Title
Efficient Optimization Algorithms for Machine Learning

Author
Askari, Armin

Publication Date
2022

Peer reviewed|Thesis/dissertation

Efficient Optimization Algorithms for Machine Learning

by

Armin Askari

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering- Electrical Engineering & Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Laurent El Ghaoui, Chair
Professor Martin Wainwright
Professor Kannan Ramchandran
Professor Peter Bickel

Fall 2022

Efficient Optimization Algorithms for Machine Learning

Abstract

Efficient Optimization Algorithms for Machine Learning

by

Armin Askari

Doctor of Philosophy in Engineering- Electrical Engineering & Computer Science

University of California, Berkeley

Professor Laurent El Ghaoui, Chair

Behind all supervised learning problems is an optimization problem. Solving these problems reliably and efficiently is a key step in any machine learning pipeline. This thesis looks at efficient optimization algorithms for a variety of machine learning problems (in particular, sparse learning problems). We first begin by looking at a new class of algorithms for training feedforward neural networks. We then look at an efficient algorithm for constructing knockoff features for statistical inference. Finally, we look at ℓ_0-penalized and constrained optimization problems and a class of efficient algorithms for training these non-convex problems while providing guarantees on the quality of the solution.

To my family – Sharareh, Vahid and my lovely sister, Ayda.

Contents

Acknowledgments

My PhD would have been nothing without the amazing people I met along the way.

I first would like to thank my advisor and mentor Laurent El Ghaoui. Laurent helped me navigate PhD at the best and worst of times. Laurent always made it easy to figure out what problems to work on because he always did research for his love of optimization. I am very appreciative that we were able to work very closely on a variety of projects – I will always remember during our sparse naive bayes project, our primary form of communication was swapping latex documents with different ideas and proofs. Laurent has taught me innumerable things about optimization that no paper or textbook could have ever and I'm grateful to have learned so much from directly working with him. I also would like to thank Alexandre d'Aspremont who was my second, informal advisor. Alex was always available to chat even though he was in France and treated me like his own student. When Alex started collaborating with us, I had the most fun ever doing research. I absolutely loved working with Laurent and Alex – from talking convex optimization for hours to complaining about reviewers, I will remember those times for the rest of my life.

I want to also thank all the professors from the EE227 A/B/C series. In particular, Gireeja Ranade made teaching undergrads bearable because of the respect she treated me with throughout the semesters. Teaching 227C with Martin Wainwright was also loads of fun, from shooting the shit in his office to him making a homework problem that took me a week to solve.

I also want to thank the late Andrew Packard and late Pravin Varaiya. Working with both of them in undergrad set me on the research path and they were both truly inspiring research mentors and even more amazing human beings. Their memory will always live on.

During my PhD, I was fortunate to intern at Voleon and I am grateful for the wonderful colleagues I had collaborated and met there. Deepak Rajan, Prem Gopalan and Martin Wainwright were amazing mentors and supervisors who I loved talking to and discussing problems with. I learned so many new things about optimization algorithms by working with them and am very grateful for the experience. Much love to CSA.

I want to thank the El Ghaoui research group – Geoff Negiar, Forest Yang, Fangda Gu, Alicia Tsai and Fabian Pedregosa. I especially want to thank Geoff, who has been there since day one when we started our PhD journey together, navigated courses, published our first papers together and figured out where our desks were too late into the semester.

I can't even imagine my years at Berkeley without all the friends I made along the way. Yeshwanth Cherapanamjeri, Ashish Kumar, Nilesh Tripuraneni, Allan Jabri, Yu Sun, Zihao Chen, Chandan Singh, Ashvin Nair, Taejoo Ahn, Melih Elibol, Evonne Ng, Isabella Huang, John Miller, Kiran Shiragur, Morris Yau – this crowd made PhD so immensely fun that I would do my PhD 10 times over if all these people would be there. They were always there for me, they were always willing to hang out, and they always knew how to have a good time. All the trips we took together, all the China Village dinners, all the FIFA, all the pool on the warped table in Cory, all the prelim cramming, all the ethiopian food that took 3 hours to prepare; I will always cherish those memories. A very special thanks to Yeshwanth and

Ashish; these guys taught me so much about myself and who I am today has been greatly shaped by interacting with them over the years. I am forever indebted to them.

I would also like to thank the Statistics PhD students for treating me like one of their own – Taejoo Ahn, Tiffany Tang, Daniel Soriano, Benji Lu. Much love also to the IEOR PhD students – Salar Fattahi, Cedric Josz, Richard Zhang, Matt Olfat, Mahan Tajrobehkar, Kevin Li.

To all the friends before PhD began – much love for Austin Chou, Janette Tang, Jenna Wong, Angela Cai and Jerry Wu for always hanging out and letting me be myself. A shout out to the friends from where it all began – Jim Kim, Edward Moon and Sasha Ricciuti for always reminding me life was bigger than research. And no amount of words would be enough to thank the TOPS crew: Peter Wen, Christian Muller, Soheil Koushan, David Liang, Avrilynn Ding, Maylynn Ding. All the trips, the late nights, the coordination to meet each other in different countries for less than 72hrs, the video chats, the moving, the sleepovers, the parties – it's been nothing short of incredible and invaluable. I am grateful for all the support

I want to thank my extended family; my cousins (Shayan Monabbati, Nikrooz Farsad, Nariman Farsad), and my aunts and uncles (Mehran Monabbati, Fereydoun Farsad, Shohreh Salimpour, Nayyer Meiboodi). I was always able to feel their steadfast love no matter where I was or what I was doing.

And last but not least, I want to thank my family – my parents (Sharareh Salimpour and Vahid Askari) and my sister Ayda. I want to thank my mom for always asking me how many white hairs I had when video chatting her, my dad for giving me his unwavering support for any decision I made, and my sister for dealing with all my nonsense over the years. You guys are everything.

Chapter 1

Overview

Machine learning models are the cornerstone of any system that uses data to make decisions and predictions. Behind these models lie optimization algorithms. After taking a real world problem, collecting data and mathematically formulating the problem, the final step between the researcher and their machine learning model is an optimization problem. In this sense, optimization algorithms are the back-bone of training almost all machine learning models.

Despite there being many early results related to optimization, the field itself emerged in the early 1900s and for the past century has seen innumerable innovations and contributions. In the past decade, recent efforts in optimization and how it pertains to machine learning have been in two main areas: non-convex optimization and optimization algorithms.

Non-convex optimization has made significant advances particularly due to the rise in deep learning and the non-convexity of those models [24, 49]. There have been countless papers on different approaches for training deep neural networks (a non-exetensive list includes [39, 56, 77, 89, 67, 69]) and different ways of "convexify-ing" these networks [113, 30, 62]. While there has been amazing success in deep neural networks in medicine, computer vision, language and robotics, the theory unfortunately has not kept pace.

The design and implementation of optimization algorithms has also made significant advances because of big data and the necessity to be able to train models reliably and efficiently [78, 43]. This has also led to considerable efforts in the field of sparse optimization, where there has been a renewed interest in algorithms that are scalable, easy to implement and are computationally inexpensive [48, 11, 60]. There has also been a renewed interest in sparse optimization with the rise of big data because sparse models are more directly interpretable than their non-sparse counterparts [73]. Some of the most advanced and theoretically rich optimization algorithms the community has developed (such as the interior point method) cannot be used in traditional machine learning settings because their complexity scales poorly with the number of data points and the dimension of the dataset. As a result, there is a need for fast, yet efficient algorithms with provable guarantees on the quality of solutions.

There are two central themes in this thesis: sparse optimization and designing problem-specific algorithms. The work relating to sparse optimization pertains to non-convex ℓ_0 constrained/penalized problems for generic supervised learning problems, constructs an al-

gorithm for approximately solving these problems, and derives theoretical bounds on the quality of the solution. The work relating to designing problem-specific algorithms takes specific instances of supervised learning problems and proposes new, hand-crafted optimization algorithms for solving them much more efficiently than using generic solvers. Each chapter is self-contained and can be read independently of the other chapters.

Lifted and Fenchel Neural Networks (Chapters 2 & 3)

In these chapters, we focus on two new ways of training feedforward neural networks. In chapter 2, we introduce lifted neural networks where the main insight is that the feedforward linking constraints in the neural network problem can be reformulated as the argmin of an optimization problem. We then use this reformulation to "lift" the dimension of the problem by relaxing the constraints and training the model using block coordinate descent. We then improve on the performance of the previous model in chapter 3 by introducing fenchel neural networks. These models relax the linking constraints via the fenchel-young inequality and can be trained as before but with an additional batching procedure.

Fast Knockoffs (Chapter 4)

In this chapter, we visit the knockoff framework for feature selection [34]. We take the semidefinite program used to construct knockoffs and create a custom optimization algorithm to solve the problem efficiently and at scale. First, we introduce a barrier formulation of the problem to handle generic covariance matrices with complexity scaling as $\mathcal{O}(p^3)$ where p is the ambient dimension. We then assume a rank-k factor model on the covariance matrix to reduce this complexity bound to $\mathcal{O}(pk^2)$. We review an efficient procedure to estimate factor models and show that under a factor model assumption, we can sample knockoff covariates with complexity linear in the dimension. We test our methods on problems with as large as 500 000 (code published at `https://github.com/qrebjock/fanok`).

Sparse non-convex programs with guarantees (Chapters 5 & 6)

In chapter 5, we take the classical naive bayes algorithm and add an ℓ_0 sparsity constraint to the learning problem. his leads to a combinatorial maximum-likelihood problem, for which we provide an exact solution in the case of binary data, or a bound in the multinomial case. We prove that our bound becomes tight as the marginal contribution of additional features decreases. Both binary and multinomial sparse models are solvable in time almost linear in problem size, representing a very small extra relative cost compared to the classical naive Bayes. Numerical experiments on text data show that the naive Bayes feature selection method is as statistically effective as state-of-the-art feature selection methods such as recursive feature elimination, l_1-penalized logistic regression and LASSO, while being orders of magnitude faster. For a large data set, having more than with 1.6 million training points

and about 12 million features, and with a non-optimized CPU implementation, our sparse naive Bayes model can be trained in less than 15 seconds.

We then extend the theory developed to analyze the non-convex, sparse naive bayes problem in chapter 6. In this chapter, we show that sparsity-constrained optimization problems over low-dimensional spaces tend to have a small duality gap. We use the Shapley–Folkman theorem to derive both data-driven bounds on the duality gap and an efficient primalization procedure to recover feasible points satisfying these bounds. These error bounds are proportional to the rate of growth of the objective with the target cardinality, which means in particular that the relaxation is nearly tight as soon as the target cardinality is large enough so that only uninformative features are added.

Chapter 2

Lifted Neural Networks

2.1 Introduction

Given current advances in computing power, dataset sizes and the availability of specialized hardware/ software packages, the popularity of neural networks continue to grow. The model has become standard in a large number of tasks, such as image recognition, image captioning and machine translation. Current state of the art is to train this model by variations of stochastic gradient descent (SGD), although these methods have several caveats. Most problems with SGD are discussed in [96].

Optimization methods for neural networks has been an active research topic in the last decade. Specialized gradient-based algorithms such as Adam and Adagrad [58, 28] are often used but were shown to generalize less than their non adaptive counterparts by [107]. Our work is related to two main currents of research aimed at improving neural network optimization: using non gradient-based approaches and initializing weights to accelerate convergence of gradient-based algorithms. To our knowledge this paper is the first to combine the two. In addition our novel formalism allow for interesting extensions towards handling constraints, robustness, optimizing network topology, etc.

[96] and [18] propose an approach similar to ours, adding variables in the training problem and using an l^2-norm penalization of equality constraints. They both break down the network training problem into easier sub-problems and use alternate minimization; however they do not exploit structure in the activation functions. For Convolutional Neural Networks (CNN), [9] model the network training problem as a difference of convex functions optimization, where each subproblem is a Support Vector Machine (SVM).

On the initialization side, [64, 40] recommend sampling from a well-chosen uniform distribution to initialize weights and biases while others either use random initialization or weights learned in other networks (transfer learning) on different tasks. [94] indicate that initialization is crucial during training and that poorly initialized networks cannot be trained with momentum. Other methods to initialize neural networks have been proposed, such as using competitive learning [70] and principal component analysis (PCA) [87]. Although

PCA produces state of the art results, it is limited to auto-encoders while our framework allows for more general learning problems. Similarly, the competitive learning approach is limited to the classification problem and works only for one layer networks while our model can easily be adapted to a broader range of network architectures. Our approach focuses on transforming the non-smooth optimization problem encountered when fitting neural network models into a smooth problem in an enlarged space; this ties to a well developed branch of optimization literature (see *e.g.* section 5.2 of [15] and references therein). Our approach can also be seen as a generalization of the parameterized rectified linear unit (PReLU) proposed by [44]. Our work can be compared to the standard practice of initializing Gaussian Mixture Models using K-Means clustering; our model uses a simpler but similar algorithm for initialization.

Chapter outline. In Section 2.2, we begin by describing the mathematical setting of neural networks and our proposed optimization problem to train the model. Section 2.3 provides an example illustrating the basic idea. Section 2.4 outlines how to encode activation functions as argmins of convex or bi-convex optimization problems. Section 2.5 then expands the approach of Section 2.3 to cover a number of useful activation functions, as well as classification tasks. Section 2.6 describes a block-coordinate descent method to solve the training problem. Section 2.7 describes numerical experiments that support a finding that the models can be used as a fast weight initialization scheme.

2.2 Background and Notation

Feedforward neural networks. We begin by establishing notation. We are given an input data matrix $X = [x_1, \ldots, x_m] \in \mathbb{R}^{n\times m}$ and response matrix $Y \in \mathbb{R}^{p\times m}$ and consider a supervised problem involving a neural network having $L \geq 1$ hidden layers. At test time, the network processes an input vector $x \in \mathbb{R}^n$ to produce a predicted value $\hat{y}(x) \in \mathbb{R}^p$ according to the prediction rule $\hat{y}(x) = x_{L+1}$ where x_{L+1} is defined via the recursion

$$x_{l+1} = \phi_l(W_l x_l + b_l), \;\; l = 0, \ldots, L, \tag{2.1}$$

with initial value $x_0 = x \in \mathbb{R}^n$ and $x_l \in \mathbb{R}^{p_l}$, $l = 0, \ldots, L$. Here, ϕ_l, $l = 1, \ldots, L$ are given activation functions, acting on a vector; the matrices $W_l \in \mathbb{R}^{p_{l+1}\times p_l}$ and vectors $b_l \in \mathbb{R}^{p_{l+1}}$, $l = 0, \ldots, L$ are parameters of the network. In our setup, the sizes $(p_l)_{l=0}^{L+1}$ are given with $p_0 = n$ (the dimension of the input) and $p_{L+1} = p$ (the dimension of the output).

We can express the predicted outputs for a given set of m data points contained in the $n \times m$ matrix X as the $p \times m$ matrix $\hat{Y}(X) = X_{L+1}$, as defined by the matrix recursion

$$X_{l+1} = \phi_l(W_l X_l + b_l \mathbf{1}^T), \;\; l = 0, \ldots, L, \tag{2.2}$$

with initial value $X_0 = X$ and $X_l \in \mathbb{R}^{p_l\times m}$, $l = 0, \ldots, L$. Here, $\mathbf{1}$ stands for the vector of ones in $\mathbb{R}^m$, and we use the convention that the activation functions act column-wise on a matrix input.

In a standard neural network, the matrix parameters of the network are fitted via an optimization problem, typically of the form

$$\begin{array}{ll} & \min_{(W_l,b_l)_0^L,(X_l)_1^L} \mathcal{L}(Y, X_{L+1}) + \sum_{l=0}^{L} \rho_l \pi_l(W_l) \\ \text{s.t.} & X_{l+1} = \phi_l(W_l X_l + b_l \mathbf{1}_m^T), \;\; l = 0, \ldots, L \\ & X_0 = X \end{array} \tag{2.3}$$

where $\mathcal{L}$ is a loss function, $\rho \in \mathbb{R}_+^{L+1}$ is a hyper-parameter vector, and π_l's are penalty functions which can be used to encode convex constraints, network structure, etc. We refer to the collections $(W_l, b_l)_{l=0}^L$ and $(X_l)_{l=1}^L$ as the (W, b)- and X-variables, respectively.

To solve the training problem (2.3), the X-variables are usually eliminated via the recursion (2.2), and the resulting objective function of the (W, b)-variables is minimized without constraints, via stochastic gradients. While this appears to be a natural approach, it does make the objective function of the problem very complicated and difficult to minimize.

Lifted models. In this paper, we develop a family of models where the X-variables are kept, and the recursion constraints (2.1) are approximated instead, via penalties. We refer to these models as "lifted" because we lift the search space of (W, b)-variables to a higher-dimensional space of (W, b, X)-variables. The training problem is cast in the form of a matrix factorization problem with constraints on the variables encoding network structure and activation functions.

Lifted models have many more variables but a much more explicit structure than the original, allowing for training algorithms that can use efficient standard machine learning libraries in key steps. The block-coordinate descent algorithm described here involves steps that are parallelizable across either data points and/or layers; each step is a simple structured convex problem.

The family of alternate models proposed here have the potential to become competitive in their own right in learning tasks, both in terms of speed and performance. In addition, such models are versatile enough to tackle problems deemed difficult in a standard setting, including robustness to noisy inputs, adaptation of activation functions to data, or including constraints on the weight matrices. Our preliminary experiments are limited to the case where the lifted model's variables are used as initialization of traditional feedforward network. However, we discuss and layout the framework for how these models can be used to tackle other issues concerning traditional networks such as robustness and optimizing how to choose activation functions at each layer.

2.3 Basic Idea

To describe the basic idea, we consider a specific example, in which all the activation functions are the ReLUs, except for the last layer. There ϕ_L is the identity for regression tasks or a softmax for classification tasks. In addition, we assume in this section that the penalty functions are of the form $\pi_l(W) = \|W\|_F^2$, $l = 0, \ldots, L$.

We observe that the ReLU map, acting componentwise on a vector input u, can be represented as the "argmin" of an optimization problem involving a jointly convex function:

$$\phi(u) = \max(0, u) = \arg\min_{v \geq 0} \|v - u\|_2. \tag{2.4}$$

As seen later, many activation functions can be represented as the "arg min" of an optimization problem, involving a jointly convex or bi-convex function.

Extending the above to a matrix case yields that the condition $X_{l+1} = \phi(W_l X_l + b_l \mathbf{1}^T)$ for given l can be expressed via an "arg min":

$$X_{l+1} \in \arg\min_{Z \geq 0} \|Z - W_l X_l - b_l \mathbf{1}^T\|_F^2.$$

This representation suggests a heuristic to solve (2.3), replacing the training problem by

$$\begin{array}{rl} \min_{(W_l, b_l), (X_l)} & \mathcal{L}(Y, W_L X_L + b_L \mathbf{1}^T) + \sum_{l=0}^{L} \rho_l \|W_l\|_F^2 \\ & + \sum_{l=0}^{L-1} \left(\lambda_{l+1} \|X_{l+1} - W_l X_l - b_l \mathbf{1}^T\|_F^2 \right) \\ \text{s.t.} & X_l \geq 0, \quad l = 1, \ldots, L-1, \quad X_0 = X. \end{array} \tag{2.5}$$

where $\lambda_{l+1} > 0$ are hyperparameters, ρ_l are regularization parameters as in (2.3) and $\mathcal{L}$ is a loss describing the learning task. In the above model, the activation function is not used in a pre-defined manner; rather, it is *adapted* to data, via the non-negativity constraints on the "state" matrices $(X_l)_{l=1}^{L+1}$. We refer to the above as a "lifted neural network" problem.

Thanks to re-scaling the variables with $X_l \to \sqrt{\lambda_l} X_l$, $W_l \to \sqrt{\lambda_{l+1}/\lambda_l} W_l$, and modifying ρ_l's accordingly, we can always assume that all entries in λ are equal, which means that our model introduces just one extra scalar hyper-parameter over the standard network (2.3).

The above optimization problem is, of course, challenging, mainly due to the number of variables. However, for that price we gain a lot of insight on the training problem. In particular, the new model has the following useful characteristics:

- For fixed (W, b)-variables, the problem is convex in the X-variables X_l, $l = 1, \ldots, L$; more precisely it is a (matrix) non-negative least-squares problem. The problem is fully *parallelizable across the data points.*
- Likewise, for fixed X-variables, the problem is convex in the (W, b)-variables and *parallelizable across layers and data points.* In fact, the (W, b)-step is a set of parallel (matrix) ridge regression problems.

These characteristics allow for efficient block-coordinate descent methods to be applied to our learning problem. Each step reduces to a basic supervised learning problem, such as ridge regression or non-negative least-squares. We describe one algorithm in more detail in section 2.6.

The reader may wonder at this point what is the prediction rule associated with our model. For now, we focus on extending the approach to broader classes of activations and loss functions used in the last layer; we return to the prediction rule issue in our more general setting in section 2.5.

2.4 Activations as $\arg\min$ Maps

In this section, we outline theory on how to convert a class of functions as the "arg min" of a certain optimization problem, which we then encode as a penalty in the training problem. We make the following assumption on a generic activation function ϕ.

> **BCR Condition.** The activation function $\phi : \mathbb{R}^k \to \mathbb{R}^j$ satisfies the bi-convex representation (BCR) condition if it can be represented as follows:
>
> $$\forall x \in \mathbb{R}^k,\ \phi(x) = \arg\min_{z \in \mathbb{R}^j} \mathcal{D}_\phi(x, z),$$
>
> where $\mathcal{D}_\phi : \mathbb{R}^k \times \mathbb{R}^j \to \mathbb{R}$ is a bi-convex function (convex in x for fixed z and vice-versa), which is referred to as a *BC-divergence* associated with the activation function.

We next examine a few examples, all based on divergences of the form

$$D_\phi(x, z) = \Phi(z) - x^T z,$$

where Φ is a convex function. This form implies that, when Φ is differentiable, ϕ is the gradient map of a convex function; thus, it is monotone.

Strictly monotone activation functions. We assume that ϕ is strictly monotone, say without loss of generality, strictly increasing. Then, it is invertible, and there exists a function, denoted ϕ^{-1}, such that the condition $x = \phi^{-1}(z)$ for $z \in \mathbf{range}(\phi)$ implies $z = \phi(x)$. Note that ϕ^{-1} is strictly increasing on its domain, which is $\mathbf{range}(\phi)$.

Define the function $\Phi : \mathbb{R} \to \mathbb{R}$, with values

$$\Phi(z) = \int_0^z \phi^{-1}(u)\, du \text{ if } z \in \mathbf{range}(\phi), \tag{2.6}$$

and $+\infty$ otherwise.

The function Φ is convex, since ϕ^{-1} is increasing. We then consider the problem

$$\min\{\ \Phi(z) - xz\ :\ z \in \mathbf{range}(\phi)\}. \tag{2.7}$$

Note that the *value* of the problem is nothing else than $\Phi^*(x)$, where Φ^* is the Fenchel conjugate of Φ.

By construction, the problem (2.7) is convex. At optimum, we have $x = \phi^{-1}(z)$, hence $z = \phi(x)$. We have obtained

$$\phi(x) = \arg\min_z\ \Phi(z) - xz\ :\ z \in \mathbf{range}(\phi).$$

Examples. As an example, consider the sigmoïd function:

$$\phi(x) = \frac{1}{1+e^{-x}},$$

with inverse

$$\phi^{-1}(z) = \log\frac{z}{1-z}, \;\; 0 < z < 1,$$

and $+\infty$ otherwise.

