_k \Big) \right] - \log p(x_d \mid \pi, \xi^x). \quad (2.36)$$

PC parameter estimation via gradient descent. Our original unconstrained PC optimization problem in Eq. (2.6) can thus be formulated for mixture models using this closed form marginal probability functions M and appropriate regularization terms R:

$$\min_{\pi,\ \xi^x,\ \xi^y} \quad -\sum_{d=1}^{D} \log p(x_d \mid \pi, \xi^x) - \lambda \sum_{d=1}^{D} \log p(y_d \mid x_d, \pi, \xi^x, \xi^y) + R(\xi). \quad (2.37)$$

We can practically solve this optimization objective via gradient descent. However, some parameters such as π live in constrained spaces like the $K-$dimensional simplex. To handle this, we apply invertible, one-to-one transformations from these constrained spaces to unconstrained real spaces and apply standard gradient methods easily.

In practice, for training supervised mixtures we use the Adam gradient descent procedure [Kingma and Ba, 2014], which requires specifying some baseline learning rate (we search over a small grid of 0.1, 0.01, 0.001) which is then adaptively scaled at each parameter dimension to improve convergence rates. We initialize parameters via random draws from reasonable ranges and run several thousand gradient update steps to achieve convergence to local optima. To be sure we find the best possible solution, we use many (at least 5, preferably more) random restarts for each possible learning rate and choose the one snapshot with the lowest training objective score.

2.5.2 Toy Example: Why Asymmetry Matters

We now consider a small example to illustrate one of our fundamental contributions: that PC training is often superior to symmetric maximum likelihood training with label replication, in terms of finding models that accurately predict labels y given data x. We will apply supervised mixture models to a simple toy dataset with data $x_d \in \mathbb{R}$ on the real line and binary labels $y_d \in \{0, 1\}$. The observed training dataset is shown in the top rows of Fig. 2.4 as a stacked histogram. We construct the data by drawing data x from three different uniform distributions over distinct intervals of the real line, which we label in order from left to right for later reference: interval A contains 175 data points $x \in [-1, 1]$, with a roughly even distribution of positive and negative labels; interval B contains 100 points $x \in [1, 1.5]$ with purely positive labels; interval C contains 75 points $x \in [1.5, 2.0]$ with purely negative labels. Stacked histograms of the data distribution, colored by the assigned label, can be found in Fig. 2.4.

We now wish to train a supervised mixture model for this dataset. To fully specify the model, we must define concrete densities and parameter spaces. For the data likelihood f, we use a 1D Gaussian distribution $\mathcal{N}(\mu_k, \sigma_k)$, with two parameters $\xi_k^x = \{\mu_k, \sigma_k\}$ for each cluster k. The mean parameter $\mu_k \in \mathbb{R}$ can take any real value, while the standard deviation is positive with a small minimum value to avoid degeneracy: $\sigma_k \in (0.001, +\infty)$. For the label likelihood g, we select a Bernoulli likelihood $\text{Bern}(\rho_k)$, which has one parameter per cluster: $\xi_k^y = \{\rho_k\}$, where $\rho_k \in (0, 1)$ defines the probability that labels produced by cluster k will be positive. For this example, we fix the model structure to exactly $K = 2$ total clusters for simplicity.

We apply very light regularization on only the π and ρ parameters:

$$R(\pi) = -\log \text{Dir}(\pi \mid 1.01, \dots 1.01), \quad R(\rho) = \sum_{k=1}^{K} -\log \text{Beta}(\rho_k \mid 1.01, 1.01). \tag{2.38}$$

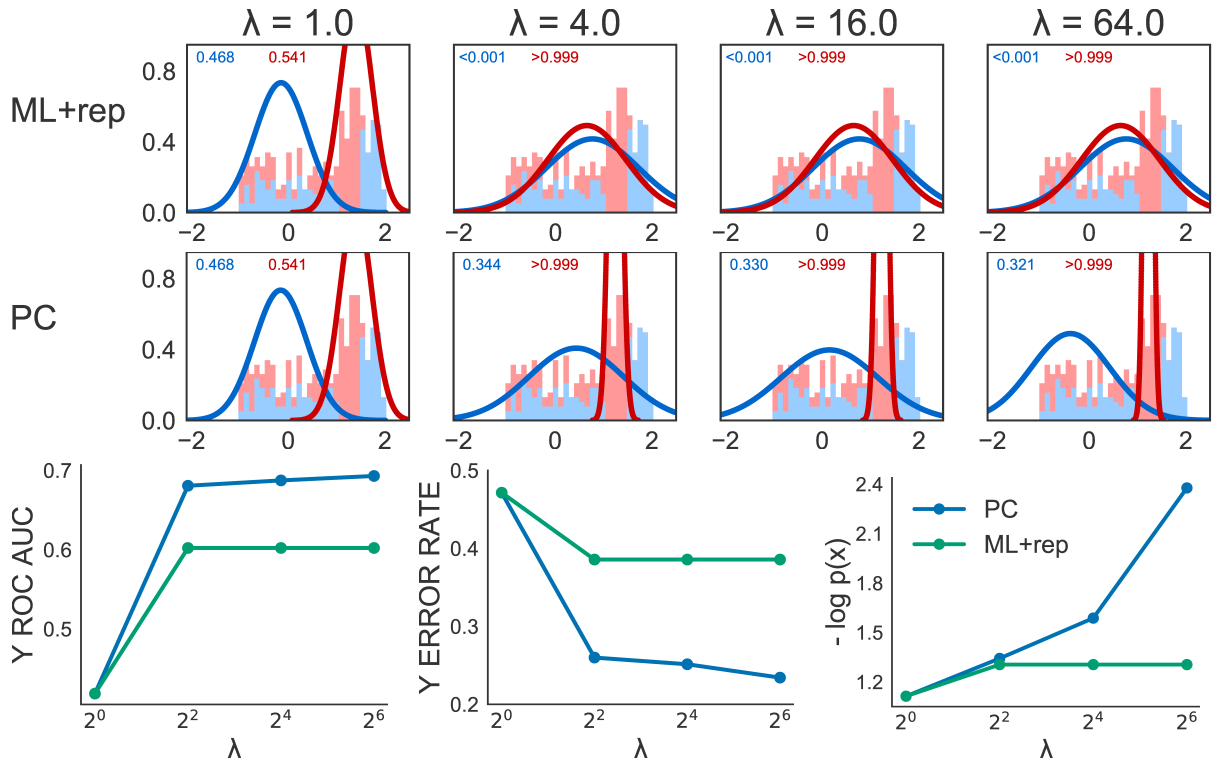


Figure 2.4: Toy example from Sec. 2.5.2: asymmetric prediction constrained (PC) training predicts labels better than symmetric joint maximum likelihood training with label replication (ML+rep). *Top rows:* Estimated 2-cluster Gaussian mixture model for each training procedure under different weight values λ, taking the best of many initializations using the relevant training objective function. Curves show the estimated 1D Gaussian distribution $\mathcal{N}(\mu_k, \sigma_k)$ for each cluster. Upper left text in each panel gives the estimated probability ρ_k that each cluster will emit a positive label. Colors are assigned so that red cluster has higher probability of emitting positive labels. Stacked histograms of 1-dimensional training dataset overlaid in background (blue shading means $y = 0$, red means $y = 1$). *Bottom row:* Area-under-the-ROC-curve and error rate scores for predicting labels y from data x on *training* data, using the best solution (as ranked by each training objective) across different weight values λ. Final panel shows negative log likelihood of data x (normalized by number of data points) across same λ values.

These choices ensure that MAP estimates of ρ_k and π are unique and always exist in numerically valid ranges (not on boundary values of exactly 0 or 1). This is helpful for the closed-form maximization step we use for the EM algorithm for the ML+rep objective.

When using this model to explain this dataset, there is a fundamental tension between explaining the data x and the labels $y|x$: no one set of parameters ξ will outrank all other parameters on both objectives. For example, standard joint maximum likelihood training

(equivalent to our PC objective when $\lambda = 1$) happens to prefer a $K = 2$ mixture model with two well-separated Gaussian clusters with means around 0 and 1.5. This gives reasonable coverage of data density $p(x)$, but has quite poor predictive performance $p(y|x)$, because the left cluster is centered over interval A (a non-separable even mix of positive and negative examples), while the right cluster explains both B and C (which together contain 100 positive and 75 negative examples).

Our PC training objective allows prioritizing the prediction of $y|x$ by increasing the Lagrange multiplier weight λ_ϵ. Fig. 2.4 shows that for $\lambda_\epsilon = 4$, the PC objective prefers the solution with one cluster (colored red) exclusively explaining interval B, which has only positive labels. The other cluster (colored blue), has wider variance to cover all remaining data points. This solution has much lower error rate (≈ 0.25 vs. ≈ 0.5) and higher AUC values (≈ 0.69 vs. ≈ 0.5) than the basic $\lambda_\epsilon = 1$ solution. Of course, the trade-off is a visibly lower likelihood of the training data $\log p(x)$, since the higher-variance blue cluster does less well explaining the empirical distribution of x. As λ_ϵ increases beyond 4, the quality of label prediction improves slightly as the decision boundaries get even sharper, but this requires the blue background cluster to drift further away from data and reduce data likelihood even more. In total, this example illustrates how PC training enables the practitioner to explore a range of possible models that tradeoff data likelihood and prediction quality.

In contrast, any amount of label replication for standard maximum likelihood training does *not* reach the prediction quality obtained by our PC approach. We show trained models for replication weights values equal to 1, 4, 16, and 64 in Fig. 2.4 (we use common notation λ for simplicity). For all values $\lambda > 1$, we see that symmetric joint "ML+rep" training finds the same solution: Gaussian clusters that are exclusively dedicated to either purely positive or purely negative labels. This occurs because at training time, both x and y are fully observed, and thus the replicated presence of y strongly cues which cluster to assign and allows completely perfect label classification. However, when we then try asymmetric

prediction of y given only x on the same training data, we see that performance is much worse: the error rate is roughly 0.4 while our PC method achieved near 0.25. It is important to stress that *no amount* of label replication would fix this, because the asymmetric task of predicting y given only x is not the focus of the symmetric joint likelihood objective.

2.5.3 Toy Example: Advantage of Semisupervised PC Training

Next, we study how our PC training objective enables useful analysis of *semi-supervised* datasets, which contain many unlabeled examples and few labeled examples. Again, we will illustrate clear advantages of our approach over standard maximum likelihood training in prediction quality.

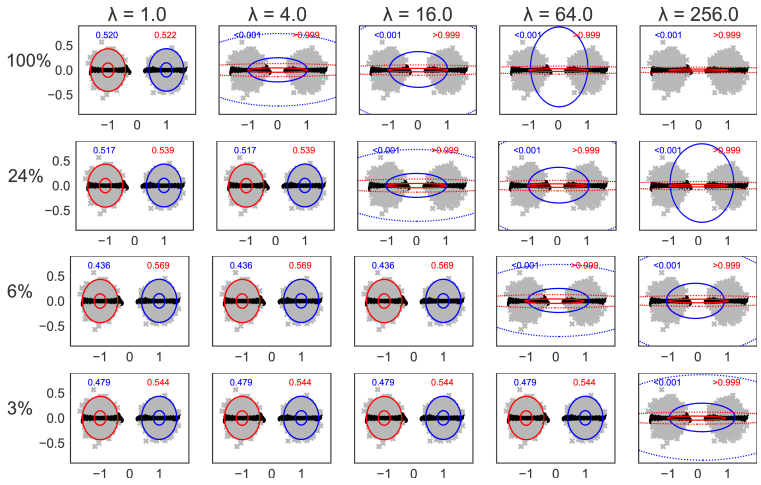
The dataset is generated in two stages. First, we generate 5000 data vectors $x_d \in \mathbb{R}^5$ drawn from a mixture of 2 well-separated Gaussians with diagonal covariance matrices:

$$x_d \sim \frac{1}{2}\mathcal{N}\left(\begin{bmatrix}-1\\0\\0\\0\\0\end{bmatrix}, \begin{bmatrix}2 & 1 & & & \\ 1 & 1 & & & \\ & & 0.5 & & \\ & & & 1 & \\ & & & & \end{bmatrix}\right) + \frac{1}{2}\mathcal{N}\left(\begin{bmatrix}+1\\0\\0\\0\\0\end{bmatrix}, \begin{bmatrix}2 & 1 & & & \\ 1 & 1 & & & \\ & & 0.5 & & \\ & & & & \end{bmatrix}\right).$$

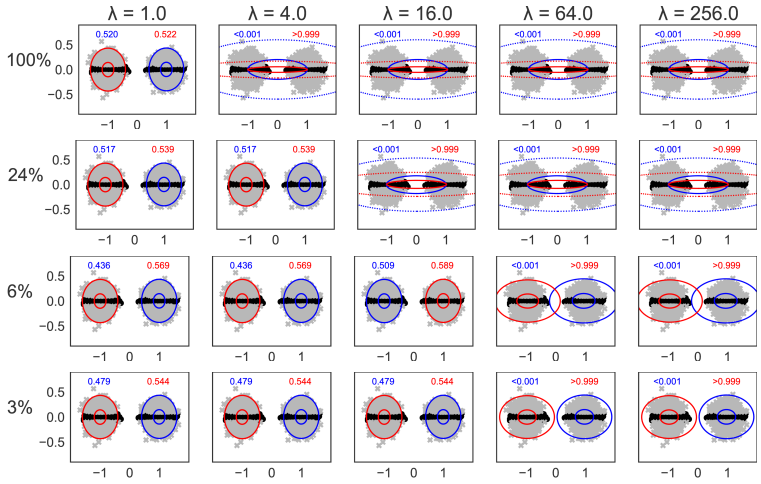
Next, we generate binary labels y_d according to a fixed threshold rule which uses only the absolute value of the second dimension of x_d:

$$y_d|x_d = \begin{cases} 1 & \text{if } |x_{d2}| < 0.1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.39}$$

While the full data vectors are 5-dimensional, we can visualize the first two dimensions of x as a scatterplot in Fig. 2.5. Each point is annotated by its binary label y: 0-labeled data points are grey 'x' markers while 1-labeled points are black 'o' markers. Finally, we make the problem *semi-supervised* by selecting some percentage b of the 5000 data points to keep labeled during training. For example if $b = 50\%$, then we train using 2500 labeled pairs



(a) PC: Prediction-constrained



(b) ML+rep: Maximum likelihood with label replication

Figure 2.5: Toy example from Sec. 2.5.3: Estimated supervised mixture models produced by PC training (a) and ML+rep (b) for semi-supervised tasks with few labeled examples. Each panel shows the 2D elliptical contours of the estimated $K = 2$ cluster Gaussian mixture model which scored best under each training objective using the indicated weight λ and percentage b of examples which have observed labels at training, which varies from 3% to 100%. Upper text in each panel gives the estimated probability ρ_k that each cluster will emit a positive label. Colors are assigned so that red cluster has higher probability of emitting positive labels. In the background of each panel is a scatter plot of the first two dimensions of data x, with each point colored by its binary label y (grey = negative, black = positive).

$\{x_d, y_d\}$ randomly selected from the full dataset as well as the remaining 2500 unlabeled data points. Our model specification is the same as the previous example: Gaussian with diagonal covariance for f, Bernoulli likelihood for g, and the same light regularization as before to allow closed-form, numerically-valid M-steps when optimizing the ML+rep objective via EM.

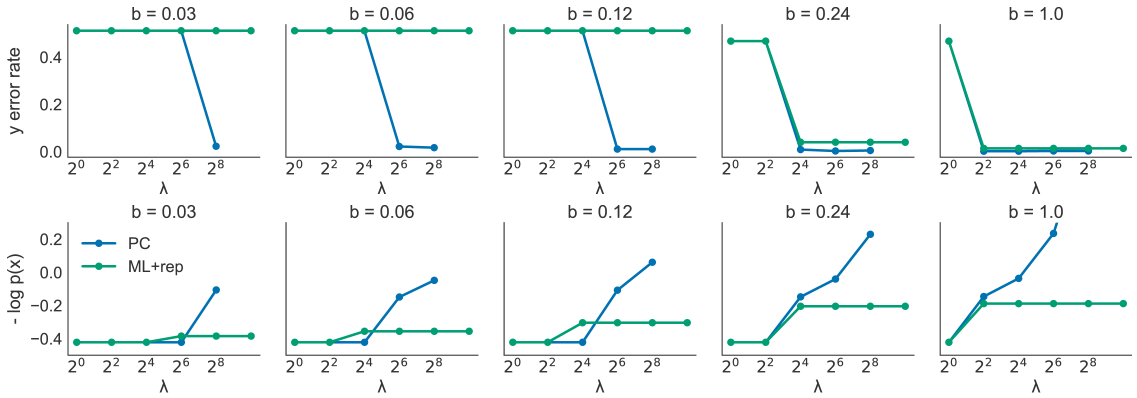


Figure 2.6: Toy example from Sec. 2.5.3: Each panel shows line plots of performance metrics as the PC or replication weight λ increases, for particular percentage of data b that is labeled. Top row shows label prediction error rate (lower is better), and bottom row shows negative data likelihood $-\log p(x)$ (lower is better). For visualizations of corresponding parameters, see Fig. 2.5.

We have deliberately constructed this dataset so that a $K = 2$ supervised mixture model is *misspecified*. Either the model will do well at capturing the data density $p(x)$ by covering the two well-separated blobs with equal-covariance Gaussians, or it will model the predictive density $p(y|x)$ well by using a thin horizontal Gaussian to model the black $y = 1$ points as well as a much larger background Gaussian to capture the rest. With only 2 clusters, no single model can do well at both.

Our PC approach provides a range of possible models to consider, one for each value of λ, which tradeoff these two objectives. Line plots showing overall performance trends for data likelihood $p(x)$ and prediction quality are shown in Fig. 2.6, while the corresponding parameter visualizations are shown in Fig. 2.5. Overall, we see that PC training when $\lambda = 1$, which is equivalent to standard ML training, yields a solution which explains the data x well but is poor at label prediction. For all tested fractions of labeled data b, as we increase λ there exists some critical point at which this solution is no longer preferred and the objective instead favors a solution with near-zero error rate for label prediction. For $b = 100\%$, we find a solution with near-zero error rate at $\lambda = 4$, while for $b = 3\%$ we see that it takes $\lambda \gg 64$.

In contrast, when we test symmetric ML training with label replication across many repli-

cation weights λ, we see big differences between plentiful labels ($b \gtrapprox 20\%$) and scarce labels ($b \lessapprox 20\%$). When enough labeled examples are available, high replication weights do favor the same near-zero error rate solution found by our PC approach. However, there is some critical value of b below which this solution is no longer favored, and instead the preferred solution for label replication is a pathological one: two well-separated clusters that explain the data well but have extreme label probabilities ρ_k. Consider the $b = 3\%, \lambda = 64.0$ solution for ML+rep in Fig. 2.5. The red cluster explains the left blob of unlabeled data x (containing about 2400 data points) as well as all positive labels y observed at training, which occur in both the left and right blobs (only 150 total labels exist, of which about half are positive). The symmetric joint ML objective weighs each data point, whether labeled or unlabeled, equally when updating the parameters ξ^h, ξ^x that control $p(x)$ no matter how much replication occurs. Thus, enough unlabeled points exert strong influence for the particular well-separated blob configuration of the data density $p(x)$, and the few labeled points can be easily explained as outliers to the two blobs. In contrast, our PC objective by construction allows upweighting the influence of the asymmetric prediction task on *all* parameters, including ξ^h, ξ^x. Thus, even when replication happens to yield good predictions when all labels are observed, it can yield pathologies with few labels that our PC easily avoids.

Chapter 3

Prediction-Constrained Hidden Markov Models

In this section, we explore prediction-constrained training of supervised Hidden Markov models for timeseries data. We now assume that each observation is a sequence of T values $x_d = \{x_{d1}...x_{dT}\}$. To simplify notation, we will assume that the T is shared across observations (i.e. all observed sequences are of the same length), however this assumption can easily be relaxed.

3.1 Hidden Markov Models

As with mixtures, HMMs [Rabiner and Juang, 1986a] assume that observed sequences are generated by a common model with K hidden, discrete states. Rather than each observation having a global state, the hidden variable is a sequence of T per-timestep state assignments $h_d = \{z_d\} = \{z_{d1}...z_{dT}\}$ drawn from a Markov process, where each state assignment z_{dt} is associated with the corresponding observation x_{dt} for time t. The global prior parameters ξ^h

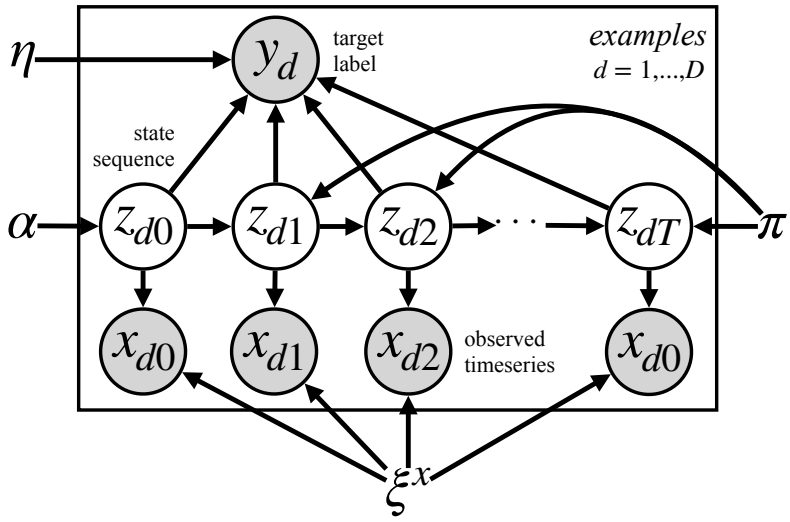


Figure 3.1: Graphical model representation of a supervised hidden Markov model.

define the initial state distribution α and a set of transition distributions π for this Markov process. So $\xi^h = \{\alpha, \pi\}$, where $\pi = \{\pi_k\}_{k=0}^K$ denotes the probabilities of transitioning from state j to state k: $\pi_{jk} = p(z_{dt} = k \mid z_{d,t-1} = j)$.

Concretely, we define our joint density as follows:

$$\log p(z_{d0} = k \mid \xi^h) = \log \alpha_k, \tag{3.1}$$

$$\log p(z_{dt} = k \mid \xi^h, z_{d,t-1}) = \log \pi_{z_{d,t-1}k} \qquad t \in \{1...T\}, \tag{3.2}$$

$$\log p(x_{dt} \mid z_{dt} = k, \xi^x) = \log p(x_{dt} \mid \xi_k^x) \qquad t \in \{1...T\}. \tag{3.3}$$

As with mixtures the data likelihood $p(x_{dt} \mid \xi_k^x)$ (also called the emission distribution) is modular and can be replaced with task-appropriate choices. In this section we consider two examples: Gaussian emissions with a state-specific mean and covariance, $\xi_k^x = \{\mu_k, \Sigma_k\}$:

$$p(x_{dt} \mid z_{nt}{=}k, \xi^x) = \mathcal{N}(x_{nt} \mid \mu_k, \Sigma_k), \tag{3.4}$$

as well as first-order autoregressive Gaussian emissions, $\xi_k^x = \{A_k, \mu_k, \Sigma_k\}$:

$$p(x_{nt} \mid x_{nt-1}, z_{nt}{=}k, \xi^x) = \mathcal{N}(x_{nt} \mid A_k x_{nt-1} + \mu_k, \Sigma_k). \tag{3.5}$$

To simplify notation, we assume that there exists an unmodeled observation x_{n0} at time zero.

3.2 Supervised Hidden Markov Models

In designing a supervised variant of the Hidden Markov Model, we note that the marginalization in equation 2.4 is intractable in general, requiring a sum over K^T terms to account for all possible state configurations. For HMMs we instead consider variants of the objective presented in equation 2.25.

Given the observed data x_n, we can efficiently compute posterior marginal probabilities, or *beliefs*, for the latent states z_d via the belief propagation or forward-backward algorithm [Rabiner and Juang, 1986a]. This algorithm computes a forward set of messages $\vec{m}_d$ at each timestep as:

$$\vec{m}_{dtk} = p(x_{dt} \mid \xi_k^x) \sum_{i=1}^{K} \vec{m}_{d(t-1)i} \left(\frac{\pi_{ki}}{\sum_{j=1}^{K} \pi_{ji}} \right), \qquad t \in \{1...T\},\ k \in \{1...K\} \tag{3.6}$$

$$\vec{m}_{d0k} = f(x_{d0} \mid \xi_k^x)\ \alpha_k. \qquad t = 0$$

The corresponding set of backwards messages $\overleftarrow{m}_d$ is computed as:

$$\overleftarrow{m}_{dtk} = \sum_{i=1}^{K} p(x_{d(t+1)} \mid \xi_i^x)\ \overleftarrow{m}_{d(t+1)i}\ \pi_{ki}, \qquad t \in \{1...T\},\ k \in \{1...K\} \tag{3.7}$$

We denote the posterior marginal probabilities by $b_{dtk} \triangleq p(z_{dt} = k \mid x_{d1} \ldots x_{d,T}, \alpha, \pi, \xi^x) \propto$

$\vec{m}_{dtk} \cdot \overleftarrow{m}_{dtk}$. Note that these beliefs are a deterministic function of x_d (which will be important for our end-to-end optimization) with computational cost $\mathcal{O}(TK^2)$. The beliefs b_{ntk} at time t take into account the full sequence x_n, including future timesteps $x_{dt'}$, $t' > t$. In some applications, predictions must be made only on the data up until time t. These *forward* beliefs $\vec{b}_{ntk} \triangleq p(z_{dt} = k \mid x_{d1}, \ldots, x_{dt}, \alpha, \pi, \xi^x)$ are computed by the forward pass of belief propagation.

We may view the complete sequence of beliefs b_d as a deterministic function of the data x_d, transition probability vectors α, π, and emission parameters ξ^x: $b_d = \mathrm{b}(x_d, \alpha, \pi, \xi^x)$. Similarly, for a specific timestep we can write $b_{dt} = \mathrm{b}_t(x_d, \alpha, \pi, \xi^x)$. Each of these functions can be computed with cost $\mathcal{O}(TK^2)$.

Now we consider the prediction of labels y given data x. Because they capture uncertainty in the hidden states z_d, the beliefs b_d are succinct (and computationally efficient) summary statistics for the data. We use beliefs as features for the prediction of labels y_d from data x_d in two scenarios: per-sequence classification, where the entire sequence d has a single label y_d, and per-timestep classification, where each timestep has its own event label $y_d = \{y_{d1}, \ldots, y_{dT}\}$.

3.2.1 Sequence classification.

In the sequence classification scenario, we seek to assign a scalar label y_d to the entire sequence. For example, in data retrieval applications we may wish to classify existing recordings of human activities into one of several predefined categories. Below, we provide two possible prediction functions that use belief features. For the probabilistic case our prediction function $\hat{y}(x_d, \alpha, \pi, \xi^x, \xi^y)$ will define the parameters of the conditional for y_d:

$$p(y_d \mid x_d, \alpha, \pi, \xi^x, \xi^y) = p(y_d \mid \hat{y}(x_d, \alpha, \pi, \xi^x, \xi^y)) \tag{3.8}$$

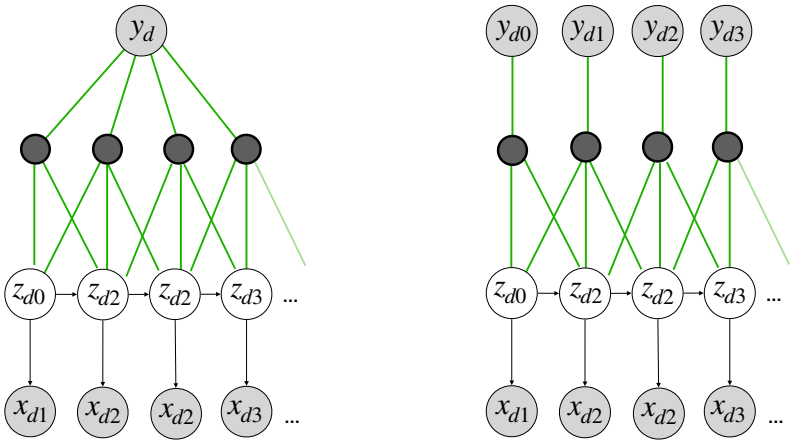


Figure 3.2: Representation of prediction models for sequence classification (*left*) and timestep classification (*right*).

First, we use the *fraction* of time spent in each state:

$$\hat{y}(x_d, \alpha, \pi, \xi^x, \xi^y) = \hat{g}(\eta^T \bar{b}(x_d, \alpha, \pi, \xi^x)), \quad \xi^y = \{\eta\} \tag{3.9}$$

$$\bar{b}(x_d, \pi, \phi) \triangleq \frac{1}{T_n} \sum_{t=1}^{T} \mathrm{b}_t(x_d, \alpha, \pi, \xi^x), \tag{3.10}$$

where $\mathrm{b}_t(x_n, \pi, \phi) = p(z_{nt} \mid x_n, \pi, \phi)$, η is a vector of regression coefficients, and $\hat{g}(\cdot)$ is an appropriate link function (e.g., a logistic $\hat{g}(w) = 1/(1 + e^{-w})$ for binary labels or a softmax for categorical labels). We demonstrate in Sec. 3.4 that predictions based on this intuitive feature match the performance of more complex, "deep" neural networks in several domains.

Second, we consider non-linear features learned from belief states. Some tasks may require this flexibility for high-quality prediction (especially if the intended HMM is misspecified). In these cases, we can replace the linear model based on averaged belief states in Eq. (5.13) with a general function that takes in the sequence of belief states, and outputs a prediction:

$$\hat{y}(x_d, \alpha, \pi, \xi^x, \xi^y) = \hat{g}(\bar{b}(x_d, \alpha, \pi, \xi^x);\ \eta), \quad \xi^y = \{\eta\} \tag{3.11}$$

where η are parameters of a parameterized differentiable function $g(\cdot; \eta)$, such as a neural network. In one of the sequence classification tasks in Sec. 3.4, we find that a prediction function incorporating a convolutional transformation of the belief sequence, followed by local max-pooling, leads to improved accuracy. The convolutional structure allows for predictions to depend on belief patterns that span several time-steps. We emphasize that the only input to this flexible prediction function are the beliefs produced by an HMM. Thus, the prediction task must be informing the learned states of this HMM.

Several previous efforts have integrated HMMs and deep neural networks. Kuehne et al. [2018] develop a per-frame activity classifier for videos where an RNN produces fine-grained likelihoods which are then fed into an HMM to infer smoothed segmentations over longer time-scales. Related efforts explore cooking videos [Malmaud et al., 2015] and sign-language sequences [Koller et al., 2017] using a similar neural likelihood approach. In contrast, our work applies an HMM to raw data and then feeds beliefs into a learned discriminator (possibly a NN). Our approach allows us to make predictions even when some data x_t is missing, and further performs end-to-end training to optimize all parameters at once to balance generative and discriminative goals, rather than the iterative alignment in Kuehne et al. [2018].

3.2.2 Event detection.

In other applications, we seek to densely label the events occurring at each timestep of a sequence, such as the prediction of medical events from hourly observations of patients in a hospital. To predict the label y_{dt} at time t, we use the beliefs b_{dt} at times in a window $t_{w_{start}} : t_{w_{end}}$ around t as features for a prediction model with parameters η:

$$\hat{y}_t(x_n, \pi, \phi, \eta) = \hat{g}(\mathrm{b}_{t_{w_{start}}:t_{w_{end}}}(x_n, \pi, \phi); \eta). \tag{3.12}$$

Here $\hat{g}(\cdot;\eta)$ could either be a generalized linear model based on the average state frequencies in the local time window, or a more complicated non-linear model as discussed for sequence classification.

Finally, we note that many prediction tasks are *offline*: the prediction is needed for post-hoc analysis and thus can incorporate information from the full data sequence. When a prediction needs to happen *online*, for example in forecasting applications, we use only the forward-beliefs $\vec{b}_{dt}$ as regression features.

While many previous efforts have used HMMs to produce features for prediction models (e.g., the forward-belief representation were used for ICU forecasting [Ghassemi et al., 2017]), prior work has trained via a two-stage procedure: first training the HMM via unsupervised likelihood maximization (without labels), and next training the prediction model $g(\cdot;\eta)$ while fixing the HMM.

Many competitive sequential prediction models are variants of *conditional random fields* (CRFs) [Lafferty et al., 2001] or *structural support vector machines* (SSVMs) [Tsochantaridis et al., 2004, Taskar et al., 2004]. These models typically assume the labels y are available for all training sequences x, and are trained to minimize a loss (log-likelihood for CRFs, hinge loss for SSVMs) in the prediction of y given x. Surveys have highlighted applications to natural language [Sutton and McCallum, 2012] and image data [Nowozin and Lampert, 2011].

3.3 Prediction-Constrained Learning of HMMs

Our *prediction-constrained* (PC) training objective for both the sequence classification and event detection is:

$$\min_{\alpha,\pi,\xi^x,\eta} -\sum_{d=1}^{D} \log p(x_d \mid \alpha, \pi, \xi^x) - R(\alpha, \pi, \xi^x, \eta) \tag{3.13}$$

$$\text{subject to:} \quad \begin{cases} \sum_{d=1}^{D} \ell\big(y_d, \hat{y}(x_d, \alpha, \pi, \xi^x, \eta)\big) \leq \epsilon \\ \quad \text{if one label per } \textit{sequence} \\ \\ \sum_{d=1}^{D} \sum_{t=1}^{T} \text{loss}\big(y_{dt}, \hat{y}_t(x_d, \alpha, \pi, \xi^x, \eta)\big) \leq \epsilon \\ \quad \text{if one label per } \textit{timestep} \end{cases}$$

As before we fit the model parameters α, π, ξ^x, η by using the KKT conditions to define an equivalent unconstrained objective that penalizes inaccurate label predictions:

$$\min_{\pi,\phi,\eta} \sum_{d=1}^{D} -\log p(x_d \mid \alpha, \pi, \xi^x) + \lambda_\epsilon \sum_{d:\mathcal{D}^S} \ell(y_d, \hat{y}(x_d, \alpha, \pi, \xi^x, \eta)) + R(\alpha, \pi, \xi^x, \eta) \tag{3.14}$$

Regularization.

In Eq. (3.14), we include a term to regularize the generative HMM parameters $\{\pi, \phi\}$ by penalizing their log density under some suitable prior distribution: $\log p(\alpha, \pi, \xi^x) = \log \text{Dir}(\alpha \mid \beta_\alpha) + \sum_{k=0}^{K} \log \text{Dir}(\pi_k \mid \beta_k) + \sum_{k=1}^{K} \log p(\xi_k^x)$. The form of the prior on the emission parameters ξ^x, will depend on the form of the likelihood. For Gaussian emissions we use a normal-inverse-Wishart prior, and for autoregressive emissions we use a matrix normal

inverse-Wishart prior. We can interpret our objective as the *maximum a posteriori* (MAP) estimation of generative parameters subject to a supervised prediction constraint.

Gradient Descent via Automatic Differentiation.

This training objective in Eq. (3.14) is differentiable with respect to the model parameters α, π, ξ^x, η and can thus be minimized via standard (stochastic) gradient descent algorithms. Efficient computation of the data log-likelihood $\log p(x_n \mid \alpha, \pi, \xi^x)$ is possible by accumulating the log-normalizers of the forward messages from belief propagation [Rabiner and Juang, 1986a]. We can compute all terms in Eq. (3.14) and their gradients via automatic differentiation software, with cost *linear* in the number of time steps.

Handling missing data.

The HMM construction allows for elegant handling of missing data within a sequence. Let x_d^O denote the observed timesteps of x_d. We can easily marginalize over the unobserved timesteps noting that $\int_{x_{dt}} p(x_{dt} \mid z_{dt} = k, \xi^x) = 1$, by definition. Modifying belief propagation simply amounts to dropping the likelihood terms $f(\cdot \mid \xi^x)$ from the messages in eq. 3.6 and eq. 3.7 for missing timesteps. Thus we can use this modified algorithm to compute $\log p(x_d^O | \alpha, \pi, \xi^x)$ and $\hat{y}(x_d^O, \alpha, \pi, \xi^x, \eta)$ and perform optimization as in the fully-observed case.

3.3.1 Toy example: PC-HMM

Figure 3.3 includes an intuitive application of the PC-HMM framework to binary sequence classification, illustrating its advantages over HMMs trained solely from unlabeled data, as well as its potential for semi-supervised learning from databases with few labeled sequences.

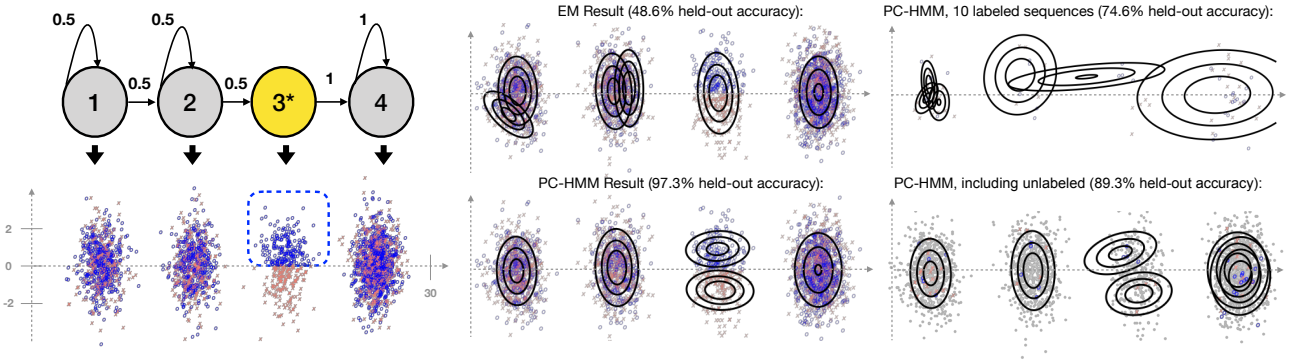


Figure 3.3: Comparison of our PC-HMM to conventional unsupervised HMM features on a synthetic binary classification task. *Left:* Many state sequences z_n of length 8 are drawn from the illustrated Markov chain, with 350 used for training and 150 held out for validation. 2D observations x_n are then drawn from evenly-spaced state-specific Gaussians. Each sequence is assigned a positive label y_n only if it has any observation x_{nt} that falls into the marked box. *Center: Fully-labeled task.* We compare a supervised PC-HMM (bottom) to the conventional approach (top) of first training an unsupervised HMM with the EM algorithm, and subsequently predicting labels given average belief state features with a linear classifier. The PC-HMM successfully learns to split the original state (yellow #3 in left diagram) that determines sequence labels, leading to more accurate predictions than features from the unsupervised HMM (97.3% accuracy vs. 48.6%). *Right: Semi-supervised task.* Only 10 randomly selected training sequences (out of 350) are labeled (colored observations). Using only these 10 sequences, the PC-HMM cannot model labels or data well (top). However, if allowed to use the 10 labeled sequences and 340 unlabeled sequences, then our PC-HMM accurately predicts test labels (bottom). All models use an overcomplete set of $K = 6$ states to avoid local optima. We select the best of 25 runs.

3.4 Applications: PC-HMM

We now assess how well our proposed PC training achieves our two key goals on real-world problems: accurate prediction of labels y given data x (even if labels are rare) and useful generative models of the sequential data x.

3.4.1 Baselines

RNN Baselines.

To establish a competitive baseline for some prediction tasks, we consider modern deep recurrent neural networks [Cho et al., 2014, Hochreiter and Schmidhuber, 1997]. We train RNNs via an aggressive randomized grid search over many possible architectures with 2 recurrent layers, varying the number of hidden units in each layer $\{10, 25, 50, 100\}$, the recurrent unit type $\{\text{gru}, \text{lstm}\}$, the activation function, and the number of dense output layers $\{1, 2\}$, batch size, learning rate and L_2 regularization strength on all weight parameters. The RNN models are trained to optimize the *class-balanced* cross-entropy loss (described in Sec. 2.2.2), using RMSprop stochastic gradient descent for up to 200 epochs with early stopping triggered whenever validation loss stops improving. For each possible model size (number of hidden units), we select the best of 50 possible hyperparameter configurations according to the validation set area under the ROC curve (AUC).

HMM Baselines.

To demonstrate that PC optimization is necessary to learn HMM states useful for prediction, we compare to a baseline that trains an HMM to maximize the unsupervised likelihood of the data, via *expectation maximization* (EM) [Rabiner and Juang, 1986b]. This HMM baseline first fits α, π, ϕ given only the data x, then trains a second-stage predictor with parameters η given belief states from the fixed HMM and labels y. An alternative supervised HMM (sHMM) baseline optimizes the $\lambda_\epsilon = 1$ special case of our PC objective. Across all HMM-based methods, to mitigate sensitivity to local optima we select the best of many independent runs from random initializations.

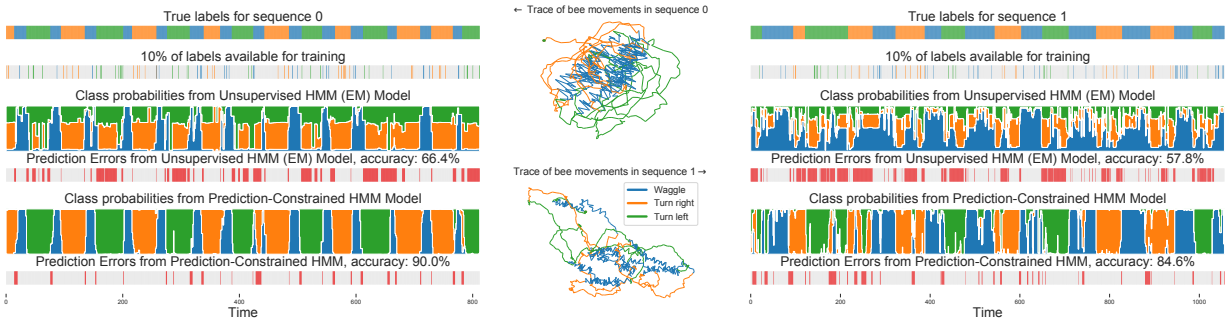


Figure 3.4: Per-timestep label completion task on the dancing bee dataset. Each method must predict the missing behavior labels for all 6 bee sequences given 10% of the labels selected at random. We heuristically set $\lambda = 100$ for the PC-HMM based on the ratio of observed labels to observed features and ran each algorithm from 15 random initializations, choosing the best using their corresponding objective function. We visualize the learned states from our PC-HMM and an unsupervised HMM on two representative sequences (*left:* sequence 0, *right:* sequence 1). Over all 6 sequences, the EM baseline achieved an overall accuracy of 61.1% on the missing labels and the PC-HMM achieved an accuracy of 87.2%.

3.4.2 Dancing Honey Bee Segmentation.

Honey bees communicate the location of food sources to other members of their hive through "waggle dances." Oh et al. [2008] tracked 6 different bees performing these dances and identified 3 distinct behaviors within each dance: *turn left*, *turn right* and *waggle* (moving straight while waggling its body). We consider the task of per-timestep prediction of these behaviors using the bee's tracked position and orientation.

We consider two tasks, a label completion task and a leave-one-sequence-out generalization task. For both tasks, we assume $K = 6$ states, an AR-Gaussian emission model, and a prediction model with two additional layers between the beliefs and the per-timestep predictions: a convolutional layer with 3 filters of width 3 and a max-pooling layer with window size 5. The small size of this data makes flexible discriminative models, such as RNNs, a poor choice.

Fig. 3.4 shows the results of a label completion task on two representative sequences. Across all sequences, the PC-HMM achieves 87.2% accuracy at label completion, compared to 61.1%

for the unsupervised HMM. On the leave-one-sequence-out task, our PC-HMM achieved **80.4%** accuracy while the EM baseline achieved only 59.4%.

3.4.3 Human activity recognition.

We next consider human activity recognition using the *activities of daily living* task from the UniMiB SHAR dataset [Micucci et al., 2017]. This dataset consists of 7759 short (256 timesteps) sequences of 3-axis accelerometer measurements captured from 30 subjects performing 9 different everyday activities, such as *walking*, *sitting down*, and *climbing stairs*. The measurement at each timestep specifies the acceleration in each of the 3 spatial dimensions at that time.

For this task, we used the Hyperopt library [Bergstra et al., 2013] to perform a wider search over the prediction model structure and hyperparameters, including the regularization hyperparameters and the number of states. We used a fixed 70%-30% train-validation split to evaluate each candidate model. Our final model uses a AR Gaussian likelihood, and a prediction model that applies a convolutional layer with 10 filters of width 6 to the belief sequence, followed by a max-pooling layer of width 9. Labels are predicted using a linear model from the time-averaged outputs of the pooling layer. We pre-processed the data for the HMM models by approximately removing gravity with a low-pass Butterworth filter [Anguita et al., 2013], then smoothing with another Butterworth filter and downsampling to effectively 10Hz.

We evaluated the performance of our final model using leave-one-out cross-validation over the 30 subjects in the dataset. Results are shown in Figure 3.5. Our best model with non-linear prediction function achieved a total accuracy of **83.0 %**. This is a substantial improvement over the 73.2% baseline accuracy from a random forest classifier reported by [Micucci et al., 2017]. A PC-HMM model using the simpler linear prediction model of Eq. (5.13) achieves

72.2% accuracy (**78.6%** accuracy with a second-order AR emission model in place of the first-order model).

3.4.4 Ventilator need in the ICU.

We consider a treatment prediction task using 16492 train, 2007 validation, and 4582 test sequences of vital signs and lab results available from the MIMIC-III public dataset of patient stays in an intensive care unit [Johnson et al., 2016a]. Following previous work on treatment onset prediction [Ghassemi et al., 2017], each sequence x_n contains 18 hourly measurements: 7 vital signs and 11 laboratory measurements. The per-sequence binary outcome y_n is the need for a mechanical ventilator (breathing tube), which was positive in 34% of the training data. Positively labeled sequences were censored one hour before the ventilator was installed to prevent label leakage into the data x_n. Negatively labeled sequences were re-sampled to have similar length distribution as the positive sequences. The average length of a sequence is 15.6 hours, with maximum length 40 hours. We train all methods with the class-balanced logistic loss, with our PCHMM using a linear prediction model given beliefs. Results are shown in Fig. 3.6. With 100% examples labeled, our 10-state PCHMM achieves an area under the precision-recall curve (AUROC) score of 0.878 which is slightly better than the RNN's 0.867. Greater advantage is seen when only 10% of examples are labeled: the 10-state PCHMM with achieves **0.848** AUROC, which beats the RNN's 0.785 and the plain HMM's 0.817.

3.4.5 ICU Mortality.

To evaluate the PC-HMM on clinical tasks that require SSL with feature missingness, we predict in-ICU mortality after the first 24 hours on the eICU dataset [Pollard et al., 2018] of vitals and labs (81% missing) from deidentified patient-stays at 59 critical care units

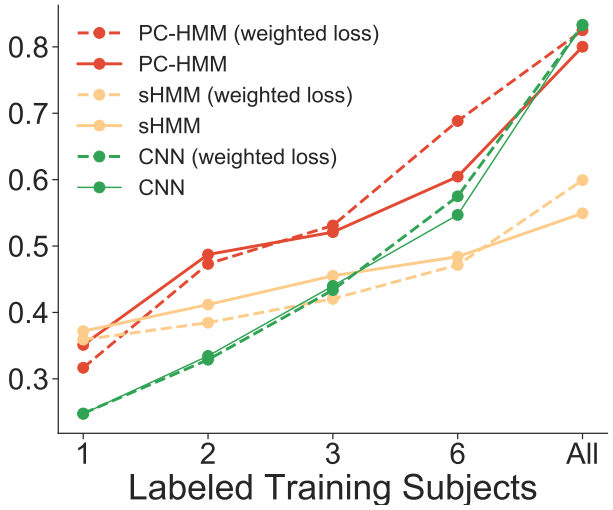
throughout the U.S. For each patient-stay, we extract 3 demographics, 8 vitals, and 6 lab measurements discretized to hourly bins using eICU Extract [Wang et al., 2020]. Our train/-valid/test splits have 43642/14509/14518 patient-stays, with $\sim 8.2\%$ resulting in death. We train the model at various percentages p of labels available for training, approximately preserving the full dataset's label imbalance. Fig. 3.7 shows the area under the precision-recall curve for each method (where $y = 1$ means death) as label availability p increases. Across all tested percentages p, the PC-HMM is competitive with deep alternatives, including BRITS [Cao et al., 2018] and GRU-D [De Brouwer et al., 2019] as well as deep SSL such as FixMatch [Sohn et al., 2020] and MixMatch [Berthelot et al., 2019b].

3.4.6 MIMIC-IV.

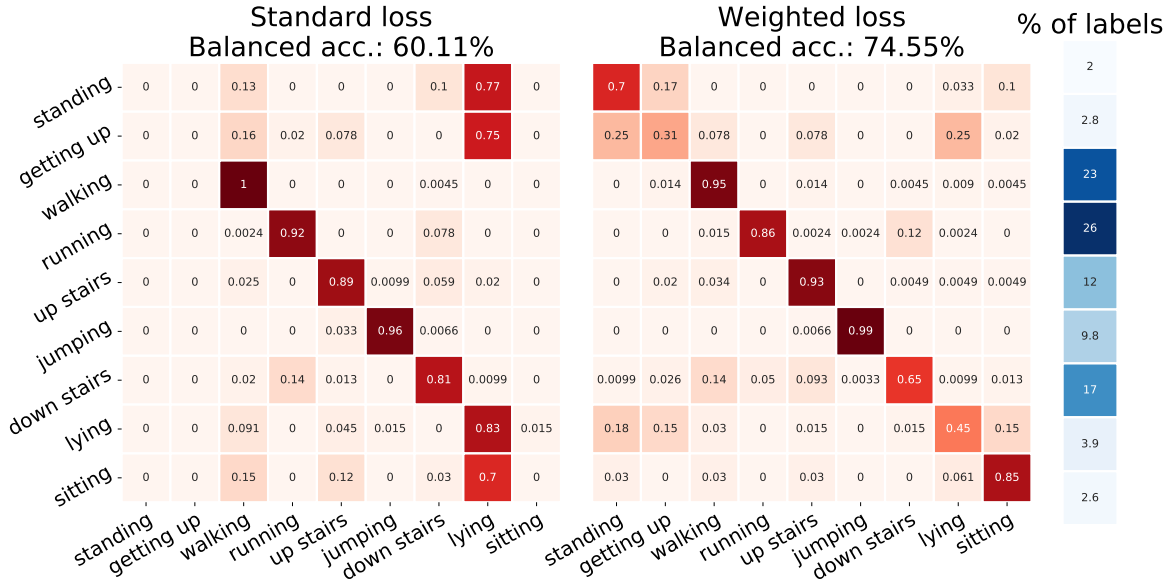
We further analyzed MIMIC-IV [Johnson et al., 2020], which contains over 60,000 de-identified ICU patient-stays from one hospital. In an effort to make the task more challenging, our preprocessing downsampled feature frequencies and death events (67% missingness, 1.2% mortality). Our train/valid./test splits have 42836/15443/15802 patient-stays, with $\sim 1.2\%$ patient stays resulting in death. Fig. 3.7 shows again that PC-HMM results are quite competitive with deep learning baselines with 100x more parameters.