Via the representation result (2.6), we obtain

$$\phi(x) = \arg\min_{0\le z\le 1} \; z\log z + (1-z)\log(1-z) - xz$$

Next consider the "leaky ReLU" function

$$\phi(x) = \begin{cases} \alpha x & \text{if } x < 0, \\ x & \text{if } x \ge 0, \end{cases}$$

where $0 < \alpha < 1$. We have

$$\phi^{-1}(z) = \begin{cases} (1/\alpha)z & \text{if } z < 0, \\ z & \text{if } z \ge 0, \end{cases}$$

with domain the full real line; thus

$$\begin{aligned} \Phi(z) &= \int_0^z \phi^{-1}(u)\, du \\ &= \frac{1}{2}\max\left(\frac{1}{\alpha}\max(0,-z)^2, \max(0,z)^2\right) \end{aligned} \tag{2.8}$$

As another example, consider the case with $\phi(x) = \operatorname{arctanh}(x)$. The inverse function is

$$\phi^{-1}(z) = \frac{1}{2}\log\frac{1+z}{1-z}, \;\; |z| \le 1$$

For any $z \in [-1,1]$, $\Phi(z)$ takes the form

$$\begin{aligned} \Phi(z) &= \frac{1}{2}\int_0^z \left(\log(1+u) - \log(1-u)\right)\, du \\ &= \frac{1}{2}\left((1-z)\log(1-z) + (1+z)\log(1+z)\right) + \text{cst}. \end{aligned}$$

Sometimes there are no closed-form expressions. For example, for the so-called "softplus" function $\phi(x) = \log(1+e^x)$, the function Φ cannot be expressed in closed form:

$$\Phi(z) = \int_0^z \log(e^u - 1)\, du, \;\; \mathbf{dom}\Phi = \mathbb{R}_+.$$

This lack of a closed-form expression does not preclude algorithms from work with these types of activation functions. The same is true of the sigmoid function.

Non-strictly monotone examples: ReLU and piece-wise sigmoid. The above expression (2.7) works in the ReLU case; we simply restrict the inverse function to the domain $\mathbb{R}_+$; specifically, we define

$$\phi^{-1}(z) = \begin{cases} +\infty & \text{if } z < 0, \\ z & \text{if } z \geq 0, \end{cases}$$

We then have $\mathbf{dom}\Phi = \mathbb{R}_+$, and for $z \geq 0$:

$$\Phi(z) = \int_0^z u \, du = \frac{1}{2} z^2.$$

We have obtained

$$\phi(x) = \arg\min_{z \geq 0} \ \Phi(z) - xz.$$

The result is consistent with the "leaky" ReLU case in the limit when $\alpha \to 0$. Indeed, in that case with Φ given as in (2.8), we observe that when $\alpha \to 0$ the domain of Φ collapses from the whole real line to $\mathbb{R}_+$, and the result follows.

In a similar vein, consider the "piecewise" sigmoid function,

$$\phi(x) = \min(1, \max(-1, x)),$$

This function can be represented as

$$\phi(x) = \arg\min_{z} \ z^2 - 2xz \ : \ |z| \leq 1.$$

Finally the sign function is represented as

$$\mathbf{sign}(x) = \arg\min \ -zx \ : \ |z| \leq 1.$$

Non-monotone examples. The approach can be sometimes extended to non-monotonic activation functions. As an example, the activation function $\phi(x) = \sin x$ has been proposed in the context of time-series. Here, we will work with

$$\Phi(z) = \int_0^z \arcsin(u) \, du = z \arcsin z + \sqrt{1 - z^2} + \text{cst.},$$

with domain $[-1, 1]$. The function is convex, and we can check that

$$\phi(x) = \arg \min_{z \, : \, |z| \leq 1} \ \Phi(x) - xz$$

Jointly convex representations. Some activation functions enjoy a stronger condition, which in turn leads to improved properties of the corresponding lifted model.

> **JCR Condition.** The activation function $\phi : \mathbb{R}^k \to \mathbb{R}^j$ satisfies the jointly convex representation (JCR) condition if it satisfies the CR condition with a jointly convex function $\mathcal{D}_\phi(x, z)$.

Note that, for the JCR condition to hold, the activation function needs to be monotone. Because of non-uniqueness, we may add a term that is not a function of the variable being optimized (i.e. in the condition above, an arbitrary function of u) to the JC-divergence in order to improve the overall structure of the problem. This is highlighted below and discussed in Section 2.6.

The JCR condition applies to several important activation functions, beyond the ReLU, for which

$$\max(x,0) = \arg\min_z \ \mathcal{D}_\phi(x,z) = \begin{cases} \|x-z\|_2^2 & \text{if } z \geq 0, \\ +\infty & \text{otherwise.} \end{cases}$$

Note that the JC-divergence for the ReLU is not unique; for example, we can replace the l_2-norm by the l_1-norm.

The "leaky" ReLU with parameter $\alpha \in (0,1)$, defined by $\phi(x) = \max(x/\alpha, x)$, can be written in a similar way:

$$\max(x/\alpha, x) = \arg\min_z \ \|x-z\|_2^2 \ : \ z \geq (1/\alpha)x.$$

The piece-wise sigmoïd, as defined below, has a similar variational representation: with $\mathbf{1}$ the vector of ones,

$$\min(1, \max(0,x)) = \arg\min_z \ \|x-z\|_2^2 \ : \ 0 \leq z \leq \mathbf{1}.$$

In order to address multi-class classification problems, it is useful to consider a last layer with an activation function that produces a probability distribution output. To this end, we may consider an activation function which projects, with respect to some metric, a vector onto the probability simplex. The simplest example is the Euclidean projection of a real vector $u \in \mathbb{R}^k$ onto the probability simplex in $\mathbb{R}^k$:

$$\phi(x) = \arg\min_z \ \|x-z\|_2^2 \ : \ z \geq 0, \ \ z^T\mathbf{1} = 1.$$

Max-pooling operators are often used in the context of image classification. A simple example of a max-pooling operator involves a p-vector input x with two blocks, $x = (x^{(1)}, x^{(2)})$, with $x(i) \in \mathbb{R}^{p_i}$, $i = 1, 2$, with $p = p_1 + p_2$. We define $\phi : \mathbb{R}^p \to \mathbb{R}^2$ by

$$\phi(x) = (\max_{1 \leq i \leq p_1} x_i^{(1)}, \max_{1 \leq i \leq p_2} x_i^{(2)}) \in \mathbb{R}^2. \tag{2.9}$$

Max-pooling operators can also be expressed in terms of a jointly convex divergence. In the above case, we have

$$\phi(x) = \arg\min_z \ \mathbf{1}^T z + \mathbf{1}^T (x - Dz)_+,$$

where D is an appropriate block-diagonal matrix of size $p \times 2$ that encodes the specifics of the max-pooling, namely in our case $D = \mathbf{diag}(\mathbf{1}_{p_1}, \mathbf{1}_{p_2})$.

Extension to matrix inputs. Equipped with a divergence function that works on vector inputs, we can readily extend it to matrix inputs with the convention that the divergence is summed across columns (data points). Specifically, if $X = [x_1, \ldots, x_m] \in \mathbb{R}^{k\times m}$, we define ϕ by $Z = \phi(X) = [z_1, \ldots, z_m] \in \mathbb{R}^{h\times m}$ as acting column-wise. We have

$$\phi(X) := [\phi(x_1), \ldots, \phi(x_m)] = \arg\min_Z \ \mathcal{D}_\phi(X, Z),$$

where, with some minor abuse of notation, we define a matrix version of the divergence, as follows:

$$\mathcal{D}_\phi([x_1, \ldots, x_m], [z_1, \ldots, z_m]) = \sum_{i=1}^m \mathcal{D}_\phi(x_i, z_i).$$

2.5 Lifted Framework

Lifted neural networks

Assume that the BCR or JCR condition is satisfied for each layer of our network and use the short-hand notation $D_l = D_{\phi_l}$ for the corresponding divergences. Condition (2.2) is then written as

$$X_{l+1} \in \arg\min_{X\in\mathcal{X}} \ D_l(X, W_l X_l + b_l \mathbf{1}^T), \quad l = 0, \ldots, L.$$

The lifted model consists in replacing the constraints (2.2) with penalties in the training problem. Specifically, the lifted network training problem takes the form

$$\begin{aligned} \min_{(W_l,b_l),(X_l)} \ & \mathcal{L}(Y, W_L X_L + b_L \mathbf{1}^T) + \sum_{l=0}^{L} \pi_l(W_l) \\ & + \sum_{l=0}^{L-1} \lambda_{l+1} D_l(W_l X_l + b_l \mathbf{1}^T, X_{l+1}) \\ & \text{s.t. } X_0 = X, \quad X_l \geq 0, \ l = 1, \ldots, L-1 \end{aligned} \tag{2.10}$$

with $\lambda_1, \ldots, \lambda_{L+1}$ given positive hyper-parameters. As with the model introduced in section 2.3, the lifted model enjoys the same *parallel and convex* structure outlined earlier. In particular, it is convex in X-variables for fixed W-variables. If we use a weaker bi-convex representation (using a bi-convex divergence instead of a jointly convex one), then convexity with respect to X-variables is lost. However, the model is still convex in X_l for a given l when all the other variables are fixed; this still allows for block-coordinate descent algorithms to be used.

As a specific example, consider a multi-class classification problem where all the layers involve ReLUs except for the last. The last layer aims at producing a probability distribution to be compared against training labels via a cross entropy loss function. The training problem

writes

$$\begin{aligned}\min_{(W_l,b_l),(X_l)} \; & -\mathbf{Tr}\, Y^T \log s(W_L X_L + b_L \mathbf{1}^T) + \sum_{l=0}^{L} \rho_l \|W_l\|_F^2 \\ & + \sum_{l=0}^{L-1} \lambda_{l+1} \|X_{l+1} - W_l X_l - b_l \mathbf{1}^T\|_F^2 \\ & \text{s.t. } X_0 = X, \;\; X_l \geq 0, \;\; l = 1, \ldots, L-1 \end{aligned} \tag{2.11}$$

where the equality constraint on X_{L+1} enforces that its columns are probability distributions. Here, $s(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}^n$ is the softmax function. We can always rescale the variables so that in fact the number of additional hyper-parameters λ_l, $l = 1, \ldots, L-1$, is reduced to just one.

Lifted prediction rule

In our model, the prediction rule will be different from that of a standard neural network, but it is based on the same principle. In a standard network, the prediction rule can be obtained by solving the problem

$$\hat{y}(x) = \min_y \; \mathcal{L}(y, x_{L+1}) \;\; : \;\; (2.2), \;\; x_0 = x,$$

where the weights are now *fixed*, and $y \in \mathbb{R}^p$ is a *variable*. Of course, provided the loss is zero whenever its two arguments coincide, the above trivially reduces to the standard prediction rule: $\hat{y}(x) = x_{L+1}$, where x_{L+1} is obtained via the recursion (2.2).

In a lifted framework, we use the same principle: solve the training problem (in our case, (2.10)), using the test point as input, fixing the weights, and letting the predicted output values be variables. In other words, the prediction rule for a given test point x in lifted networks is based on solving the problem

$$\begin{aligned}\hat{y} = \arg\min_{y,(x_l)} \; & \mathcal{L}(y, W_L x_L + b_L) \\ & + \sum_{l=0}^{L-1} \lambda_{l+1} D_l(W_l x_l + b_l, x_{l+1}) \\ & \text{s.t. } x_0 = x. \end{aligned} \tag{2.12}$$

The above prediction rule is a simple convex problem in the variables y and x_l, $l = 1, \ldots, L$. In our experiments, we have found that applying the standard feedforward rule of traditional networks is often enough.

2.6 Block-Coordinate Descent Algorithm

In this section, we outline a block-coordinate descent approach to solve the training problem (2.10).

Updating (W, b)-variables

For fixed X-variables, the problem of updating the W-variables, *i.e.* the weighting matrices $(W_l, b_l)_{l=0}^{L}$, is parallelizable across both data points and layers. The sub-problem involving updating the weights at a given layer $l = 0, \ldots, L$ takes the form

$$(W_l^+, b_l^+) = \arg\min_{W,b} \ \lambda_{l+1} D_l(WX_l + b\mathbf{1}^T, X_{l+1}) + \pi_l(W).$$

The above is a convex problem, which can be solved via standard machine learning libraries. Since the divergences are sums across columns (data points), the above problem is indeed parallelizable across data points.

For example, when the activation function at layer l is a ReLU, and the penalty π_l is a squared Frobenius norm, the above problem reads

$$(W_l^+, b_l^+) = \arg\min_{W,b} \ \lambda_{l+1} \|WX_l + b\mathbf{1}^T - X_{l+1}\|_F^2 + \rho_l \|W\|_F^2$$

which is a standard (matrix) ridge regression problem. Modern sketching techniques for high-dimensional least-squares can be employed, see for example [108, 84].

Updating X-variables

In this step we minimize over the matrices $(X_l)_{l=1}^{L+1}$. The sub-problem reads exactly as (2.10), with now the (W, b)-variables fixed. By construction of divergences, the problem is decomposable across data points. When JCR conditions hold, the joint convexity of each JC-divergence function allows us update all the X-variables at once, by solving a convex problem. Otherwise, the update must be done cyclically over each layer, in a block-coordinate fashion.

For $l = 1, \ldots, L$, the sub-problem involving X_l, with all the other X-variables X_j, $j \neq l$ fixed, takes the form

$$\begin{aligned} X_l^+ = \arg\min_{Z} \ & \lambda_{l+1} D_l(W_l Z + b_l \mathbf{1}^T, X_{l+1}) \\ & + \lambda_l D_{l-1}(Z, X_{l-1}^0) \end{aligned} \tag{2.13}$$

where $X_{l-1}^0 := W_{l-1}X_{l-1} + b_{l-1}\mathbf{1}^T$. By construction, the above is a convex problem, and is again parallelizable across data points.

Let us detail this approach in the case when the layers $l, l+1$ are both activated by ReLUs. The sub-problem above becomes

$$\begin{aligned} X_l^+ = \arg\min_{Z \geq 0} \ & \lambda_{l+1} \|X_{l+1} - W_l Z - b_l \mathbf{1}^T\|_F^2 + \\ & \lambda_l \|Z - W_{l-1}X_{l-1} - b_{l-1}\mathbf{1}^T\|_F^2 \end{aligned}$$

The above is a (matrix) non-negative least-squares, for which many modern methods are available, see [53, 54] and references therein. As before, the problem above is fully parallelizable across data points (columns), where each data point gives rise to a standard (vector)

non-linear least-squares. Note that the cost of updating all columns can be reduced by taking into account that all column's updates share the same coefficient matrix W_l.

The case of updating the last matrix X_{L+1} is different, as it involves the output and the loss function $\mathcal{L}$. The update rule for X_{L+1} is indeed

$$X_{L+1}^+ = \arg\min_Z \ \mathcal{L}(Y, Z) + \lambda_{L+1} D_L(X_L^0, Z), \tag{2.14}$$

where $X_L^0 := W_L X_L + b_L \mathbf{1}^T$. Again the above is parallelizable across data points.

In the case when the loss function $\mathcal{L}$ is a squared Frobenius norm, and with a ReLU activation, the update rule (2.14) takes the form

$$X_{L+1} = \arg\min_{Z \geq 0} \ \|Z - Y\|_F^2 + \lambda_L \|Z - X_L^0\|_F^2,$$

which can be solved analytically:

$$X_{L+1}^+ = \max\left(0, \frac{1}{1+\lambda_{L+1}} Y + \frac{\lambda_{L+1}}{1+\lambda_{L+1}} X_L^0\right).$$

In the case when the loss function is cross-entropy, and the last layer generates a probability distribution via the probability simplex projection, the above takes the form

$$\begin{aligned} X_{L+1} = \arg\min_Z \ & -\mathbf{Tr}\, Y^T \log Z + \lambda_{L+1} \|Z - X_L^0\|_F^2 \\ & Z \geq 0, \ \ Z^T \mathbf{1} = \mathbf{1} \end{aligned} \tag{2.15}$$

where we use the notation log in a component-wise fashion. The above can be solved as a set of parallel bisection problems. See Appendix A.

2.7 Numerical Experiments

Although lifted models in their own right can be used for supervised learning tasks, their main success so far has been using them to initialize traditional networks. In this section, we examine this and see if the lifted models can generate good initial guesses for standard networks.

MNIST

The model described in this paper was compared against a traditional neural network with equivalent architectures on the MNIST dataset [63]. For the classification problem, the dataset was split into 60,000 training samples and 10,000 test samples with a softmax cross entropy loss. This is a similar model to the one specified in (2.5), with the only difference that the last layer loss is changed from an ℓ_2 loss to a softmax cross entropy loss as seen in (2.11). In addition to comparing the models, the weights and biases learned in the augmented

Learning rate η = 1e–5

Architecture	Our Model	NN[Normal]	NN[Xavier]	NN [σ-scale]	NN [Lifted]
300	0.90 ± 0.01	0.92 ± 0.01	0.92 ± 0.01	0.92 ± 0.00	$\mathbf{0.96 \pm 0.00}$
300/100	0.88 ± 0.01	0.92 ± 0.00	0.93 ± 0.00	0.93 ± 0.00	$\mathbf{0.97 \pm 0.00}$
500/150	0.87 ± 0.01	0.93 ± 0.00	0.94 ± 0.00	0.94 ± 0.01	$\mathbf{0.97 \pm 0.01}$
500/200/100	0.85 ± 0.00	0.93 ± 0.00	0.94 ± 0.01	0.94 ± 0.00	$\mathbf{0.96 \pm 0.01}$
400/200/100/50	0.77 ± 0.02	0.92 ± 0.01	$\mathbf{0.94 \pm 0.00}$	0.94 ± 0.01	0.92 ± 0.03

Learning rate η =1e–6

Architecture	Our Model	NN[Normal]	NN[Xavier]	NN [σ^2-scale]	NN [Lifted]
300	0.90 ± 0.01	0.80 ± 0.01	0.84 ± 0.01	0.84 ± 0.01	$\mathbf{0.88 \pm 0.02}$
300/100	0.88 ± 0.01	0.79 ± 0.01	0.84 ± 0.01	0.84 ± 0.01	$\mathbf{0.90 \pm 0.02}$
500/150	0.87 ± 0.01	0.82 ± 0.01	0.85 ± 0.00	0.86 ± 0.00	$\mathbf{0.89 \pm 0.02}$
500/200/100	0.85 ± 0.00	0.82 ± 0.02	0.86 ± 0.01	0.85 ± 0.01	$\mathbf{0.93 \pm 0.05}$
400/200/100/50	0.77 ± 0.02	0.75 ± 0.05	0.84 ± 0.02	0.82 ± 0.02	$\mathbf{0.96 \pm 0.00}$

Table 2.1: Accuracy rate on the test set using different networks with the best result in boldface. The architectures indicate the number of hidden layers and the number of hidden units per layer. NN[x] indicates a standard neural network initialized with method x: *Normal* for normally distributed intialization of all weight variables with $\mu = 0$ and $\sigma^2 = 0.1$, *Xavier* for initialization highlighted in [40], σ^2-*scale* for variance scaling initialization and *Lifted* for initializing with the weights and biases learned from a lifted NN. All bias variables were initialized to 0.1 except for the Lifted case in which the bias vectors are optimized during pretraining. The neural networks were trained for 17 epochs using mini-batch gradient descent in Tensorflow [71]. The lifted model achieves test accuracy as high as 90 % on MNIST.

neural network were used as initialization parameters for training a standard neural net of the same architecture to compare their performance, both in classification and convergence during training. For all models, ReLU activations were used. ℓ_2 regularization was used for all layers and the regularization parameters $\rho = 10^{-3}$ were held constant throughout all training procedures. The λ parameters for the lifted model were selected using Bayesian Optimization. The lifted model was trained using the block-coordinate descent scheme outlined in Section 2.6. The standard feedforward networks were trained in Tensorflow using a constant learning rate; reasons for this are highlighted in [107]. Table 1 summarizes the accuracy rates for the different architectures for 2 different learning rates. Figure 2.1 illustrates the test set accuracy versus number of epochs for two different architectures.

Remark 1. *Although our model does not perform as well as the other models on this task, using it as initialization results in increased accuracy for almost all network architectures.*

In particular, in Figure 2.1 we see that with our initialization, the test accuracy both

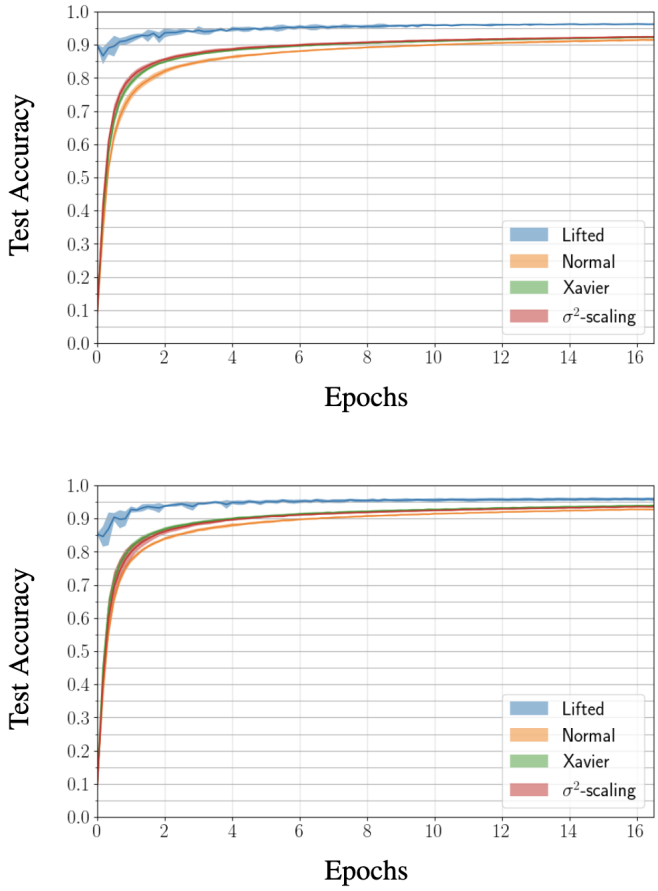

Figure 2.1: Plot of test accuracy vs number of training epochs on a held-out validation set during training for two different architectures. The shaded area on the plots indicated uncertainty to 2 standard deviations across 5 different experiments. The batch size was fixed at 100 and the learning rate was $\eta = 1\text{e}{-5}$. **Top:** One layer neural network with 300 hidden units and ReLU activation. **Bottom:** Neural network composed of 3 ReLU layers with 500, 200, and 100 hidden units respectively.

converges more quickly and to higher values compared with the other initializations: in fact, across all experiments the lifted initlization starts within 90% of its final accuracy. This seems to indicate that the lifted model we train on is a close approximation to a standard feedforward network and our weights learned are already near optimal for these networks. Although after a few passes of the dataset the other models converge, we usually observed a constant gap between the test set accuracy using our initialization versus the others.