3.4.7 Interpreting learned PC-HMM models.

Our PC-HMM framework lets us interpret cohorts of vulnerable populations by visualizing the emission distributions for the states that have the highest predictor coefficients η_k. On eICU data, Fig. 3.8 shows our PC-HMM identifies states representing high blood urea nitrogen (BUN) and high creatinine (common indicators of kidney failure [Baha et al., 2021]) as the most vulnerable to in-ICU mortality. The model transitions to these 'high-risk' states as soon as high values of creatinine and BUN are seen in a patient who eventually dies.



(a) Cross-validation accuracy



(b) Confusion matrix for test data using cross-entropy and weighted losses

Figure 3.5: Performance on the *activities of daily living* task, evaluated using 30-fold cross-validation, split by human subject. We show the performance for the PC-HMM model varying the number of labeled test subjects available in each fold of the training data (each fold uses unique labeled subjects). We compare these results against a CNN architecture for this task [Li et al., 2018b] that we found strictly outperformed our RNN baseline. Using a class-balanced loss, the PC-HMM outperforms the CNN in most cases and is better able to distinguish rare classes at test time. It is also clear that an objective equivalent to maximizing the joint likelihood p(y,x) ($\lambda = 1$), is outperformed by a prediction constrained objective with further emphasis on the discriminative term.

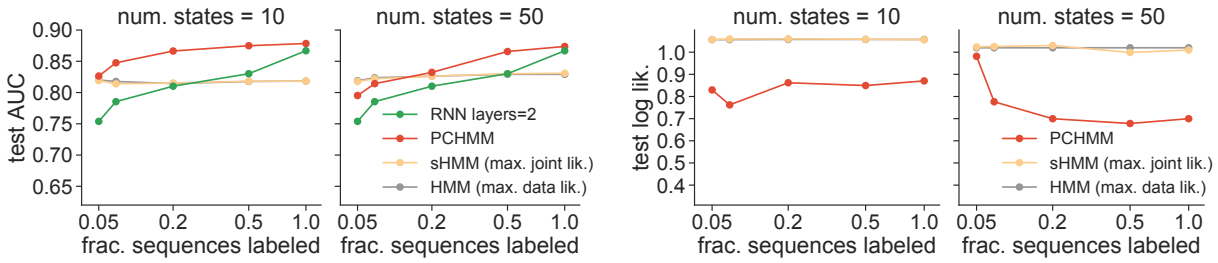


Figure 3.6: Performance on ICU task: model vital signs x and predict need-for-ventilator y, using 5%, 10%, 20%, 50%, and 100% of labels. *Left:* AUC scores (prediction quality, higher is better) on heldout test set, for HMMs with $K = 10$ and $K = 50$ states. *Right:* Predictive likelihood (generative quality, higher is better) on test set.

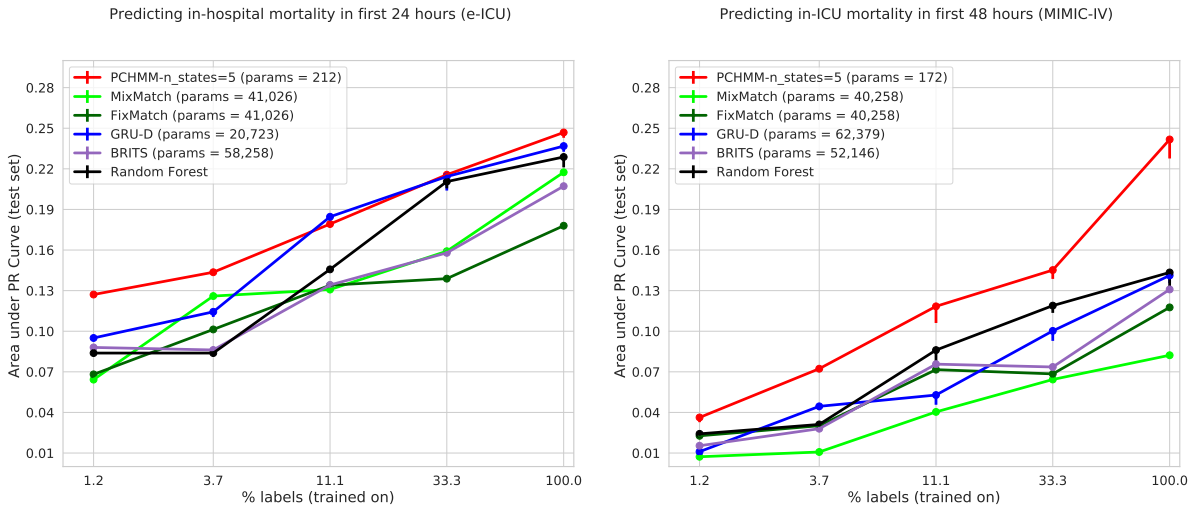


Figure 3.7: AUPRC (higher is better) versus amount of labeled data for in-hospital mortality early warning classifiers on two large EHR datasets (left: eICU, right: MIMIC-IV). X-axis: Percentage of all training sequences available with labels (SSL methods treat remaining sequences as unlabeled; other methods discard them). Y-axis: Area under precision-recall curve (AUPRC, higher is better). SSL methods, including our PC-HMM as well as MixMatch and FixMatch, learn from both labeled and unlabeled data. GRU-D, BRITS, and Random Forest use the labeled set only. The PC-HMM matches or beats the other models across all tested labeled set sizes, despite needing fewer parameters and only $1/10^{th}$ of the training time as the other models.

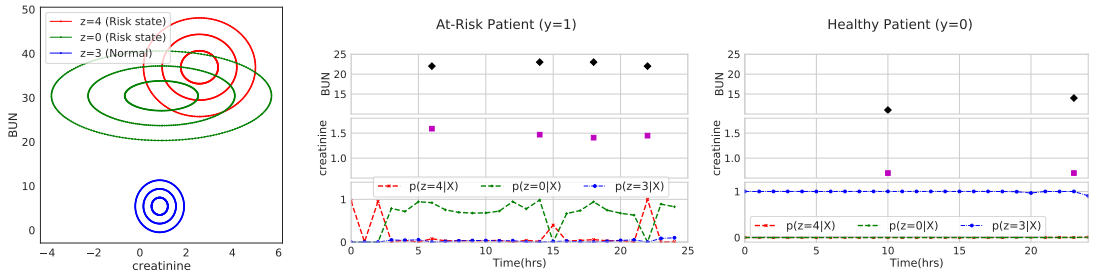


Figure 3.8: Interpretation of learned PC-HMM on eICU data. ***Left***: Gaussian emission distributions of states with highest and lowest η_k (predictor weights). The red and green states represent high risk (high η_k); the blue state indicates low risk. We show only 3 states and 2 features for clarity. High risk states seem to represent populations with high blood urea nitrogen (BUN) levels (above 20mg/dL) and high creatinine levels (above 1.45mg/dL), which could indicate kidney problems [Baha et al., 2021]. ***Center***: For a patient labeled $y = 1$ (who eventually dies), high BUN and high creatinine are observed throughout the stay, and thus beliefs b_t give highest probability to high risk states (green and red). The probability of state 4 (red) peaks initially and towards the end of the stay because of high heart rate (> 150 bpm) and high blood glucose (> 170 mg/dL). ***Right***: For a patient labeled $y = 0$ with normal creatinine and BUN levels, our PC-HMM says the most likely state is low-risk (blue).

Chapter 4

Prediction-Constrained Topic Models

4.1 Latent Dirichlet Allocation

The *latent Dirichlet allocation* topic model [Blei, 2012] (LDA) finds structure in a collection of D documents, or more generally, D examples of count vectors. Each document d is represented by a count vector x_d of V discrete words or features: $x_d \in \mathbb{Z}_+^V$. The LDA model generates these counts via a document-specific mixture of K topics:

$$\begin{aligned} \pi_d|\alpha &\sim \text{Dir}(\pi_d \mid \alpha), \\ x_d|\pi_d, \phi &\sim \text{Mult}(x_d \mid \textstyle\sum_{k=1}^K \pi_{dk}\phi_k, N_d). \end{aligned} \tag{4.1}$$

The latent random variable $h_d = \{\pi_d\}$ is a document-topic probability vector, where π_{dk} is the probability of topic k in document d and $\sum_{k=1}^K \pi_{dk} = 1$. The vectors $\xi^x = \{\phi_1 ... \phi_K\}$ are topic-word probability vectors, where ϕ_{kv} gives the probability of word v in topic k and $\sum_{v=1}^V \phi_{kv} = 1$. N_d is the (observed) size of document d: $N_d = \sum_v x_{dv}$. LDA assumes π_d has

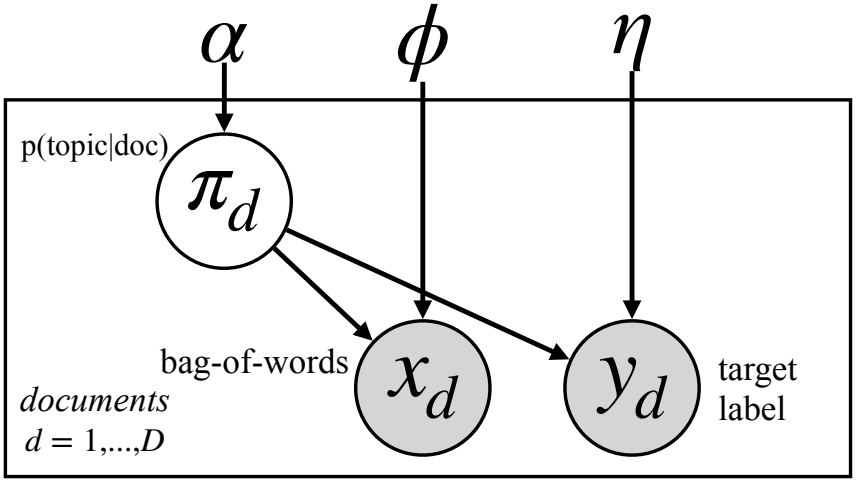


Figure 4.1: Graphical model representation of a supervised latent Dirichlet allocation.

a symmetric Dirichlet prior, with parameters $\xi^h = \{\alpha\} > 0$. Our log joint density is:

$$\begin{aligned} \log p(\pi_d|\alpha) &= \log \mathrm{Dir}(\pi_d \mid \alpha), \\ \log p(x_d|\pi_d, \phi) &= \log \mathrm{Mult}(x_d \mid \textstyle\sum_{k=1}^{K} \pi_{dk}\phi_k, N_d). \end{aligned} \tag{4.2}$$

4.2 Supervised Topic Models

Suppose document d also has a binary label $y_d \in \{0, 1\}$. Standard supervised topic models assume labels and word counts are conditionally independent given document-topic probabilities π_d:

$$y_d \mid \pi_d, \eta \sim \mathrm{Bern}(y_d \mid \hat{g}(\textstyle\sum_{k=1}^{K} \pi_{dk}\eta_k)), \tag{4.3}$$

where $\hat{g}(z) = (1 + e^{-z})^{-1}$ is the logit function, and $\xi^y = \{\eta\} \in \mathbb{R}^K$ is a vector of real-valued regression weights. Non-binary labels can be predicted via a generalized linear model [McAuliffe and Blei, 2008]. In some experiments, we model vectors of binary labels $y_d \in \{0, 1\}^L$ with L conditionally independent logistic regressions.

The sLDA model of McAuliffe and Blei [2008] represents the count likelihood of Eq. (4.2) via N_d independent assignments $z_{dn} \sim \text{Cat}(\pi_d)$ of word tokens to topics, and generates labels $y_d \sim \text{Bern}(y_d \mid \hat{g}(\sum_{k=1}^{K} \bar{z}_{dk}\eta_k))$, where $\bar{z}_d = N_d^{-1} \sum_n z_{dn}$ and $E[\bar{z}_d] = \pi_d$. To enable more efficient inference algorithms, we analytically marginalize the topic assignments z_d away in Eq. (4.2,4.3).

There also exist "upstream" variants of supervised topic models [Lacoste-Julien et al., 2009, Mimno and McCallum, 2008] in which the document-topic probabilities π_d have a distribution that is conditioned on the label y_d. We focus on "downstream" topic models as in Eq. (4.3) because they are more easily learned from data in which not all documents have labels y_d.

4.3 Prediction-constrained Training of LDA

For prediction-constrained training of supervised topic models, we focus on training linear predictors with a probabilistic loss as in prior work on supervised LDA:

$$\hat{y}(\pi_d, \xi^y) \triangleq \hat{g}(\eta^T \pi_d), \quad \xi^y = \{\eta\}. \tag{4.4}$$

$$\ell(y_d, \hat{y}(\pi_d, \eta)) = -\log p(y_d \mid \hat{y}(\pi_d, \eta)). \tag{4.5}$$

We note that as with mixture models and HMMs, the framework could easily be applied to non-probabilistic losses and non-linear prediction functions. Our prediction-constrained objective for supervised LDA is therefore:

$$\begin{aligned} \min_{\alpha,\phi,\eta} \; & -\Big[\sum_{d=1}^{D} \log p(x_d \mid \phi, \alpha)\Big] + R(\alpha, \phi, \eta) \\ \text{subject to} \quad & -\sum_{d=1}^{D} \log p(y_d \mid x_d, \alpha, \phi, \eta) \leq \epsilon. \end{aligned} \tag{4.6}$$

The corresponding unconstrained objective is:

$$\min_{\alpha,\phi,\eta} -\sum_{d=1}^{D} \Big[\log p(x_d \mid \alpha, \phi) + \lambda_\epsilon \log p(y_d | x_d, \alpha, \phi, \eta) \Big] + R(\alpha, \phi, \eta). \tag{4.7}$$

Computing $p(x_d \mid \alpha, \phi)$ and $p(y_d \mid x_d, \alpha, \phi, \eta)$ requires the marginalization of π_d over its simplex domain Δ^K:

$$\begin{aligned} p(x_d \mid \alpha, \phi) &= \int \mathrm{Mult}\big(x_d \mid \textstyle\sum_{k=1}^{K} \pi_{dk}\phi_k\big)\mathrm{Dir}\big(\pi_d \mid \alpha\big) d\pi_d, \\ p(y_d \mid x_d, \phi, \eta) &= \int \mathrm{Bern}\big(y_d \mid \sigma(\pi_d^T \eta)\big) p\big(\pi_d \mid x_d, \alpha, \phi, \alpha\big) d\pi_d. \end{aligned} \tag{4.8}$$

Because $p(y_d \mid x_d, \alpha, \phi, \eta)$ integrates over $p(\pi_d \mid x_d, \alpha, \phi)$, the posterior of π_d given *only* words x_d, our PC objective encodes the asymmetry of label prediction tasks.

Unfortunately, these integrals are intractable. To gain traction, we first contemplate an objective that *instantiates* π_d rather than marginalizing π_d away:

$$\min_{\pi,\phi,\eta} -\sum_{d=1}^{D} \Big[\log p(\pi_d \mid \alpha) + \log p(x_d \mid \pi_d, \phi) + \lambda \log p(y_d \mid \pi_d, \eta) \Big] + R(\alpha, \phi, \eta) \tag{4.9}$$

As discussed in previous sections, solutions to this objective would lead to replicated joint training and its poor predictions of y_d given x_d alone. Since we wish to train under the same asymmetric conditions present at test time, where we have x_d but not y_d, we *fix* π_d to a deterministic embedding of the words x_d to the topic simplex. We choose this mapping to produce the *maximum a posteriori* (MAP) estimate of π_d given x_d: $\pi_d = \mathrm{argmax}_{\pi_d \in \Delta^K} \log p(\pi_d \mid x_d, \phi, \alpha)$. As we show in Sec. 4.3.1, this MAP estimate can be found deterministically via a tractable function: $\pi_d \leftarrow \mathrm{MAP}(x_d, \phi, \alpha)$.

Our chosen MAP embedding is a feasible approximation to the full posterior $p(\pi_d | x_d, \phi, \alpha)$ needed in Eq. (4.8), with approximation accuracy increasing as the number of observed words

N_d grows. We can now write a *tractable* PC training objective for sLDA:

$$-\sum_{d=1}^{D}\Big[\log p\big(\text{MAP}(x_d,\phi,\alpha)\mid\alpha\big)+\log p\big(x_d\mid\text{MAP}(x_d,\phi,\alpha),\phi\big) \\ +\lambda_\epsilon\log p\big(y_d\mid\text{MAP}(x_d,\phi,\alpha),\eta\big)\Big]+R(\alpha,\phi,\eta). \tag{4.10}$$

Note that is this a special case of Eq. (2.31), where $q(h_d) \triangleq \delta(h_d - \pi_d)$ and $\hat{v}(x_d \mid \xi) \triangleq \text{MAP}(x_d, \phi, \alpha)$, where $\delta(\cdot)$ is the Dirac delta function.

While this objective is similar to BP-sLDA [Chen et al., 2015], the key difference is that the prediction constraint of Eq. (4.6) leads to a multiplier λ_ϵ that balances the generative and discriminative objectives. In contrast, Chen et al. [2015] consider only a fully unsupervised objective (labels y are ignored) and a fully supervised objective (the distribution of x is ignored). If documents are partially labeled, the objectives of Eq. (4.6) and (4.10) can be naturally generalized to only include prediction constraints for observed labels.

4.3.1 Inference and Learning for PC-LDA

We first show how to evaluate the PC objective of Eq. (4.10) by describing an algorithm that computes the embedding $\text{MAP}(x_d, \phi, \alpha)$. We then differentiate through the entire objective to allow gradient-based optimization of the topic-word probability vectors $\{\phi_k\}_{k=1}^K$, document-topic probability prior $\{\alpha\}$ and regression coefficients $\{\eta_k\}_{k=1}^K$.

MAP via Exponentiated Gradient.

Sontag and Roy [2011] define the document-topic MAP estimation problem for LDA as $\text{argmax}_{\pi_d \in \Delta^K} \ell(\pi_d, x_d, \phi, \alpha)$, where

$$\ell(\pi_d, x_d, \phi, \alpha) = \log \text{Mult}(x_d \mid \pi_d^T \phi) + \log \text{Dir}(\pi_d \mid \alpha). \tag{4.11}$$

This problem is convex for $\alpha \geq 1$ and non-convex otherwise. For the convex case, they apply an *exponentiated gradient* algorithm [Kivinen and Warmuth, 1997] that iteratively re-scales elements of the probability vector with exponentiated derivatives of the objective ℓ:

$$\begin{aligned} &\text{init:} \quad \pi_d^0 \leftarrow \left[\tfrac{1}{K} \dots \tfrac{1}{K}\right], \\ &\text{repeat:} \ \pi_{dk}^t \leftarrow \tfrac{p_{dk}^t}{\sum_{j=1}^K p_{dj}^t}, \quad p_{dk}^t = \pi_{dk}^{t-1} \cdot e^{\nu \nabla \ell(\pi_{dk}^{t-1})}. \end{aligned} \tag{4.12}$$

With small enough step size $\nu > 0$, exponentiated gradient (EG) converges to the MAP solution. We define our embedding function $\pi_d \leftarrow \text{MAP}(x_d, \phi, \alpha)$ to be the deterministic outcome of T EG iterations. In experiments, we use $T \approx 100$ and $\nu \approx 0.005$.

When $\alpha < 1$, the sparsity-promoting Dirichlet prior may lead to multimodal posteriors on the simplex $\pi_d \in \Delta^K$. But as noted by Taddy [2012], if we instead use a softmax [MacKay, 1997] representation of π_d (the natural parameters of the corresponding exponential family), the posterior is log-concave with a single mode. Elegantly, the softmax-basis MAP for a particular $\alpha < 1$ equals the simplex MAP estimate under a modified Dirichlet prior, $p(\pi_d \mid x_d, \phi, \alpha + 1)$. Using this "add one" trick, exponentiated gradient gives optimal *natural parameter* MAP estimates even when $\alpha < 1$.

Parameter Learning via SGD.

To optimize the objective in Eq. (4.10), we realize first that the iterative MAP embedding in Eq. (4.12) is *differentiable* with respect to the parameters α, ϕ and η. This means the *entire* objective is differentiable and modern gradient descent methods may be applied to learn α, ϕ, η from data, using standard transformations of constrained parameters $\{\alpha, \phi\}$ from the simplex to the reals. Once the loss function is specified via unconstrained parameters, we perform automatic differentiation to compute gradients and then optimize via the Adam algorithm [Kingma and Ba, 2014]. For scalability, we can perform stochastic updates from minibatches of data.

Previously, Chen et al. [2015] optimized a purely discriminative objective via mirror descent directly on the constrained parameters ϕ, using a C# implementation with manually-derived gradient computations. In contrast, our approach allows many useful extensions (such as multi-label binary classification) without need to derive and implement gradient calculations by hand.

Efficient gradients via implicit differentiation.

It is in fact not necessary to apply automatic differentiation to each individual operation in the exponentiated gradient optimization. Instead, we can use the fact that the MAP is a fixed point of the exponentiated gradient update:

$$\mathrm{MAP}(x_d, \alpha, \phi) = \mathrm{EG}\left(\mathrm{MAP}(x_d, \alpha, \phi),\ x_d, \alpha, \phi \right), \tag{4.13}$$

where EG is the vector-wise exponentiated gradient update defined in Eq. (4.12), such that $\pi_d^t \leftarrow \mathrm{EG}(\pi_d^{t-1}, x_d, \alpha, \phi)$, and apply the *implicit function theorem* [Krantz and Parks, 2002] to find the Jacobian of the the MAP with respect to α, ϕ : $\nabla_{\alpha,\phi}\mathrm{MAP}(x_d, \alpha, \phi)$. Letting

$\pi_d^* = \text{MAP}(x_d, \alpha, \phi)$ and differentiating both sides of Eq. (4.13) we get:

$$\nabla_{\alpha,\phi}\text{MAP}(x_d, \alpha, \phi) = \left[I - \nabla_{\pi_d^*}\text{EG}\left(\pi_d^*, x_d, \alpha, \phi\right)\right]^{-1} \nabla_{\alpha,\phi}\text{EG}\left(\pi_d^*, x_d, \alpha, \phi\right). \tag{4.14}$$

This form allows us to compute the gradients of out MAP embedding function with respect to the global parameters using only the final output of the exponentiated gradient optimization. This approach saves significant memory over applying automatic differentiation to the iterative computation which would require storing all of the intermediate values computed during optimization.

4.3.2 Toy example: PC-LDA

To study trade-offs between models of $p(x)$ and $p(y|x)$, we built a toy dataset that is deliberately *misspecified*: neither the unsupervised LDA maximum likelihood solution nor the supervised sLDA maximum likelihood solution performs better than chance at label prediction. We look at 500 training documents, each with $V = 9$ possible vocabulary words that can be arranged in a 3-by-3 grid to indicate bar-like co-occurrence structure, as illustrated in Fig. 4.2. Each binary label y_d is unrelated to the observed x_d vector except for a rare signal word (top-left corner).

Baselines.

Our discriminative baselines include logistic regression, the fully supervised BP-sLDA algorithm of Chen et al. [2015], the unsupervised Gibbs sampler for LDA [Griffiths and Steyvers, 2004] from the Mallet toolbox [McCallum, 2002], and the supervised MED-sLDA Gibbs sampler [Zhu et al., 2013] which is reported to improve on an earlier variational method [Zhu et al., 2012]. To be fair to all methods, we tune relevant hyperparameters (L_2 regulariza-

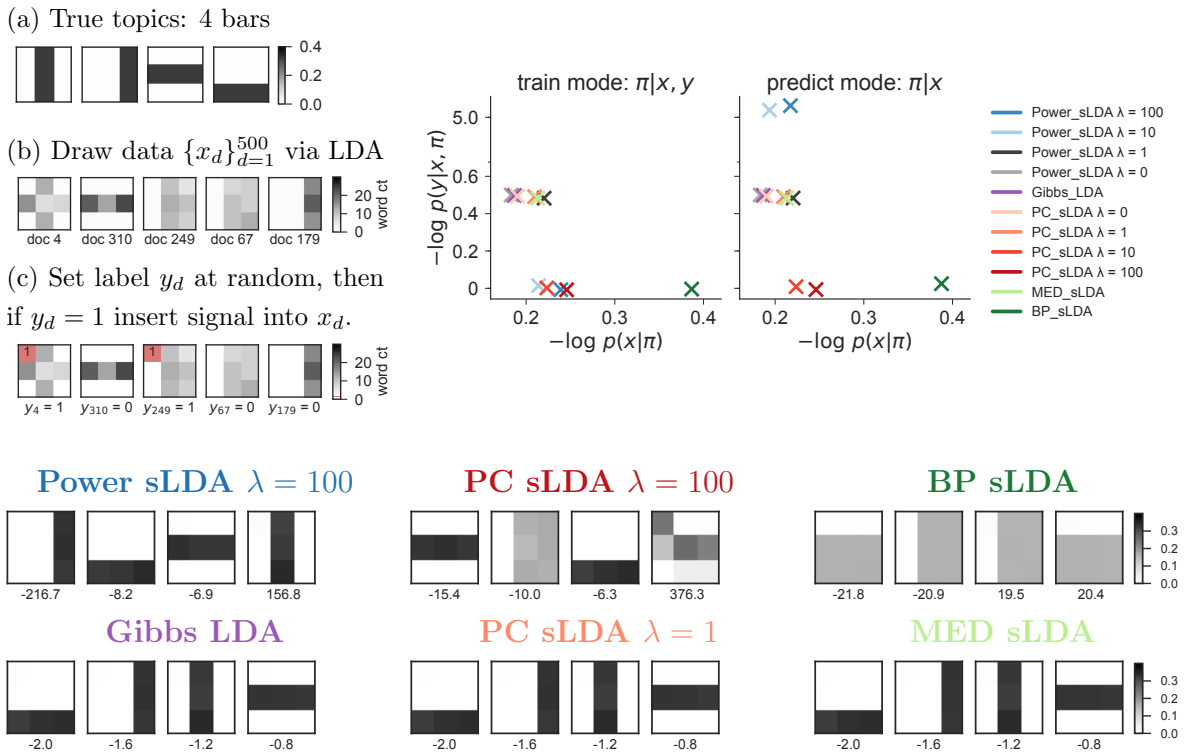


Figure 4.2: 3×3 bars task: The advantages of PC training under misspecification. Given only $K = 4$ topics, the goal is to simultaneously model the bar-like topic structure (as in Griffiths and Steyvers [2004]) of observed counts while making accurate binary label predictions using learned topic features. *Top Left:* Illustration of the true generative process for 5 example documents. Each document d has a binary label y_d and a count vector x_d over 9 possible vocabulary symbols arranged in a 3×3 square grid. To generate document d, we first draw x_d as a mixture of 4 true "bar" topics, as in LDA. Next, we draw $y_d \sim \text{Bern}(0.2)$, so it is *independent* of x_d and thus any sLDA model is misspecified. Finally, if $y_d = 1$ we set the top-left word $x_{d0} = 1$, otherwise $x_{d0} = 0$. Thus, there is a clear signal to predict y_d well given x_d but it relies on *none* of the bar topics. *Top Right:* Each method's best solution (as ranked by its training objective) is located on a 2-dimensional fitness landscape. The x-axis is negative log likelihood of data x averaged per token (lower is better). The y-axis is the negative log likelihood of labels y averaged per document (lower is better). These metrics are computed on the *training* set. We show these scores under two possible modes for estimating the document-topic vector π_d. *Train mode* finds the supervised MAP estimate $\max_{\pi_d} \log p(\pi_d|x_d, y_d, \phi, \eta, \alpha)$. *Predict mode* finds the unsupervised MAP estimate $\max_{\pi_d} \log p(\pi_d|x_d, \phi, \alpha)$. This distinction highlights the key difference between PC-sLDA with $\lambda_\epsilon > 1$, which deliberately trains topics to be good at labels-from-data prediction, and label replication (Power-sLDA with $\lambda > 1$), which trains models that do well in training mode but fail in a predictive setting (even on the same training data). *Bottom Rows:* Learned topic-word parameters for each method, labeled with regression coefficient η_k for each topic.

tion strength for regression, MED-sLDA regularization weight, step sizes, etc.) on validation data. For our toy example, we also compare to a coordinate ascent algorithm for the maximum-likelihood sLDA objective in Eq. (4.9) (Power-sLDA), across different values of the label replication factor $\lambda \geq 0$. Power-sLDA $\lambda = 0$ is equivalent to unsupervised LDA; Power-sLDA $\lambda = 1$ is the standard sLDA of McAuliffe and Blei [2008].

All baselines support documents with one binary label $y_d \in \{0, 1\}$. Third-party MED-sLDA and BP-sLDA code does not support multiple binary labels per document, but our PC-sLDA does. In these cases, we either train MED-sLDA on only one label (e.g., only wifi for the Yelp task) or omit it.

Results.

We see that our PC-sLDA with $\lambda_\epsilon \geq 10$ is the only method to find a topic with high probability on the signal word (top left corner), which is key to good discrimination. Most other methods, such as sLDA or MED-sLDA, are indistinguishable from the unsupervised LDA solution. Label replication (Power-sLDA $\lambda > 1$) suffers the most under misspecification, yielding solutions with terrible generalization performance. Purely discriminative BP-sLDA discriminates well but learns very poor generative models with no useful bar structure.

4.4 Applications: PC-LDA

4.4.1 Movie and restaurant review analysis.

We compare both the predictive and generative performance of PC-LDA training to existing methods on 2 real-world datasets of review text:

- **Movie reviews.** Each of the 4004/500/501 documents is a published movie review by a professional critic [Pang and Lee, 2005], with $V = 5338$ terms. Each review has one binary label, where $y_d = 1$ means the critic gave the film more than 2 of 4 stars.

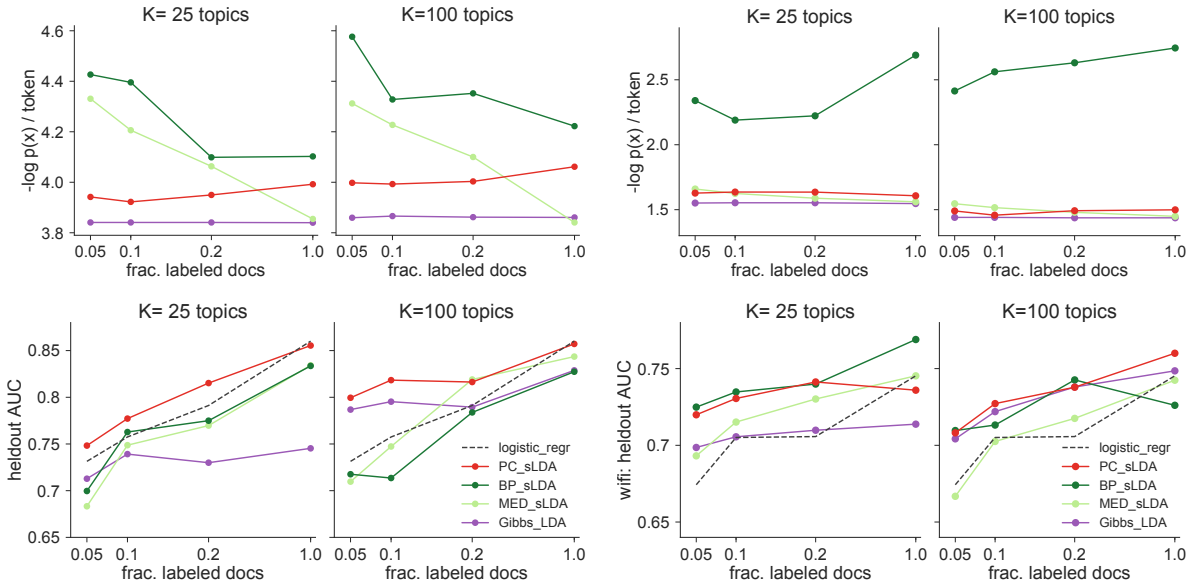
- **Yelp reviews.** Each of the 23159/2895/2895 documents [Yelp Dataset Challenge, 2016] aggregates all text reviews for a single restaurant, using $V = 10,000$ vocabulary terms. Each document also has 7 possible binary labels y_d: takes-reservations, delivery, alcohol, good-for-kids, expensive, outdoor-patio, and wifi.

In our comparisons we artificially include only a small fraction (0.05, 0.10, or 0.20) of available training labels, chosen at random. Fully supervised methods (e.g. BP-sLDA, MED-sLDA) are *only given* documents (x_d, y_d) from this subset, because third-party code does not allow using unlabeled data at training. Our PC-sLDA as well as Gibbs-LDA uses the entire partially-labeled training set.

All topic models are run from multiple random initializations of ϕ, η (for fairness, all methods use same predefined initializations of these parameters). We record point estimates of topic-word parameters ϕ and regression weights η at defined intervals throughout training. For all methods, at each parameter snapshot ϕ, η we evaluate *discriminative* prediction quality via area-under-the-ROC-curve (AUC) using the predicted probability $\Pr(y_d = 1|x_d) = \sigma(\eta^T \text{MAP}(x_d, \phi, \alpha))$. We evaluate *generative* model quality via a variational evidence lower bound on heldout per-token log likelihood: $(\sum_d N_d)^{-1} \sum_{d=1}^{D} \log p(x_d|\phi, \alpha)$. For all methods, we select the best snapshot on the validation set (early stopping) by minimizing the hueristically-chosen score:

$$-\sum_{d=1}^{D} \Big[10 * \text{AUC}(y_d, x_d, \phi, \eta) - \text{PerTokELBO}(x_d|\phi, \alpha) \Big]. \tag{4.15}$$

When datasets are fully labeled, we sensibly find that purely discriminative methods like



(a) Movie reviews (b) Yelp reviews

Figure 4.3: Movie and Yelp tasks: Performance metrics vs. fraction of labeled training documents used for 25 and 100 topics. An extended version is in the supplement. *Top row:* Heldout generative performance (negative likelihood, lower is better). *Bottom row:* Heldout discriminative performance (AUC, higher is better). Note that improvements over supervised learning algorithms, including logistic regression, are particularly large when the fraction of labeled documents is small.

logistic regression (LR) or BP-sLDA often achieve the highest AUC values. But our PC-sLDA is consistently competitive, matching LR on the Movie task in Fig. 4.3.

PC-sLDA predictions remain good when few documents have labels. For the Movie task, PC-sLDA dominates the AUC metric for small fractions of labels (0.05, 0.1), beating even LR when K=100. In this regime, unsupervised Gibbs-LDA with $K = 100$ topics has better AUC than BP-sLDA and MED-sLDA, demonstrating the value of unlabeled data for prediction. On Yelp, PC-sLDA predictions at small fractions are better than all but BP-sLDA.

The top row of Fig. 4.3 shows trends in heldout data negative log likelihood (lower is better). As expected, unsupervised Gibbs-LDA consistently achieves the best scores, because explaining data is its sole objective. MED-sLDA also does reasonably, in some cases better than PC-sLDA, but usually in these cases MED-sLDA has worse AUC than PC-sLDA.

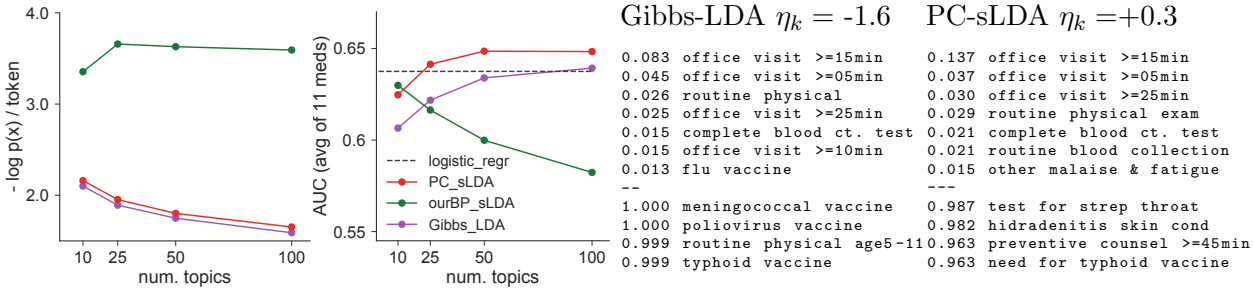


Figure 4.4: Antidepressant prediction task. *Left:* Heldout negative likelihood (generative performance, lower is better). *Center:* Heldout AUC (discriminative, higher is better). We use our own implementation of BP-sLDA for this multiple binary label prediction task. Both PC-sLDA and BP-sLDA numbers here are the results of runs initialized from Gibbs. While BP-sLDA exhibits severe overfitting (see supplement), our PC-sLDA improves on the baseline Gibbs predictions reliably. *Right:* Comparison of topic #11 of $K = 25$ for both Gibbs-LDA and our PC-sLDA when initialized from Gibbs. We show the regression coefficient η_k for this topic when predicting patient success with drug citalopram. The top row is ranked by $p(\text{word}|\text{topic})$. The bottom row is ranked by $p(\text{topic}|\text{word})$, indicating potential *anchor words*. The original Gibbs topic is mostly about routine preventative care and vaccination. PC-sLDA training evolves the topic to emphasize longer duration encounters focused on counseling or behavior change, mixed together with a few infection words.

BP-sLDA is consistently poor, having per-token likelihoods about 0.1-1.0 nats higher than others on full training sets. These results show that the solely discriminative approach of BP-sLDA cannot explain the data well. In contrast, our PC-sLDA can capture essential data properties while still predicting labels accurately.

4.4.2 Antidepressant prediction.

Finally, we predict which subset of 11 common antidepressants would successfully treat an individual's major depressive disorder given a count vector x_d of the patient's electronic health record (EHR) code history. These are real de-identified data from tertiary care hospitals, split into 29774/3721/3722 documents (one per patient) with $V = 5126$ codewords which represent past diagnoses (ICD-9), procedures (CPT), and medications. Fig. 4.4 shows the results of our comparison.

PC-LDA's learned topic-word probabilities ϕ are interpretable for prediction. A key point of our work is that our PC training estimates topic-word parameters ϕ to focus more on the label prediction than unsupervised training would. On the Antidepressant task, Fig. 4.4 shows that PC-sLDA initialized from Gibbs indeed causes an original Gibbs topic to significantly evolve its regression weight η_k and associated top words. The original Gibbs topic is mostly about routine outpatient preventative care and vaccination. The evolved PC-sLDA topic prefers long-duration primary care encounters focused on behavior change ("counseling"). With clinical collaborators, we hypothesize that this more focused topic leads to a positive η_k value because the drug citalopram is often a treatment of choice for such patients (i.e., uncomplicated MDD diagnosed and treated in primary care).

Chapter 5

Prediction-Constrained Variational Autoencoders

5.1 Variational Autoencoders

The *variational autoencoder* (VAE) [Kingma and Welling, 2014] is a *deep* latent variable model that generates data via a neural-network parameterized likelihood together with a Gaussian latent variable, $h_d = \{z_d\}$. In its most basic form the generative process can be summarized as:

$$\begin{aligned} z_d &\sim \mathcal{N}(0, I) \\ x_d &\sim \mathcal{N}(\mu(z_d, \theta),\ \sigma(z_d, \theta)). \end{aligned} \tag{5.1}$$

Here $\mu(\cdot)$ and $\sigma(\cdot)$ are parameterized functions that take in the value of the hidden variable and a set of global parameters defining the function $\xi^x = \{\theta\}$, and output the parameters (mean and variance) of a Gaussian distribution for the observation, x_d. In the case where $\theta = \{W, \sigma\}$, $\mu(z_d, \theta) = W z_d$, and $\sigma(z_d, \theta) = \sigma I$, this model is equivalent to probabilistic PCA

[Tipping and Bishop, 1999].

Notation.

For the rest of this work we will adopt the shorthand $f_\theta(\cdot)$ for a parameterized function, such as a neural network, with parameters θ, such that $f_\theta(x) \triangleq f(x, \theta)$. In general, functions of this form are assumed to be non-linear neural network functions. In some cases we adopt a further shorthand for distributions, $p_\theta(x \mid z)$, to refer to a distribution where the parameters are defined by a parameterized function of z, such that $p_\theta(x \mid z) \triangleq p(x \mid f_\theta(z))$. We use both forms interchangeably as needed for clarity. Distributions derived from $p_\theta(x|z)$ may similarly adopt this notation: $p_\theta(x) \triangleq \int_z p(x \mid f_\theta(z))p(z)dz$ and $p_\theta(z|x) \triangleq \frac{p(x|f_\theta(z))p(z)}{\int_z p(x|f_\theta(z))p(z)dz}$.

5.1.1 Generalized likelihoods.

We note that the Gaussian likelihood for x_d can be replaced with a generic parameterized distribution $\mathcal{F}(\cdot)$ so long as it can be easily evaluated and differentiated. In this case $\mu_\theta(\cdot)$ and $\sigma_\theta(\cdot)$ would be replaced by an parameterized function $f_\theta(\cdot)$ that outputs appropriate parameters for $\mathcal{F}$ given z_d and θ. We refer to this as the *decoder* network. In our experiments we consider both the Gaussian case and alternatives such as the continuous Bernoulli distribution [Loaiza-Ganem and Cunningham, 2019] and discretized mixture of logistics [Salimans et al., 2017]. We also note that the our assumption of a fixed Gaussian prior with no learneable parameters ($\xi^h = \emptyset$, in our general LVM notation), can similarly be replaced with a more expressive form for $p(z_d \mid \xi^h)$, but we do not explore this case.

For a VAE with likelihood family $\mathcal{F}$, the joint log likelihood factorizes as:

$$\begin{aligned} \log p(z_d) &= \log \mathcal{N}(z_d \mid 0, I), \\ \log p_\theta(x_d \mid z_d) &= \log \mathcal{F}\left(x_d \mid f_\theta\left(z_d\right)\right) \end{aligned} \tag{5.2}$$

5.1.2 Variational inference for VAEs

For models with non-linear decoder $[f_\theta(\cdot)]$, marginalizing over z_d is typically intractable, thus directly optimizing the marginal likelihood $\log p_\theta(x_d)$ is impossible. Instead we consider a lower bound on the log marginal likelihood via an approximate posterior $q(z_d \mid \nu_d) \approx p_\theta(z_d \mid x_d)$, with parameters ν_d. This bound can be written as:

$$\begin{aligned} \log p_\theta(x_d) &= \log \int_{z_d} p_\theta(x_d, z_d) dz_d \\ &\geq \mathbb{E}_{q(z_d|\nu_d)}\left[\log \frac{p_\theta(x_d, z_d)}{q(z_d \mid \nu_d)}\right] \\ &= \mathbb{E}_{q(z_d|\nu_d)}\Bigg[\log p_\theta(x_d|z_d)\Bigg] - KL\Bigg(q(z_d|\nu_d) \;\Bigg|\Bigg|\; p(z_d)\Bigg), \end{aligned} \tag{5.3}$$

where $KL(\cdot||\cdot)$ is the Kullback-Leibler divergence. This is known as the *evidence lower bound objective* or ELBO. We use $\mathcal{L}(x_d, \theta, \nu_d)$ to denote our optimization objective which is the negative ELBO. To fit the global parameters of our model θ we jointly optimize negative ELBO over the global parameters θ and local variational parameters ν.

$$\min_{\theta,\nu} \mathcal{L}(x, \theta, \nu) = \sum_{d=1}^{D} \mathbb{E}_{q(z_d|\nu_d)}\Bigg[- \log p_\theta(x_d \mid z_d)\Bigg] + KL\Bigg(q(z_d \mid \nu_d) \;\Bigg|\Bigg|\; p(z_d)\Bigg) \tag{5.4}$$

As discussed in section 2.4.2, we can also write this in terms of an inner optimization function $\hat{\nu}(x_d, \theta)$ that produces the optimal variational parameters for a given input and global

parameter:

$$\hat{\nu}(x_d, \theta) = \underset{\nu_d}{\operatorname{argmin}}\ \mathcal{L}(x_d, \theta, \nu_d). \tag{5.5}$$

In this setting, we can rewrite our objective as:

$$\min_{\theta} \sum_{d=1}^{D} \mathbb{E}_{q(z_d|\hat{\nu}(x_d,\theta))}\Big[-\log p_\theta(x_d \mid z_d)\Big] + KL\Big(q\big(z_d \mid \hat{\nu}(x_d, \theta)\big)\ \Big\|\ p(z_d)\Big). \tag{5.6}$$

As discussed previously, this lets us distinguish between prediction-constrained and posterior-regularized objectives.

While $q(z_d \mid \nu_d)$ can be any distribution with the same support as $p(z_d)$, in practice we match the form of the prior, using a Gaussian: $q(z_d \mid \nu_d) = \mathcal{N}(z_d \mid \mu_{\nu_d}, \sigma_{\nu_d})$, where $\nu_d = \{\mu_{\nu_d}, \sigma_{\nu_d}\}$.

5.1.3 Amortized inference for VAEs.

Optimizing individual variational parameters for each datum x_d can be prohibitively expensive in both computational and memory costs. To circumvent this issue the original authors [Kingma and Welling, 2014] propose an *amortized inference* approach. In this framework, observation-specific parameters, ν_d, are replaced with a parameterized function $\nu_\phi(x_d)$ that predicts the optimal variational parameters given the input. The global inference network parameters ϕ can be jointly trained with the generative parameters θ using the corresponding objective:

$$\min_{\theta,\phi}\ \mathcal{L}(x, \theta, \phi) = \sum_{d=1}^{D} \mathbb{E}_{q_\phi(z_d|x_d)}\Big[-\log p\big(x_d \mid f_\theta(z_d)\big)\Big] + KL\Big(q_\phi\big(z_d \mid x_d\big)\ \Big\|\ p(z_d)\Big). \tag{5.7}$$

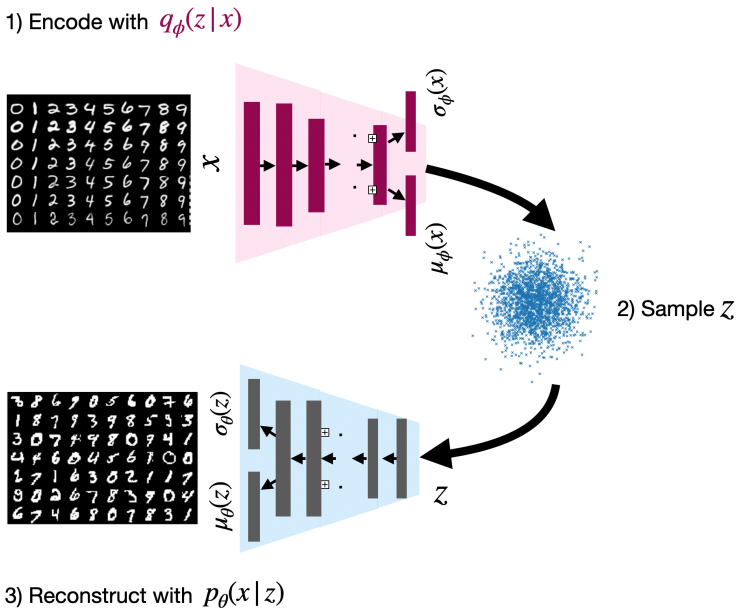


Figure 5.1: Computational flow diagram of a variational autoencoder.

Consistent with our notation, $q_\phi(z_d|x_d) \triangleq q(z_d|\nu_\phi(x_d))$. In effect, these networks are trained to approximate the inner optimization function $\hat{\nu}(\cdot)$:

$$\nu_\phi(x_d) \approx \underset{\nu_d}{\text{argmin}}\ \mathcal{L}(x_d, \theta, \nu_d), \quad \text{for} \quad x_d \in \{x_1 ... x_D\} \tag{5.8}$$

5.1.4 Hierarchical VAEs

Hierarchical variational autoencoders (HVAEs, Sønderby et al. [2016]) extend variational autoencoders by partitioning the latent code into L disjoint groups $z_d = (z_{d0}, z_{d1}, ...z_{dL})$, increasing the expressiveness of the generative model for complex data like images [Vahdat and Kautz, 2020, Child, 2021]. HVAEs generate these stochastic codes sequentially as $p_\theta(x_d|z_d) = p_\theta(z_{d0})(\prod_{\ell=1}^{L} p_\theta(z_{d\ell} \mid z_{d,<\ell}))p_\theta(x_d \mid z_{dL})$, with a similar encoder: $q_\phi(z_d|x_d) = q_\phi(z_{d0} \mid x_d)\prod_{\ell=1}^{L} q_\phi(z_{d\ell} \mid z_{d<\ell}, x_d)$. Each conditional in the decoder $p_\theta(z_{d\ell}|z_{d<\ell})$, and the

encoder $q_\phi(z_{d\ell} \mid z_{d,<\ell}, x)$, is typically Gaussian with mean and variance determined by (non-linear) neural networks.

The HVAE ELBO [Vahdat and Kautz, 2020] equals:

$$\mathcal{L}_H(x_d, \theta, \phi) = E_{q_\phi(z_d|x_d)}\Big[\log p_\theta(x_d \mid z_d)\Big] - \mathrm{KL}\Big(q_\phi(z_{d0} \mid x_d) \mid\mid p_\theta(z_{d1})\Big) \tag{5.9}$$

$$- \sum_{\ell=1}^{L} E_{q_\phi(z_{d,<\ell}|x)}\Big[\mathrm{KL}\Big(q_\phi(z_{d\ell} \mid z_{d,<\ell}, x) \mid\mid p_\theta(z_\ell \mid z_{d,<\ell})\Big)\Big], \tag{5.10}$$

where $q_\phi(z_{d,<\ell} \mid x_d) = \prod_{i=1}^{\ell-1} q(z_{di} \mid z_{d,<i}, x_d)$ is the approximate posterior up to latent group $(\ell - 1)$. Reparameterization is then used to provide Monte Carlo gradient estimates.

We can rewrite the conditional prior and approximate posterior for layer ℓ to make the set of relevant networks explicit:

$$p_\theta(z_{d\ell} \mid z_{d,<\ell}) = \mathcal{N}\big(z_{d\ell} \mid \ \mu_{\theta_\ell}(z_{d,<\ell}),\ \sigma_{\theta_\ell}(z_{d,<\ell})\big), \tag{5.11}$$

$$q_\phi(z_{d\ell} \mid z_{d,<\ell}, x_d) = \mathcal{N}\big(z_{d\ell} \mid \mu_{\phi_\ell}(h_{\phi_\ell}(x_d), g_{\phi_\ell}(z_{d,<\ell})\big),\ \sigma_{\phi_\ell}(...)).$$

Here, h_{ϕ_ℓ} and g_{ϕ_ℓ} are networks that extract feature representations of the observation x_d and the previous layers $z_{d,<\ell}$, respectively. These features determine the mean and scale of the conditional Gaussian posterior via μ_{ϕ_ℓ}, σ_{ϕ_ℓ}. Networks μ_{θ_ℓ}, σ_{θ_ℓ} similarly generate the prior parameters for layer ℓ.