2.8 Solving for the last layer with cross entropy loss

In this section, we consider problem (15), which is of the form

$$\min_Z \ -\mathbf{Tr}\, Y^T \log Z + \lambda \|Z - X^0\|_F^2 \ : \ Z^T\mathbf{1} = \mathbf{1}, \ \ Z \geq 0, \tag{2.16}$$

where we use the notation log in a component-wise fashion, and $X^0 \in \mathbb{R}^{p\times m}$ and $Y \in \{0,1\}^{p\times m}$, $Y^T\mathbf{1} = \mathbf{1}$, $\lambda > 0$ are given. The above can be easily solved by dual matrix bisection. Indeed, the problem can be decomposed across columns of Y (that is, across data points). The problem for a single column has the following form:

$$p^* := \min_z \ -\sum_{i=1}^p y_i \log z_i + \lambda \|z - x^0\|_2^2 \ : \ z \geq 0, \ \ z^T\mathbf{1} = 1,$$

where vectors $y \in \{0,1\}^p$, $y^T\mathbf{1} = 1$ and $x^0 \in \mathbb{R}^p$ are given. Dualizing the equality constraint, we obtain a Lagrangian of the form

$$\mathcal{L}(z,\nu) = 2\nu + \sum_{i=1}^p \left(z_i^2 - \frac{y_i}{\lambda}\log z_i - 2z_i(\nu + x_i^0)\right),$$

where ν is a (scalar) dual variable. At the optimum z^*, we have

$$\forall\, i \ : \ 0 = \frac{1}{2}\frac{\partial \mathcal{L}(z,\nu)}{\partial z_i}(z^*,\nu) = z_i^* - \frac{y_i}{2\lambda z_i^*} - (\nu + x_i^0),$$

leading to the unique non-negative solution

$$z_i^* = \frac{x_i^0 + \nu}{2} + \sqrt{\left(\frac{x_i^0+\nu}{2}\right)^2 + \frac{y_i}{2\lambda}}, \ \ i = 1,\ldots,p,$$

where the dual variable ν is such that $\mathbf{1}^T z^* = 1$. We can locate such a value ν by simple bisection.

The bisection scheme requires initial bounds on ν. For the upper bound, we note that the property $z^* \leq \mathbf{1}$, together with the above optimality condition, implies

$$\nu \leq 1 - \max_{1\leq i\leq p}\left(x_i^0 + \frac{y_i}{2\lambda}\right).$$

For the lower bound, let us first define $\mathcal{I} := \{i \ : \ y_i \neq 0\}$, $k = |\mathcal{I}| \leq p$. At optimum, we have

$$\forall\, i \in \mathcal{I} \ : \ \ -\log z_i^* \leq -\sum_{j\in\mathcal{I}} y_j \log z_j^* \leq p^* \leq \theta,$$

$$\theta := -\sum_{i\in\mathcal{I}} y_i \log z_i^0 + \lambda\|z^0 - x^0\|_2^2$$

where $z^0 \in \mathbb{R}^p$ is any primal feasible point, for example $z_i^0 = 1/k$ if $i \in \mathcal{I}$, 0 otherwise. We obtain

$$\forall\, i \in \mathcal{I} \;:\; z_i^* \geq z_{\min} := e^{-\theta}.$$

The optimality conditions imply

$$0 = \frac{1}{2}\sum_i \frac{\partial \mathcal{L}(z^*, \nu)}{\partial z_i} = -\frac{1}{2\lambda}\sum_{i\in\mathcal{I}} \frac{y_i}{z_i^*} + 1 - \mathbf{1}^T x^0 - p\nu$$

and therefore:

$$p\nu = 1 - \mathbf{1}^T x^0 - \frac{1}{2\lambda}\sum_{i\in\mathcal{I}} \frac{y_i}{z_i^*} \geq 1 - \mathbf{1}^T x^0 - \frac{\mathbf{1}^T y}{2\lambda} e^{\theta}.$$

To conclude, we have

$$\frac{1}{p}\left(1 - \mathbf{1}^T x^0 - \frac{\mathbf{1}^T y}{2\lambda} e^{\theta}\right) =: \underline{\nu} \leq \nu \leq \overline{\nu} := 1 - \max_{1\leq i\leq p} \left(x_i^0 + \frac{y_i}{2\lambda}\right).$$

To solve the original (matrix) problem (2.16), we can process all the columns in parallel (matrix) fashion, updating a *vector* $\nu \in \mathbb{R}^m$.

Chapter 3

Fenchel Neural Networks

3.1 Introduction

Deep neural networks (DNNs) have become the preferred model for supervised learning tasks after their success in various fields of research. However, due to their highly non-convex nature, DNNs pose a difficult problem during training time; the optimization landscape consists of many saddle points and local minima which make the trained model generalize poorly [21, 23]. This has motivated regularization schemes such as weight decay [59], batch normalization [47], and dropout [92] so that the solutions generalize better to the test data.

In spite of this, backprop used along with stochastic gradient descent (SGD) or similar variants like Adam [57] suffer from a variety of problems. One of the most notable problems is the vanishing gradient problem which slows down gradient-based methods during training time. Several approaches have been proposed to deal with the problem; for example, the introduction of rectified linear units (ReLU). However, the problem persists. For a discussion on the limitations of backprop and SGD, we direct the reader to Section 2.1 of [97].

One approach to deal with this problem is to introduce auxiliary variables that increase the dimension of the problem. In doing so, the training problem decomposes into multiple, local sub-problems which can be solved efficiently without using SGD or Adam; in particular, the methods of choice have been block coordinate descent (BCD) [3, 61, 114, 19] and the alternating direction method of multipliers (ADMM) [97, 115]. By lifting the dimension of the problem, these models avoid many of the problems DNNs face during training time. They also offer new avenues towards interpretability and robustness of networks by allowing for the penalization of the additional variables.

While these methods, which we refer to as "lifted" models for the remainder of the paper, offer an alternative to the original problem with some added benefits, they have their limitations. Most notably, traditional DNNs are still able to outperform these methods in spite of the difficult optimization landscape. As well, most of the methods are unable to operate in an online manner or adapt to continually changing data sets which is prevalent in most reinforcement learning settings [95]. Finally, by introducing auxiliary variables, the

dimensionality of the problem greatly increases, making these methods very difficult to train with limited computational resources.

To address the problems listed above, we propose Fenchel lifted networks, a biconvex formulation for deep learning based on Fenchel's duality theorem that can be optimized using BCD. We show that our method is a rigorous *lower bound* for the learning problem and admits a natural batching scheme to adapt to changing data sets and settings with limited computational power. We compare our method against other lifted models and against traditional fully connected and convolutional neural networks. We show that we are able to outperform the former and that we can compete with or even outperform the latter.

Chapter outline. In Section 3.2, we give a brief overview of related works on lifted models. In Section 3.3 we introduce the notation for the remainder of the paper. Section 3.4 introduces Fenchel lifted networks, their variants and discusses how to train these models using BCD. Section 3.5 compares the proposed method against fully connected and convolutional networks on MNIST and CIFAR-10.

3.2 Related Work

Lifted methods Related works that lift the dimension of the training problem are primarily optimized using BCD or ADMM. These methods have experienced recent success due to their ability to exploit the structure of the problem by first converting the constrained optimization problem into an unconstrained one and then solving the resulting sub-problems in parallel. They do this by relaxing the network constraints and introducing penalties into the objective function. The two main ways of introducing penalties into the objective function are either using quadratic penalties [94, 97, 61] or using equivalent representations of the activation functions [3, 114].

As a result, these formulations have many advantages over the traditional training problem, giving superior performance in some specific network structures [19, 114]. These methods also enjoy great potential for parallelization as shown by [97]; the authors parallelize training over different cores and show a linear scaling between reduction in training time and the number of cores used. However, there has been little evidence showing that these methods can compete with traditional DNNs which shadows the nice structure these formulations bring about.

An early example of auxiliary variables being introduced into the training problem is the method of auxiliary coordinates (MAC) by [19] which uses quadratic penalties to enforce network constraints. They test their method on auto encoders and show that their method is able to outperform SGD. Followup work by [20, 97] demonstrate the huge potential for parallelizing these methods. [61] gives some convergence guarantee on a modified problem.

Another class of models that lift the dimension of the problem do so by representing activation functions in equivalent formulations. [75, 3, 114, 68] explore the structure of activation functions and use $\arg\min$ maps to represent activation functions. In particular,

[3] show how a strictly monotone activation function can be seen as the arg min of a specific optimization problem. Just as with quadratic penalties, this formulation of the problem still performs poorly compared to traditional neural networks. In an independent and concurrent work by [68], the authors arrive at a lifted formulation of the training problem via the use of proximal operators. While the approach in aforementioned work appears unrelated to the work presented here, they are in fact deeply related (for a more complete discussion, see 3.9).

3.3 Background and Notation

Feedforward neural networks. We are given an input data matrix of m data points $X = [x_1, x_2, ..., x_m] \in \mathbb{R}^{n\times m}$ and a response matrix $Y \in \mathbb{R}^{p\times m}$. We consider the supervised learning problem involving a neural network with $L \geq 1$ hidden layers. The neural network produces a prediction $\hat{Y} \in \mathbb{R}^{p\times m}$ with the feed forward recursion $\hat{Y} = W_L X_L + b_L \mathbf{1}^\top$ given below

$$X_{l+1} = \phi_l(W_l X_l + b_l \mathbf{1}^\top), \quad l = 0, \ldots, L-1. \tag{3.1}$$

where $\phi_l, l = 0, \ldots, L$ are the activation functions that act column-wise on a matrix input, $\mathbf{1} \in \mathbb{R}^m$ is a vector of ones, and $W_l \in \mathbb{R}^{p_{l+1}\times p_l}$ and $b_l \in \mathbb{R}^{p_{l+1}}$ are the weight matrices and bias vectors respectively. Here p_l is the number of output values for a single data point (i.e., hidden nodes) at layer l with $p_0 = n$ and $p_{L+1} = p$. Without loss of generality, we can remove $b_l \mathbf{1}^\top$ by adding an extra column to W_l and a row of ones to X_l. Then (3.1) simplifies to

$$X_{l+1} = \phi_l(W_l X_l), \quad l = 0, \ldots, L-1. \tag{3.2}$$

In the case of fully connected networks, ϕ_l is typically sigmoidal activation functions or ReLUs. In the case of Convolutional Neural Networks (CNNs), the recursion can accommodate convolutions and pooling operations in conjunction with an activation. For classification tasks, we typically apply a softmax function after applying an affine transformation to X_L.

The initial value for the recursion is $X_0 = X$ and $X_l \in \mathbb{R}^{p_l\times m}$, $l = 0, \ldots, L$. We refer to the collections $(W_l)_{l=0}^L$ and $(X_l)_{l=1}^L$ as the W and X-variables respectively.

The weights are obtained by solving the following constrained optimization problem

$$\begin{aligned} \min_{(W_l)_{l=0}^L, (X_l)_{l=1}^L} \ & \mathcal{L}(Y, W_L X_L) + \sum_{l=0}^{L} \rho_l \pi_l(W_l) \\ \text{s.t.} \ & X_{l+1} = \phi_l(W_l X_l), \ l = 0, \ldots, L-1 \\ & X_0 = X \end{aligned} \tag{3.3}$$

Here, $\mathcal{L}$ is a loss function, $\rho \in \mathbb{R}_+^{L+1}$ is a hyper-parameter vector, and π_l's are penalty functions used for regularizing weights, controlling network structures, etc. In (3.3), optimizing over the X-variables is trivial; we simply apply the recursion (3.2) and solve the resulting

unconstrained problem using SGD or Adam. After optimizing over the weights and biases, we obtain a prediction $\hat{Y}$ for the test data X by passing X through the recursion (3.2) one layer at a time.

Our model. We develop a family of models where we approximate the recursion constraints (3.2) via penalties. We use the arg min maps from [3] to create a biconvex formulation that can be trained efficiently using BCD and show that our model is a lower bound of (3.3). Furthermore, we show how our method can naturally be batched to ease computational requirements and improve the performance.

3.4 Fenchel lifted networks

In this section, we introduce Fenchel lifted networks. We begin by showing that for a certain class of activation functions, we can equivalently represent them as biconvex constraints. We then dualize these constraints and construct a lower bound for the original training problem. We show how our lower bound can naturally be batched and how it can be trained efficiently using BCD.

Activations as bi-convex constraints

In this section, we show how to convert the equality constraints of (3.3) into inequalities which we dualize to arrive at a *relaxation* (lower bound) of the problem. In particular, this lower bound is biconvex in the W-variables and X-variables. We make the following assumption on the activation functions ϕ_l.

> **BC Condition** The activation function $\phi : \mathbb{R}^p \to \mathbb{R}^q$ satisfies the BC condition if there exists a *biconvex* function $B_\phi : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}_+$, such that
>
> $$v = \phi(u) \iff B_\phi(v, u) \leq 0.$$

We now state and prove a result that is at the crux of Fenchel lifted networks.

Theorem 2. *Assume $\phi : \mathbb{R} \to \mathbb{R}$ is continuous, strictly monotone and that $0 \in$ range(ϕ) or $0 \in$ domain(ϕ). Then ϕ satisfies the BC condition.*

Proof. Without loss of generality, ϕ is strictly increasing. Thus it is invertible and there exists ϕ^{-1} such that $u = \phi^{-1}(v)$ for $v \in \text{range}(\phi)$ which implies $v = \phi(u)$. Now, define $F : \mathbb{R}^p \to \mathbb{R}$ as

$$F(v) := \int_z^v \phi^{-1}(\xi)\, d\xi$$

where $z \in \text{range}(\phi)$ and is either 0 or satisfies $\phi^{-1}(z) = 0$. Then we have

$$F^*(u) = \int_{\phi^{-1}(z)}^{u} \phi(\eta)\, d\eta$$
$$B(v, u) = F(v) + F^*(u) - uv \tag{3.4}$$

where F^* is the Fenchel conjugate of F. By the Fenchel-Young inequality, $B(v, u) \geq 0$ with equality if and only if

$$v^* = \arg\max_v \; uv - F(v) \;:\; v \in \text{range}(\phi)$$

By construction, $v^* = \phi(u)$. Note furthermore since ϕ is continuous and strictly increasing, so is ϕ^{-1} on its domain, and thus F, F^* are convex. It follows that $B(v, u)$ is a biconvex function of (u, v).

We simply need to prove that $F^*(u)$ above is indeed the Fenchel conjugate of F. By definition of the Fenchel conjugate we have that

$$F^*(u) = \max_v \; uv - F(v) \;:\; v \in \text{range}(\phi)$$

It is easy to see that $v^* = \phi(u)$. Thus

$$\begin{aligned} F^*(u) &= u\phi(u) - F(\phi(u)) \\ &= u\phi(u) - \int_z^{\phi(u)} \phi^{-1}(\xi)\, d\xi \\ &= \int_z^{\phi(u)} \xi \frac{d}{d\xi}\phi^{-1}(\xi)\, d\xi \\ &= \int_{\phi^{-1}(z)}^{u} \phi(\eta)\, d\eta \end{aligned}$$

where the third equality is a consequence of integration by parts, and the fourth equality we make the subsitution $\eta = \phi^{-1}(\xi)$ □

Note that Theorem 2 implies that activation functions such as sigmoid and tanh can be equivalently written as a biconvex constraint. Although the ReLU is not strictly monotone, we can simply restrict the inverse to the domain $\mathbb{R}_+$; specifically, for $\phi(x) = \max(0, x)$ define

$$\phi^{-1}(z) = \begin{cases} +\infty & \text{if } z < 0, \\ z & \text{if } z \geq 0, \end{cases}$$

Then, we can rewrite the ReLU function as the equivalent set of biconvex constraint

$$v = \max(0, u) \iff \begin{cases} \frac{1}{2}v^2 + \frac{1}{2}u_+^2 - uv \leq 0 \\ v \geq 0 \end{cases}$$

where $u_+ = \max(0, u)$. This implies

$$B_\phi(v,u) = \begin{cases} \frac{1}{2}v^2 + \frac{1}{2}u_+^2 - uv & \text{if } v \geq 0 \\ +\infty & \text{otherwise} \end{cases} \tag{3.5}$$

Despite the non-smoothness of u_+, for fixed u or fixed v, (3.5) belongs in C^1 – that is, it has continuous first derivative and can be optimized using first order methods. We can trivially extend the result of Theorem 2 for matrix inputs: for matrices $U, V \in \mathbb{R}^{p\times q}$, we have

$$B_\phi(V,U) = \sum_{i,j} B_\phi(V_{ij}, U_{ij}).$$

Lifted Fenchel model

Assuming the activation functions of (3.3) satisfy the hypothesis of Theorem 2, we can reformulate the learning problem equivalently as

$$\begin{aligned} \min_{(W_l)_{l=0}^L, (X_l)_{l=1}^L} \; & \mathcal{L}(Y, W_L X_L) + \sum_{l=0}^{L} \rho_l \pi_l(W_l) \\ \text{s.t. } & B_l(X_{l+1}, W_l X_l) \leq 0, \; l = 0, \ldots, L-1 \\ & X_0 = X, \end{aligned} \tag{3.6}$$

where B_l is the short-hand notation of B_{ϕ_l}. We now dualize the inequality constraints and obtain the lower bound of the standard problem (3.3) via Lagrange relaxation

$$\begin{aligned} G(\lambda) := \min_{(W_l)_{l=0}^L, (X_l)_{l=1}^L} \; & \mathcal{L}(Y, W_L X_L) + \sum_{l=0}^{L} \rho_l \pi_l(W_l) \\ & + \sum_{l=0}^{L-1} \lambda_l B_l(X_{l+1}, W_l X_l) \\ \text{s.t. } & X_0 = X, \end{aligned} \tag{3.7}$$

where $\lambda_l \geq 0$ are the Lagrange multipliers. The maximum lower bound can be achieved by solving the dual problem

$$p^* \geq d^* = \max_{\lambda \geq 0} G(\lambda) \tag{3.8}$$

where p^* is the optimal value of (3.3). Note if all our activation functions are ReLUs, we must also include the constraint $X_l \geq 0$ in the training problem as a consequence of (3.5). Although the new model introduces L new parameters (the Lagrange multipliers), we can

show that using variable scaling we can reduce this to only *one* hyperparameter (for details, see 3.6). The learning problem then becomes

$$\begin{aligned} G(\lambda) := \min_{(W_l)_{l=0}^L, (X_l)_{l=1}^L} & \mathcal{L}(Y, W_L X_L) + \sum_{l=0}^{L} \rho_l \pi_l(W_l) \\ & + \lambda \sum_{l=0}^{L-1} B_l(X_{l+1}, W_l X_l) \\ \text{s.t. } & X_0 = X. \end{aligned} \tag{3.9}$$

In a regression setting where the data is generated by a one layer network, we are able to provide global convergence guarantees of the above model (for details, see 3.7).

Comparison with other methods. For ReLU activations, $B(v, u)$ as in (3.5) differs from the penalty terms introduced in previous works. In [3, 114] they set $B(v, u) = \|v - u\|_2^2$ and in [97, 19] they set $B(v, u) = \|v - u_+\|_2^2$. Note that $B(v, u)$ in the latter is not biconvex. While the $B(v, u)$ in the former is biconvex, it does not perform well at test time. [68] set $B(v, u)$ based on a proximal operator that is similar to the BC condition.

Convolutional model. Our model can naturally accommodate average pooling and convolution operations found in CNNs, since they are linear operations. We can rewrite $W_l X_l$ as $W_l * X_l$ where $*$ denotes the convolution operator and write $\text{Pool}(X)$ to denote the average pooling operator on X. Then, for example, the sequence Conv $\to$ Activation can be represented via the constraint

$$B_l(X_{l+1}, W_l * X_l) \leq 0, \tag{3.10}$$

while the sequence Pool $\to$ Conv $\to$ Activation can be represented as

$$B_l(X_{l+1}, W_l * \text{Pool}(X_l)) \leq 0. \tag{3.11}$$

Note that the pooling operation changes the dimension of the matrix.

Prediction rule.

In previous works that reinterpret activation functions as arg min maps [3, 114], the prediction at test time is defined as the solution to the optimization problem below

$$\begin{aligned} \hat{y} = \arg\min_{y, (x_l)} & \mathcal{L}(y, W_L x_L) + \lambda \sum_{l=0}^{L-1} B_l(x_{l+1}, W_l x_l) \\ \text{s.t. } & x_0 = x, \end{aligned} \tag{3.12}$$

where x_0 is test data point, $\hat{y}$ is the predicted value, and x_l, $l = 1, \ldots, L$ are the intermediate representations we optimize over. Note if $\mathcal{L}$ is a mean squared error, applying the traditional feed-forward rule gives an optimal solution to (3.12). We find empirically that applying the standard feed-forward rule works well, even with a cross-entropy loss.

Batched model

The models discussed in the introduction usually require the entire data set to be loaded into memory which may be infeasible for very large data sets or for data sets that are continually changing. We can circumvent this issue by batching the model. By sequentially loading a part of the data set into memory and optimizing the network parameters, we are able to train the network with limited computational resources. Formally, the batched model is

$$\begin{aligned} \min_{(W_l)_{l=0}^L, (X_l)_{l=1}^L} \quad & \mathcal{L}(Y, W_L X_L) + \sum_{l=0}^{L} \rho_l \pi_l(W_l) \\ +\lambda \sum_{l=0}^{L-1} B_l(W_l X_l, X_{l+1}) & + \sum_{l=0}^{L} \gamma_l \|W_l - W_l^0\|_F^2 \\ & \text{s.t. } X_0 = X, \end{aligned} \tag{3.13}$$

where X_0 contains only a batch of data points instead of the complete data set. The additional term in the objective $\gamma_l \|W_l - W_l^0\|_F^2, \quad l = 0, \ldots, L$ is introduced to moderate the change of the W-variables between subsequent batches; here W_l^0 represents the optimal W variables from the previous batch and $\gamma \in \mathbb{R}_+^{L+1}$ is a hyperparameter vector. The X-variables are reinitialized each batch by feeding the new batch forward through the equivalent standard neural network.

Block-coordinate descent algorithm

The model (3.9) satisfies the following properties:

- For fixed W-variables, and fixed variables $(X_j)_{j \neq l}$, the problem is convex in X_l, and is decomposable across data points.
- For fixed X-variables, the problem is convex in the W-variables, and is decomposable across layers.

The non-batched and batched Fenchel lifted network are trained using block coordinate descent algorithms highlighted in Algorithms 1 and 2. By exploiting the biconvexity of the problem, we can alternate over updating the X-variables and W-variables to train the network.

Note Algorithm 2 is different from Algorithm 1 in three ways. First, re-initialization is required for the X-variables each time a new batch of data points are loaded. Second,

Algorithm 1 Non-batched BCD Algorithm

Initialize $(W_l)_{l=0}^L$.
Initialize X_0 with input matrix X.
Initialize $X_1, \ldots, X_L$ with neural network feed forward rule.
repeat
  $X_L \leftarrow \arg\min_Z \ \mathcal{L}(Y, W_L Z) + \lambda B_{L-1}(Z, X_{L-1}^0)$
  for $l = L-1, \ldots, 1$ **do**
    $X_l \leftarrow \arg\min_Z \ B_l(X_{l+1}, W_l Z) + B_{l-1}(Z, X_{l-1}^0)$
  end for
  $W_L \leftarrow \arg\min_W \ \mathcal{L}(Y, W X_L) + \rho_l \pi_l(W)$
  for $l = L-1, \ldots, 0$ **do**
    $W_l \leftarrow \arg\min_W \ \lambda B_l(X_{l+1}, W X_l) + \rho_l \pi_l(W)$
  end for
until convergence

Algorithm 2 Batched BCD Algorithm

Initialize $(W_l)_{l=0}^L$.
repeat
  $(W_l^0)_{l=0}^L \leftarrow (W_l)_{l=0}^L$
  Re-initialize X_0 with a batch sampled from input matrix X.
  Re-initialize $X_1, \ldots, X_L$ with neural network feed forward rule.
  for alternation$= 1, \ldots, K$ **do**
    $X_L \leftarrow \arg\min_Z \ \mathcal{L}(Y, W_L Z) + \lambda B_{L-1}(Z, X_{L-1}^0)$
    for $l = L-1, \ldots, 1$ **do**
      $X_l \leftarrow \arg\min_Z \ \lambda B_l(X_{l+1}, W_l Z) + \lambda B_{l-1}(Z, X_{l-1}^0)$
    end for
    $W_L \leftarrow \arg\min_W \ \mathcal{L}(Y, W X_L) + \rho_L \pi_L(W) + \gamma_l \|W - W_L^0\|_F^2$
    for $l = L-1, \ldots, 0$ **do**
      $W_l \leftarrow \arg\min_W \ \lambda B_l(X_{l+1}, W X_l) + \rho_l \pi_l(W) + \gamma_l \|W - W_l^0\|_F^2$
    end for
  end for
until convergence

the sub-problems for updating W-variables are different as shown in Section 3.4. Lastly, an additional parameter K is introduced to specify the number of training alternations for each batch. Typically, we set $K = 1$.