5.2 Prediction-constrained training of VAEs

For prediction-constrained training of supervised VAEs, we once again focus on training linear predictors with a probabilistic loss:

$$\hat{y}(z_d, \xi^y) \triangleq \hat{g}(\eta^T z_d), \quad \xi^y = \{\eta\}. \tag{5.12}$$

$$\ell(y_d, \hat{y}(z_d, \eta)) = -\log p(y_d \;\; | \;\; \hat{y}(z_d, \eta)). \tag{5.13}$$

Here again ξ^y are the global parameters of the prediction model which, for the linear case, take the form of a vector of regression weights η. $\ell(\cdot)$ is the predictions loss, which we assume is probabilistic. $\hat{g}(\cdot)$ is a link function that transforms $\eta^T z_d$ into constrained parameters appropriate for the form of $p(y_d| \;\; \cdot \;\;)$. We note again that the framework could easily be applied to non-probabilistic losses and non-linear prediction functions. Our prediction-constrained objective for supervised VAEs is therefore:

$$\begin{aligned} \min_{\theta,\eta} \;\; & -\left[\sum_{d=1}^{D} \log p_\theta(x_d)\right] + R(\theta, \eta) \\ \text{subject to} \quad & -\sum_{d=1}^{D} \log p_\theta(y_d \mid x_d, \eta) \leq \epsilon. \end{aligned} \tag{5.14}$$

The corresponding unconstrained objective is:

$$\min_{\theta,\eta} -\sum_{d=1}^{D} \left[\log p_\theta\big(x_d\big) + \lambda_\epsilon \log p_\theta\big(y_d \mid x_d, \eta\big)\right] + R(\theta, \eta). \tag{5.15}$$

As with prediction-constrained LDA, the necessary marginalization over z_d is intractable. We instead consider the form from Eq. (2.31), using the inner optimiation defined in Eq.

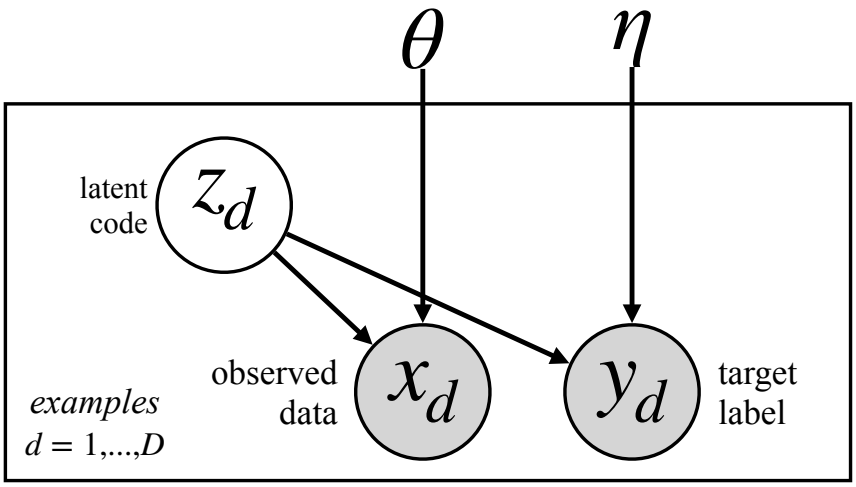


Figure 5.2: Graphical model representation of downstream supervised variational autoencoders.

(5.5) for inference:

$$\min_{\theta,\eta} -\sum_{d=1}^{D} \mathbb{E}_{q(z_d|\hat{\nu}(x_d,\theta))}\Big[\log p_\theta\big(x_d, z_d\big) + \lambda_\epsilon \log p\big(y_d \mid z_d, \eta\big)\Big] + R(\theta,\eta), \tag{5.16}$$

In practice, the inner optimization is extremely expensive to compute for VAEs, necessitating the application of amortized variational inference for many applications. With amortization, the prediction-constrained VAE objective becomes:

$$\begin{aligned}\min_{\theta,\eta,\phi} -\sum_{d=1}^{D} \mathbb{E}_{q_\phi(z_d|x_d)}\Big[\log p_\theta(x_d \mid z_d) + \lambda_\epsilon \log p(y_d \mid z_d, \eta)\Big] \\ + KL\Big(q_\phi\big(z_d \mid x_d\big) \,\Big\|\, p(z_d)\Big) + R(\theta,\eta).\end{aligned} \tag{5.17}$$

This objective does not perfectly capture asymmetric properties of the original prediction-constrained objective, as the training of ϕ is informed by y, however the posterior approximation network still only takes x_d explicitly as input. Subsequent sections will introduce methods to address the effects of this limitation.

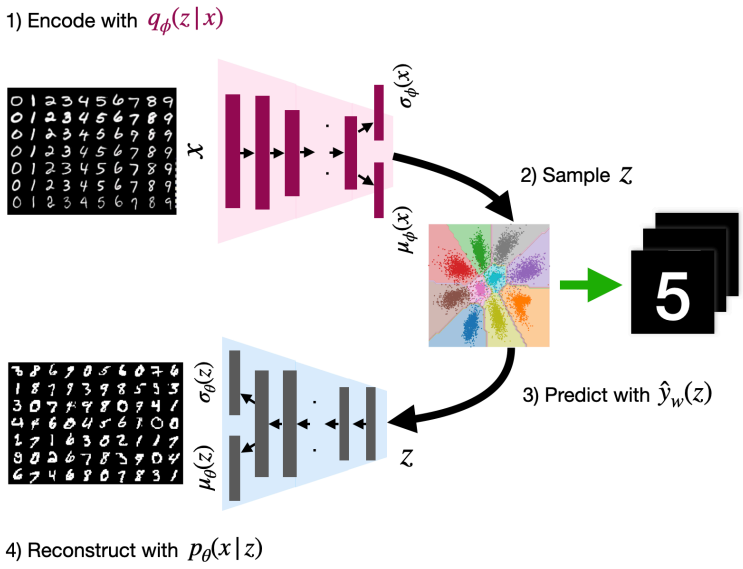


Figure 5.3: Computational flow diagram of a prediction-constrained variational autoencoder.

Parameter learning with SGD

As with other models, we learn the parameters θ, η, ϕ of the prediction-constrained VAE by using stochastic gradient descent to optimize the objective in Eq. (5.17). We also note that we can optimize Eq. (5.16) with SGD, using implicit differentiation (as with LDA) to save computational resources.

5.2.1 Semi-supervised learning with prediction-constrained VAEs

As with other prediction-constrained models we can train the prediction-constrained VAE on partially-labeled data using the modification of the objective in Eq. (2.9) and likewise optimizing with stochastic gradient descent.

Several prior works in semi-supervised deep learning [Kingma et al., 2014b, Oliver et al., 2018] have employed a balanced stochastic gradient optimization approach where training

batches are selected to have equal numbers of labeled and unlabeled examples. This prevents instances where batches have no labeled examples and reduces the variance of stochastic training. We can implement a similar scheme for the prediction-constrained VAE without changing the expectation of the objective.

Starting with the generic unconstrained PC-VAE objective for semi-supervised learning,

$$\sum_{d:\mathcal{D}^U \cup \mathcal{D}^S} \mathcal{L}\big(x_d, \theta, \phi\big) + \sum_{d:\mathcal{D}^S} \lambda_\epsilon \mathbb{E}_{q_\phi(z_d|x_d)}\Big[\ell\big(y_d, \hat{y}(z_d, \eta)\big)\Big], \tag{5.18}$$

we can rewrite an equivalent objective, separating out the supervised and unsupervised terms:

$$\frac{|\mathcal{D}^S|}{|\mathcal{D}^S|}\sum_{d:\mathcal{D}^S}\Big[\mathcal{L}\big(x_d, \theta, \phi\big) + \lambda_\epsilon \mathbb{E}_{q_\phi(z_d|x_d)}\big[\ell\big(y_d, \hat{y}(z_d, \eta)\big)\big]\Big] + \frac{|\mathcal{D}^U|}{|\mathcal{D}^U|}\sum_{d:\mathcal{D}^U}\mathcal{L}\big(x_d, \theta, \phi\big). \tag{5.19}$$

Rewriting in terms of expectations over the labeled and unlabeled datasets, leads to a natural approach for optimizing via balanaced batches:

$$|\mathcal{D}^S|\ \mathbb{E}_{d:\mathcal{D}^S}\Big[\mathcal{L}\big(x_d, \theta, \phi\big) + \lambda_\epsilon \mathbb{E}_{q_\phi(z_d|x_d)}\big[\ell\big(y_d, \hat{y}(z_d, \eta)\big)\big]\Big] + |\mathcal{D}^U|\ \mathbb{E}_{d:\mathcal{D}^U}\Big[\mathcal{L}\big(x_d, \theta, \phi\big)\Big]. \tag{5.20}$$

We see that in our stochastic optimization, we can get an unbiased estimate of the objective and its gradient by sampling labeled and unlabeled batches separately. With batch size B for both labeled and unlabeled batches our estimated objective becomes:

$$\frac{|\mathcal{D}^S|}{B}\sum_{d:B^S}\Big[\mathcal{L}\big(x_d, \theta, \phi\big) + \lambda_\epsilon \mathbb{E}_{q_\phi(z_d|x_d)}\big[\ell\big(y_d, \hat{y}(z_d, \eta)\big)\big]\Big] + \frac{|\mathcal{D}^U|}{B}\sum_{d:B^U}\Big[\mathcal{L}(x_d, \theta, \phi)\Big] \tag{5.21}$$

$$B^S \sim \mathcal{D}^S,\ B^U \sim \mathcal{D}^U$$

We may also normalize by an additional scale factor of $\frac{1}{|\mathcal{D}^S|+|\mathcal{D}^U|}$ to approximately remove dependence on the dataset size.

5.3 Comparisons to Prior Work of Semi-supervised VAEs

5.3.1 Advantages over two-stage VAEs for semi-supervised learning

VAEs have previously been used for SSL via a *two-stage* "VAE + GLM" strategy [Kingma et al., 2014b]. First, a VAE is trained to maximize the unsupervised likelihood in Eq. (5.3) of *all* features x (both labeled $\mathcal{D}^S$ and unlabeled $\mathcal{D}^U$). Second, fixing ϕ and using only $\mathcal{D}^S$, a label-from-code *predictor* $\hat{y}(z_d, \eta)$ is learned that maps latent codes z_d to prediction scores. Our experiments use a *generalized linear model* (GLM) with weights η trained to minimize the cross-entropy loss, $\sum_{d:\mathcal{D}^S} \mathbb{E}_{q_\phi(z_d|x_d)} [\ell_S(y_d, \hat{y}(z_d, \eta))]$. While "VAE + GLM" is a common baseline, labels are only used in the second stage, and thus *misspecified* generative models (including deep neural network models of images or text) often lead to inaccurate predictions.

5.3.2 Advantages over label-conditioned VAEs

Motivated by limitations of two-stage SSL, Kingma et al. [2014b] proposed a VAE-inspired "M2" model for *joint* generative modeling of categorical labels y and data x. M2 first generates labels y with frequencies π, and then features x: $p_\theta(x_d, y_d, z_d) = \mathcal{N}(z_d \mid 0, I) \cdot \text{Cat}(y_d \mid \pi) \cdot \mathcal{F}(x_d \mid f_\theta(y_d, z_d))$. M2 inference sets $q_\phi(y_d, z_d \mid x_d) = q_{\phi^{y|x}}(y_d \mid x_d) q_{\phi^{z|x,y}}(z_d \mid x_d, y_d)$, where $\phi = \{\phi^{y|x}, \phi^{z|x,y}\}$.

Kingma et al. [2014b] train M2 to minimize the joint objective:

$$\min_{\theta,\phi^{y|x},\phi^{z|x,y}} \quad \sum_{d:\mathcal{D}^S} \mathcal{L}^S(x_d, y_d, \theta, \phi^{z|x,y}) + \sum_{d:\mathcal{D}^U} \mathcal{L}^U(x_d, \theta, \phi^{y|x}, \phi^{z|x,y}). \tag{5.22}$$

Like unsupervised VAEs, Eq. (5.22) and its gradients may be approximated via samples from the variational posterior. The first, "supervised" term bounds the feature-and-label *joint* likelihood:

$$\log p_\theta(x_d, y_d) \geq \mathcal{L}^S(x_d, y_d, \theta, \phi^{z|x,y}) = \mathbb{E}_{q_{\phi^{z|x,y}}(z_d|x_d,y_d)}\left[\log \frac{p_\theta(x_d,y_d,z_d)}{q_{\phi^{z|x,y}}(z_d|x_d,y_d)}\right].$$

The second, "unsupervised" term is a variational bound for the features-only likelihood $\log p_\theta(x_d)$:

$$\begin{aligned}\mathcal{L}^U(x, \theta, \phi) &= \mathbb{E}_{q_\phi(y_d,z_d|x_d)}\left[\log \frac{p_\theta(x_d, y_d, z_d)}{q_\phi(y_d, z_d|x_d)}\right] \\ &= \sum_{y\in\mathcal{Y}} q_{\phi^{y|x}}(y_d \mid x_d)\big(\mathcal{L}^S(x_d, y_d, \theta, \phi^{z|x,y}) - \log q_{\phi^{y|x}}(y_d \mid x_d)\big).\end{aligned} \tag{5.23}$$

Here $\mathcal{Y}$ is the set of possible categorical labels.

Several recent SSL methods adapt the same generative and inference networks as M2 [Li et al., 2018a, Feng et al., 2021, Gordon and Hernández-Lobato, 2020]. The multi-layer ADGM and SDGM [Maaløe et al., 2016] extend this structure with an auxiliary latent variable.

All of these models are *direct label-conditioned* (DLC): x depends explicitly on y in the generative network, and y depends directly on x in the inference network. Let notation $q_\phi(y|x, \cdot)$ generically refer to the model-dependent variational factor used for to predict labels y by DLC models.

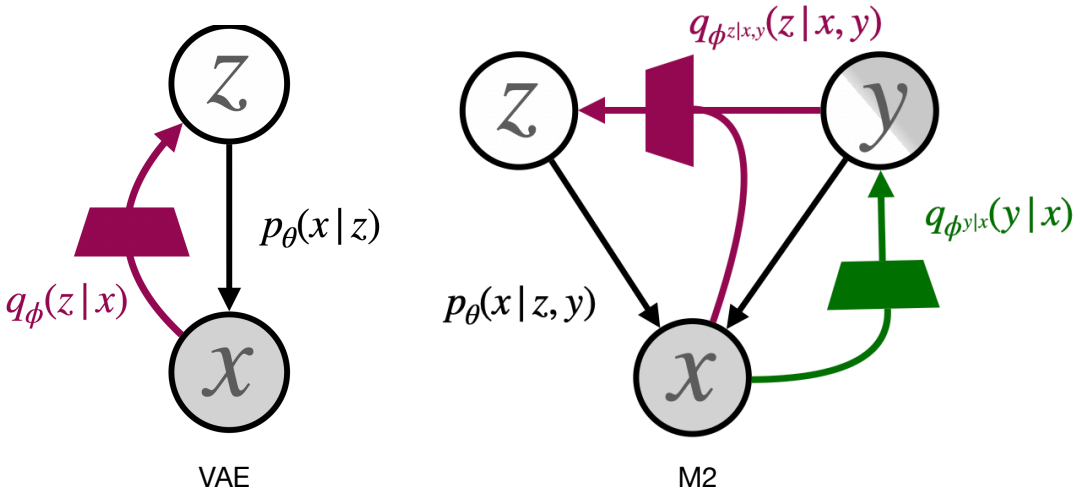


Figure 5.4: Illustraion of the "M2" model. *Left:* A standard VAE, highlighting the inference model used. *Right:* The M2 model of Kingma et al. [2014a], showing both inference networks used.

DLC models require label marginalization.

When labels are unobserved, computing the DLC variational bound requires an expensive marginalization over all $|\mathcal{Y}|$ possible class labels in Eq. (5.23). Experiments show that the practical runtime of DLC models is roughly $|\mathcal{Y}|/2$ times longer than our proposed approach. Futhermore the runtime of this marginalization is *exponential* the the number of unobserved labels associated with each observation. Some prior work has resorted to biased gradient estimates due to this cost; Li et al. [2018a] approximate y by point estimate; Feng et al. [2021] sample y via a continuous relaxation of $q_\phi(y_d|x_d, \cdot)$.

The DLC prediction dilemma and heuristic fix.

For test data x, DLC models predict labels via the inference model's discriminator $q_{\phi^{y|x,\cdot}}(y_d \mid x_d, \cdot)$. However, the discriminator's parameters $\phi^{y|x,\cdot}$ are only informed by the *unlabeled* data

via the objective $\mathcal{L}^U$ of Eq. (5.23); it is *not* used to compute $\mathcal{L}^S$. We cannot expect accurate predictions from a parameter that does not touch *any* labeled training data.

To partially address this issue, Kingma et al. [2014b] and other DLC methods use a weighted objective:

$$\max_{\theta,\phi} \sum_{d:\mathcal{D}^S} \left(\alpha \log q_{\phi^{y|x}}(y_d \mid x_d) + \lambda \mathcal{L}^S(x_d, y_d, \theta, \phi^{z|x,y})\right) + \sum_{d:\mathcal{D}^U} \mathcal{L}^U\left(x_d, \theta, \phi^{y|x}, \phi^{z|x,y}\right). \tag{5.24}$$

This objective pushes the inference discriminator $q_{\phi^{y|x}}(y_d|x_d)$ to predict training labels via an extra loss term, weighted by hyperparameter $\alpha > 0$. Kingma et al. [2014b] fixed $\lambda = 1$ and tuned α using validation data; Siddharth et al. [2017] tuned λ to balance supervised and unsupervised data.

While the $\mathcal{L}^S$ and $\mathcal{L}^U$ terms in Eq. (5.24) have a rigorous justification as maximizing the data likelihood, the first term ($\alpha \log q(y_d \mid x_d)$) is not justified by the generative model; it is an extra regularizer for the inference network [Shu et al., 2018]. Problematically, when the training data is fully labeled, we remove $\mathcal{L}^U$ and the discriminator is trained only via this additional loss. In DLC models without auxiliary variables, this implies a complete *decoupling* of generator parameters $\theta, \phi^{z|x,y,\cdot}$ from discriminator parameters $\phi^{y|x,\cdot}$ during supervised training. This is deeply unsatisfying: we want a single model informed by *both* generative and discriminative goals, not two separate models.

DLC models are fragile.

Intuitively, DLC models can improve semi-supervised classification over purely discriminative baselines *only if* improvements to the discriminative variational factor $q_\phi(y_d \mid x_d, \cdot)$ are necessary to optimize the reconstruction loss $\mathbb{E}_{q_\phi(y,z,\cdot|x)}[\log p_\theta(x|z,y,\cdot)]$. However, as network ($\theta$, ϕ) or code (z) capacity increases, DLC models can improve reconstruction while functionally

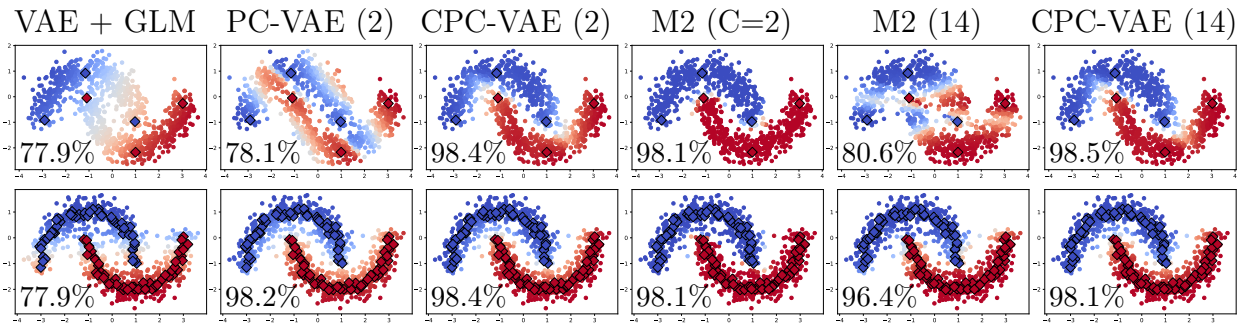


Figure 5.5: Predictions on half-moon classification (accuracy in corner) for semi-supervised VAE learning. Dots are 2-dim. feature vectors colored by predicted probability of mostly likely label, labeled examples are shown as larger diamonds. Titles indicate encoding size $C = 2$ or $C = 14$. M2 [Kingma et al., 2014b] accuracy *deterioriates* when capacity increases from $C = 2$ to 14 (drop from 98.1% to 80.6% accuracy). Our CPC-VAE is reliable at any capacity via constraints that ensure prediction quality. *Top row:* Learning from 6 labeled examples (diamonds) and 994 unlabeled examples. *Bottom row:* Learning from 100 labeled examples and 900 unlabeled examples.

decoupling generative and discriminative learning, such that $p_\theta(x_d \mid z_d, y_d, \cdot) \approx p_\theta(x_d \mid z_d, \cdot)$ and $q_\phi(z_d|x_d, y_d, \cdot) \approx q_\phi(z_d|x_d, \cdot)$. This is specifically encouraged because y can provide only a few bits of information, and because $q_\phi(y \mid x, \cdot)$ must satisfy the additional heuristic prediction loss.

Fig. 5.5 provides an intuitive example where, even with tuning of the prediction weight α in Eq. (5.24), increased latent code capacity causes DLC predictions to deteriorate. We show similar fragility for DLC models of real images in Table 5.3. This instability is a serious obstacle to creating DLC variants of state-of-the-art "very deep" VAEs [Sønderby et al., 2016, Child, 2021, Vahdat and Kautz, 2020], which have orders-of-magnitude more latent variables than observations.

5.3.3 Other related work on SSL of VAEs

Gordon and Hernández-Lobato [2020] indirectly couple discriminative and generative VAEs via a joint prior. Such "parameter coupling" [Lasserre et al., 2006] still requires expensive

label marginalization. Our experiments show directly integrating generative parameters in predictions improves accuracy.

Other direct label-conditioned models.

Li et al. [2018a] propose MMCVA, a variant of M2 whose discriminator is trained via *max-margin* principles, augmented with several additional losses from the SSL literature. They do not analyze the relative importance of the various loss terms. The SHOT-VAE [Feng et al., 2021] is a DLC model that seeks to better justify the discriminative loss in Eq. (5.24) by applying label smoothing [Szegedy et al., 2016], inducing a "smoothed" variational objective containing a KL-divergence incorporating $\log q_\phi(y \mid x)$. The experiments of Feng et al. suggest that the SSL accuracy of SHOT-VAE is primarily due to their addition of a variant of Mixup data augmentation [Zhang et al., 2017]. The biases induced by this augmentation significantly hurt generative performance. Maaløe et al. [2019] introduce BIVA, a very deep VAE aimed at generative quality. SSL is not the focus of the paper; only a single MNIST result is shown via a DLC architecture, and no SSL code is available.

Indirect label-conditioned models.

Some joint likelihood-based methods for SSL use *indirect label-conditioned* (ILC) models where y is an ancestor, but *not* a direct parent, of x in the generative process. The M1+M2 model [Kingma et al., 2014b] is a greedily trained ILC model, where a pre-trained unsupervised VAE encodes features for a separate M2 model. Kingma et al. found that end-to-end training of M1+M2 was ineffective. The *Characteristic-Capturing VAE* (CCVAE, [Joy et al., 2021]) optimizes a bound on $p(x_d, y_d, z_d) = p(y_d)p(z_d|y_d)p(x_d|z_d)$. The CCVAE has improved class-specific generative performance, but Joy et al. acknowledge that it does not significantly improve SSL accuracy. Ilse et al. [2020] propose DIVA, a similar ILC model that

(a) Unsupervised VAE (b) CPC-VAE (c) M2

Figure 5.6: Class-conditional samples of the 10 possible digit classes in the MNIST dataset. Each column shows multiple samples from one specific digit class. From left to right, each panel shows samples from a standard unsupervised VAE, our CPC-VAE, and model M2 [Kingma et al., 2014b]. All models use a 2-dimensional latent code, and are trained on the MNIST dataset with 100 labeled examples (10 per class).

captures shifts in the data domain via multiple independent latent spaces. The SeGMA model [Smieja et al., 2021] maps classes to Gaussian mixture components in the latent code space, and adapts Wasserstein autoencoders to simplify training with mixtures.

5.4 Class-conditional Generation with PC-VAEs

A standard VAE generates data by sampling $z_s \sim \mathcal{N}(0, I)$, and then sampling $x_s \sim \mathcal{F}(f_\theta(z_s))$. For the PC-VAE, we can further sample images conditioned on a particular class label. We consider several ways to accomplish this. In all cases we consider how to sample z_d given a label of a particular discrete class, $y_d = c$.

5.4.1 MCMC sampling.

For cases where our prediction-constraint is probabilistic and we have an explicit model $p(y_d|z_d, \eta)$, we can tractably compute the posterior density $p(z_d|y_d = c, \eta)$ up to a constant:

$$p(z_d \mid y_d = c, \eta) \propto p(y_d = c, z_d \mid \eta) = p(y_d = c \mid z_d, \eta)p(z_d) \tag{5.25}$$

The gradient of the log-posterior with respect to z_d is therefore:

$$\nabla_{z_d} \log p(z_d \mid y_d = c, \eta) = \nabla_{z_d} \log p(y_d = c \mid z_d, \eta) + \nabla_{z_d} \log p(z_d). \tag{5.26}$$

Gradient-based MCMC methods such as Hamiltonian Monte Carlo [Neal et al., 2011] can therefore be applied to sample from the posterior efficiently. Unfortunately, this approach does not account for our asymmetric goals; it may be desirable to sample observations that would be *confidently* predicted to be a certain class.

5.4.2 Rejection sampling.

A simple, general method to sample class-conditional observations is to use rejection sampling. Our rejection sampler works by repeatedly sampling $z_s \sim \mathcal{N}(0, I)$ until a sample meets the criteria: $p(y_s = c \mid z_c, \eta) > 1 - \epsilon_s$, for some target threshold ϵ_s. We typically use $\epsilon_s = 0.05$ in our experiments (see fig. 5.18).

5.4.3 Model-based sampling.

An alternative approach is to fit a post-hoc model of $p(z|y)$, using the trained VAE and classifer. We accomplish this by fitting a multivariate Normal distribution for each class, such that

$$p(z_s|y = c) = \mathcal{N}(\mu_c, \Sigma_c) \tag{5.27}$$

Figure 5.7: Class-conditional samples of the 10 possible digit classes in the SVHN dataset. The generative model was trained on the fully labeled SVHN dataset with prediction and consistency constraints. Samples were chosen via rejection sampling in the latent space with a threshold of 95% confidence in the target class.

The mean and covariance of each class-conditional distribution are estimated from the set of training examples predicted as that class, so

$$\mu_c = \frac{1}{|\mathcal{D}^c|} \sum_{d:\mathcal{D}^c} z_d, \tag{5.28}$$

$$\Sigma_c = \frac{1}{|\mathcal{D}^c| - 1} \sum_{d:\mathcal{D}^c} \big(z_d - \mu_c\big)\big(z_d - \mu_c\big)^T,$$

$$\mathcal{D}^c := \left\{ z_d \;\middle|\; \operatorname{argmax}\Big(\hat{y}\big(z_d, \eta\big)\Big) = c,\ z_d \sim q_{\phi^{z|x}}(z_d \mid x_d),\ x_d \in \mathcal{D}^U \cup \mathcal{D}^S, \right\}.$$

With this model, class conditional sampling is trivial. We utilize this approach for the results in table 5.1.

5.4.4 Results of Class-conditional Generation.

Fig. 5.10 shows 2-dimensional latent space encodings of the MNIST dataset [LeCun et al., 2010] using several different models. We provide a complementary visualization of generative models in Fig. 5.6, where we compare class-conditional samples for three of these models. The unsupervised VAE's encodings of some classes (e.g., 2's and 4's and 8's and 9's) are not separated, and samples thus frequently appear to be the wrong class. Model M2 [Kingma et al., 2014b] explicitly encodes the class label as a latent variable, but nevertheless many sampled images do not visually match the conditioned class. In contrast, for our CPC-VAE model almost all samples are easily recognized as the target class.

We illustrate class-conditional samples for our CPC-VAE model of SVHN [Netzer et al., 2011] in Fig. 5.7, and for our CPC-VAE models of Celeb-A in Fig. 5.17. The SVHN samples show rich variability while clearly retaining the digit identity. For both the standard and very-deep CPC-VAE models of Celeb-A [Liu et al., 2015], the corresponding class can be easily determined from the sampled images. We also see the clearly superior image detail and realism that the state-of-the-art very-deep CPC-VAE architecture provides, which leads to substantially improved reclassification accuracy in Table 5.1.

5.5 Consistency-Constrained Variational Autoencoders

While PC-VAEs are effective given sufficient labels, they may generalize poorly when labels are very sparse (see Fig. 5.5). This fundamental problem arises because in the PC objective of Eq. (5.14), the parameters η of the predictor $\hat{y}(z, \eta)$ are only informed by the small labeled training dataset. We show that a complementary *consistency constraint* arises naturally from our modeling assumptions, and is an effective approach to SSL of VAEs. This approach has connections to consistency regularization methods for discriminative SSL [Miyato et al.,

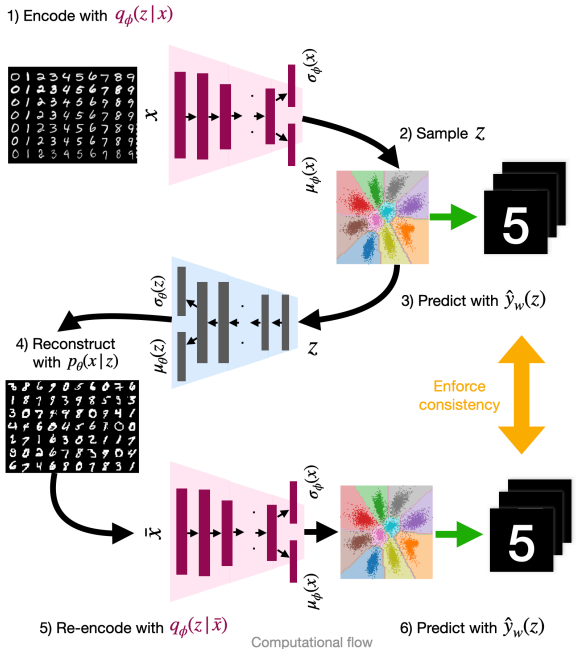


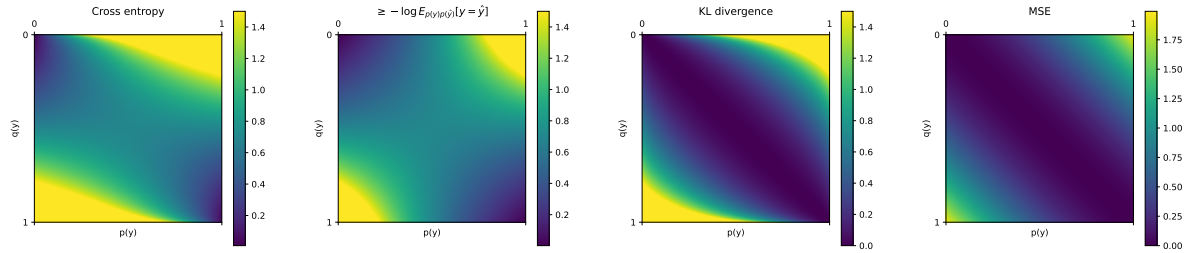
Figure 5.8: Computational flow diagram of a consistency-constrained variational autoencoder.

2019, Oliver et al., 2018].

5.5.1 Consistency Constraints via Generative Models

Let $x_d \sim p_\theta(\cdot \mid z_d)$ and $\bar{x}_d \sim p_\theta(\cdot \mid z_d)$ be two observations sampled from the *same* code z_d. Even if the true label y of x is uncertain, we know that for this VAE to be useful for prediction, $\bar{x}_d$ must have the *same* label as x_d. We formalize this (and dramatically boost performance) via a *consistency constraint* requiring label predictions for common-code data pairs $(x, \bar{x})$ to approximately match (see Fig. 5.11).

Given x_d, we predict labels $\hat{y}(z_d, \eta)$ via codes $z_d \sim q_\phi(z_d \mid x_d)$. Alternatively, given x_d we can *simulate* alternative features $\bar{x}_d$ with matching code z_d by sampling from the inference and generative models, and then predict the label for $\bar{x}_d$. We force the label predictions for x_d, and for $\bar{x}_d$, to be similar via a (cross-entropy) consistency penalty $\ell_C(\hat{y}(z_d, \eta), \hat{y}(\bar{z}_d, \eta))$



$\rho_p \log \rho_q + (1-\rho_p)\log(1-\rho_q)$ $\quad$ $\log(\rho_p\rho_q + (1-\rho_p)(1-\rho_q))$ $\quad$ $KL(p(y)||q(y))$ $\quad$ $(\rho_p - \rho_q)^2$

Figure 5.9: Visual comparison of consistency loss functions $\ell_C(p(y), q(y))$ for binary classification models. We use $p(y)$ and $q(y)$ to denote two Bernoulli distributions over $y \in \{0, 1\}$. Each plot shows $\ell_C(p(y), q(y))$ as a function of $p(y = 1) \triangleq \rho_p$ on the x-axis and $q(y = 1) \triangleq \rho_q$ on the y-axis. From left, **cross entropy**, **log-probability of equality**, **KL-divergence**, **MSE**.

applied to both *unlabeled* and *labeled* data as follows:

$$\begin{aligned}
\mathcal{C}^U(x_d, \theta, \phi, \eta) &\triangleq \mathbb{E}_{q_\phi(z_d|x_d)}\Big[\mathbb{E}_{p_\theta(\bar{x}_d|z_d)}\Big[\mathbb{E}_{q_\phi(\bar{z}_d|\bar{x}_d)}\Big[\ell_C\big(\hat{y}(z_d, \eta), \hat{y}(\bar{z}_d, \eta)\big)\Big]\Big]\Big], \\
\mathcal{C}^S(x_d, y_d, \theta, \phi, \eta) &\triangleq \mathbb{E}_{q_\phi(z_d|x_d)}\Big[\mathbb{E}_{p_\theta(\bar{x}_d|z_d)}\Big[\mathbb{E}_{q_\phi(\bar{z}_d|\bar{x}_d)}\Big[\ell_C\big(y_d, \hat{y}(\bar{z}_d, \eta)\big)\Big]\Big]\Big].
\end{aligned} \tag{5.29}$$

Fig. 5.9 shows alternative consistency losses for binary classification.

Consistent PC: Unconstrained objective.

We use multiplier $\gamma > 0$ to enforce consistency constraints for both unlabeled and labeled features, yielding the objective:

$$\min_{\theta,\phi,\eta} \sum_{d:\mathcal{D}^U \cup \mathcal{D}^S} \mathcal{L}(x_d, \theta, \phi) + \sum_{d:\mathcal{D}^U} \gamma \mathcal{C}^U(x_d, \theta, \phi, \eta) + \sum_{d:\mathcal{D}^S} +\lambda_\epsilon \mathcal{P}(x_d, y_d, \phi, \eta) + \gamma \mathcal{C}^S(x_d, y_d; \theta, \phi, \eta),$$

where $\mathcal{L}$ is the unsupervised bound, $\mathcal{P}$ is the prediction loss: $\mathcal{P}(x_d, y_d, \phi, \eta) \triangleq \mathbb{E}_{q_\phi(z_d|x_d)}\Big[\log p(y_d \mid z_d, \eta)\Big]$, and $\mathcal{C}^U, \mathcal{C}^S$ are the consistency costs.

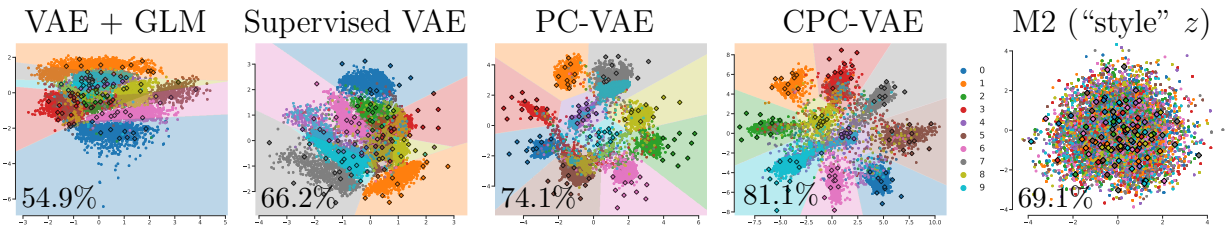


Figure 5.10: Learned 2-dim. encodings of MNIST (accuracy in corner). All methods use 49,900 unlabeled and 100 labeled images. We show each image's most likely encoding z, colored by true label y, with decision boundaries if possible. *Baselines:* 2-stage unsupervised VAE + GLM (Sec. 5.3.1) and a "supervised" VAE maximizing joint likelihood $\log p(x, y)$ (a PC-VAE with $\lambda = 1$). *Ours:* PC-VAE with $\lambda = 25$ and CPC-VAE. *Competitor:* M2 [Kingma et al., 2014b] intentionally decouples label y from "style" encoding z.

The decision network of Fig. 5.11 formalizes the integration of consistency constraints with the VAE generative model. All decisions incorporate the same inference network $q_\phi(z|x)$, explicitly coupling generative and discriminative learning. Figure 5.5 and Table 5.3 show that our CPC-VAE is more robust to over-parameterized architectures than DLC models.

Our CPC framework is agnostic to the choice of prediction and consistency penalties. While we focus on SSL image classification, extensions to continuous or multi-label prediction are promising.

Encoding visualizations.

To provide intuition for model differences, Fig. 5.10 shows VAE encodings of MNIST digits with latent dimension $C = 2$, given only 10 labeled examples per class. M2 has poor accuracy and (by design) no relation between encoding z and label y, while our CPC-VAE is more accurate and interpretable than all baselines.

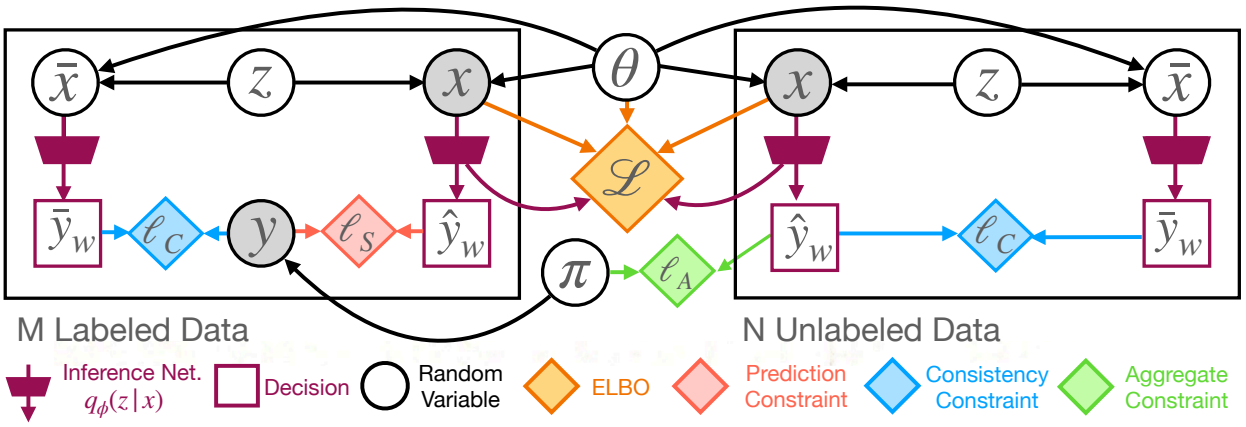


Figure 5.11: Formalization of our CPC-VAE as a decision network [Cowell et al., 2006]. Circular nodes are random variables, including latent codes z and observed features x. Shaded nodes are observed, including the class labels y for some data (left). Square decision nodes indicate label predictions $\hat{y}_w$ via the inference network $q_\phi(z \mid x)$, and diamonds indicate losses. *(a) Generative likelihood:* Like standard VAEs, we seek generative parameters θ and variational posteriors q_ϕ that maximize the variational bound $\mathcal{L}$ (*orange*). *(b) Prediction accuracy:* Unlike previous semi-supervised VAEs, we do not model the probability of labels y given z or x. Instead, we treat label prediction as a decision problem, with task-motivated loss ℓ_S (*red*) that constrains the encoder $q_\phi(z|x)$ (and thus the generative model). *(c) Prediction consistency:* For unlabeled data (right), we know that two observations x and $\bar{x}$ generated from the same latent code z should have identical labels; otherwise, the model cannot have high accuracy. The loss ℓ_C (*blue*) enforces this *consistency*. *(d) Aggregate consistency:* The predicted label frequencies for unlabeled data should be close to the empirical frequencies π of labeled data. The loss ℓ_A (*green*) enforces this, penalizing degenerate solutions to ℓ_C that use the same label $\hat{y}_w$ for most unlabeled data.

5.5.2 Aggregate label consistency.

Because the unsupervised consistency constraint $\mathcal{C}^U$ may be trivially satisfied by predicting the same class for all ambiguous examples, consistency-based training is susceptible to local optima where classes "collapse". Without careful balancing of constraint multipliers, CPC-VAEs may become trapped in degenerate solutions early in training. Discriminative consistency regularizers may also be susceptible to this issue.

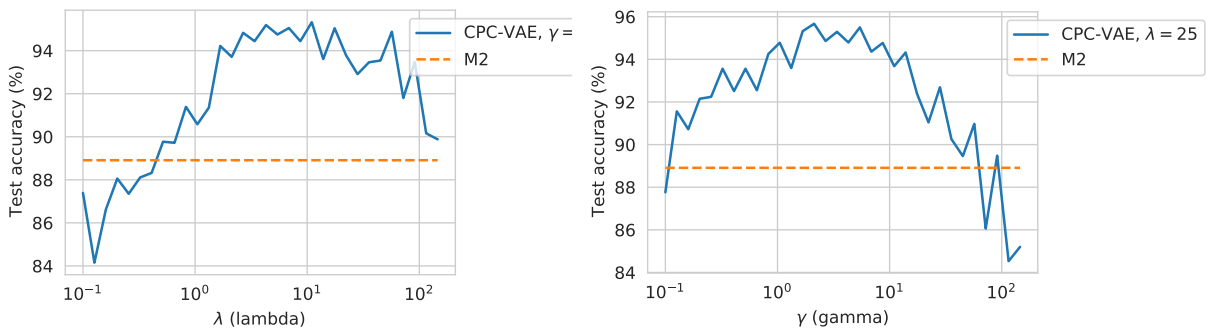
We address class collapse via a simple but natural assumption: unknown labels should approximately follow a known target distribution π. This target π may be the empirical distribution of the labeled data $\mathcal{D}^S$, or another distribution to facilitate domain adaptation. We then regularize predictions via an *aggregate label consistency* constraint that forces the distribution of label predictions to be close to π,

$$\ell_A\left(\pi, \mathbb{E}_{d\sim\mathcal{D}^U}\left[\mathbb{E}_{q(z_d|x_d)}\left[\hat{y}(z_d,\eta)\right]\right]\right), \tag{5.30}$$

where ℓ_A is the cross-entropy. Importantly, this aggregate consistency constraint does not penalize *individual* predictions for being confident. Prior work on SSL classifiers [Grandvalet and Bengio, 2004, Oliver et al., 2018] suggests that encouraging confident predictions on unlabeled data may be useful.

5.5.3 Sensitivity to constraint multiplier hyperparameters

We compare the test accuracy for our consistency-constrained model for MNIST over a range of values for both λ (the prediction constraint multiplier) and γ (the consistency constraint multiplier) in Figure 5.12. All runs used our best consistency-constrained model for MNIST using dense networks.



(a) Sensitivity to λ (b) Sensitivity to γ

Figure 5.12: Sensitivity of test accuracy to the constraint (Lagrange multiplier) hyperparameters λ and γ, evaluated on MNIST [LeCun et al., 2010].

We see that the resulting test accuracy smoothly varies across several orders of magnitude, with the optimal result being at or near the values we chose for our experiments. Performance is superior to the M2 baseline model for a wide range of hyperparameter values.

5.5.4 Training time comparison

Figure 5.13 below provides an empirical comparison of the average training time cost per step using the MNIST models summarized in our main paper's Table 2. Our CPC-VAE implementation runs both the encoder and decoder networks twice to compute the objective (once

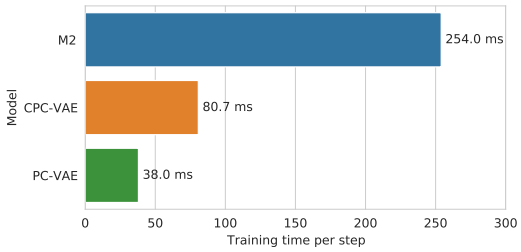


Figure 5.13: Comparison of training time per update step of stochastic gradient descent. Each model was trained on the semi-supervised MNIST with 100 labels using hyperparameter settings identical to those used in Table 2. Experiments were run on an RTX Titan GPU, using a common codebase built on top of Tensorflow [Abadi et al., 2015] that implements all methods. Each time reported is the average training step time over the second epoch.

for the standard VAE loss and an additional time to compute the consistency reconstruction and prediction), thus the runtime is approximately twice that of the PC-VAE: the PC-VAE requires 38 milliseconds per training step, while the CPC-VAE requires 80.7 milliseconds.

Furthermore, our empirical findings show that training M2 is more expensive than our proposed CPC-VAE in practice, which we expect given the theoretical runtime analysis. The M2 model must run the encoder and decoder networks once *per class* in order to compute the loss, due to the marginalization of the labels required for the unsupervised loss in Eq. (4). This increases the runtime by a factor equivalent to the number of classes. In our empirical test, we see that the training time per step is 6.7x that of the PC-VAE model, close to the 10x slowdown we would expect for the 10 digit classes of MNIST. In our experiments, we did not find substantial differences in the size of networks or number of training steps needed to train each of these models effectively.

5.5.5 Related Work on Constrained Learning

Li et al. [2018a]'s MMVA uses an objective like (5.14) for training VAEs from *fully-supervised* data with a max-margin loss. They favor a different DLC architecture, MMCVA, for semi-supervised learning.

Our unconstrained PC objective (5.15) has connections to the multi-conditional objective of McCallum et al. [2006], which was extended to deep generative models (both explicit VAEs and implicit GANs) by Kuleshov and Ermon [2017]. This prior work does not use consistency, does not present our constraint-based view of SSL, and exclusively uses *implicit GANs* in SSL experiments.

Recent non-generative image classifiers have used loss functions that encourage both accuracy and a notion of consistency on unlabeled data. *Virtual Adversarial Training* enforces

consistency under adversarial perturbations [Miyato et al., 2019]. *MixMatch* enforces consistency under feature interpolations [Berthelot et al., 2019b]. *Unsupervised Data Augmentation* (UDA, Xie et al. [2020]) achieves state-of-the-art vision and text SSL classification by enforcing label consistency on augmented perturbations of unlabeled features, but requires highly-engineered augmentation routines (e.g., image processing libraries). In contrast, we learn a generative model that samples features whose label predictions need to be consistent. Our CPC-VAE applies to new domains where augmentation routines are unavailable; the learned generator provides augmentations. Other related work includes *FixMatch* [Sohn et al., 2020], *ReMixMatch* [Berthelot et al., 2019a], and *SimMatch* [Zheng et al., 2022].

Consistent with the literature on SSL of VAEs, we do *not* compare to discriminative methods that require external data augmentation libraries. Conceivably such methods could be used to boost the SSL performance of all generative models, including our own CPC-VAE. But note that in our experiments, we find that one method using data augmentation (the SHOT-VAE) has poor generative performance.

More broadly, "cycle-consistency" has improved generative adversarial learning for images [Zhu et al., 2017, Zhou et al., 2016] and biomedical data [McDermott et al., 2018]. Others have developed cycle-consistent [Jha et al., 2018, Cemgil et al., 2020] and transform-consistent [Sinha and Dieng, 2021] objectives for VAEs that make the encodings z consistent. Miller et al. [2019] consider feature-to-label prediction in VAEs and enforce consistency with reconstructed predictions on fully-labeled data. In contrast, our work focuses on SSL and enforces consistency in code-to-label prediction.

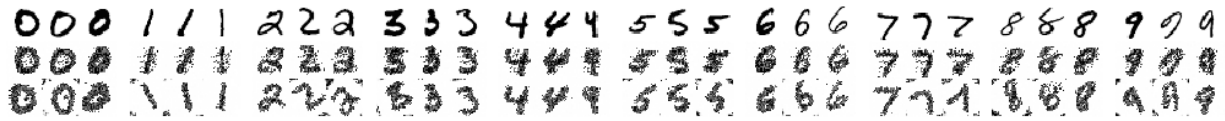


Figure 5.14: Sampled reconstructions used to compute the consistency loss during training. *Top:* Original image. *Middle:* Sampled reconstructions using a "Noise-Normal" likelihood. *Bottom:* Sampled reconstructions with spatial affine transformations sampled from the prior.

Figure 5.15: Visualization of spatial transform CPC-VAE reconstructions (trained with full labels). Each triplet shows *left:* the original image, *center:* the reconstructed image, and *right:* the "aligned" reconstruction obtained by setting the affine transform dimensions of the latent code to the prior mean. We see that the model learns a canonical orientation for each digit.

5.6 Generative Model Innovations

By design, the performance of our CPC-VAE is tied to the quality of the generative model, and we may boost semi-supervised learning performance by incorporating generative model advances (unlike more fragile DLC models). Conversely, if samples are poor consistency constraints may be ineffective. Our experiments explore two extensions of the basic VAE: affine transformations for poorly-aligned data, and "very deep" VAEs with many stochastic layers [Child, 2021].

5.6.1 Spatial Transformer VAE.

Our CPC-VAE framework for SSL exploits the spatial transformer's interpretable structure. Assuming small spatial transforms preserve the image class, our consistency constraints match z_* but sample z_t when generating $\bar{x}$ to improve diversity.

In our experiments, the first 6 latent dimensions z_{dt} are associated with affine transformation parameters capturing image translation, rotation, scaling, and shear:

- $z_{dt}^{(1)} \rightarrow$ *horizontal translation*,
- $z_{dt}^{(2)} \rightarrow$ *vertical translation*,
- $z_{dt}^{(3)} \rightarrow$ *rotation*,
- $z_{dt}^{(4)} \rightarrow$ *shear*,
- $z_{dt}^{(5)} \rightarrow$ *horizontal scale*,
- $z_{dt}^{(6)} \rightarrow$ *vertical scale*.

The remainder of the latent code, z_{d*}, generates parameters for independent per-pixel likelihoods.

To constrain our transformations to a fixed range of plausible values, we construct M_{dt} using parameters $\bar{z}_{dt}^{(i)} = \tanh(z_{dt}^{(i)})$ that are first mapped to the interval $[-1, +1]$, and then linearly rescaled to an appropriate range via hyperparameters $\alpha^{(1)}, \ldots, \alpha^{(6)}$. Figure 5.16 illustrates that the induced prior for $\bar{z}_{dt}^{(i)}$ is heaviest for extreme values, encouraging aggressive augmentation when sampling from the prior. The mapping function could be changed to modify this distribution for other applications.