Updating X-variables

For fixed W-variables, the problem of updating X-variables can be solved by cyclically optimizing $X_l, \quad l = 1, \ldots, L$, with $(X_j)_{j \neq l}$ fixed. We initialize our X-variables by feeding forward through the equivalent neural network and update the X_l's backward from X_L to X_1 in the spirit of backpropagation.

We can derive the sub-problem for $X_l, \quad l = 1, \ldots, L-1$ with $(X_j)_{j \neq l}$ fixed from (3.6). The sub-problem writes

$$X_l^+ = \arg\min_Z B_l(X_{l+1}, W_l Z) + B_{l-1}(Z, X_{l-1}^0) \tag{3.14}$$

where $X_{l-1}^0 := W_{l-1} X_{l-1}$. By construction, the sub-problem (3.14) is convex and parallelizable across data points. Note in particular when our activation is a ReLU, the objective function in (3.14) is in fact strongly convex and has a continuous first derivative.

For the last layer (i.e., $l = L$), the sub-problem derived from (3.6) writes differently

$$X_L^+ = \arg\min_Z \mathcal{L}(Y, W_L Z) + \lambda B_{L-1}(Z, X_{L-1}^0) \tag{3.15}$$

where $X_{L-1}^0 := W_{L-1} X_{L-1}$. For common losses such as mean square error (MSE) and cross-entropy, the subproblem is convex and parallelizable across data points. Specifically, when the loss is MSE and we use a ReLU activation at the layer before the output layer, (3.15) becomes

$$X_L^+ = \arg\min_{Z \geq 0} ||Y - W_L Z||_F^2 + \frac{\lambda}{2} ||Z - X_{L-1}^0||_F^2$$

where $X_{L-1}^0 := W_{L-1} X_{L-1}$ and we use the fact that X_{L-1}^0 is a constant to equivalently replace B_{L-1} as in (3.5) by a squared Frobenius term. The sub-problem is a non-negative least squares for which specialized methods exist [54].

For a cross-entropy loss and when the second-to-last layer is a ReLU activation, the sub-problem for the last layer takes the convex form

$$\begin{aligned} X_L^+ = &\arg\min_{Z \geq 0} - \mathbf{Tr}\, Y^\top \log s(W_L Z) + \\ &\frac{\lambda}{2} ||Z - X_{L-1}^0||_F^2, \end{aligned} \tag{3.16}$$

where $s(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ is the softmax function and log is the element-wise logarithm. [3] show how to solve the above problem using bisection.

Updating W-variables

With fixed X-variables, the problem of updating the W-variables can be solved in parallel across layers.

Sub-problems for non-batched model. The problem of updating W_l at intermediate layers becomes

$$W_l = \arg\min_W \lambda B_l(X_{l+1}, WX_l) + \rho_l \pi_l(W). \tag{3.17}$$

Again, by construction, the sub-problem (3.17) is convex and parallelizable across data points. Also, since there is no coupling in the W-variables between layers, the sub-problem (3.17) is parallelizable across layers.

For the last layer, the sub-problem becomes

$$W_L = \arg\min_W \mathcal{L}(Y, WX_L) + \rho_L \pi_L(W). \tag{3.18}$$

Sub-problems for batched model. As shown in Section 3.4, the introduction of regularization terms between W and values from a previous batch require the sub-problems (3.17, 3.18) be modified. (3.17) now becomes

$$\begin{aligned} W_l = \arg\min_W \ & \lambda B_l(X_{l+1}, WX_l) + \rho_l \pi_l(W) \\ & + \gamma_l \|W - W_l^0\|_F^2, \end{aligned} \tag{3.19}$$

while (3.18) becomes

$$\begin{aligned} W_L = \arg\min_W \ & \mathcal{L}(Y, WX_L) + \rho_L \pi_L(W) \\ & + \gamma_L \|W - W_L^0\|_F^2. \end{aligned} \tag{3.20}$$

Note that these sub-problems in the case of a ReLU activation are strongly convex and parallelizable across layers.

3.5 Numerical Experiments

In this section, we compare Fenchel lifted networks against other lifted models discussed in the introduction and against traditional neural networks. In particular, we compare our model against the models proposed by [97], [61] and [3] on MNIST. Then we compare Fenchel lifted networks against a fully connected neural network and LeNet-5 [65] on MNIST. Finally, we compare Fenchel lifted networks against LeNet-5 on CIFAR-10. For a discussion on hyperparameters and how model paramters were selected, see Appendix 3.8.

Fenchel lifted networks vs. lifted models

Here, we compare the non-batched Fenchel lifted network against the models proposed by [97][1], [61][2] and [3]. The former model is trained using ADMM and the latter ones using

[1]Code available in `https://github.com/PotatoThanh/ADMM-NeuralNetworks`

[2]Code available in `https://github.com/deeplearning-math/bcd_dnn`

the BCD algorithms proposed in the respective papers. In Figure 3.1, we compare these models on MNIST with a 784-300-10 architecture (inspired by [65]) using a mean square error (MSE) loss.

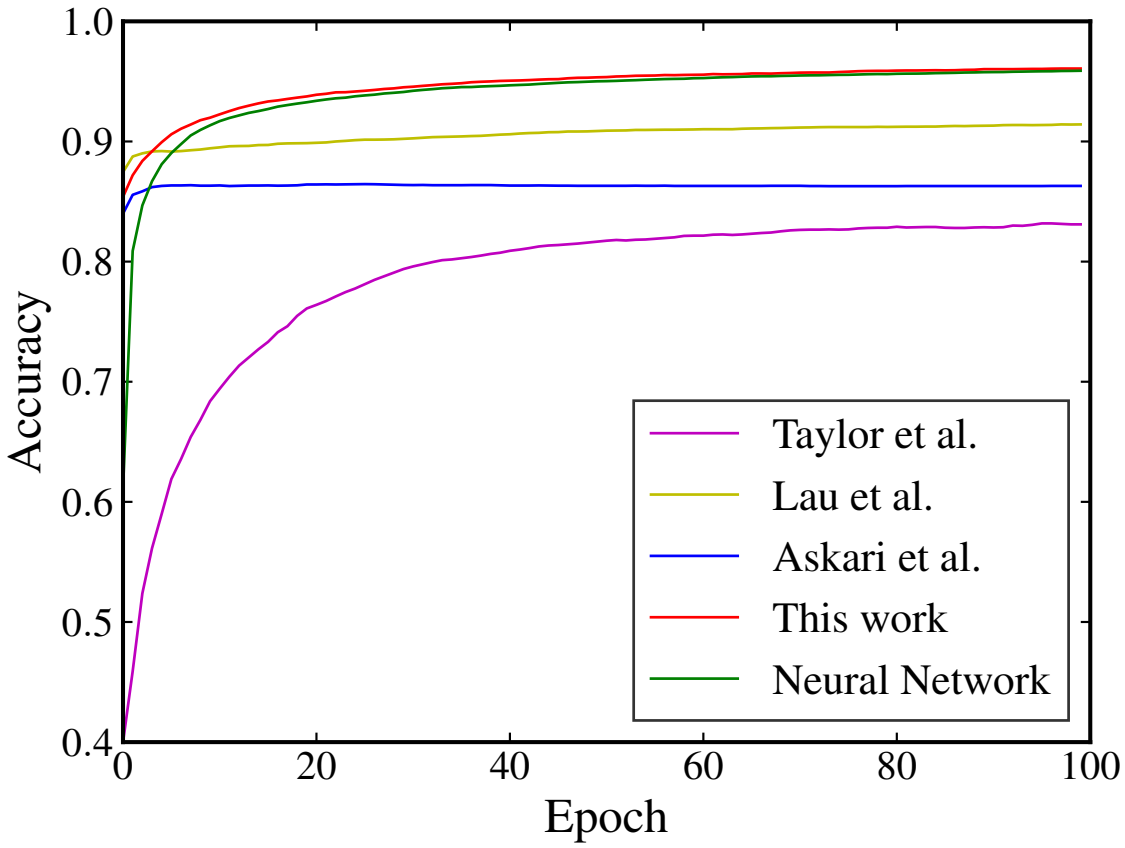

Figure 3.1: Test set performance of different lifted methods with a 784-300-10 network architecture on MNIST with a MSE loss. Final test set performances: **Taylor et al.** 0.834, **Lau et al.** 0.914, **Askari et al.** 0.863, **Neural Network** 0.957, **This work** 0.961.

After multiple iterations of hyperparameter search with little improvement over the base model, we chose to keep the hyperparameters for [97] and [61] as given in the code. The hyperparameters for [3] were tuned using cross validation on a hold-out set during training. Our model used these same parameters and cross validated the remaining hyperparameters. The neural network model was trained using SGD. The resulting curve of the neural network is smoothed in Figure 3.1 for visual clarity. From Figure 3.1 it is clear that Fenchel lifted networks vastly outperform other lifted models and achieve a test set accuracy on par with traditional networks.

Fenchel lifted networks vs. neural networks on MNIST

For the same 784-300-10 architecture as the previous section, we compare the batched Fenchel lifted networks against traditional neural networks trained using first order methods. We use a cross entropy loss in the final layer for both models. The hyperparameters for our model are tuned using cross validation. Figure 3.2 shows the results.

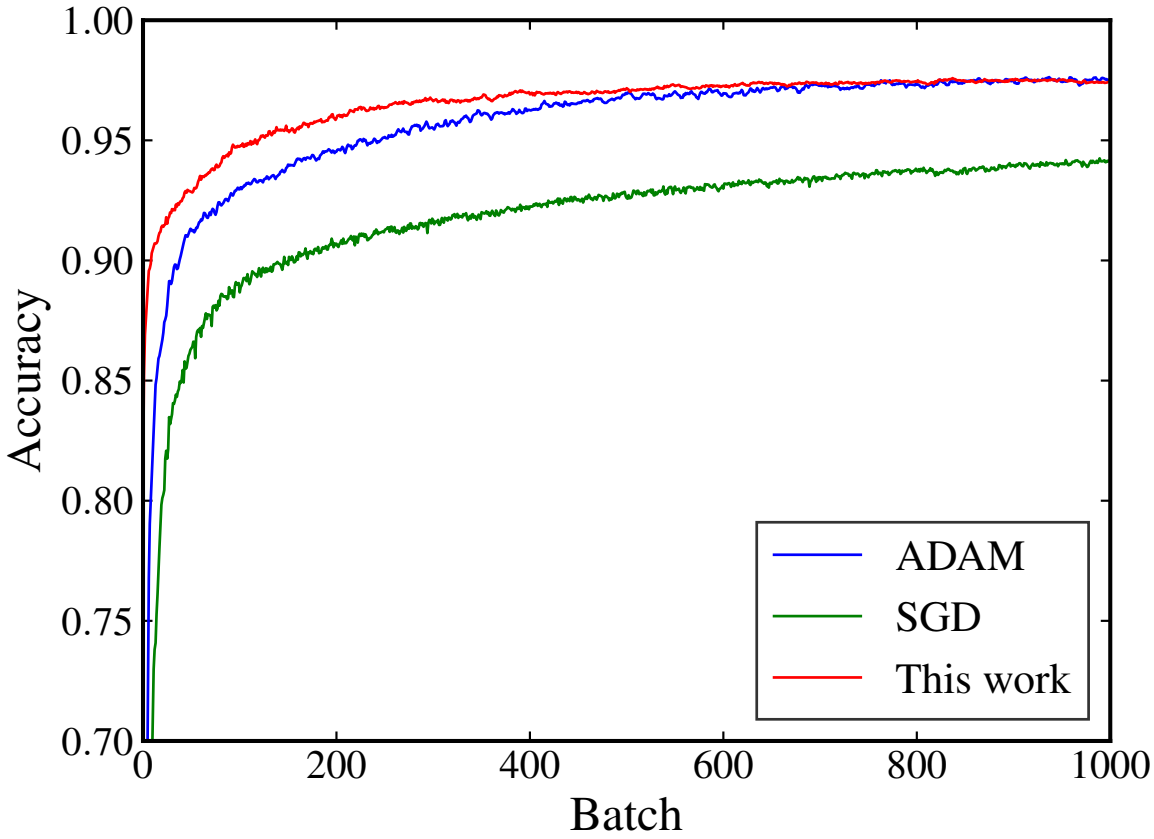

Figure 3.2: Test set performance of Fenchel lifted networks and fully connected networks trained using Adam and SGD on a 784-300-10 network architecture on MNIST with cross entropy loss. Total training time was 10 epochs. Final test set performances: **SGD** 0.943, **Adam** 0.976, **This work** 0.976.

As shown in Figure 3.2, Fenchel lifted networks learn faster than traditional networks as shown by the red curve being consistently above the blue and green curve. Although not shown, between batch 600 and 1000, the accuracy on a training batch would consistently hit 100% accuracy. The advantage of the Fenchel lifted networks is clear in the early stages of

training, while towards the end the test set accuracy and the accuracy of an Adam-trained network converge to the same values.

We also compare Fenchel lifted networks against a LeNet-5 convolutional neural network on MNIST. The network architecture is 2 convolutional layers followed by 3 fully-connected layers and a cross entropy loss on the last layer. We use ReLU activations and average pooling in our implementation. Figure 3.3 plots the test set accuracy for the different models.

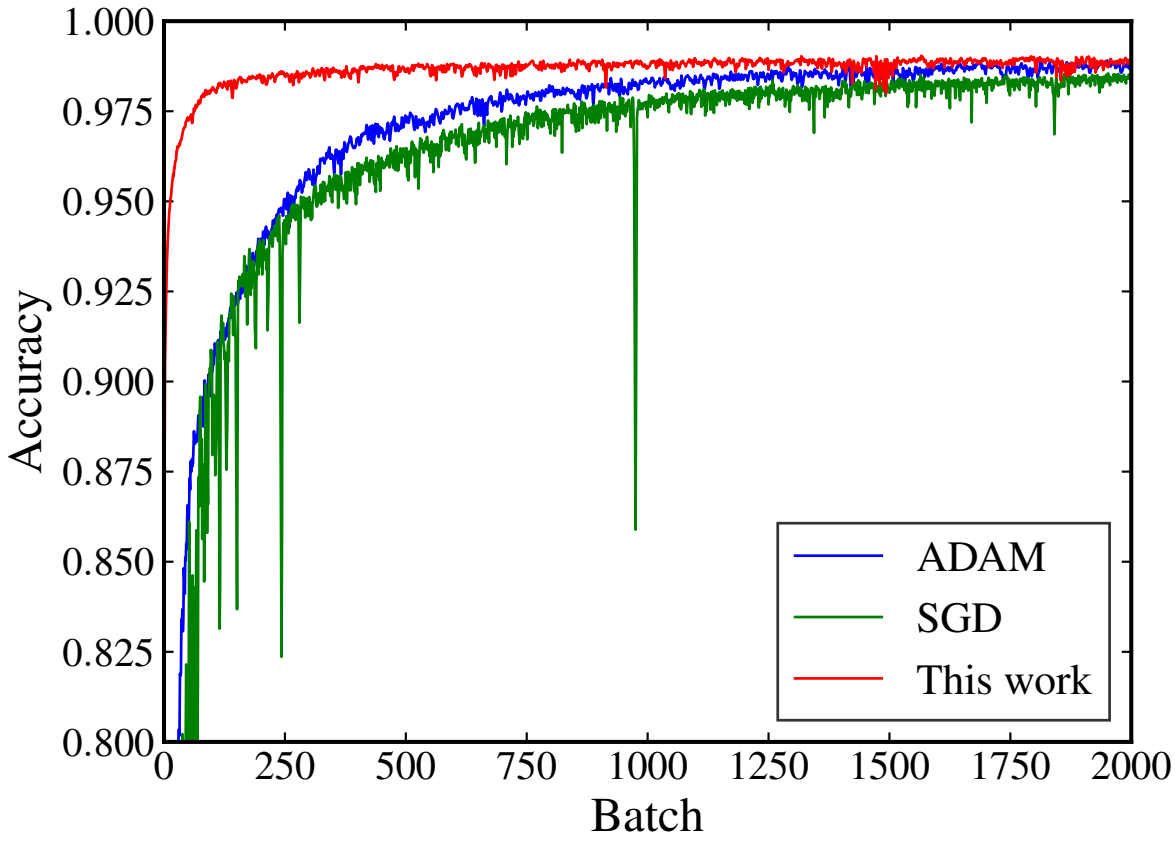

Figure 3.3: Test set performance of Fenchel lifted networks and LeNet-5 trained using Adam and SGD on MNIST with a cross entropy loss. Total training time was 20 epochs. Final test set performances: **SGD** 0.986, **Adam** 0.989, **This work** 0.990.

In Figure 3.3, our method is able to nearly converge to its final test set accuracy after only 2 epochs while Adam and SGD need the full 20 epochs to converge. Furthermore, after the first few batches, our model is attaining over 90% accuracy on the test set while the other methods are only at 80%, indicating that our model is doing something different (in

a positive way) compared to traditional networks, giving them a clear advantage in test set accuracy.

Fenchel lifted networks vs CNN on CIFAR-10

In this section, we compare the LeNet-5 architechture and with Fenchel lifted networks on CIFAR-10. Figure 3.4 compares the accuracies of the different models.

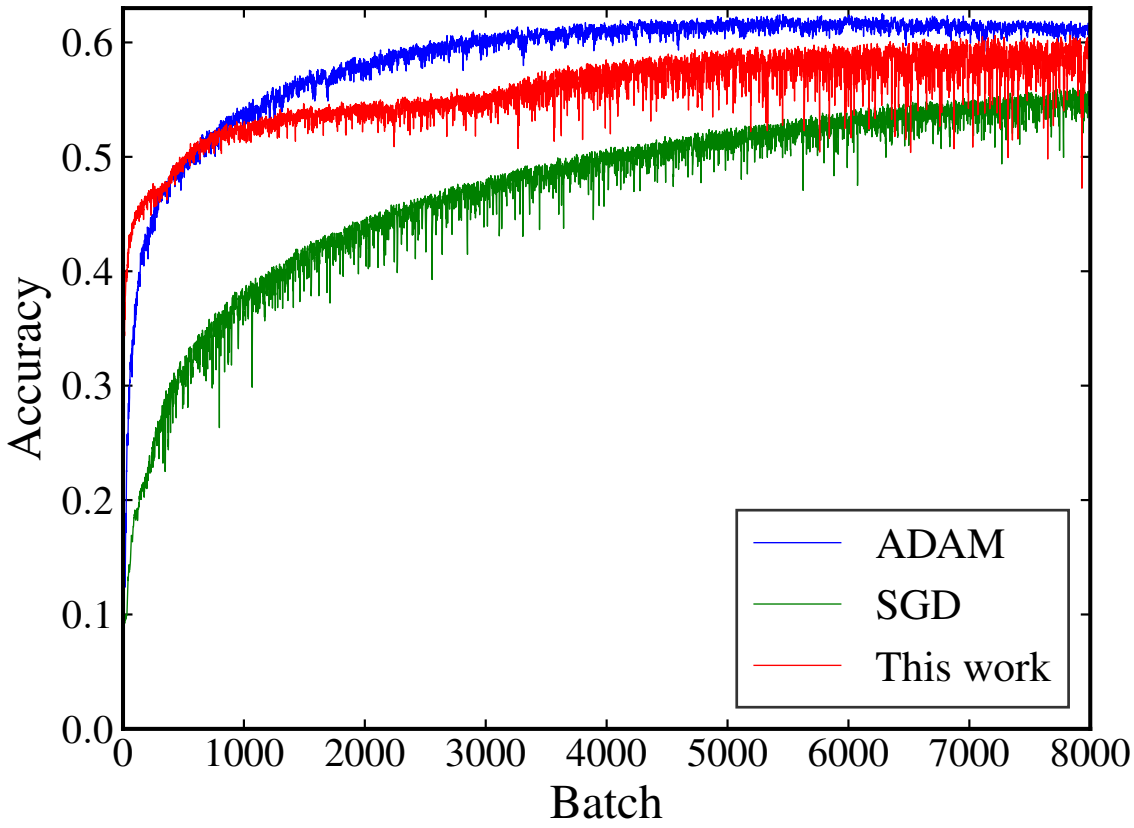

Figure 3.4: Test set performance of Fenchel lifted networks and LeNet-5 trained using Adam and SGD on CIFAR-10 with a cross entropy loss. Total training time was 80 epochs. Final test set performance: **SGD** 0.565, **Adam** 0.625, **This work** 0.606

In this case, the Fenchel lifted network still outperforms the SGD trained network and only slightly under performs compared to the Adam trained network. The larger variability in the accuracy per batch for our model can be attributed to the fact that in this experiment, when updating the W-variables, we would only take one gradient step instead of solving

(3.19) and (3.20) to completion. We did this because we found empirically solving those respective sub-problems to completion would lead to poor performance at test time.

3.6 Variable Scaling

Note that the new model (3.9) has introduced $L+1$ more hyperparameters. We can use variable scaling and the dual formulation to show how to effectively reduce this to only *one* hyperparameter. Consider the model with ReLU activations, that is, the biconvex function as in (3.5) and regularization functions $\pi_l(W_l) = \|W_l\|_F^2$ for $l = 0, \dots, L$. Note that B_ϕ is homogeneous of degree 2, that is for any U, V and γ we have

$$\gamma B_\phi(V, U) = B_\phi(\sqrt{\gamma}V, \sqrt{\gamma}U)$$

Define $\lambda_{-1} = 1$ and the scalings

$$\bar{X}_l := \sqrt{\lambda_{l-1}} X_l, \;\; \bar{W}_l := \sqrt{\frac{\lambda_l}{\lambda_{l-1}}} W_l,$$

Then (3.9) becomes

$$\begin{aligned} G(\lambda) := \min_{(\bar{W}_l)_{l=0}^{L}, (\bar{X}_l)_1^{L+1}} \; & \mathcal{L}(Y, \sqrt{\lambda_L}(\bar{W}_L \bar{X}_L)) \\ & + \sum_{l=0}^{L} \rho_l \pi_l(\sqrt{\frac{\lambda_{l-1}}{\lambda_l}} W_l) + \sum_{l=0}^{L-1} B_l(\bar{X}_{l+1}, \bar{W}_l \bar{X}_l) \\ \text{s.t. } & \bar{X}_0 = X, \; \bar{X}_l \geq 0, \; l = 0, \dots, L \end{aligned} \tag{3.21}$$

Using the fact $\pi_l(W_l) = \|W_l\|_F^2$ and defining $\bar{\rho}_l = \rho_l \dfrac{\lambda_{l-1}}{\lambda_l}$ we have

$$\begin{aligned} G(\lambda) := \min_{(\bar{W}_l)_{l=0}^{L}, (\bar{X}_l)_1^{L+1}} \; & \mathcal{L}(Y, \sqrt{\lambda_L}(\bar{W}_L \bar{X}_L)) \\ & + \sum_{l=0}^{L} \bar{\rho}_l \|\bar{W}_l\|_F^2 + \sum_{l=0}^{L-1} B_l(\bar{X}_{l+1}, \bar{W}_l \bar{X}_l) \\ \text{s.t. } & \bar{X}_0 = X, \; \bar{X}_l \geq 0, \; l = 0, \dots, L \end{aligned} \tag{3.22}$$

where $G(\lambda)$ is now only a function of one variable λ_L as opposed to L variables. Note that this argument for variable scaling still works when we use average pooling or convolution operations in conjunction with a ReLU activation since they are linear operations. Note furthermore that the same scaling argument works in place of any norm due to the homogeneity of norms – the only thing that would change is how $\bar{\rho}$ is scaled by λ_{l-1} and λ_l.