Given these latent transformation parameters, we define an affine transformation matrix M_{dt}:

$$M_{dt} = \begin{bmatrix} 1 & 0 & \alpha^{(1)}\bar{z}_{dt}^{(1)} \\ 0 & 1 & \alpha^{(2)}\bar{z}_{dt}^{(2)} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\alpha^{(3)}\bar{z}_{dt}^{(3)}) & -\sin(\alpha^{(3)}\bar{z}_{dt}^{(3)}\alpha^{(4)}\bar{z}_{dt}^{(4)}) & 0 \\ \sin(\alpha^{(3)}\bar{z}_{dt}^{(3)}) & \cos(\alpha^{(3)}\bar{z}_{dt}^{(3)}\alpha^{(4)}\bar{z}_{dt}^{(4)}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} (\alpha^{(5)})^{\bar{z}_{dt}^{(5)}} & 0 & 0 \\ 0 & (\alpha^{(6)})^{\bar{z}_{dt}^{(6)}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.31}$$

To determine the parameters of the likelihood function for the pixel at coordinate (i, j), we

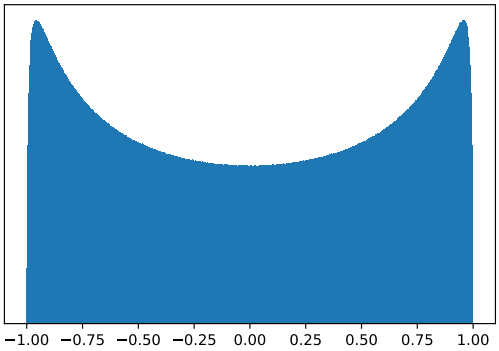


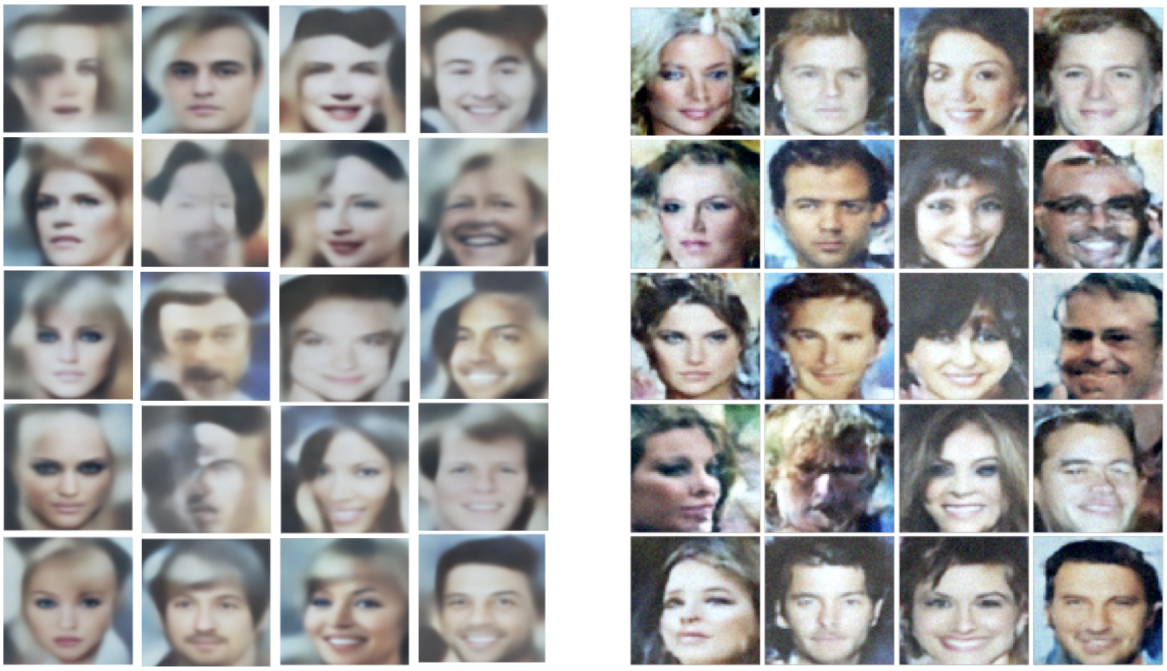
Figure 5.16: Prior distribution for latent parameters $\bar{z}_{dt}^{(i)} = \tanh(z_{dt}^{(i)})$ used to represent affine transformations.

use the generative model (or decoder) output at the pixel (i', j') for which

$$\begin{bmatrix} i \\ j \\ 1 \end{bmatrix} = M_{dt} \begin{bmatrix} i' \\ j' \\ 1 \end{bmatrix} . \tag{5.32}$$

This corresponds to applying horizontal and vertical scaling, followed by rotation and shear, followed by translation. As (i', j') may not correspond to integer coordinates, a *spatial transformer layer* [Jaderberg et al., 2015] uses bilinear interpolation of the non-transformed likelihood parameters, appropriately padding the boundaries of the decoder output. Related approaches in prior work have been proposed by: Skafte and Hauberg [2019], Bidart and Wong [2019], Bepler et al. [2019] and Ding et al. [2020].

Our approach could also be extended to other defined transforms of images or other types of data such as color shifts.



(a) Single stochastic-layer VAE (b) "Very-deep" VAE

Figure 5.17: Comparison of class-conditional samples of Celeb-A from a standard VAE and the deep VAE. From left to right, classes are *woman-neutral, man-neutral, woman-smiling, man-smiling*. Both models were trained as a semi-supervised CPC-VAE with 1000 labels. Samples from the single-layer VAE show the mean output for each pixel, samples from the very-deep model are fully sampled including at the pixel level.

5.6.2 Very Deep VAEs.

We can apply prediction and consistency constraints to a state-of-the-art "very deep" hierarchical VAE [Child, 2021], which defines an encoder-decoder architecture with "ladder" connections [Sønderby et al., 2016] that enable learning with dozens of latent stochastic layers. Child's Deep VAE achieves generative likelihoods that match or exceed state-of-the-art autoregressive models, while also enabling fast generative sampling in a coarse-to-fine fashion. Efficient simulation is critical for learning with our consistency constraints, to allow stochastic estimation of (5.29).

In the original very-deep VAE, the initial "bottom-up" encoder produces parameters for the variational posterior of the topmost stochastic layer $q_\phi(z_{d0} \mid x_d)$, while intermediate out-

Class Samples *Consistency Reconstructions*

Method	$\log p(x)$	FID	Reclass. Acc.
Deep CPC	***-1.47***	***42.1***	***92.9***
CPC-VAE	-4.27	53.7	**88.0**
SDGM	-5.50	70.4	25.6
M2	-4.43	65.1	69.0
VAE + GLM	**-4.25**	**51.6**	80.0
CC-VAE		75.7	82.2
SHOT-VAE		207.5	54.7

Figure 5.18: Evaluation of CPC-VAE's generative performance on CelebA. Due to the enormous computational cost of experiments in Child [2021], we use a smaller-scale variant of the very-deep architecture that nevertheless improves substantially on conventional VAEs. **Left image:** Samples from the learned (very) Deep CPC-VAE model conditioned on class (by column: *neutral woman, neutral man, smiling woman, smiling man*). Samples are chosen via rejection sampling in the latent space with a threshold of 95% confidence in the target class. **Right image:** Reconstructions of test images. Each pair shows an image and sample sharing only the deepest stochastic layer, sampling other layers. Our consistency constraints force these pairs, which differ in more subtle ways than standard image transforms, to have the same label. **Table:** Quantitative evaluation of generative performance on CelebA. *Held-out likelihood (*$\log p(x)$*, higher is better)*: We report log-likehoods per dimension, estimated using 500 importance samples [Burda et al., 2015] on 2000 test images. For CC-VAE and SHOT-VAE, author-provided code did not produce directly comparable likelihoods. *FID* (lower is better): We compare generated (unconditional) samples from each model to the test set using the Frechet Inception Distance metric [Heusel et al., 2017]. *Reclassification accuracy (higher is better):* Following Joy et al. [2021], we ask whether generative models (trained with only 1000 labels) produce samples from a target class recognizable by an independent classifier. We use a discriminative WRN trained on *all* CelebA training data, reaching 90% accuracy on CelebA's test set. CPC-VAE and VAE+GLM class-conditional samples are generated by post-hoc fitting of a model of $p(z \mid y)$, similar to the CCVAE; see Appendix for details.

puts from this network are used as ladder connections influencing corresponding variational distributions for intermediate stochastic layers. Figure 5.19(left) illustrates this structure, where $d_{dK}, \ldots, d_{d1}, d_{d0}$ are (deterministic) layer outputs from the bottom up encoder network. We let K equal the number of intermediate stochastic layers (2 for the simplified model in Fig. 5.19, many more in our experiments).

For our very-deep CPC-VAE we modify this structure by splitting the bottom-up encoder into two separate networks. We retain the ladder-structured bottom up network (parameters

ϕ) to influence the approximate posteriors for intermediate layers ($q_\phi(z_{d1}|x_d), q_\phi(z_{d2}|x_d), \ldots$). We do not modify the architecture of this network, but we do not use its final output $q_\phi(z_{d0}|x_d)$. Instead we introduce a separate encoder network for $q_\phi(z_{d0}|x_d)$, with parameters denoted ϕ^0. For this network we use the same WRN architecture employed in our single stochastic layer VAE experiments. Figure 5.19 (right) illustrates this modified encoder structure. The generative model remains unchanged, as it is not influenced by ladder connections. We find that this model architecture helps encourage consistency that affects the entire generative hierarchy, leading to higher test accuracy. Note that this change also does not affect the bottom-up factorization of the variational distribution. As in prior work, $p_\theta(z_d)$ and $q_\phi(z_d|x_d)$ are factorized as follows:

$$
\begin{aligned}
p_\theta(z_d) &= p(z_{d0}) p_\theta(z_{d1}|z_{d0}) \cdots p_\theta(z_{dK} \mid z_{d,<K})), \\
q_{\phi,\phi^0}(z) &= q_{\phi^0}(z_{d0}|x_d) q_\phi(z_{d1}|z_{d0}, x_d) \ldots q_\phi(z_{dK}|z_{d,<K}, x_d).
\end{aligned}
\tag{5.33}
$$

PC-VAE Architecture:

For the PC-VAE and CPC-VAE the prediction constraint is applied only to a subset of the latent variables, specifically those of the topmost stochastic layer (z_{d0}), so that the constraint only affects a small number of global latent variables. With this structure, implicit label information is accessible at every scale in the generative process. Our (constrained) PC-VAE objective becomes:

$$
\max_{\theta,\phi,\eta} \quad \sum_{d:\mathcal{D}^U \cup \mathcal{D}^S} \mathcal{L}^{\text{VAE}}(x_d, \theta, \phi^0, \phi, \quad \text{subj. to:} \quad \frac{1}{M} \sum_{x,y \in \mathcal{D}^S} \underbrace{\mathbb{E}_{q_\phi(z_{d0}|x)}[\ell_S(y_d, \hat{y}(z_{d0}, \eta))]}_{\mathcal{P}(x_d, y_d; \phi^0, \eta)} \leq \epsilon.
\tag{5.34}
$$

This design simplifies the classification structure and limits over-fitting. Due to the multi-

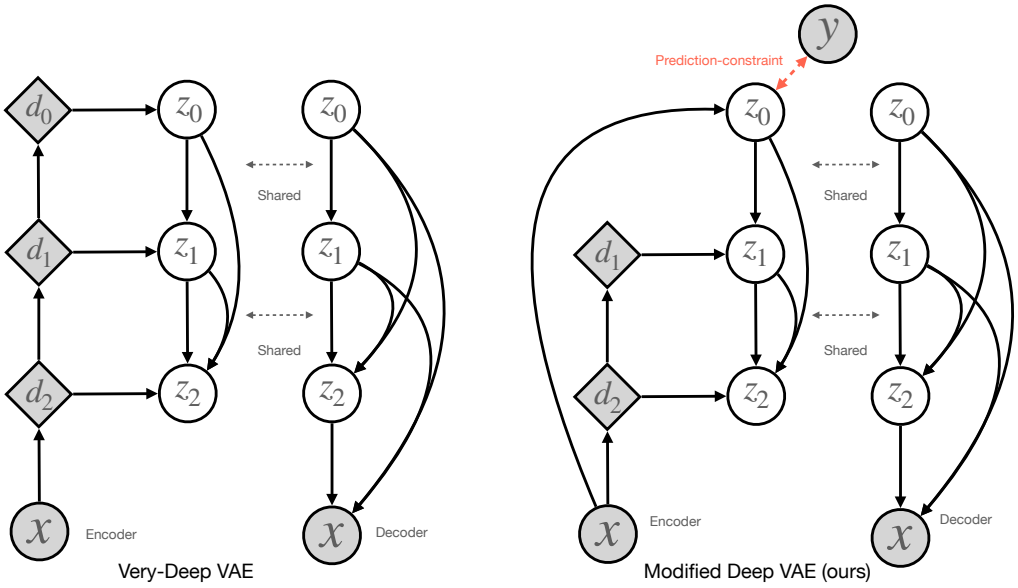


Figure 5.19: Comparison of the VAE model architecture proposed by [Child, 2021] and the modified version used by our CPC-VAE, which applies prediction and consistency constraints to z_0. Shaded nodes indicate observed variables, while diamond nodes are deterministic intermediate outputs of the encoder network. As in [Child, 2021], portions of the network structure are shared between the encoder and decoder.

scale structure of the very-deep VAE, latent variables lower in the hierarchy are highly localized, making them less suitable as features for predicting global image classes.

CPC-VAE Architecture:

When applying our consistency constraint, we follow our assumption that z_{d0} should fully determine the class of an image and thus we condition our consistency reconstruction only on the topmost stochastic layer, z_{d0}. In practice, this preserves the global representation relevant to the class label, while allowing consistency reconstructions to exhibit significant local variations. The distribution for "neighboring" images $\bar{x}_d$ assumed to have consistent

class labels is defined as:

$$p_\theta(\bar{x}_d|z_{d0}) \propto p_\theta(\bar{x}_d|z_{d,\leq K})p_\theta(z_{dK}|z_{d,<K})...p_\theta(z_{d1}|z_{d0}), \tag{5.35}$$

$$q_{\phi,\theta}(\bar{x}_d|x_d) = p_\theta(\bar{x}_d|z_{d0})q_\phi(z_{d0}|x_d). \tag{5.36}$$

Our corresponding supervised and unsupervised consistency losses are then defined as:

$$\mathcal{C}^U(x_d\theta, \phi^0, \phi, \eta) \triangleq \mathbb{E}_{q_\phi(z_{d0}|x_d)}\left[\mathbb{E}_{p_\theta(\bar{x}_d|z_{d0})}\left[\mathbb{E}_{q_\phi(\bar{z}_{d0}|\bar{x}_d)}\left[\ell_C(\hat{y}(z_d,\eta), \hat{y}(\bar{z}_{d0},\eta))\right]\right]\right], \tag{5.37}$$

$$\mathcal{C}^S(x_d, y_d, \theta, \phi^0, \phi, \eta) \triangleq \mathbb{E}_{q_\phi(z_{d0}|x_d)}\left[\mathbb{E}_{p_\theta(\bar{x}_d|z_{d0})}\left[\mathbb{E}_{q_\phi(\bar{z}_{d0}|\bar{x}_d)}\left[\ell_C(y_d, \hat{y}(\bar{z}_{d0},\eta))\right]\right]\right]. \tag{5.38}$$

5.7 Applications: PC-VAE

We compare our CPC-VAE to baselines on two goals: generative modeling of images x and classification accuracy of y given x. We use networks based on the WRN-28-2 architecture [Zagoruyko and Komodakis, 2016], train with Adam [Kingma and Ba, 2014], balance minibatches with 50% labeled and 50% unlabeled data, and search hyperparameters via Optuna [Akiba et al., 2019] to maximize validation accuracy.

Datasets.

For the CelebA dataset [Liu et al., 2015] with 1000 labeled and 159,770 unlabeled images, we predict 4 classes that combine gender (woman/man) and facial expression (neutral/smiling). We also test Street-View Housing Numbers (SVHN, Netzer et al. [2011], $|\mathcal{D}^S|$=1000, $|\mathcal{D}^U|$=62257, 10 classes) and MNIST (LeCun et al. [2010], $|\mathcal{D}^S|$=100, $|\mathcal{D}^U|$=49900, 10 classes).

SSL?	Gen?	Source	Method	MNIST (100)	SVHN (1000)	Celeb (1000)
✓	✓	ours	**CPC-VAE**	**98.86** (±0.18)	94.22 (±0.62)	**86.22**
✓	✓	Tab. 1-2 of Kingma et al.	M1 + M2	96.67 (±0.14)	63.98 (±0.10)	*79.28*
✓	✓	Tab. 6-7 of Li et al.	MMCVA	98.76 (±0.54)	**95.05** (±0.18)	-
✓	✓	Tab. 2 of Maaløe et al.	SDGM	98.68 (±0.07)	83.39 (±0.24)	*83.56*
✓	✓	Tab. 3 of Feng et al.	SHOT-VAE	96.88 (±0.22)	71.18 (±0.49)	*77.10*
✓	✓	Tab. 2 of Smieja et al.	SeGMA	87.3	77.6	75.4
✓	✓	Tab. 6 of Joy et al.	CCVAE	92.7 (200 labels)	-	*84.20*
✓		Tab. 3-4 of Miyato et al.	VAT	98.64 (±0.03)	94.23 (±0.32)	*81.48*
		ours	Discrim.	73.91 (±1.45)	87.70 (±1.02)	76.10

Table 5.2: SSL image classification results. We report mean test set accuracy (+/- std. dev.) across 10 runs on distinct random samples of the labeled set; only 1 run is feasible on large CelebA. Check in first column indicates the method uses both unlabeled and labeled data. Check in second column indicates the method is a generative model. Italicized entries indicate our own experimental results using the cited methods, matching architectures and pre-processing to our CPC-VAE. CPC-VAE results on CelebA do not use spatial transforms.

Method	MNIST (100)	Method	MNIST (100)	Method	MNIST (100)
CPC (2 Layer)	**96.68** (±0.54)	M2[§](1 L, α=0.1, B)	88.03 (±1.71)	MMVA (1 L, α=$*$)	80.50 (±2.56)
CPC (2 L, w/o A)	94.27 (±3.78)	M2 (2 L, α=0.1, B)	83.32 (±5.22)	MMVA (2 L, α=$*$)	83.50 (±2.51)
CPC (2 L, w/o ST)	91.93 (±1.65)	M2 (4 L, α=0.1, B)	47.05 (±8.13)	MMVA (2 L, α=.1)	58.27 (±5.82)
CPC (4 L, w/o ST)	93.78 (±2.25)	M2 (4 L, α=$*$, B)	68.15 (±3.43)		
PC (2 L)	80.49 (±3.31)	M2 (1 L, α=0.1, N)	73.93 (±8.12)	VAE + GLM (2 L)	72.90 (±1.98)

Table 5.3: Ablation study on MNIST comparing our SSL VAEs to M2 [Kingma et al., 2014b] and MMVA [Li et al., 2018a]. We use our own implementation, except for entry marked [§] from Kingma et al. [2014b]. We use a common MLP architecture with $C = 50$ and 1000 units per hidden layer. We indicate the likelihood: Noise-Normal (N, used by all CPC runs) or Bernoulli (B, used by M2 and MMVA). *Left:* Our innovations (consistency, spatial transforms (ST), and aggregate loss (A)) improve accuracy. *Center:* M2's accuracy deteriorates with larger networks, even after tuning α (α=$*$) instead of Kingma et al.'s default (α=0.1). CPC results are stable as size increases. *Right:* MMVA results are *worse* than CPC's.

SSL assessment in Table 5.2.

We compare the classification accuracy of our CPC-VAE to 6 state-of-the-art SSL VAE approaches, as well as two discriminative neural net baselines: *VAT*, a SSL method [Miyato et al., 2019], and *Discrim.*, a WideResNet trained only on the labeled data.

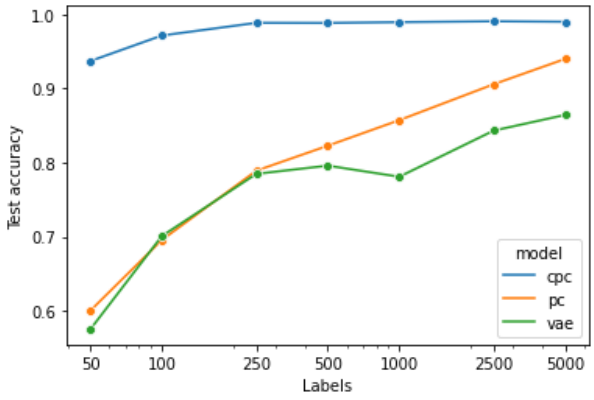


Figure 5.20: Comparison of consistency-constrained, prediction-constrained and 2-stage VAE test accuracy as a function of the number of labels observed. We train all models on the MNIST dataset using the full dataset as unlabeled data.

Ablation studies in Table 5.3.

We compare variants of CPC, M2, and MMVA [Li et al., 2018a] for SSL training on MNIST using a common architecture. We try CPC without consistency and without spatial transformations. For M2 and MMVA, we vary the number of layers and α penalties.

Generative assessment in Fig. 5.18 and Tab. 5.1.

This figure shows the quality of CelebA samples from our Deep CPC-VAE. The table compares our CPC-VAE and Deep CPC-VAE to other SSL VAEs on three metrics for generative model quality.

5.7.1 Findings and Analysis of Results

CPC-VAEs improve SSL classification accuracy.

In Tab. 5.2, CPC achieves the top accuracy among all methods (6 VAEs, 2 discriminative-only baselines) on two datasets: CelebA (86.22%) and MNIST (98.96%). On SVHN, CPC is close (within 1%) to MMCVA and matches the non-generative VAT.

CPC-VAEs generate high-quality images.

Samples and reconstructions from our Deep CPC-VAE are visually rich (Fig. 5.18) and achieve top predictive likelihoods by a wide margin. Among not-very-deep VAEs, our CPC-VAE achieves predictive likelihoods and FID scores in Table 5.1 closest to an unsupervised VAE. This suggests that prediction and consistency constraints have only minor impact on generative performance, toward our goal of *simultaneous* generation and discrimination.

CPC-VAE samples can be reclassified well.

Likelihoods do not fully capture generative quality [Theis et al., 2016], so to evaluate how well our models capture label characteristics, we use a *reclassification* paradigm adapted from Joy et al. [2021]. Table 5.1 reports the reclassification accuracy of using an independent classifier to recognize samples as examples of the intended class. Our CPC-VAE gets 88% and the (very) Deep CPC-VAE gets 92.9% accuracy; no other method tops 82%.

Flexible architectures improve CPC predictions, but may harm DLC models.

Adding spatial transforms boosts CPC accuracy on MNIST from 89.5% to 97.5%, while more expressive (WRN) networks grant a similar performance boost. In contrast, M2 is *fragile*

with accuracy that gets *worse* as generative capacity increases, dropping from 83% with 2 layers to 68% with 4 layers in Tab. 5.3.

Both consistency and prediction constraints are needed for high accuracy.

In Tab. 5.3, using only prediction constraints (PC) gets 80% accuracy on MNIST; adding consistency yields 97.5%. On CelebA, consistency improves accuracy from 78% to 86%.

Chapter 6

Future Directions

Throughout this thesis we have shown that prediction-constrained training effectively leverages latent variable models for better semi-supervised learning, more interpretable prediction and label-aware generation. In this final chapter we explore possible avenues for expanding on and improving prediction constrained training in future work.

6.1 Prediction-Guided Imputation.

In many real-world applications data is partially-observed with missing features, as shown in the ICU mortality-prediction experiments in chapter 3. In this case, vital signs and lab measurements are only observed at a subset of times during each patient's stay, and not every patient receives the same set of tests. As discussed in that chapter, the prediction-constrained latent variable model framework provides a robust method for making accurate predictions in the presence of missing features. A complementary task is *imputation*, predicting the values of missing features given the observed features. This means modeling the conditional distribution $p(x_M|x_O)$, where x_O is the feature subset of the full data observation x that is

observed and x_M is the missing subset.

Latent variable models have been widely used for imputation, particularly in the image domain via Markov random field and variational autoencoder models. Recent work has shown that a particularly effective approach is to apply traditional, non-amortized variational inference to variational autoencoders. This approach, known as *query-adaptive variational inference* (QAVI) [Agarwal et al., 2023], fits a per-image approximate posterior for the joint set missing and latent features: $q(x_M, z)$. This posterior is optimized using a lower bound on the marginal likelihood of *observed* features,

$$\log p(x_O) \geq \mathbb{E}_{q(x_M,z)} \left[\log \frac{p(x_O, x_M, z)}{q(x_M, z)} \right]. \tag{6.1}$$

Future work could combine this effective inpainting technique with prediction and consistency-constrained VAEs, allowing for label-informed imputations. For example, given a supervised VAE trained with prediction constraints, one approach could be to optimize the imputation approximate posterior subject to the same prediction-constraint on the latent representation. This would yield the following constrained objective:

$$\max_{q(x_M,z)} \mathbb{E}_{q(x_M,z)} \left[\log \frac{p(x_O, x_M, z)}{q(x_M, z)} \right] \tag{6.2}$$

$$\text{subject to } \mathbb{E}_{q(x_M,z)} \left[\log p(y_s|z) \right] \leq \epsilon_s, \tag{6.3}$$

where y_s is the target label. This approach could potentially be applied to both single-layer and hierarchical prediction and consistency constrained VAEs.

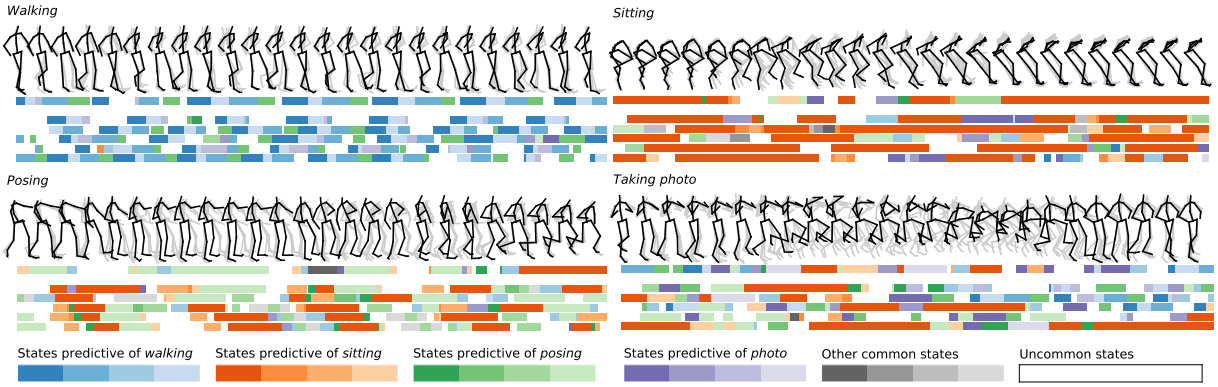


Figure 6.1: The SVAE SLDS segments each sequence of human motion, which we display as a sequence of discrete colors. **Discrete variables are interpretable:** Below each segmentation, we show 5 segmentations of *other* subjects performing the same action, noting similarity across semantically similar series. **Discrete variables are compact representations:** Samples from the generative model conditioned on ground-truth segmentations yield the stick figures in grey, which track closely with the observed data.

6.2 Prediction-Constrained Structured Variational Autoencoders.

The *structured variational autoencoder* (SVAE) [Johnson et al., 2016b] is a framework for combining discrete latent variable models, such as mixture models, topic models and hidden Markov models, with the *deep* neural network-defined likelihoods of variational autoencoders. In general, the generative model for a structured variational autoencoder variant of a discrete latent variable model has the same structure as the corresponding non-deep model, but the original set of observations, denoted as x in previous sections, are assumed to be unobserved and a new variable w now denotes the observed data. Analogous to the original variational autoencoder, w is modeled as conditional on x through a neural-network parameterized likelihood: $p_\theta(w \mid x) = \mathcal{F}(w \mid f_\theta(x))$, for a given likelihood family $\mathcal{F}$ and a function f with parameters θ.

The joint distribution of the SVAE thus factorizes as $p_\theta(w, x, h \mid \xi) = p(h \mid \xi^h)p(x \mid h, \xi^x)p_\theta(w \mid x)$. The corresponding evidence lower bound objective (ELBO) for the ap-

proximate posterior $q_\lambda(h, x) = q_{\lambda^h}(h)q_{\lambda^x}(x)$ is:

$$\log p(w) \geq \mathbb{E}_{q_{\lambda^h}(h)q_{\lambda^x}(x)} \left[\log \frac{p(h \mid \xi^h)p(x \mid h, \xi^x)p_\theta(w \mid x)}{q_{\lambda^h}(h)q_{\lambda^x}(x)} \right]. \tag{6.4}$$

The structured VAE uses a modified form of amortized variational inference that may be referred to as *likelihood-amortized* variational inference. In this approach the variational factors for a given observation w are fit according to a modified objective:

$$\lambda^h, \lambda^x = \underset{\lambda^h, \lambda^x}{\operatorname{argmax}} \, \mathbb{E}_{q_{\lambda^h}(h)q_{\lambda^x}(x)} \left[\log \frac{p(h \mid \xi^h)p(x \mid h, \xi^x)\ell_\phi(x|w)}{q_{\lambda^h}(h)q_{\lambda^x}(x)} \right]. \tag{6.5}$$

Here $\ell_\phi(x|w)$ is the amortized inference network with parameters ϕ that takes in the observation w and outputs the parameters of an appropriate conjugate likelihood for x. This *pseudo-objective* can then be optimized with respect to the variational parameters λ^h, λ^x with coordinate-ascent variational inference. Treating this internal variational optimization as a function of model parameters and observation, we can re-write the ELBO as:

$$\log p(w) \geq \mathbb{E}_{q_\phi(h,x|w,\xi)} \left[\log \frac{p(h \mid \xi^h)p(x \mid h, \xi^x)p_\theta(w \mid x)}{q_\phi(h, x \mid w, \xi)} \right]. \tag{6.6}$$

This objective can be optimized end-to-end using standard gradient descent.

A natural extension to prediction and consistency-constrained VAEs, would be to apply prediction (and consistency) constraints to the structured VAE model. In this setup the constrained objective would be:

$$\min_{\phi,\xi} -\mathbb{E}_{q_\phi(h,x|w,\xi)} \left[\log \frac{p(h \mid \xi^h)p(x \mid h, \xi^x)p_\theta(w \mid x)}{q_\phi(h, x \mid w, \xi)} \right] \tag{6.7}$$

$$\text{subject to: } -\mathbb{E}_{q_\phi(h,x|w,\xi)} \left[\log p(y \mid h, \eta) \right] \leq \epsilon. \tag{6.8}$$

This model could allow for natural extensions of prediction-constrained variational autoencoders to structured data, such as timeseries or geo-spatial data.

Fig. 6.1 shows an example of results from a structured variational autoencoder with a switching linear dynamical system discrete model (SVAE-SLDS). This model was train and evalutated on the H3.6M dataset [Ionescu et al., 2014]. The discrete states are an interpretable, compact representation of each sequence that could be used for classification.

Bibliography

M. Abadi, A. Agarwal, P. Barham, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

S. Agarwal, G. Hope, A. Younis, and E. Sudderth. A decoder suffices for query-adaptive variational inference. In *The 39th Conference on Uncertainty in Artificial Intelligence*, 2023. URL `https://openreview.net/forum?id=xqNCTr9kqg`.

T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

S. H. Al-Harbi and V. J. Rayward-Smith. Adapting k-means for supervised clustering. *Applied Intelligence*, 24(3):219–226, 2006.

C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1-2):5–43, 2003.

D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. A Public Domain Dataset for Human Activity Recognition using Smartphones. In *ESANN*, 2013.

A. D. Baha, T. Z. I. Fendoglu, N. Kokturk, H. Kilic, H. C. Hasanoglu, S. Arslan, M. Gulhan, N. Ogan, E. E. Akpinar, and A. Alhan. The effect of blood urea nitrogen/albumin ratio in the short-term prognosis of chronic obstructive pulmonary disease. *Erciyes Medical Journal*, 43(2):184–189, 2021.

T. Bepler, E. Zhong, K. Kelley, E. Brignole, and B. Berger. Explicitly disentangling image content from translation and rotation with spatial-vae. *Advances in Neural Information Processing Systems*, 2019.

J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International Conference on Machine Learning*, 2013.

D. Berthelot, N. Carlini, E. D. Cubuk, A. Kurakin, K. Sohn, H. Zhang, and C. Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv preprint arXiv:1911.09785*, 2019a.

D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. Raffel. MixMatch: A Holistic Approach to Semi-Supervised Learning. In *Advances in Neural Information Processing Systems*, 2019b. URL `http://arxiv.org/abs/1905.02249`.

R. Bidart and A. Wong. Affine variational autoencoders. In *International Conference on Image Analysis and Recognition*, pages 461–472. Springer, 2019.

D. M. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

D. M. Blei and J. D. Lafferty. Dynamic topic models. In *International Conference on Machine Learning*, 2006.

K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann. The balanced accuracy and its posterior distribution. In *International Conference on Pattern Recognition*, 2010.

Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li. BRITS: Bidirectional Recurrent Imputation for Time Series. In *Advances in Neural Information Processing Systems*, 2018. URL `https://papers.nips.cc/paper/2018/file/734e6bfcd358e25ac1db0a4241b95651-Paper.pdf`.

T. Cemgil, S. Ghaisas, K. Dvijotham, S. Gowal, and P. Kohli. The autoencoding variational autoencoder. *Advances in Neural Information Processing Systems*, 33, 2020.

M.-W. Chang, L. Ratinov, and D. Roth. Guiding semi-supervision with constraint-driven learning. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2007.

J. Chen, J. He, Y. Shen, L. Xiao, X. He, J. Gao, X. Song, and L. Deng. End-to-end learning of LDA by mirror-descent back propagation over a deep architecture. In *Neural Information Processing Systems*, 2015.

R. Child. Very deep VAEs generalize autoregressive models and can outperform them on images. *ICLR conference paper, arXiv:2011.10650v2*, 2021.

K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Empirical Methods in Natural Language Processing*, 2014. URL `http://arxiv.org/abs/1406.1078`.

R. G. Cowell, P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic networks and expert systems: Exact computational methods for Bayesian networks*. Springer Science & Business Media, 2006.

E. De Brouwer, J. Simm, A. Arany, and Y. Moreau. GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

A. Dhurandhar, M. Ackerman, and X. Wang. Uncovering group level insights with accordant clustering. *arXiv preprint 1704.02378*, 2017.

T. M. DiCicco and R. Patel. Machine classification of prosodic control in dysarthria. *Journal of medical speech-language pathology*, 18(4):35, 2010.

Z. Ding, Y. Xu, W. Xu, G. Parmar, Y. Yang, M. Welling, and Z. Tu. Guided variational autoencoder for disentanglement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7920–7929, 2020.

C. F. Eick, N. Zeidat, and Z. Zhao. Supervised clustering-algorithms and benefits. In *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pages 774–776. IEEE, 2004.

B. S. Everitt and D. Hand. *Finite mixture distributions*. Chapman and Hall, 1981. ISBN 0412224208.

H.-Z. Feng, K. Kong, M. Chen, T. Zhang, M. Zhu, and W. Chen. SHOT-VAE: Semi-supervised Deep Generative Models With Label-aware ELBO Approximations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, number 8, 2021. URL `https://ojs.aaai.org/index.php/AAAI/article/view/16909`.

T. Finley and T. Joachims. Supervised clustering with support vector machines. In *Proceedings of the 22nd international conference on Machine learning*, pages 217–224. ACM, 2005.

N. Flammarion, B. Palaniappan, and F. Bach. Robust discriminative clustering with sparse regularizers. *arXiv preprint 1608.08052*, 2016.

K. Ganchev, J. Graça, J. Gillenwater, and B. Taskar. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049, Aug. 2010.

Z. Ghahramani and G. E. Hinton. Parameter estimation for linear dynamical systems. Technical Report CRG-TR-96-2, University of Toronto Dept. of Computer Science, 1996.

Z. Ghahramani and M. I. Jordan. Supervised learning from incomplete data via an em approach. In *Neural Information Processing Systems*, 1993.

M. Ghassemi, M. Wu, M. C. Hughes, P. Szolovits, and F. Doshi-Velez. Predicting intervention onset in the icu with switching state space models. *AMIA Summits on Translational Science Proceedings*, 2017:82, 2017.

J. Gordon and J. M. Hernández-Lobato. Combining deep generative and discriminative models for Bayesian semi-supervised learning. *Pattern Recognition*, 100, 2020.

J. Graça, K. Ganchev, and B. Taskar. Expectation maximization and posterior constraints. In *Neural Information Processing Systems*, 2008.

Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04, page 529–536, Cambridge, MA, USA, 2004. MIT Press.

M. Grbovic, N. Djuric, S. Guo, and S. Vucetic. Supervised clustering of label ranking data using label preference information. *Machine learning*, 93(2-3):191–225, 2013.

G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical report, California Institute of Technology, 2007.

T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 2004.

L. A. Hannah, D. M. Blei, and W. B. Powell. Dirichlet process mixtures of generalized linear models. *Journal of Machine Learning Research*, 12(Jun):1923–1953, 2011.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems*, 30, 2017.

S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8), 1997. URL `https://www.bioinf.jku.at/publications/older/2604.pdf`.

M. Ilse, J. M. Tomczak, C. Louizos, and M. Welling. DIVA: Domain Invariant Variational Autoencoders. In *Proceedings of the Third Conference on Medical Imaging with Deep Learning*, pages 322–348. PMLR, 2020. URL `https://proceedings.mlr.press/v121/ilse20a.html`.

C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, jul 2014.

O. A. Ismaili, V. Lemaire, and A. Cornuéjols. Supervised pre-processings are useful for supervised clustering. In *Analysis of Large and Complex Data*, pages 147–157. Springer, 2016.

T. S. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. Technical Report AITR-1668, Artificial Intelligence Laboratory at the Massachusetts Institute of Technology, 1999a. URL `http://people.csail.mit.edu/tommi/papers/maxent.ps`.

T. S. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. In *Neural Information Processing Systems*, 1999b.

M. Jaderberg, K. Simonyan, A. Zisserman, and k. kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, 2015. URL `http://papers.nips.cc/paper/5854-spatial-transformer-networks.pdf`.

T. Jebara. *Discriminative, generative and imitative learning*. PhD thesis, Massachusetts Institute of Technology, 2001.

T. Jebara and A. Pentland. Maximum conditional likelihood via bound maximization and the CEM algorithm. In *Neural Information Processing Systems*, 1999.

A. H. Jha, S. Anand, M. Singh, and V. S. R. Veeravasarapu. Disentangling Factors of Variation with Cycle-Consistent Variational Auto-encoders. In *European Conference on Computer Vision (ECCV)*. Springer International Publishing, 2018.

A. Johnson, L. Bulgarelli, T. Pollard, S. Horng, L. A. Celi, and R. Mark. Mimic-iv. *PhysioNet. Available online at: https://physionet. org/content/mimiciv/1.0/(accessed August 23, 2021)*, 2020.

A. E. Johnson, T. J. Pollard, L. Shen, L.-w. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3, 2016a.

M. Johnson, D. K. Duvenaud, A. Wiltschko, R. P. Adams, and S. R. Datta. Composing graphical models with neural networks for structured representations and fast inference. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, pages 2946–2954. Curran Associates, Inc., 2016b. URL `http://papers.nips.cc/paper/6379-composing-graphical-models-with-neural-networks-for-structured-representations-a` `pdf`.

M. J. Johnson, D. K. Duvenaud, A. Wiltschko, R. P. Adams, and S. R. Datta. Composing graphical models with neural networks for structured representations and fast inference. *Advances in neural information processing systems*, 29, 2016c.

T. Joy, S. Schmon, P. Torr, S. N, and T. Rainforth. Capturing Label Characteristics in VAEs. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=wQRlSUZ5V7B`.

C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. In *AAAI Conference on Artificial Intelligence*, 2006.

D. Kingma and M. Welling. Auto-encoding variational Bayes. In *The International Conference on Learning Representations (ICLR)*, 2014.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, 2014. URL `http://arxiv.org/abs/1412.6980`.

D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Neural Information Processing Systems*, 2014a.

D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, 2014b. URL `https://papers.nips.cc/paper/5352-semi-supervised-learning-with-deep-generative-models.pdf`.

J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

O. Koller, S. Zargaran, and H. Ney. Re-sign: Re-aligned end-to-end sequence modelling with deep recurrent CNN-HMMs. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

S. G. Krantz and H. R. Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002.

A. Krause, P. Perona, and R. G. Gomes. Discriminative clustering by regularized information maximization. In *Advances in neural information processing systems*, pages 775–783, 2010.

H. Kuehne, A. Richard, and J. Gall. A hybrid rnn-hmm approach for weakly supervised temporal action segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018. doi: 10.1109/TPAMI.2018.2884469.

H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Traces and emergence of nonlinear programming*, pages 247–258. Springer, 2013.

V. Kuleshov and S. Ermon. Deep Hybrid Models: Bridging Discriminative and Generative Approaches. In *Uncertainty in Artificial Intelligence*, page 10, 2017.

S. Lacoste-Julien, F. Sha, and M. I. Jordan. DiscLDA: Discriminative learning for dimensionality reduction and classification. In *Neural Information Processing Systems*, 2009.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann, 2001.

J. A. Lasserre, C. M. Bishop, and T. P. Minka. Principled Hybrids of Generative and Discriminative Models. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, CVPR '06, pages 87–94, USA, 2006. IEEE Computer Society.

Y. LeCun, C. Cortes, and C. Burges. MNIST handwritten digit database, 2010. URL `http://yann.lecun.com/exdb/mnist/`.

C. Li, J. Zhu, and B. Zhang. Max-Margin Deep Generative Models for (Semi-)Supervised Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(11):2762–2775, 2018a.

F. Li, K. Shirahama, M. Nisar, L. Köping, and M. Grzegorzek. Comparison of feature learning methods for human activity recognition using wearable sensors. *Sensors*, 18(2): 679, 2018b.

Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

S. Liverani, D. Hastie, L. Azizi, M. Papathomas, and S. Richardson. PReMiuM: An R package for profile regression mixture models using Dirichlet processes. *Journal of Statistical Software*, 64(7):1–30, 2015. URL https://www.jstatsoft.org/v064/i07.

G. Loaiza-Ganem and J. P. Cunningham. The continuous bernoulli: fixing a pervasive error in variational autoencoders. *Advances in Neural Information Processing Systems*, 2019.

L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary Deep Generative Models. *arXiv:1602.05473 [cs, stat]*, 2016. URL http://arxiv.org/abs/1602.05473.

L. Maaløe, M. Fraccaro, V. Liévin, and O. Winther. Biva: A very deep hierarchy of latent variables for generative modeling. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019.

D. J. C. MacKay. Ensemble learning for hidden Markov models. Technical report, Department of Physics, University of Cambridge, 1997.

J. Malmaud, J. Huang, V. Rathod, N. Johnston, A. Rabinovich, and K. Murphy. What's cookin'? interpreting cooking videos using text, speech and vision. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

G. S. Mann and A. McCallum. Simple, robust, scalable semi-supervised learning via expectation regularization. In *International Conference on Machine Learning*, 2007.

G. S. Mann and A. McCallum. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of Machine Learning Research*, 11(Feb):955–984, 2010.

J. D. McAuliffe and D. M. Blei. Supervised topic models. In *Neural Information Processing Systems*, pages 121–128, 2008.

A. McCallum, C. Pal, G. Druck, and X. Wang. Multi-Conditional Learning: Generative/Discriminative Training for Clustering and Classification. In *AAAI Conference on Artificial Intelligence*, 2006. URL https://www.aaai.org/Papers/AAAI/2006/AAAI06-069.pdf.

A. K. McCallum. MALLET: Machine learning for language toolkit. mallet.cs.umass.edu, 2002.

M. B. A. McDermott, T. Yan, T. Naumann, N. Hunt, H. Suresh, P. Szolovits, and M. Ghassemi. Semi-Supervised Biomedical Translation with Cycle Wasserstein Regression GANs. In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, page 8, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/viewFile/16938/15951.

D. Micucci, M. Mobilio, and P. Napoletano. Unimib shar: A dataset for human activity recognition using acceleration data from smartphones. *Applied Sciences*, 7(10), 2017. ISSN 2076-3417. doi: 10.3390/app7101101. URL http://www.mdpi.com/2076-3417/7/10/1101.

A. C. Miller, Z. Obermeyer, J. P. Cunningham, and S. Mullainathan. Discriminative Regularization for Latent Variable Models with Applications to Electrocardiography. In *International Conference on Machine Learning*, page 10, 2019. URL `https://proceedings.mlr.press/v97/miller19a/miller19a.pdf`.

D. Mimno and A. McCallum. Topic models conditioned on arbitrary features with Dirichlet-multinomial regression. In *Uncertainty in Artificial Intelligence*, 2008.

T. Miyato, S.-I. Maeda, M. Koyama, and S. Ishii. Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1979–1993, 2019. URL `https://ieeexplore.ieee.org/document/8417973/`.

J. Molitor, M. Papathomas, M. Jerrett, and S. Richardson. Bayesian profile regression with an application to the national survey of children's health. *Biostatistics*, 11(3):484–498, 2010.

R. M. Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. URL `http://ufldl.stanford.edu/housenumbers`.

K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Learning to classify text from labeled and unla- beled documents. In *AAAI Conference on Artificial Intelligence*, 1998.

S. Nowozin and C. H. Lampert. Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6(3–4):185–365, 2011.

S. M. Oh, J. M. Rehg, T. Balch, and F. Dellaert. Learning and inferring motion patterns using parametric segmental switching linear dynamic systems. *International Journal of Computer Vision*, 77(1-3):103–124, 2008.

A. Oliver, A. Odena, C. A. Raffel, E. D. Cubuk, and I. Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems*, pages 3235–3246, 2018.

B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2005.

B. Peralta, P. Espinace, and A. Soto. Enhancing k-means using class labels. *Intelligent Data Analysis*, 17(6):1023–1039, 2013.

B. Peralta, A. Caro, and A. Soto. A proposal for supervised clustering with dirichlet process using labels. *Pattern Recognition Letters*, 80:52–57, 2016.

T. J. Pollard, A. E. Johnson, J. D. Raffa, L. A. Celi, R. G. Mark, and O. Badawi. The eicu collaborative research database, a freely available multi-center database for critical care research. *Scientific data*, 5(1):1–13, 2018.

L. R. Rabiner and B.-H. Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986a.

L. R. Rabiner and B.-H. Juang. An introduction to hidden Markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986b.

D. Ramage, D. Hall, R. Nallapati, and C. D. Manning. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 248–256. Association for Computational Linguistics, 2009.

R. G. Ramani and S. G. Jacob. Improved classification of lung cancer tumors based on structural and physicochemical properties of proteins using data mining models. *PloS one*, 8(3):e58772, 2013.

Y. Ren, Y. Wang, and J. Zhu. Spectral learning for supervised topic models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

B. Shahbaba and R. Neal. Nonlinear models using dirichlet process mixtures. *Journal of Machine Learning Research*, 10(Aug):1829–1850, 2009.

R. Shu, H. H. Bui, S. Zhao, M. J. Kochenderfer, and S. Ermon. Amortized Inference Regularization. In *Advances in Neural Information Processing Systems*, 2018. URL `https://proceedings.neurips.cc/paper/2018/file/1819932ff5cf474f4f19e7c7024640c2-Paper.pdf`.

R. H. Shumway and D. S. Stoffer. An approach to time series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis*, 3(4):253–264, 1982.

N. Siddharth, B. Paige, J.-W. van de Meent, A. Desmaison, N. D. Goodman, P. Kohli, F. Wood, and P. H. S. Torr. Learning Disentangled Representations with Semi-Supervised Deep Generative Models. In *Advances in Neural Information Processing Systems*, 2017. URL `http://arxiv.org/abs/1706.00400`.

S. Sinha and A. B. Dieng. Consistency regularization for variational auto-encoders. *arXiv preprint arXiv:2105.14859*, 2021.

N. Skafte and S. Hauberg. Explicit disentanglement of appearance and perspective in generative models. *Advances in Neural Information Processing Systems*, 32, 2019.

M. Smieja, M. Wolczyk, J. Tabor, and B. C. Geiger. SeGMA: Semi-Supervised Gaussian Mixture Autoencoder. *IEEE transactions on neural networks and learning systems*, 32(9): 3930–3941, 2021.

K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E. D. Cubuk, A. Kurakin, H. Zhang, and C. Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020.

C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder Variational Autoencoders. In *Advances in Neural Information Processing Systems*, 2016. URL `https://proceedings.neurips.cc/paper/2016/file/6ae07dcb33ec3b7c814df797cbda0f87-Paper.pdf`.

D. Sontag and D. Roy. Complexity of inference in latent dirichlet allocation. In *Neural Information Processing Systems*, 2011.

C. Sutton and A. McCallum. An introduction to conditional random fields. *Foundations and Trends in Machine Learning*, 4(4):267—-373, 2012.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

M. Taddy. On estimation and selection for topic models. In *Artificial Intelligence and Statistics*, 2012.

B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Neural Information Processing Systems*, pages 25–32, 2004.

L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. In *4th International Conference on Learning Representations*, 2016. URL `http://arxiv.org/abs/1511.01844`.

M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *International Conference on Machine Learning*, pages 104–112, 2004.

A. Vahdat and J. Kautz. Nvae: A deep hierarchical variational autoencoder. *arXiv preprint arXiv:2007.03898*, 2020.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

C. Wang, D. Blei, and F.-F. Li. Simultaneous image classification and annotation. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2009.

S. Wang, M. B. McDermott, G. Chauhan, M. Ghassemi, M. C. Hughes, and T. Naumann. Code for eicu extract. `https://github.com/MLforHealth/MIMIC_Extract/tree/eICU_Extract`, 2020.

Y. Wang and J. Zhu. Spectral methods for supervised topic models. In *Advances in Neural Information Processing Systems*, 2014.

Y. J. Wang and G. Y. Wong. Stochastic blockmodels for directed graphs. *Journal of the American Statistical Association*, 82(397):8–19, 1987.

Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le. Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems*, 2020.

Yelp Dataset Challenge. Yelp dataset challenge. `https://www.yelp.com/dataset_challenge`, 2016. Accessed: 2016-03.

J. Yoon, A. M. Alaa, M. Cadeiras, and M. van der Schaar. Personalized donor-recipient matching for organ transplantation. *arXiv preprint arXiv:1611.03934*, 2016.

S. Zagoruyko and N. Komodakis. Wide residual networks. In E. R. H. Richard C. Wilson and W. A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, September 2016. ISBN 1-901725-59-6. doi: 10.5244/C.30.87. URL `https://dx.doi.org/10.5244/C.30.87`.

C. Zhang and H. Kjellström. How to supervise topic models. In *ECCV Workshop on Graphical Models in Computer Vision*, 2014.

H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

M. Zheng, S. You, L. Huang, F. Wang, C. Qian, and C. Xu. Simmatch: Semi-supervised learning with similarity matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14471–14481, 2022.

T. Zhou, P. Krahenbuhl, M. Aubry, Q. Huang, and A. A. Efros. Learning Dense Correspondence via 3D-Guided Cycle Consistency. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 117–126, Las Vegas, NV, USA, 2016. IEEE.

J. Zhu, A. Ahmed, and E. P. Xing. MedLDA: maximum margin supervised topic models. *The Journal of Machine Learning Research*, 13(1):2237–2278, 2012.

J. Zhu, N. Chen, H. Perkins, and B. Zhang. Gibbs max-margin topic models with fast sampling algorithms. In *International Conference on Machine Learning*, 2013.

J. Zhu, N. Chen, and E. P. Xing. Bayesian inference with posterior regularization and applications to infinite latent svms. *Journal of Machine Learning Research*, 15(1):1799–1847, 2014.

J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, Venice, 2017. IEEE.