Another way to show that we only require one hyperparameter λ is to note the equivalence

$$B_l(v,u) \leq 0 \ \ \forall l \iff \sum_l B_l(v,u) \leq 0$$

Then we may replace the L biconvex constraints in (3.6) by the equivalent constraint $\sum_l B_l(v,u) \leq 0$. Since this is only one constraint, when we dualize we only introduce *one* Lagrange multiplier λ.

3.7 One-layer Regression Setting

In this section, we show that for a one layer network we are able to convert a non-convex optimization problem into a convex one by using the BC condition described in the main text.

Consider a regression setting where $Y = \phi(W^*X)$ for some fixed $W^* \in \mathbb{R}^{p\times n}$ and a given data matrix $X \in \mathbb{R}^{n\times m}$. Given a training set (X,Y) we can solve for W by solving the following non-convex problem

$$\min_W \ \|Y - \phi(WX)\|_F^2. \tag{3.23}$$

We could also solve the following relaxation of (3.23) based on the BC condition

$$\min_W \ B_\phi(Y, WX) \tag{3.24}$$

Note (3.24) is trivially convex in W by definition of $B_\phi(\cdot,\cdot)$. Furthermore, by construction $B_\phi(Y,WX) \geq 0$ and $B_\phi(Y,WX) = 0$ if and only if $Y = \phi(WX)$. Since $Y = \phi(W^*X)$, it follows W^* (which is the minimizer of (3.23)) is a global minimizer of the convex program (3.24). Therefore, we can solve the original non-convex problem (3.23) to global optimality by instead solving the convex problem presented in (3.24).

3.8 Hyperparameters for Experiments

For all experiments that used batching, the batch size was fixed at 500 and $K = 1$. We observed empirically that larger batch sizes improved the performance of the lifted models. To speed up computations, we set $K = 1$ and empirically find this does not affect final test set performance. For batched models, we do not use $\pi_l(\cdot)$ since we explicitly regularize through batching (see (3.13)) while for the non-batched models we set $\pi_l(W_l) = \|W_l\|_F^2$ for all l. For models trained using Adam, the learning rate was set to $\eta = 10^{-3}$ and for models trained using SGD, the learning rate was set to $\eta = 10^{-2}$. The learning rates were a hyperparamter that we picked from $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ to give the best final test performance for both

Adam and SGD.

For the network architechtures described in the experimental results, we used the following hyperparamters:

- Fenchel Lifted Network for LeNet-5 architecture
  1. $\rho_1 = 1e-4, \lambda_1 = 5$
  2. $\rho_2 = 1e-2, \lambda_2 = 5$
  3. $\rho_3 = 1, \quad \lambda_3 = 1$
  4. $\rho_4 = 1, \quad \lambda_4 = 1$
  5. $\rho_5 = 1$
- Fenchel Lifted Network for 784-300-10 architecture (batched)
  1. $\rho_1 = 1, \quad \lambda_1 = 0.1$
  2. $\rho_2 = 100$
- Fenchel Lifted Network for 784-300-10 architecture (non-batched)
  1. $\rho_1 = 1e-2, \ \lambda_1 = 0.1$
  2. $\rho_2 = 10$

For all weights the initialization is done through Xavier initialization implemented in TensorFlow. The ρ variables are chosen to balance the change of variables across layers in iterations. Although the theory in Appendix A states we can collapse all λ hyperparameters into a single hyperparameter, due to time constraints, we were unable to implement this change upon submission. We also stress that the hyperparamter search over the ρ's were very coarse and a variety of ρ values worked well in practice; for simplicity we only present the ones we used to produce the plots in the experimental results.

3.9 Fenchel Conjugates and Proximal Operators

Here we discuss the similarities between [68] and the approach of this paper (for simplicity, we only concern ourselves with the ReLU activation since it is convex). In what follows, when we refer to equation numbers, they are the equation numbers in [68]. First we derive an elementary result relating conjugate functions and proximal operators.

Lemma 1. *Let $\lambda > 0$ and let $f(x)$ be a closed, convex and proper function. Define $\tilde{f}(x) = \lambda f(x) + \frac{1}{2}\|x\|_2^2$ and let $f^*(y)$ be the fenchel conjugate of $f(x)$. Furthermore, define the proximal operator as $\boldsymbol{prox}_{\lambda f}(x) = \arg\min_y f(y) + \frac{1}{2\lambda}\|x-y\|_2^2$ and for a given x, let $y^*(x) = \arg\max_y x^\top y - \tilde{f}(y)$. Then $\boldsymbol{prox}_{\lambda f}(x) = y^*(x)$.*

Proof.

$$\begin{aligned}\arg\min_y f(y) + \frac{1}{2\lambda}\|y - x\|_2^2 &= \arg\min_y f(y) + \frac{1}{2\lambda}\|x\|_2^2 - \frac{1}{\lambda}x^\top y + \frac{1}{2\lambda}\|y\|_2^2 \\ &= \arg\min_y \left(\lambda f(y) + \frac{1}{2}\|y\|_2^2\right) - x^\top y \\ &= \arg\max_y x^\top y - \tilde{f}(y)\end{aligned}$$

The left hand side is exactly $\texttt{prox}_{\lambda f}(x)$ and the right hand side is exactly $y^*(x)$. Note furthermore that the problem defining $\texttt{prox}_{\lambda f}(x)$ is strongly convex and hence there is only one unique global optima and similarly for the problem defining $y^*(x)$. □

The above lemma shows the natural connection between proximal operators and fenchel conjugates. We now highlight this in the case of the ReLU function $\phi(x) = \max(0, x)$ and make the connection explicit. Below we consider the scalar case, and the multivariate case is a simple generalization of the argument below.

As in [68], if we set $f(x) = \int_0^x \phi^{-1}(z) - z \; dz$ as defined below (11) and set $g(x) = \int_0^x \phi(z) - z \, dz$ as defined below (18) in the aforementioned reference and, we then have

$$\begin{aligned}f(x) &= \int_0^x \phi^{-1}(z) - z \; = 0 \\ g(x) &= \int_0^x \phi(z) - z \, dz = \frac{1}{2}\max(x, 0)^2 - \frac{1}{2}x^2\end{aligned}$$

where we use the fact that $\phi^{-1}(z) = z$ for $z \in [0, \infty)$ and set $\phi^{-1}(z) = +\infty$ otherwise. Modulo hyperparameters in their objective function, the term inside the summand in (18) (in the scalar case), reduces to

$$\begin{aligned}&f(x^i) + \frac{1}{2}(x^i - w^{i-1}x^{i-1})^2 + g(w^{i-1}x^{i-1}) \\ =&0 + \frac{1}{2}(x^i - w^{i-1}x^{i-1})^2 + \frac{1}{2}(w^{i-1}x^{i-1})_+^2 - \frac{1}{2}(w^{i-1}x^{i-1})^2 \\ =&\frac{1}{2}(x^i)^2 - \langle w^{i-1}x^{i-1}, x^i\rangle + \frac{1}{2}(w^{i-1}x^{i-1})_+^2 \\ =&B_\phi(x^i, w^{i-1}x^{i-1})\end{aligned}$$

Hence

$$B_\phi(v, u) = f(v) + \frac{1}{2}\|v - u\|_2^2 + g(u)$$

As a result, the term in the summand the authors use in (18) is equivalent to the fenchel lifted formulation.

Chapter 4

Fast Knockoffs

4.1 Introduction

Feature selection is a key preprocessing step in prediction tasks. Pruning out irrelevant variables both improves test performance by reducing noise and helps interpretation by focusing the prediction task on a short list of important variables. In many cases, the variable selection step is in fact more important than the prediction itself. The tradeoff between prediction performance and model size is typically very favorable. However, feature selection needs to select among an exponential number of hypotheses (the subset of selected variables) using a limited number of samples, and is thus naturally exposed to false discoveries. A lot of effort has been focused on controlling the false discovery rate (FDR) in feature selection, with notably [**Benj95**] controlling FDR using p-values. These results work well in settings where p-values are readily available and has been extended, in part, to more sophisticated feature selection procedures in what is known as post selection inference (see e.g. [**Berk13**, **Lee16**]). This requires computing p-values after complex prediction tasks, which is far from trivial.

A more flexible alternative is provided by the *knockoff framework* developed in [**Barb15**, **Cand18**, **Barb19**]. In this setting, we first generate *knockoff* covariates whose distribution roughly matches that of the true covariates, except that knockoffs are designed to be conditionally independent of the response, and hence should never be selected by a feature selection procedure. This last fact helps in controlling the false discovery rate. The procedure in [**Cand18**] shows how to design knockoffs in the Gaussian case and requires solving a semidefinite program (SDP) of dimension p equal to the ambient dimension. While the knockoff framework does not explicitly control power, the SDP optimally decorrelates true covariates and their knockoff, which empirically improves power. The current package provided by the authors of [**Cand18**] uses generic interior point methods (IPM), which scale roughly as $\mathcal{O}(p^{4.5})$, which can be reduced to $\mathcal{O}(p^{3.5})$ using problem structure [13]. Feature selection is naturally a high dimensional problem, making generic IPM solvers ill suited for the task. Simple tricks produce simple feasible solutions to the knockoff SDP, but at the

expense of a loss in power. Clustering the covariance matrix also allows [**Cand18**] to solve much larger problems, but the limitations on maximum block size remains.

In this chapter we propose *Fast Knockoffs* (FANOK) where we use problem structure to derive a block coordinate descent method and solve a barrier formulation as in, *e.g.* [22, 105]. Iterations require low rank Cholesky updates which can be handled efficiently. This allows us to produce a first algorithm which handles generic covariance matrices, and has a complexity scaling as $\mathcal{O}(p^3)$ where p is the ambient dimension. We then derive another method which assumes a rank k factor (low rank) model on the covariance matrix to reduce this complexity bound to $\mathcal{O}(pk^2)$. The low rank algorithm is potentially unstable in very particular scenarios, but we do not observe instabilities in practice. We also derive efficient procedures to both estimate factor models and sample knockoff covariates with complexity linear in the dimension. We test our methods on problems with p as large as 500 000.

Notation

Let $[p] = \{1, \dots, p\}$. Given $M \in \mathbb{R}^{p \times p}$ and two sets of indices $I, J \subseteq [p]$, $M_{I,J}$ denotes the $|I| \times |J|$ matrix obtained by keeping the $|I|$ rows and $|J|$ columns indexed by I and J respectively. For simplicity, an integer j denotes the set $\{j\}$, j^c denotes the set $[p] \setminus \{j\}$ and ":" denotes either all rows or columns in the matrix subscript context. For example,

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \implies M_{1^c,1^c} = \begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix}, \quad M_{1^c,1} = \begin{bmatrix} 4 \\ 7 \end{bmatrix}, \quad M_{1^c,:} = \begin{bmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad M_{1,1} = 1\,.$$

For $s \in \mathbb{R}^p$, $D = \mathbf{diag}(s)$ denotes a $p \times p$ diagonal matrix with $D_{ii} = s_i$. For $M \in \mathbb{R}^{p \times p}$, $m = \mathbf{diag}(M)$ denotes a vector in $\mathbb{R}^p$ with $m_i = M_{ii}$. Unless otherwise stated, x_j and x'_j denote the j^{th} column and row of a matrix X respectively. $\mathbf{S}_p$ denotes the set of $p \times p$ symmetric matrices.

Primer on Knockoffs

Given random covariates and a random response $(x, y) \in \mathbb{R}^p \times \mathbb{R}$, the knockoff framework of [**Barb15**, **Cand18**, **Barb19**] seeks to control the false discovery rate in feature selection by constructing a new family of random variables $\tilde{x} \in \mathbb{R}^p$ called *knockoffs* which have a joint distribution comparable to their counterparts x but are independent of the response y. As a result, these knockoff variables should not be selected by any reasonable feature selection procedure. The knockoff framework controls the FDR by keeping the features which are more strongly selected than their knockoff counterpart (which usually requires solving a LASSO-type problem; see Section 3.2 of [**Cand18**]).

More specifically, the *model-X knockoff* framework of [**Cand18**] formally defines knockoffs as a new family of random variables $\tilde{x} \in \mathbb{R}^p$ such that $\tilde{x} \perp\!\!\!\perp y \mid x$, and for any $S \subset [p]$, we

have

$$\begin{bmatrix} x \\ \tilde{x} \end{bmatrix}_{\text{swap}(S)} \stackrel{d}{=} \begin{bmatrix} x \\ \tilde{x} \end{bmatrix},$$

where $[x^\top,\ \tilde{x}^\top]^\top_{\text{swap}(S)}$ is obtained from $[x^\top,\ \tilde{x}^\top]^\top$ by swapping the jth entries of x and $\tilde{x}$ for all $j \in S$. In the Gaussian case where $x \sim \mathcal{N}(\mu_{\text{pop}}, \Sigma_{\text{pop}})$ this invariance property means that $[x^\top,\ \tilde{x}^\top]^\top$ is also Gaussian with covariance matrix given by

$$\begin{bmatrix} \Sigma_{\text{pop}} & \Sigma_{\text{pop}} - \mathbf{diag}(s) \\ \Sigma_{\text{pop}} - \mathbf{diag}(s) & \Sigma_{\text{pop}} \end{bmatrix} \succeq 0$$

for some $s \in \mathbb{R}^p$ such that the matrix is positive semidefinite (PSD), *i.e.*, such that $0 \preceq \mathbf{diag(s)} \preceq 2\Sigma_{\text{pop}}$.

Without loss of generality, we assume that $\mu_{\text{pop}} = 0$. Since the population covariance matrix Σ_{pop} is never observed in practice, we instead use an estimate Σ from the observed samples. By rescaling the data, without loss of generality we assume Σ is a correlation matrix. Given an observation x, Gaussian knockoffs are then sampled from the conditional distribution $\tilde{x} \mid x \sim \mathcal{N}(\mu, \Omega)$ such that

$$\begin{aligned} \mu &= x - \mathbf{diag}(s)\Sigma^{-1}x\,, \\ \Omega &= \mathbf{diag}(s)\left(2\mathbf{I}_p - \Sigma^{-1}\,\mathbf{diag}(s)\right). \end{aligned} \tag{4.1}$$

For the remainder of the paper, let $X \in \mathbb{R}^{p\times n}$ denote the *scaled* data matrix for the response vector $y \in \mathbb{R}^n$. After we sample all the knockoffs and aggregate them into the knockoff matrix $\tilde{X} \in \mathbb{R}^{p\times n}$, we compute a feature statistic in order to do feature selection. Intuitively, we want to construct knockoffs that are not "too similar" to the original features (*i.e.*, with low $\mathbf{E}\,\langle x_i', \tilde{x}_i'\rangle = 1 - s_i$). To do so, we maximize the entries of s, solving the following SDP

$$\begin{array}{ll} \text{maximize} & \mathbf{1}^\top s \\ \text{subject to} & \mathbf{diag}(s) \preceq 2\Sigma \\ & 0 \le s \le 1 \end{array} \tag{4.2}$$

In this paper, we are concerned with solving (4.2) as efficiently as possible.

4.2 Solving for Second Order Knockoffs

Solving the semidefinite program in (4.2) using generic interior point methods [76, 45, 13] has complexity $\mathcal{O}(p^{4.5})$ or $\mathcal{O}(p^{3.5})$ exploiting structure, which precludes their use on large-scale examples. In what follows, we will describe a coordinate ascent method that better exploits the structure of the problem. Each barrier problem has complexity $\mathcal{O}(p^3)$, but when the covariance matrix Σ is assumed to have a *diagonal plus low-rank* (aka factor model) structure, this complexity can be reduced to $\mathcal{O}(pk^2)$ where $k \ll p$. Most of the results in this section require the estimated covariance Σ to be positive definite (we require all its

principal submatrices to be invertible). Hence, we will assume in this section that $\Sigma \succ 0$. This is the case almost surely when $n \geq p$, but Σ is not invertible when $p > n$. In this later scenario, we use the Ledoit-Wolf estimator which is always positive definite (see Section 4.2 for more details).

A Basic Coordinate Ascent Algorithm

Here, as in [8, 106], we exploit the fact that the feasible set of program (4.2) has a product structure amenable to block coordinate ascent to derive an efficient algorithm for maximizing a barrier formulation of (4.2) written

$$\begin{array}{ll} \text{maximize} & \mathbf{1}^\top s + \lambda \, \log\det\left(2\Sigma - \mathbf{diag}(s)\right) \\ \text{subject to} & 0 \leq s \leq 1 \end{array} \tag{4.3}$$

in the variable $s \in \mathbb{R}^p$, where $\lambda > 0$ is a barrier parameter. Note that the dual of (4.2) writes

$$\begin{array}{ll} \text{minimize} & 2\,\mathbf{Tr}(\Lambda\Sigma) + \mathbf{1}^\top \eta \\ \text{subject to} & \mathbf{diag}(\Lambda) + \eta \geq 1 \\ & \Lambda \succeq 0, \ \eta \geq 0 \end{array}$$

and could be solved by adapting the block-coordinate method as in [106]. Here however, we solve the primal problem (4.2) for two reasons. First, in the barrier formulation, the primal is a lower dimensional convex problem than the dual (p versus $\mathcal{O}(p^2)$) with non-coupling constraints. Second, we are focused on getting solutions of the primal variables, not the dual variables. Hence we focus on a block coordinate algorithm for solving (4.3). We first recall the following key fact.

Lemma 2. *For any symmetric, invertible matrix $M \in \boldsymbol{S}_p$ and any $j \in [p]$,*

$$\det(M) = \left(M_{j,j} - M_{j^c,j}^\top M_{j^c,j^c}^{-1} M_{j^c,j}\right) \cdot \det\left(M_{j^c,j^c}\right) .$$

On the barrier problem (4.3), Lemma 2 yields

$$\log\det\left(2\Sigma - \mathbf{diag}(s)\right) = \log\left(2\Sigma_{j,j} - s_j - 4\Sigma_{j^c,j}^\top Q_j^{-1} \Sigma_{j^c,j}\right) + \log\det\left(Q_j\right) ,$$

where $Q_j = 2\Sigma_{j^c,j^c} - \mathbf{diag}(s_{j^c})$ does not depend on s_j. Using this decomposition, maximizing over s_j in (4.3) and leaving all other entries fixed, the first order optimality condition gives

$$s_j^\star := \min\left(1, \max\left(2\Sigma_{j,j} - 4\Sigma_{j^c,j}^\top Q_j^{-1}\Sigma_{j^c,j} - \lambda, 0\right)\right) . \tag{4.4}$$

Applying this result iteratively yields the block coordinate ascent method detailed in Algorithm 3.

Algorithm 3 Coordinate ascent with log-barrier

1: **Input:** A covariance matrix $\Sigma \in \mathbf{S}_p$, barrier coefficient $\lambda > 0$, decay $\mu < 1$, $s^{(0)} = 0 \in \mathbb{R}^p$.
2: Set $s = s^{(0)}$.
3: **repeat**
4: **for** $j = 1, \ldots, p$ **do**
5: Form $Q_j = 2\Sigma_{j^c, j^c} - \mathbf{diag}\, s_{j^c}$
6: Compute $s_j = \min\left(1, \max\left(2\Sigma_{j,j} - 4\Sigma_{j^c,j}^\top Q_j^{-1}\Sigma_{j^c,j} - \lambda,\, 0\right)\right)$
7: **end for**
8: $\lambda = \mu\lambda$
9: **until** stopping criteria
10: **Output:** A solution s.

Iteration Complexity

In Algorithm 3, the bottleneck is the inversion of the matrix $Q_j \in \mathbf{S}_{p-1}^+$ in line 6 which is $\mathcal{O}(p^3)$, making the total complexity of Algorithm 3 $\mathcal{O}(n_{\text{iters}}p^4)$. We can however reduce the cost of Algorithm 3 to $\mathcal{O}(n_{\text{iters}}p^3)$ by carefully updating Q_j^{-1} between subsequent coordinates.

Lemma 3. *Let $s \geq 0$ and $A = 2\Sigma - \boldsymbol{diag}(s)$. Then, for any $j \in [p]$, Q_j^{-1} can be computed as the inverse of a rank-3 update on A.*

Proof. Up to a permutation, we can assume without loss of generality that $j = 1$. We can write

$$\begin{bmatrix} 1 & 0 \\ 0 & Q_j^{-1} \end{bmatrix} = \left(A + e_j e_j^\top (1 + s_j + 2\Sigma_{jj}) - 2e_j\Sigma_j^\top - 2\Sigma_j e_j^\top\right)^{-1},$$

where e_j and Σ_j is the j^{th} Euclidean basis vector and column of Σ respectively. □

Using the Sherman-Woodbury-Morrisson (SWM) formula [41]

$$(A + UV^T)^{-1} = A^{-1} - A^{-1}U(\mathbf{I} + V^T A^{-1} U)^{-1} V^T A^{-1}, \tag{4.5}$$

updating Q_j^{-1} has complexity $\mathcal{O}(p^2)$. Note that A enjoys a rank-1 modification when a coordinate of s is updated. After the initial inversion of Σ, each update of Q_j thus becomes an $\mathcal{O}(p^2)$ operation and looping over all coordinates gives us a time complexity of $\mathcal{O}(n_{\text{iters}}p^3)$.

Stable Updates

Despite this improvement in complexity, the biggest practical problem with the aforementioned scheme is the numerical instabilities present using the SWM formula [110]. In order to circumvent this issue, we propose Algorithm 4 which is a modification of Algorithm 3 that uses Cholesky decompositions instead of matrix inversions. The key idea in Algorithm 4 is to keep a Cholesky factorization of $A \equiv 2\Sigma - \mathbf{diag}(s)$ at any time, performing stable rank

one update on A in $\mathcal{O}(p^2)$ after updating a coordinate of s. Hence, given $A = LL^\top$, we can solve a triangular system directly instead of inverting a matrix, as stated in the following lemma.

Lemma 4. *For a given index $j \in [p]$, note $\tilde{y} \in \mathbb{R}^p$ the jth column of 2Σ with the jth entry set to zero, that is,*

$$\tilde{y}_i = \begin{cases} 2\Sigma_{i,j} & \text{if } i \neq j\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

We claim that $s_j^\star$ can be computed in the optimality condition (4.4) from $\tilde{y}$, with

$$4\Sigma_{j^c,j}^\top Q_j^{-1}\Sigma_{j^c,j} = \frac{\zeta \tilde{y}^\top A^{-1}\tilde{y}}{\zeta + \tilde{y}^\top A^{-1}\tilde{y}} \tag{4.6}$$

where $\zeta = 2\Sigma_{j,j} - s_j$.

Proof. To prove (4.6), we can assume, up to a permutation and without loss of generality, that $j = p$, so that

$$\tilde{y}^\top A^{-1}\tilde{y} = \begin{bmatrix} 2\Sigma_{j^c,j} \\ 0 \end{bmatrix}^\top A^{-1} \begin{bmatrix} 2\Sigma_{j^c,j} \\ 0 \end{bmatrix}, \tag{4.7}$$

where

$$A = \begin{bmatrix} Q_j & 2\Sigma_{j^c,j} \\ 2\Sigma_{j^c,j}^\top & \zeta \end{bmatrix}, \qquad A^{-1} = \begin{bmatrix} B & * \\ * & * \end{bmatrix},$$

and inverse of A has the block structure given above where $B = Q_j^{-1} + \frac{4}{\beta} Q_j^{-1}(\Sigma_{j^c,j}\Sigma_{j^c,j}^\top)Q_j^{-1}$ and $\beta = \zeta - 4\Sigma_{j^c,j}^\top Q_j^{-1}\Sigma_{j^c,j}$. Plugging this into (4.7) and simplifying, we arrive at

$$\tilde{y}^\top A^{-1}\tilde{y} = 4\Sigma_{j^c,j}^\top Q_j^{-1}\Sigma_{j^c,j} + \frac{4(2\Sigma_{j^c,j}^\top Q_j^{-1}\Sigma_{j^c,j})^2}{\zeta - 4\Sigma_{j^c,j}^\top Q_j^{-1}\Sigma_{j^c,j}}\,,$$

which yields (4.6). □

Let x be the solution of the system $Lx = \tilde{y}$, whose computation amounts to forward substitution and requires only $\mathcal{O}(p^2)$ steps given L. Then $\|x\|_2^2 = \tilde{y}^\top (LL^\top)^{-1}\tilde{y}$, which means that the update can easily be computed using

$$4\Sigma_{j^c,j}^\top Q_j^{-1}\Sigma_{j^c,j} = \frac{\zeta \|x\|_2^2}{\zeta + \|x\|_2^2}$$

according to Lemma 4. After computing $s_j^\star$ and updating s, we perform a rank one Cholesky update of L to maintain the equality $LL^\top = 2\Sigma - \mathbf{diag}\, s$.

Hence, Algorithm 4 has the same worst-case complexity as the scheme proposed in Section 4.2, but is both faster and more stable in practice. Despite this computational improvement, the complexity $\mathcal{O}(n_{\text{iters}}p^3)$ is still prohibitive for large p. To make coordinate ascent scale, we assume in what follows that Σ has a low-rank factor model structure and adapt the method.

Algorithm 4 Stable coordinate ascent

Input: Σ, barrier coefficient $\lambda > 0$, decay $\mu < 1$, $s^{(0)} = \mathbf{0}_p$
$s = s^{(0)}$
$L = \textsc{Cholesky}2\Sigma - \mathbf{diag}\, s$
repeat
  for $j = 1, \ldots, p$ **do**
    Construct $\tilde{y}$ with $\tilde{y}_j = 0$ and $\tilde{y}_{j^c} = 2\Sigma_{j^c,j}$
    Solve $Lx = \tilde{y}$
    $\zeta = 2\Sigma_{j,j} - s_j$
    $c = \frac{\zeta \|x\|_2^2}{\zeta + \|x\|_2^2}$
    $s_j = \min\left(1, \max\left(2\Sigma_{j,\,j} - c - \lambda,\, 0\right)\right)$
    $\textsc{CholeskyUpdate}L$ {Maintain equality $LL^\top = 2\Sigma - \mathbf{diag}\, s$}
  end for
  $\lambda = \mu\lambda$
until stopping criteria

Coordinate Ascent under Factor Model

The complexity of Algorithm 4 can be drastically reduced, from $\mathcal{O}(n_{\text{iters}} p^3)$ to $\mathcal{O}\left(n_{\text{iters}} p k^2\right)$ assuming the low-rank factor model

$$\Sigma = D + UU^\top\,, \tag{4.8}$$

where $D \succeq 0$ is a diagonal matrix, and $U \in \mathbb{R}^{p\times k}$ where $k \ll p$ (see Section 4.2 for details on how efficiently estimate such a model). Under this assumption, for a given $j \in [p]$, noting again $A = 2\Sigma - \mathbf{diag}(s)$, we have via SWM (4.5),

$$A^{-1} = \tilde{D}^{-1} - 2\tilde{D}^{-1}U\left(I_k + 2U^\top \tilde{D}^{-1}U\right)^{-1} U^\top \tilde{D}^{-1}\,, \tag{4.9}$$

where $\tilde{D} = 2D - \mathbf{diag}(s)$. The computational gain comes from solving a $k \times k$ linear system as opposed to a $p \times p$ one. Recall that at each iteration j, only the jth coordinate of s is updated using

$$s_j \leftarrow \min(1, \max(\alpha^\star, 0)) \quad \text{where} \quad \alpha^\star = 2\Sigma_{j,\,j} - \frac{\zeta\kappa}{\zeta + \kappa} - \lambda\,,$$

as shown in Lemma 4, where $\kappa = \tilde{y}^\top A^{-1}\tilde{y}$. The following lemma helps computing the coefficient κ.

Lemma 5. *For a given index $j \in [p]$, let $\tilde{y} \in \mathbb{R}^p$ be the jth column of 2Σ with the jth entry set to zero. Then,*

$$\tilde{y}^\top A^{-1}\tilde{y} = s_j - 2\Sigma_{j,j} + (s_j - 2\Sigma_{j,j})^2 e_j^\top A^{-1} e_j\,.$$

Proof. The identity $AA^{-1} = \mathbf{I}$ yields that $A^{-1}(\tilde{y} + (2\Sigma_{j,j} - s_j)e_j) = e_j$, or equivalently, $A^{-1}\tilde{y} = e_j + (s_j - 2\Sigma_{j,j})A^{-1}e_j$. Hence,

$$\begin{aligned}\tilde{y}^\top A^{-1}\tilde{y} &= (s_j - 2\Sigma_{j,j})\tilde{y}^\top A^{-1}e_j \\ &= (s_j - 2\Sigma_{j,j})e_j^\top \left[e_j + (s_j - 2\Sigma_{j,j})A^{-1}e_j\right] \\ &= s_j - 2\Sigma_{j,j} + (s_j - 2\Sigma_{j,j})^2 e_j^\top A^{-1}e_j\,,\end{aligned}$$

where the second equality comes from the fact that A is symmetric. □

Combining Lemma 5 with equation (4.9) gives

$$\begin{aligned}\kappa &= \tilde{y}^\top A^{-1}\tilde{y} \\ &= s_j - 2\Sigma_{j,j} + (s_j - 2\Sigma_{j,j})^2 e_j^\top A^{-1}e_j \\ &= -\zeta + \frac{\zeta^2}{2D_{j,j} - s_j} - 2\frac{\zeta^2}{(2D_{j,j} - s_j)^2}U_{j,:}M^{-1}U_{j,:}^\top\,,\end{aligned}$$

where $\zeta = s_j - 2\Sigma_{j,j}$ and $M = I_k + 2U^\top \tilde{D}^{-1}U$. Finally noting $\xi = 2D_{j,j} - s_j$ we obtain the update rule

$$\begin{aligned}\alpha^\star &= 2\Sigma_{j,j} - \frac{\zeta\kappa}{\zeta + \kappa} - \lambda \\ &= 2\Sigma_{j,j} - \frac{-\zeta^2 + \frac{\zeta^3}{\xi} - 2\frac{\zeta^3}{\xi^2}U_{j,:}M^{-1}U_{j,:}^\top}{\frac{\zeta^2}{\xi} - 2\frac{\zeta^2}{\xi^2}U_{j,:}M^{-1}U_{j,:}^\top} - \lambda \\ &= 2\Sigma_{j,j} - \frac{-\xi^2 + \zeta\xi - 2\zeta U_{j,:}M^{-1}U_{j,:}^\top}{\xi - 2U_{j,:}M^{-1}U_{j,:}^\top} - \lambda \\ &= 2\Sigma_{j,j} + 2\frac{(2D_{j,j} - s_j)(D_{j,j} - \Sigma_{j,j}) - (s_j - 2\Sigma_{j,j})U_{j,:}M^{-1}U_{j,:}^\top}{2D_{j,j} - s_j - 2U_{j,:}M^{-1}U_{j,:}^\top} - \lambda\,.\end{aligned}$$

QR updates The bottleneck in the computation of the update coefficient is the inversion of the $k \times k$ matrix $M = I_k + 2U^\top \tilde{D}^{-1}U$. To this end, we notice that an update of the jth coordinate of s leads to a rank one update of M. This means that we can efficiently compute $\alpha^\star$ by performing successive rank one updates on $k \times k$ matrices at each iteration. Indeed, forming M and computing a QR decomposition costs $\mathcal{O}(pk^2)$ operations.[1] From this, the term $U_{j,:}(QR)^{-1}U_{j,:}^\top$ can efficiently be computed using the fact that Q is orthogonal and R triangular. Finally, after s_j has been updated, we perform a rank one update on the QR decomposition of $\mathbf{I}_k + 2M$.

The algorithm taking advantage of the factor model structure is summarized in Algorithm 5. One nuance to using the SWM formula in Equation (4.9) is the fact that $\tilde{D}_j$ can be nearly singular. In theory, this would preclude solving the SDP to arbitrary accuracy. In practice, this does not seem to be problem as numerical instabilities rarely occur (see Section 4.4).

[1]The matrix M is not necessarily psd, hence Cholesky decompositon is not possible.

Algorithm 5 Coordinate Ascent using Factor Model

Input: approximation $\Sigma = D + UU^\top$, barrier coefficient $\lambda > 0$, decay $\mu < 1$, $s^{(0)} = 0_p$
$s = s^{(0)}$
$M = U^\top (2D - \mathbf{diag}\, s)^{-1} U$
$Q, R \leftarrow \textsc{DecomposeQR}\mathbf{I}_k + 2M$
repeat
  for $j = 1, \ldots, p$ **do**
    $z \leftarrow j$th column of $U^\top$
    Solve $Rx = z$
    $a \leftarrow x^\top Q^\top z$
    $\alpha^\star \leftarrow 2\Sigma_{j,j} + 2\frac{(2d_j - s_j)(d_j - \Sigma_{j,j}) - (s_j - 2\Sigma_{j,j})a}{2d_j - s_j - 2a} - \lambda$
    $s_j \leftarrow \min(1, \max(\alpha^\star, 0))$
    $\textsc{UpdateQR}Q, R$ {Maintain $QR = \mathbf{I}_k + 2M$}
  end for
  $\lambda = \mu\lambda$
until stopping criteria

Estimating Factor Models

In this section, we review how to efficiently compute a factor model for the correlation matrix written $\Sigma = D + UU^\top$ [7]. The factor model assumption is somewhat natural for two reasons. First, many natural data sets have correlation matrices that are numerically low rank [**Udel19**] which can be well approximated by factor models. Second, classical regularized estimators of the correlation matrix such as the Ledoit-Wolfe estimator [**Ledo04**] are directly written as factor models.

The factor model is estimated by solving the following non-convex optimization problem

$$(D^*, U^*) = \arg \min_{D,U \in \mathbb{R}^{p \times k}} \left\{ \|\Sigma - D - UU^\top\|_F^2 \ : \ D \succeq 0 \text{ diagonal} \right\}, \tag{4.10}$$

where $k \ll p$ is a user-specified rank. Note that when $k = p$, $D^* = 0$ and $U = V\Lambda^{1/2}$ where $\Sigma = V\Lambda V^\top$. While (4.10) is non-convex, we use an alternating minimization scheme for solving it to (local) optimality. Given $UU^\top$, solving for D is direct, we simply set $D_{ii} = \max(0, \Sigma_{ii} - U_{ii}^2)$. Now, given D, getting the optimal U reduces to projecting $\Sigma - D$ onto the space of rank k PSD matrices. The optimal U is given by $U^* = V\Lambda^{1/2}$ where $V \in \mathbb{R}^{p \times k}$ are the top k eigenvectors of $\Sigma - D$ associated with the top k eigenvalues and $\Lambda \in \mathbb{R}^{k \times k}$ is a diagonal matrix with $\Lambda_{ii} = \max(0, \lambda_i)$ for $i = 1, \ldots k$ (note $\Sigma - D$ need not be PSD).

In the setting where $n \ll p$, the empirical covariance tends to be far from the population covariance matrix and is ill-conditioned. To alleviate this, [66] uses Stein shrinkage to compute a better estimate of Σ. We use the regularized covariance (also known as the

Ledoit-Wolf estimator)

$$\Sigma = (1-\delta)\Sigma + \delta\mu I_p, \quad \mu = \mathbf{Tr}(\Sigma)/p, \quad \delta^\star = \frac{1}{n^2}\frac{\sum_{i=1}^n (x_i^\top x_i)^2 - n\,\mathbf{Tr}(\Sigma^2)}{\mathbf{Tr}(\Sigma^2) - \mathbf{Tr}(\Sigma)^2/p} \tag{4.11}$$

where $\delta^\star$ is the optimal shrinkage parameter. These traces may be approximated with stochastic Lanczos quadrature [100] without explicitly evaluating Σ and Σ^2.

When p is extremely large, Σ may be too costly to store in memory. In order to circumvent this issue, since we are only interested in computing the top k eigenvector–eigenvalue pairs, we can simply compute the top left singular vectors of X instead of performing a spectral decomposition on Σ when $\Sigma = \frac{1}{n}XX^\top$ or when Σ is the Ledoit-Wolf estimator [42, 111].

4.3 Sampling factor model Knockoffs

In this section, we detail how to generate the knockoff matrix $\tilde{X} \in \mathbb{R}^{p\times n}$ once an optimal solution s to the semidefinite program (4.2) has been found. Each column $\tilde{x}_i$ is sampled according to the Gaussian conditional distribution in (4.1). This means sampling $\tilde{x}_i \mid x_i \sim \mathcal{N}(\mu_i, \Omega)$ such that

$$\mu_i = x_i - \mathbf{diag}(s)\Sigma^{-1}x_i \quad \text{and} \quad \Omega = 2\,\mathbf{diag}(s) - \mathbf{diag}(s)\Sigma^{-1}\,\mathbf{diag}(s)\,.$$

Naively sampling from $\mathcal{N}(\mu_i, \Omega)$ via $\tilde{x}_i = \mu_i + Lv$ where $v \sim \mathcal{N}(0, \mathbf{I}_p)$ and where L satisfies $LL^\top = \Omega$ has complexity $\mathcal{O}(p^3)$ (the cost associated with the Cholesky decomposition).

We now show that if we have a factor model assumption on Σ, we can sample knockoffs in time linear in p. Suppose that Σ has a factor model structure as in (4.8); that is, $\Sigma = D + UU^\top$ where $D \succ 0$ is a diagonal matrix and $U \in \mathbb{R}^{p\times k}$ (with $k \ll p$). We show how to factorize Ω and sample the knockoff matrix $\tilde{X}$ in $\mathcal{O}\left(npk^2\right)$ steps using $\mathcal{O}\left(p(n+k)\right)$ memory. Using the factor model assumption and the SWM formula (4.5), we have

$$\Sigma^{-1} = D^{-1} - D^{-1}UNN^\top U^\top D^{-1}$$

where $N \in \mathbb{R}^{k\times k}$ is the Cholesky factorization of $(\mathbf{I}_k + U^\top D^{-1}U)^{-1}$. Setting $S = \mathbf{diag}(s)$ gives

$$\Omega = 2S - S\Sigma^{-1}S = C + ZZ^\top$$

where $C = 2S - SD^{-1}S$ is diagonal (but not necessarily PSD) and $Z = SD^{-1}UN \in \mathbb{R}^{p\times k}$ is low-rank. Forming C and Z takes at most $\mathcal{O}\left(pk^2\right)$ operations and $\mathcal{O}\left(pk\right)$ memory. Notice that the mean μ_i is easily computed in $\mathcal{O}\left(npk^2\right)$ operations and without additional memory as follows

$$\mu_i = x_i - \Sigma^{-1}Sx_i = x_i - D^{-1}Sx_i + D^{-1}UNZ^\top x_i\,.$$

For this reason, the problem reduces to sampling from $\mathcal{N}\left(0, \Omega\right)$ efficiently. To do so, we adopt the $L\Delta L^\top$ factorization procedure presented by [90], which means decomposing Ω in the following way

$$\Omega = C + ZZ^\top = L\left(Z,\, B\right)\,\Delta\,L\left(Z,\, B\right)^\top\,, \tag{4.12}$$

where $L(Z,\, B)$ (denoted L in the sequel) has the following structure

$$L(Z,\, B) = \begin{pmatrix} 1 & & & \\ \langle z_2', b_1' \rangle & 1 & & \\ \vdots & & \ddots & \\ \langle z_p', b_1' \rangle & \dots & \langle z_p', b_{p-1}' \rangle & 1 \end{pmatrix}.$$

Here z_i', b_i' denote the i^{th} row of Z and B respectively, with $B \in \mathbb{R}^{p \times k}$, and $\Delta \in \mathbf{S}_p^+$ is diagonal. With a sample v from $\mathcal{N}(0, \mathbf{I}_p)$, $\tilde{x}_i$ can be computed by setting $\tilde{x}_i = \mu_i + L\sqrt{\Delta}\, v$. The advantages of using the $L\Delta L^\top$ decomposition are that *(i)* we do not require $C \succeq 0$ and *(ii)* we never have to store the full matrices L or B to compute the product $L\sqrt{\Delta}\, v$.

Forming B and Δ Given a diagonal plus low-rank covariance $\Omega = C + ZZ^\top$, Algorithm 6 (from [90]) forms the matrices $B \in \mathbb{R}^{p \times k}$ and $\Delta \in \mathbb{R}^{p \times p}$ such that $\Omega = L(Z, B)\ \Delta\ L(Z, B)^\top$. It requires $\mathcal{O}(pk)$ steps and $\mathcal{O}(pk)$ additional memory (if only the diagonal of Δ is stored).

Algorithm 6 Forming the Cholesky factorization matrices B and Δ

Input: $\Omega = C + ZZ^\top$
$M = \mathbf{I}_k$, $B = 0$
for $j = 1, \dots, p$ **do**
  $t = Mz_j$
  $\Delta_{j,j} = C_{j,j} + z_j^\top t$ {always non-negative}
  if $\Delta_{j,j} > 0$ **then**
    $b_j = t/\Delta_{j,j}$
    $M = M - tt^\top/\Delta_{j,j}$
  else
    $b_j = 0$ {b_j may be anything, choose 0 for simplicity}
  end if
end for
Output: Δ and B as in (4.12).

Fast multiplication By virtue of the specific structure of L (and diagonality of Δ), given the matrices B, Δ and a vector $v \in \mathbb{R}^p$, Algorithm 7 computes the product $u = L(Z, B)\, \Delta v$ in only $\mathcal{O}(pk)$ operations (instead of the $\mathcal{O}(p^2)$ normally required for a matrix-vector product) and $\mathcal{O}(p+k)$ memory. A low asymptotic complexity is possible thanks to the special structure

Algorithm 7 Fast Cholesky multiplication

Input: B, Δ such that $\Omega = L(Z,B)\ \Delta\ L(Z,B)^\top$ and a vector $v \in \mathbb{R}^p$
$w = \mathbf{0}_k$
for $j = 1, \ldots, p$ **do**
  $u_j = \sqrt{\Delta_{j,j}} v_j + z_j^\top w$
  $w = w + \sqrt{\Delta_{j,j}} v_j b_j$
end for
Output: $u = L(Z,B)\,\Delta v$

of $L(Z,B)$. More precisely note that for any $j \in [p]$

$$\begin{aligned} u_j &= \left(L(Z,\ B)\sqrt{\Delta} v\right)_j \\ &= \sqrt{\Delta_{j,j}} v_j + \sum_{i=1}^{j-1} z_j^\top b_i \sqrt{\Delta_{i,i}} v_i \\ &= \sqrt{\Delta_{j,j}} v_j + z_j^\top w_j \end{aligned}$$

where $w_j = \sum_{i=1}^{j-1} b_i \sqrt{\Delta_{i,i}} v_i$. The buffer vector w may be updated iteratively which allows to compute u at low cost.

Sampling knockoffs In practice, we derive an iterative procedure that never stores B nor L in memory to compute $\tilde{x}_i$. Instead, their rows are computed on the fly and requires only $\mathcal{O}(p + k^2)$ memory. We combine Algorithms 6 and 7 in order to sample knockoffs. From Algorithm 7, it is clear that neither B, Δ nor $L(Z,B)$ need to be fully computed and stored in memory. Instead, the rows of B and the diagonal of Δ may be computed iteratively, as shown in Algorithm 8, which has a time complexity of $\mathcal{O}(pk^2)$ and uses $\mathcal{O}(k^2)$ memory. Here a single value is sampled from $\mathcal{N}(0, \Omega)$; it may be easily extended to sample the n required knockoffs. Finally, n columns $\tilde{x}_i$ have to be sampled to form $\tilde{X} \in \mathbb{R}^{p \times n}$, which requires $\mathcal{O}(npk^2)$ steps and $\mathcal{O}(p(n+k))$ memory.

Spectrum of Ω If Σ can only be *approximated* by a factor model (see Section 4.2) we have $\Sigma \approx D + UU^\top$ and the careful reader may notice that s computed via Algorithm 5 (which by construction satisfies $\mathbf{diag}(s) \preceq 2(D + UU^\top)$) need not satisfy $\mathbf{diag}(s) \preceq 2\Sigma$. This in turn implies Ω is not PSD. In order to circumvent this problem, we propose two procedures: the *hybrid* approach, and the *low rank* approach. In the hybrid approach, after obtaining $\hat{s}$ from Algorithm 5, as in [**Cand18**], we solve

$$\gamma^* = \arg\max_\gamma\ \gamma\ :\ \mathbf{diag}(\gamma \hat{s}) \preceq 2\Sigma$$

which is a minimum eigenvalue problem that can be solved efficiently via bisection over γ. This then ensures that $\Omega \succeq 0$ when $s = \gamma^* \hat{s}$. In the low rank approach, we *assume*

Algorithm 8 Fast Gaussian sampling

Input: $\Omega = C + ZZ^\top$ and $v \in \mathbb{R}^p$ a sample from $\mathcal{N}(0, \mathbf{I}_p)$
$M = I_k$
$w = \mathbf{0}_k$
for $j = 1, \ldots, p$ **do**
  $t = Mz_j$
  $\delta_j = C_{j,j} + z_j^\top t$ {δ_j is always non-negative}
  $u_j = \sqrt{\delta_j} v_j + z_j^\top w$
  if $\delta_j > 0$ **then**
    $b'_j = t/\delta_j$ {jth row of B}
    $M = M - tt^\top/\delta_j$
    $w = w + \sqrt{\delta_j} v_j w_j$
  end if
end for
Output: $u \in \mathbb{R}^p$ sampled from $\mathcal{N}(0, \Omega)$

$\Sigma = D + UU^\top$ and sample our knockoffs accordingly. While not theoretically justified, we show in Section 4.4 how this model is able to outperform most of the other methods in both speed and performance while still controlling FDR.

4.4 Numerical Results

In this section, we perform a series of experiments to benchmark the aforementioned algorithms. All experiments utilized a standard workstation. For the plots below, all error bars represent one standard deviation. Unless referring to our algorithms, all other functions used were from `Scikit-Learn` [81] and `Nilearn` [1] (library for machine learning on neuro-imaging data).

Benchmarks

We first generate random covariance matrices and compare CPU time and quality of solutions in solving (4.2) using `SCS` (a first order method) and `CVXOPT` (an IPM) interfaced with `CVXPY` [79, 2, 26] and solving (4.3) using coordinate ascent. We set $\Sigma = D + V\Lambda V^\top$ where $D \in \mathbb{R}^{p\times p}, V \in \mathbb{R}^{p\times k}, \Lambda \in \mathbb{R}^{k\times k}$ where $D = 10^{-3} I_p$, $\Lambda_{ii} \sim U[0,1]$, $V_{ij} \sim \mathcal{N}(0,1)$, and $k = \lceil 0.05p \rceil$. Figure 4.1 shows the results of the experiment for increasing p and the optimality of the generated solution. In addition to comparing the optimality of the methods based on objective functions, we check the feasibility of the solutions generated by the solutions. Figure 4.2 plots the minimum eigenvalue of $2\Sigma - \mathbf{diag}(s)$ versus the dimension. If the minimum eigenvalue is negative, then the solution generated is infeasible. We see that with default tolerances, `CVXOPT` and `SCS` generate infeasible solutions whereas our models stay

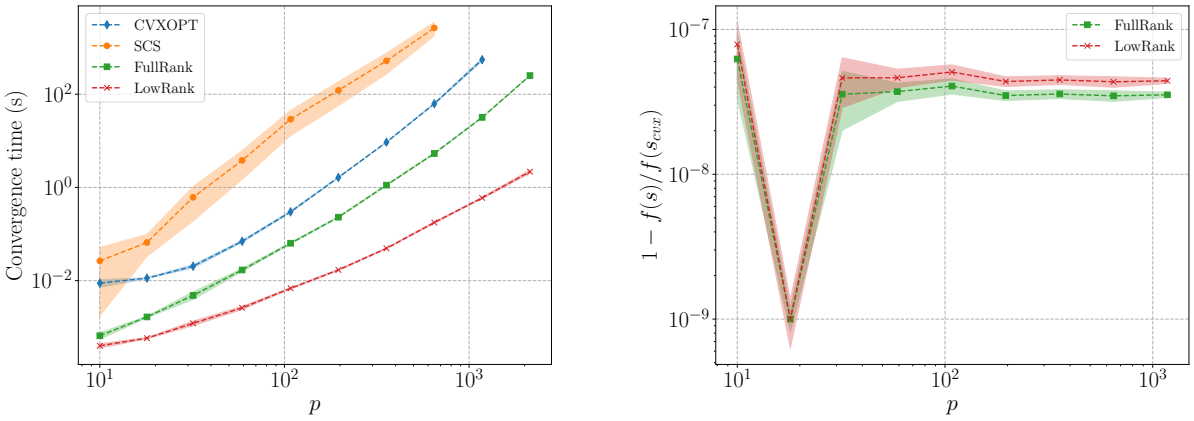

Figure 4.1: (*Left*) Convergence time versus dimension for solving (4.2) using `CVXOPT` and `SCS` in `CVXPY` and using Algorithms 4 and 5 to solve (4.3). (*Right*) Objective values reached by the full and low rank algorithms relative to that generated using IPMs. Here $f(s) = \mathbf{1}^\top s$.

feasible. We noticed that decreasing the default tolerances of `CVXOPT` and `SCS` did not help much in this regard and significantly increased the run time of the methods.

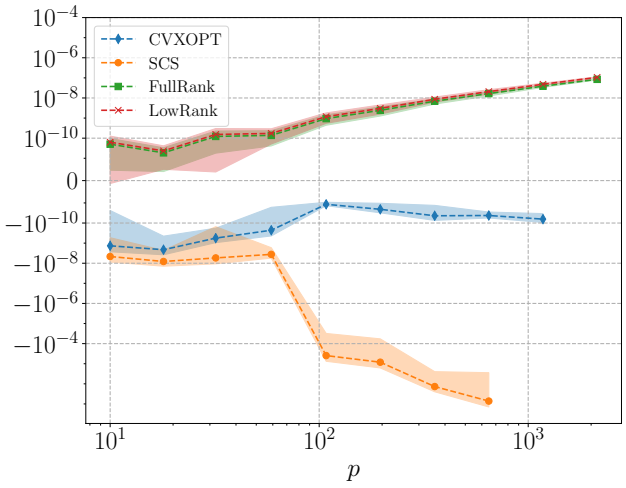

Figure 4.2: Feasibility plot of $\lambda_{\min}(2\Sigma - \mathbf{diag}(s))$ versus dimension.

In Figure 4.1, coordinate ascent provides substantial computational gains compared to using `SCS` or `CVXOPT`. Solving the full rank model is consistently one (resp. two) orders of magnitude faster than `CVXOPT` (resp. `SCS`) and the low rank model for $p = 500$ is four orders

of magnitude faster than SCS. The slopes also indicate that for larger p, SCS and CVXOPT become prohibitively slow while the low rank model can comfortably handle $p \sim 10^5$. The right panel in Figure 4.1 shows that the solution computed by our solver is indeed close to the CVXOPT solution (while SCS produced infeasible solutions). For the full rank and low rank models, our convergence criteria was $\lambda \leq \epsilon$ where $\epsilon = 10^{-8}$. For SCS, eps $= 10^{-6}$ and for CVXOPT the default tolerances were used.

Complexity

SDP under factor model We now check empirically the complexity bounds of Algorithm 5 derived in Section 4.2 (under the factor model assumption). We focus on the time spent per cycle of the for loop in Algorithm 5. We run two sets of experiments: one where we fix k and increase p and another where we fix p and increase k. For both experiments, we generate covariance matrices as above (Section 4.4). The results are plotted in Figure 4.3. This shows a favorable linear rate when $k \in [10^1, 10^2]$ and the theoretically derived quadratic rate when $k \in [10^2, 10^3]$. Empirically, we observed that 5 to 50 cycles are enough to converge to a tolerance threshold of 10^{-6} on all the covariance matrices we experimented.

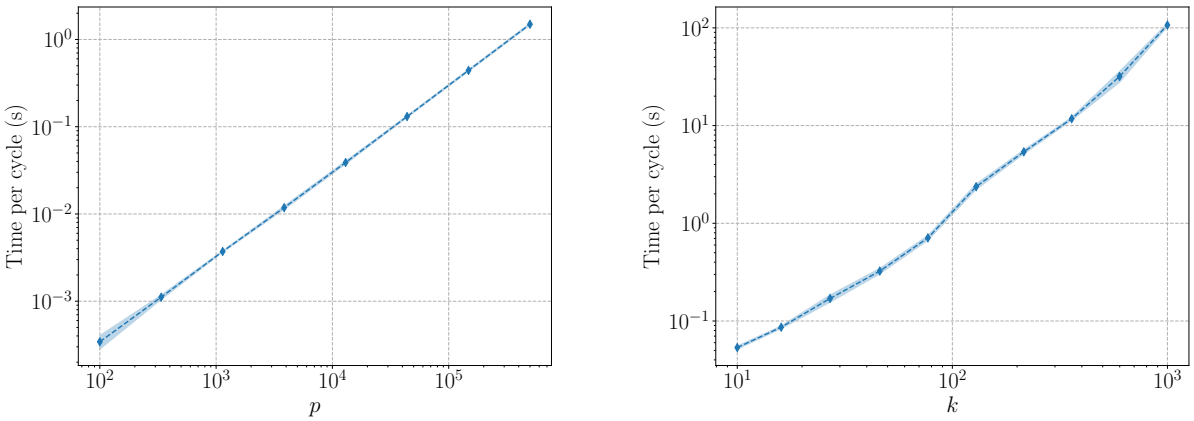

Figure 4.3: (*Left*) Time per cycle versus dimension p with $k = 25$. This shows a linear dependence on p. (*Right*) Time per cycle versus k with $p = 50\,000$.

Sampling knockoffs We also benchmark the complexity of Algorithm 8 to sample from $\mathcal{N}(0, \Omega)$ when $\Omega = C + ZZ^\top$ where $C \in \mathbb{R}^{p \times p}$ is diagonal and $Z \in \mathbb{R}^{p \times k}$. In Figure 4.4 we compare it to the classical approach consisting in finding the Cholesky decomposition of Ω, which is our baseline. As seen in Figure 4.4, Algorithm 8 enjoys a linear dependence on p, a favorable (sub)linear rate when $k \in [1, 10^2]$ and a quadratic dependence on k (as expected)

when $k \in [10^2, 10^3]$. Hence, using the special structure of the covariance, knockoffs may be sampled in linear time.

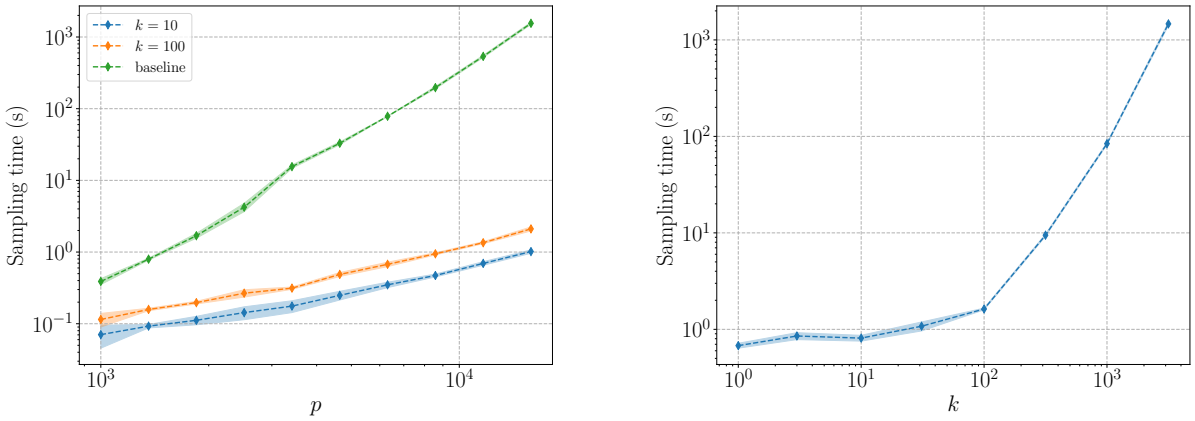

Figure 4.4: *(Left)* Sampling time versus dimension. *(Right)* Sampling time versus rank for $p = 25\,000$. Note we use a plain Python/`NumPy` implementation for our algorithm.

FDR Control on Synthetic Data

We now compare FDR control and power using different methods of solving (4.2) at scale. For computational reasons, the two main current methods for constructing knockoffs in high dimension either use an equicorrelated (Equi) construction or an approximate semidefinite program (ASDP) construction (for more details see [**Cand18**]). The Equi and ASDP constructions are approximations to the solution of (4.2) and in this experiment, we compare the quality of knockoffs (measured via false discovery rate and power) generated using the above methods with the knockoffs generated via coordinate ascent, in the full rank and factor model settings.

We run a similar experiment to that in Figure 5 of [**Cand18**]. We generate Σ with $\Sigma = D + VV^\top$ where $D_{ii} \sim U[0,1], V_{ij} \sim \mathcal{N}(0, 1/k)$ and $V \in \mathbb{R}^{p\times k}$. We then generate $X \in \mathbb{R}^{p\times n}$ where the ith column of X is generated according to $x_i \sim \mathcal{N}(\mathbf{0}_p, \Sigma)$. We then set $y = X^\top\beta + \epsilon$ where $\epsilon_i \sim \mathcal{N}(0,1)$ and β has a fixed number of nonzero regression coefficients each having equal magnitudes and random signs. We then estimate a factor model from the empirical covariance (see Section 4.2) with rank equal to k, solve the appropriate SDP, sample the knockoffs 100 times and finally compare the FDR and power of the various methods in Figure 4.5. Note that in Section 4.1, the results assumes that we know the *true* covariance Σ for knockoff generation and sampling. In this experiment, we are estimating Σ from X.

The target FDR rate is set to 10%. The results of Figure 4.5 confirm the fundamental trade off between maximizing power and minimizing FDR – if the FDR is very low, we do not expect the method to have much power. However, since the knockoff procedure simply

provides a bound on the FDR, we are interested in comparing which procedure provides the most power. We observe in Figure 4.5 that the approximate solutions produced using Equi and ASDP constructions tend to be more conservative in their FDR control (which is well below the 10% target) and often have significantly less power than the optimal full and low rank SDP solutions. Overall, these optimal SDP solutions have an empirical FDR closer to the target (sometimes marginally above due to model estimation error) and exhibit more power, probably because the knockoffs are less correlated. Surprisingly, the low rank solutions have more power than the full rank ones even when their FDR match, which might be explained by the implicit regularization effect of the low rank structure.

The above setup favours the low rank factor model setting because we explicitly set $\Sigma = D + VV^\top$. In Figure 4.6, we run the same experiment as in Figure 4.5 but we now set $\Sigma = VV^\top$ with $V_{ij} \sim \mathcal{N}(0, 1/p)$ and $V \in \mathbb{R}^{p \times p}$ (i.e Σ is full rank and does not inherently posses a factor model structure). We then estimate a factor model from the regularized covariance (see Section 4.2) with rank equal to $k = \{20, 400\}$.

Figure 4.6 shows that despite the true covariance not having a factor model structure, the FDR is still controlled and the power is maintained. Furthermore, based on the right hand plot in Figure 4.7, a rank 20 factor model is a much poorer approximation to Σ than a rank 400 approximation. However, we see in Figure 4.6 that in spite of this, the rank 20 approximation still controls FDR and has power comparable to that of other methods.

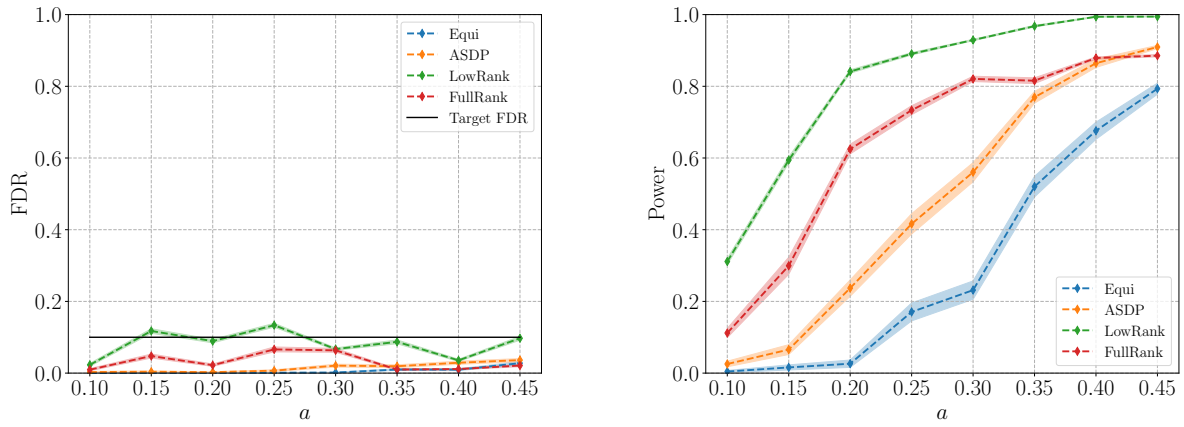

Figure 4.5: $(n, p, k) = (1000, 500, 50)$, $\|\beta\|_0 = 50$ and each entry has equal amplitude. Each point represents 100 trials (the same X and β is used for each amplitude; the randomness is over the knockoff sampling). Error bars represent the std divided by the square root of the number of trials in order to make it a 68% confidence interval. (*Left*) FDP versus amplitude. (*Right*) Power versus amplitude.

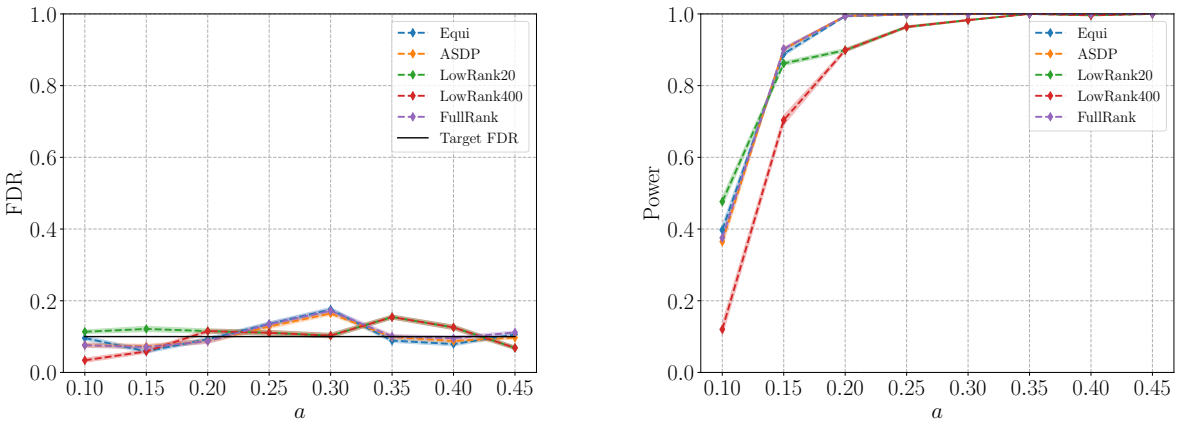

Figure 4.6: Same setup as the experiment for Figure 4.5 but now we estimate a rank $k = \{20, 400\}$ factor model from a full rank $\Sigma = VV^\top$ with $V \in \mathbb{R}^{p \times p}$. (*Left*) FDP versus amplitude. (*Right*) Power versus amplitude.

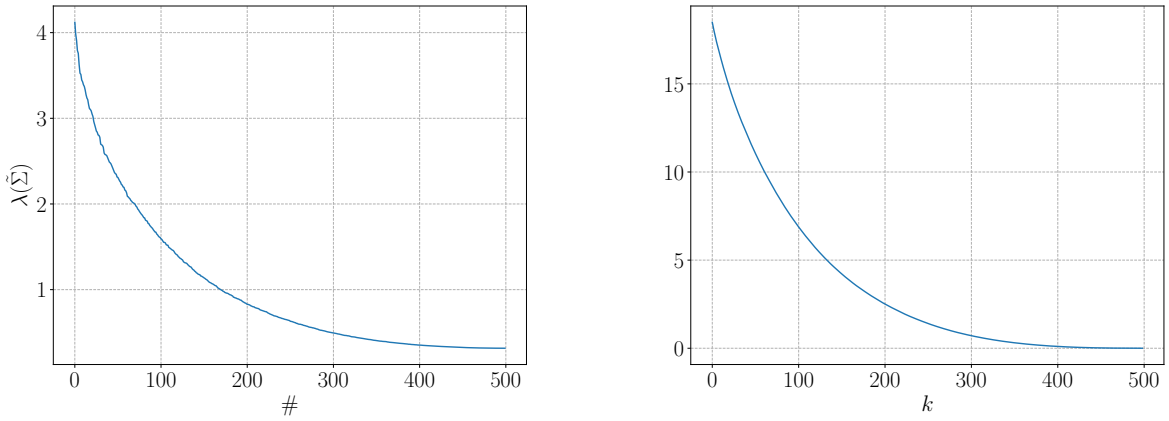

Figure 4.7: (Left) Spectrum of eigenvalues of Σ (defined via (4.11)) used in estimating a rank 20 and rank 400 factor model in Figure 4.6. (Right) Error $\|\Sigma - D - UU^\top\|_F$ in factor model approximation of Σ with increasing rank k.

fMRI feature selection

We now test the low-rank factor model on the Human Connectome Project (HCP) [31] dataset for feature selection. Composed of brain connectivity maps, the dataset contains brain activity from 1 496 healthy patients that was measured while they were shown pictures of either humans faces or geometric shapes. We derive a binary classification task from the

fMRI data consisting of identifying the category of the picture shown to each patient given their brain activity. More specifically, we apply the knockoff filter to find which regions of the brain are the most discriminative for classification.

Preprocessing Connectivity maps are volumes of size $91 \times 109 \times 91$. Among these 902 629 voxels only 212 445 are in the brain envelope. We first extract them because they contain the functional information of the brain. Since fMRI data is by nature very noisy and extremely high dimensional, we then perform a spacial clustering step resulting in $p = 5\,000$ components. This step averages the noise and reduces the data dimension. To do so, we make use of the package `Nilearn` which provides parcellation algorithms. We employed the Ward clustering method [51] because it is known to perform well in terms of accuracy [98]. In the following experiments, the factor model is then computed for the shrunk (Ledoit-Wolf) covariance matrix (see Section 4.2) with $k = 50$. Figure 4.8 shows the error of the factor model as a function of the rank. We observe a very rapid decrease for small ranks (< 50) followed by a long plateau. This justifies the choice $k = 50$.

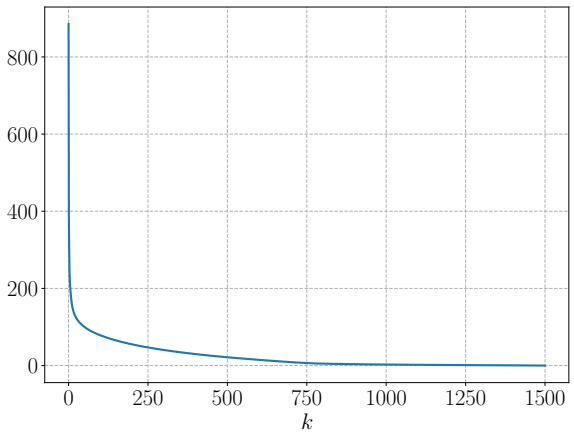

Figure 4.8: Error $\|\Sigma - D - UU^T\|_F$ made by the factor model of rank k with respect to the shrunk (Ledoit-Wolf) covariance matrix Σ for fMRI data.

Sparse Center Classifiers As the knockoff framework offers a lot of freedom regarding the choice of the covariates statistics, we chose to derive them from sparse centroid classifiers [16], primarily because it can be computed very efficiently as compared to the LCD [**Cand18**] statistic, but also because we found them to be more effective than the LCD statistic for this classification task. More specifically, for any L_0 penalty coefficient $\lambda \geq 0$ we define the

sparse centroids parameters $\left(\hat{\theta}^+(\lambda),\ \hat{\theta}^-(\lambda)\right)$ as the solutions of the optimization problem

$$\left(\hat{\theta}^+(\lambda),\ \hat{\theta}^-(\lambda)\right) = \underset{\theta^+,\theta^- \in \mathbb{R}^p}{\arg\min} \frac{1}{n_+} \sum_{j \in \mathcal{J}^+} \|x_j - \theta^+\|_2^2 + \frac{1}{n_-} \sum_{j \in \mathcal{J}^-} \|x_j - \theta^-\|_2^2 + \lambda \|\theta^+ - \theta^-\|_0 \,,$$

where $\mathcal{J}^{\pm}$ denotes an index set corresponding to the ± 1 labeled data points and $n_{\pm} = |\mathcal{J}^{\pm}|$. Following the same idea as the LSM statistic [**Cand18**], we define $Z_j = \sup\{\lambda \geq 0 \mid \hat{\theta}^+(\lambda) \neq \hat{\theta}^-(\lambda)\}$ for all $j \in [2p]$. Finally, our statistic takes the following form

$$W_j = |Z_j| - |Z_{j+p}| \,, \tag{4.13}$$

using the difference function which is antisymmetric. The knockoff filter controls the FDR only if the statistics obey the *flip-sign* property as explained in Section 3.2 of [**Cand18**]. It is easy to verify that the statistics defined in Equation (4.13) satisfy the requirements.

Estimating the factor model, solving (4.3), sampling knockoffs and computing the covariates statistics takes roughly 20 seconds. Figure 4.9 shows the brain regions that were selected with a FDR target of 10%. We cannot evaluate power or FDR here since the ground

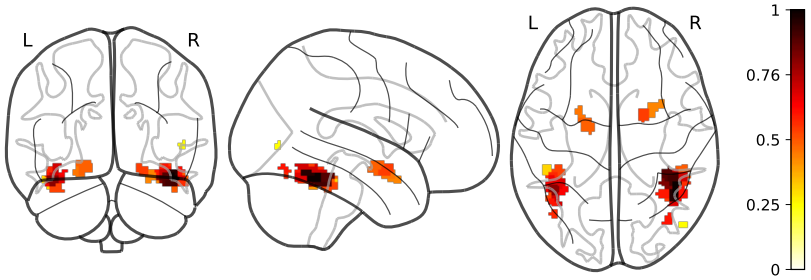

Figure 4.9: Discoveries (34 in total) and their weights obtained by applying the knockoff filter with the low-rank factor model. In comparison, Equi-knockoffs did not result in any discoveries.

truth is not known. Note however that the discoveries are quite symmetric and concentrated in a few locations. Since the results were obtained without combining knockoffs with any additional structured penalty constraint to enforce localization or symmetry, this suggests that the features are indeed meaningful.

LCD statistics We also experimented LCD statistics as they were shown to be robust in many settings [**Cand18**]. The computation takes roughly 5 minutes (as opposed to 2 seconds for the centroids) and the procedure selects approximately the same regions and the same features. Figure 4.10 shows the features that were selected with a FDR target of 10% using LCD statistic.

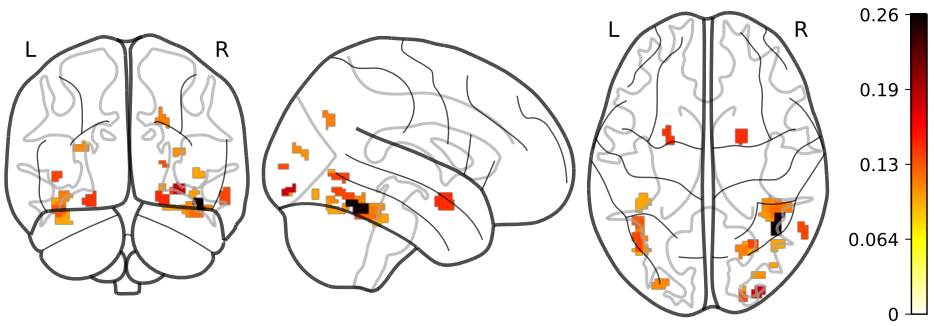

Figure 4.10: Discoveries (26 in total) and their weights obtained by applying the knockoff filter with the low-rank factor model and LCD statistic.

Synthetic response As a correctness check, we run the following experiment to assess the quality of the selection for fMRI data. We generate a random synthetic response $y = \text{Bernoulli}(\text{sigmoid}(X\beta))$ where β has 20 nonzero coefficients each having equal amplitudes a and random signs. With this construction, the ground truth is known and we can compute the power and FDR of the selection procedure. We perform knockoffs feature selection with LCD statistics and we repeat this process 20 times (always changing the ground truth) for a range of amplitudes. We plot the power and FDR for Equi, ASDP and low-rank knockoffs in Figure 4.11. This shows that low-rank knockoffs perform very well in terms of power. The

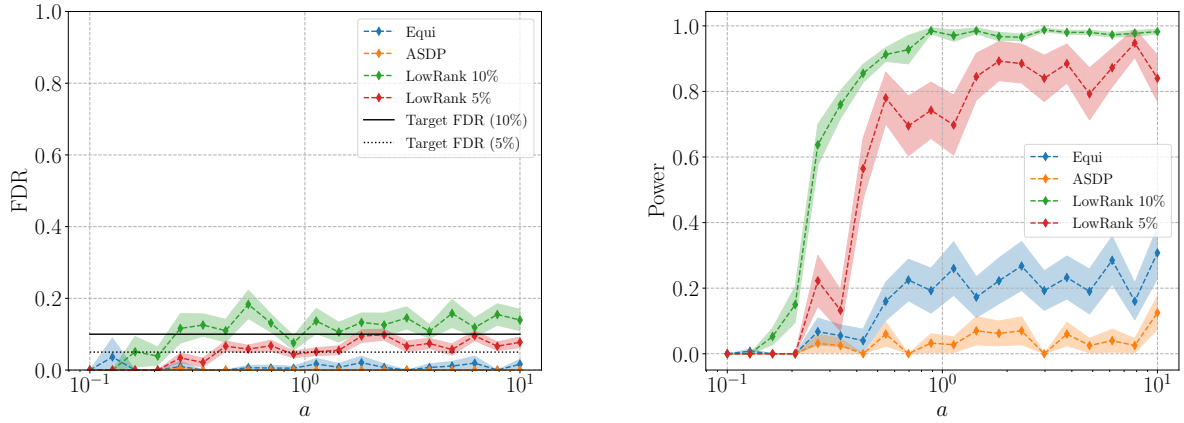

Figure 4.11: fMRI feature selection for a synthetic binary response, with a target FDR of 10% for Equi, ASDP and low-rank knockoffs (blue, orange and green). Additionally low-rank knockoffs with a target FDR of 5% is plotted in red. *(Left)* FDR versus amplitude. *(Right)* Power versus amplitude.

target FDR is slightly exceeded, most likely because both the Gaussian and the low-rank

assumptions are overly optimistic. In contrast, Equi and ASDP knockoffs have a very low power, even for larger amplitudes. For this particular problem, ASDP has excessively low power compared to the other two methods. In this experiment, we used 10 blocks for the ASDP solver and it finds a solution roughly 4 times lower than Equi, at a much higher computational cost. Even with only 2 blocks, the solution is slightly smaller than that of Equi. In order to correct the FDR excess of low-rank, we can be more conservative and run knockoffs with a lower target FDR, say 5%, and hope to meet the actual 10% target. We plot the results of this approach in red in Figure 4.11. This shows that the FDR is controlled better, and the power still compares favorably to the other methods. While not plotted, we observed that using Sparse Center Classifiers statistics (Section 4.4) yields similar results, even though the power is slightly lower.

Chapter 5

Sparse Naive Bayes

5.1 Introduction

Modern, large-scale data sets call for classification methods that scale mildly (e.g. linearly) with problem size. In this context, the classical naive Bayes model remains a very competitive baseline, due to its linear complexity in the number of training points and features. In fact, it is sometimes the only feasible approach in very large-scale settings, particularly in text applications, where the number of features can easily be in the millions.

Feature selection, on the other hand, is a key component of machine learning pipelines, for two main reasons: i) to reduce effects of overfitting by eliminating noisy, non-informative features and ii) to provide interpretability. In essence, feature selection is a combinatorial problem, involving the selection of a few features in a potentially large population. State-of-the-art methods for feature selection employ some heuristic to address the combinatorial aspect, and the most effective ones are usually computationally costly. For example, LASSO [99] or l_1-SVM models [32] are based on solving a l_1-penalized convex problem in order to achieve sparsity (at the expense of tuning a hyper parameter to attain a desired sparsity level).

Since naive Bayes corresponds to a linear classification rule, feature selection in this setting is directly related to the sparsity of the vector of classification coefficients. This chapter is devoted to a sparse variant of naive Bayes. Our main contributions are as follows.

- We formulate a sparse naive Bayes problem that involves a direct constraint on the cardinality of the vector of classification coefficients, leading to an interpretable naive Bayes model. No hyper-parameter tuning is required in order to achieve the target cardinality.

- We derive an exact solution of sparse naive Bayes in the case of binary data, and an approximate upper bound for general data, and show that it becomes increasingly tight as the marginal contribution of features decreases. Both models can be trained very

efficiently, with an algorithm that scales almost linearly with the number of features and data points, just like classical naive Bayes.

- We show in experiments that our model significantly outperforms simple baselines (*e.g.*, thresholded naive Bayes, odds ratio), and achieves similar performance as more sophisticated feature selection methods, at a fraction of the computing cost.

Related Work on Naive Bayes Improvements. A large body of literature builds on the traditional naive Bayes classifier. A non-extensive list includes the seminal work by [36] introducing Weighted naive Bayes; Lazy Bayesian Learning by [116]; and the Tree-Augmented naive Bayes method by [37]. The paper [103] improves the computational complexity of the aforementioned methods, while maintaining the same accuracy. For a more complete discussion of modifications to naive Bayes, we refer the reader to [50] and the references therein.

Related Work on Naive Bayes and Feature Selection. Of particular interest to this work are methods that employ feature selection. [55] use information-theoretic quantities for feature selection in text classification, while [72] compare a host of different methods and shows the comparative efficacy of the Odds Ratio method. These methods often use ad hoc scoring functions to rank the importance of the different features. [33] uses the mutual information to select features in a fast way while [112] employs a weighting approach for selecting relevant features. [12] achieve soft variable selection by introducing bayesian regularization into the training problem.

5.2 Background on Naive Bayes

In this paper, for simplicity only, we consider a two-class classification problem; the extension to the general multi-class case is straightforward.

Notation. For an integer m, $[m]$ is the set $\{1, \ldots, m\}$. The notation $\mathbf{1}$ denotes a vector of ones, with size inferred from context. The cardinality (number of non-zero elements) in a m-vector x is denoted $\|x\|_0$, whereas that of a finite set $\mathcal{I}$ is denoted $|\mathcal{I}|$. Unless otherwise specified, functional operations (such as $\max(0, \cdot)$) on vectors are performed element-wise. For $k \in [n]$, we say that a vector $w \in \mathbb{R}^n$ is k-sparse or has sparsity level $\alpha\%$ if at most k or $\alpha\%$ of its coefficients are nonzero respectively. For two vectors $f, g \in \mathbb{R}^m$, $f \circ g \in \mathbb{R}^m$ denotes the elementwise product. For a vector z, the notation $s_k(z)$ is the sum of the top k entries. Finally, $\mathbf{Prob}(A)$ denotes the probability of an event A.

Data Setup. We are given a non-negative data matrix $X \in \mathbb{R}_+^{n \times m} = [x^{(1)}, x^{(2)}, \ldots, x^{(n)}]^\top$ consisting of n data points, each with m dimensions (features), and a vector $y \in \{-1, 1\}^n$ that encodes the class information for the n data points, with C_+ and C_- referring to the

positive and negative classes respectively. We define index sets corresponding to each class C_+, C_-, and their respective cardinality, and data averages:

$$\begin{aligned}
&\mathcal{I}_\pm := \{i \in [n] \ : \ y_i = \pm 1\}, \\
&n_\pm = |\mathcal{I}_\pm|, \\
&f_\pm := \sum_{i \in \mathcal{I}_\pm} x^{(i)} = \pm(1/2) X^\top (y \pm \mathbf{1})
\end{aligned}$$

Naive Bayes. We are interested in predicting the class label of a test point $x \in \mathbb{R}^m$ via $\hat{y}(x) = \arg\max_{\epsilon \in \{-1,1\}} \mathbf{Prob}(C_\epsilon \,|\, x)$. To calculate the latter posterior probability, we employ Bayes' rule and then use the "naive" assumption that features are independent of each other: $\mathbf{Prob}(x \mid C_\epsilon) = \prod_{j=1}^m \mathbf{Prob}(x_j \mid C_\epsilon)$, leading to

$$\hat{y}(x) = \arg \max_{\epsilon \in \{-1,1\}} \ \log \mathbf{Prob}(C_\epsilon) + \sum_{j=1}^m \log \mathbf{Prob}(x_j | C_\epsilon). \tag{5.1}$$

In (5.1), we need to have an explicit model for $\mathbf{Prob}(x_j|C_i)$; in the case of binary or integer-valued features, we use Bernoulli or categorical distributions, while in the case of real-valued features we can use a Gaussian distribution. We then use the maximum likelihood principle (MLE) to determine the parameters of those distributions. Using a categorical distribution, $\mathbf{Prob}(C_\pm)$ simply becomes $n_\pm/n$.

Bernoulli Naive Bayes. With binary features, that is, $X \in \{0,1\}^{n \times m}$, we choose the following conditional probability distributions parameterized by two non-negative vectors $\theta^+, \theta^- \in [0,1]^m$. For a given vector $x \in \{0,1\}^m$,

$$\mathbf{Prob}(x_j \mid C_\pm) = (\theta_j^\pm)^{x_j} (1 - \theta_j^\pm)^{1 - x_j}, \ \ j \in [m],$$

hence

$$\sum_{j=1}^m \log \mathbf{Prob}(x_j \mid C_\pm) = x^\top \log \theta^\pm + (\mathbf{1} - x)^\top \log(\mathbf{1} - \theta^\pm).$$

Training a classical Bernoulli naive Bayes model reduces to the problem

$$(\theta_*^+, \theta_*^-) = \arg \max_{\theta^+, \theta^- \in [0,1]^m} \mathcal{L}_{\text{bnb}}(\theta^+, \theta^-; X) \tag{5.2}$$

where the loss is a concave function

$$\begin{aligned}
\mathcal{L}_{\text{bnb}}(\theta^+, \theta^-) = &\sum_{i \in \mathcal{I}_+} \log \mathbf{Prob}(x^{(i)} \mid C_+) \\
&+ \sum_{i \in \mathcal{I}_-} \log \mathbf{Prob}(x^{(i)} \mid C_-) \\
=&f^{+\top} \log \theta^+ + (n_+ \mathbf{1} - f^+)^\top \log(\mathbf{1} - \theta^+) \\
&+ f^{-\top} \log \theta^- + (n_- \mathbf{1} - f^-)^\top \log(\mathbf{1} - \theta^-)
\end{aligned} \tag{5.3}$$

Note that problem (5.2) is decomposable across features and the optimal solution is simply the MLE estimate, that is, $\theta_*^{\pm} = f^{\pm}/n_{\pm}$. From (5.1), we get a linear classification rule: for a given test point $x \in \mathbb{R}^m$, we set $\hat{y}(x) = \mathbf{sign}(v + w_b^\top x)$, where

$$\begin{aligned} v &:= \log \frac{\mathbf{Prob}(C_+)}{\mathbf{Prob}(C_-)} + \mathbf{1}^\top \Big(\log(\mathbf{1} - \theta_*^+) - \log(\mathbf{1} - \theta_*^-) \Big) \\ w_b &:= \log \frac{\theta_*^+ \circ (\mathbf{1} - \theta_*^-)}{\theta_*^- \circ (\mathbf{1} - \theta_*^+)}. \end{aligned} \tag{5.4}$$

Multinomial naive Bayes. With integer-valued features, that is, $X \in \mathbb{N}^{n \times m}$, we choose the following conditional probability distribution, again parameterized by two non-negative m-vectors $\theta^{\pm} \in [0,1]^m$, but now with the constraints $\mathbf{1}^\top \theta^{\pm} = 1$: for given $x \in \mathbb{N}^m$,

$$\begin{aligned} \mathbf{Prob}(x \mid C_{\pm}) &= \frac{(\sum_{j=1}^m x_j)!}{\prod_{j=1}^m x_j!} \prod_{j=1}^m (\theta_j^{\pm})^{x_j} \\ \Rightarrow \log \mathbf{Prob}(x \mid C_{\pm}) &= x^\top \log \theta^{\pm} + \log \left(\frac{(\sum_{j=1}^m x_j)!}{\prod_{j=1}^m x_j!} \right) \end{aligned}$$

While it is essential that the data be binary in the Bernoulli model seen above, the multinomial one can still be used if x is non-negative real-valued, and not integer-valued. Training the classical multinomial model reduces to the problem

$$\begin{aligned} (\theta_*^+, \theta_*^-) = \arg \max_{\theta^+, \theta^- \in [0,1]^m} & \mathcal{L}_{\text{mnb}}(\theta^+, \theta^-) \\ \mathbf{1}^\top \theta^+ = \mathbf{1}^\top \theta^- &= 1 \end{aligned} \tag{5.5}$$

where the loss is a again a concave function

$$\begin{aligned} \mathcal{L}_{\text{mnb}}(\theta^+, \theta^-) &= \sum_{i \in \mathcal{I}_+} \log \mathbf{Prob}(x^{(i)} \mid C_+) \\ &\quad + \sum_{i \in \mathcal{I}_-} \log \mathbf{Prob}(x^{(i)} \mid C_-) \\ &= f^{+\top} \log \theta^+ + f^{-\top} \log \theta^- \end{aligned} \tag{5.6}$$

Again, problem (5.5) is decomposable across features, with the added complexity of equality constraints on $\theta^{\pm}$. The optimal solution is the MLE estimate $\theta_*^{\pm} = f^{\pm}/(\mathbf{1}^\top f^{\pm})$. As before, we get a linear classification rule: for a given test point $x \in \mathbb{R}^m$, we set $\hat{y}(x) = \mathbf{sign}(v + w_m^\top x)$, where

$$v := \log \mathbf{Prob}(C_+) - \log \mathbf{Prob}(C_-), \quad w_m := \log \theta_*^+ - \log \theta_*^- \tag{5.7}$$

5.3 Naive Feature Selection

In this section, we incorporate sparsity constraints into the aforementioned models.

Naive Bayes with Sparsity Constraints

For a given integer $k \in [m]$, with $k < m$, we seek to obtain a naive Bayes classifier that uses at most k features in its decision rule. For this to happen, we need the corresponding coefficient vector, denoted w_b and w_m for the Bernoulli and multinomial cases, and defined in (5.4) and (5.7) respectively, to be k-sparse. For both Bernoulli and multinomial models, this happens if and only if the difference vector $\theta_*^+ - \theta_*^-$ is sparse. By enforcing k-sparsity on the difference vector, the classifier uses less than m features for classification, making the model more interpretable.

Sparse Bernoulli Naive Bayes. In the Bernoulli case, the sparsity-constrained problem becomes

$$\begin{aligned}(\theta_*^+, \theta_*^-) = \arg \max_{\theta^+, \theta^- \in [0,1]^m} \ & \mathcal{L}_{\text{bnb}}(\theta^+, \theta^-; X) \\ \|\theta^+ - \theta^-\|_0 \leq k & \end{aligned} \tag{SBNB}$$

where $\mathcal{L}_{\text{bnb}}$ is defined in (5.3). Here, $\|\cdot\|_0$ denotes the l_0-norm, or cardinality (number of non-zero entries) of its vector argument, and $k < m$ is the user-defined upper bound on the desired cardinality.

Sparse Multinomial Naive Bayes. In the multinomial case, in light of (5.5), our model is written

$$\begin{aligned}(\theta_*^+, \theta_*^-) = \arg \max_{\theta^+, \theta^- \in [0,1]^m} \ & \mathcal{L}_{\text{mnb}}(\theta^+, \theta^-; X) \\ \mathbf{1}^\top \theta^+ = \mathbf{1}^\top \theta^- = 1 & \\ \|\theta^+ - \theta^-\|_0 \leq k & \end{aligned} \tag{SMNB}$$

where $\mathcal{L}_{\text{mnb}}$ is defined in (5.6).

Main Results

Due to the inherent combinatorial and non-convex nature of the cardinality constraint, and the fact that it couples the variables $\theta^\pm$, the above sparse training problems look much more challenging to solve when compared to their classical counterparts, (5.2) and (5.5). We will see in what follows that this is not the case.

Sparse Bernoulli Case. The sparse counterpart to the Bernoulli model, (SBNB), can be solved efficiently in *closed form*, with complexity comparable to that of the classical Bernoulli problem (5.2).

Theorem 3 (Sparse Bernoulli naive Bayes)**.** *Consider the sparse Bernoulli naive Bayes training problem* (SBNB)*, with binary data matrix $X \in \{0,1\}^{n\times m}$. The optimal values of the variables are obtained as follows. Set*

$$\begin{aligned} v &:= (f^+ + f^-) \circ \log\Big(\frac{f^+ + f^-}{n}\Big) \\ &\quad + (n\mathbf{1} - f^+ - f^-) \circ \log\Big(\mathbf{1} - \frac{f^+ + f^-}{n}\Big) \qquad (5.8) \\ w &:= w^+ + w^- \qquad (5.9) \\ w^\pm &:= f^\pm \circ \log\frac{f^+}{n_\pm} + (n_\pm \mathbf{1} - f^\pm) \circ \log\Big(\mathbf{1} - \frac{f^\pm}{n_\pm}\Big). \end{aligned}$$

Then identify a set $\mathcal{I}$ of indices with the k largest elements in $w - v$, and set θ^+_*, θ^-_* according to

$$\theta^+_{*_i} = \theta^-_{*_i} = \frac{1}{n}(f^+_i + f^-_i),\ \forall i \in \mathcal{I}, \quad \theta^\pm_{*_i} = \frac{f^\pm_i}{n_\pm},\ \forall i \notin \mathcal{I}. \qquad (5.10)$$

Proof. For completeness we also include the following proof in Appendix 5.5. First note that an ℓ_0-norm constraint on a m-vector q can be reformulated as

$$\|q\|_0 \le k \Longleftrightarrow \exists\, \mathcal{I} \subseteq [m],\ \ |\mathcal{I}| \le k\ :\ \forall\, i \notin \mathcal{I},\quad q_i = 0.$$

Hence problem (SBNB) is equivalent to

$$\begin{aligned} \max_{\theta^+,\theta^- \in [0,1]^m, \mathcal{I}} & \mathcal{L}_{\text{bnb}}(\theta^+, \theta^-; X) \\ \text{s.t. } & \theta^+_i = \theta^-_i \ \ \forall i \notin \mathcal{I},\ \ \mathcal{I} \subseteq [m],\ \ |\mathcal{I}| \le k \qquad (5.11) \end{aligned}$$

where the complement of the index set $\mathcal{I}$ encodes the indices where variables θ^+, θ^- agree. Then (5.11) becomes

$$p^* := \max_{\mathcal{I}\subseteq[m],\, |\mathcal{I}|\le k} \ \Big(\sum_{i\notin\mathcal{I}} h^\pm_i\Big) + \Big(\sum_{i\in\mathcal{I}} h^+_i + h^-_i\Big) \qquad (5.12)$$

where

$$\begin{aligned} h^\pm_i &= \max_{\theta_i\in[0,1]} (f^+_i + f^-_i)\log\theta_i + (n - f^+_i - f^-_i)\log(1-\theta_i) \\ h^+_i &= \max_{\theta^+_i\in[0,1]} f^+_i \log\theta^+_i + (n_+ - f^+_i)\log(1-\theta^+_i) \\ h^-_i &= \max_{\theta^-_i\in[0,1]} f^-_i \log\theta^-_i + (n_- - f^-_i)\log(1-\theta^-_i) \end{aligned}$$

and where we use the fact that $n_+ + n_- = n$. All the above expressions for $h_i^{\pm}, h_i^+, h_i^-$ have closed form values and solutions

$$\begin{aligned} \theta_i = \theta^+_{*\,i} = \theta^-_{*\,i} &= \frac{1}{n}(f_i^+ + f_i^-), \;\; \forall i \notin \mathcal{I} \\ \theta^{\pm}_{*_i} &= \frac{f_i^{\pm}}{n_{\pm}}, \;\; \forall i \in \mathcal{I} \end{aligned} \tag{5.13}$$

Plugging the above inside the objective of (5.11) results in a Boolean formulation, with a Boolean vector u of cardinality $\leq k$ such that $\mathbf{1} - u$ encodes indices for which entries of θ^+, θ^- agree:

$$p^* := \max_{u \in \mathcal{C}_k} (\mathbf{1} - u)^\top v + u^\top w,$$

where, for $k \in [m]$:

$$\mathcal{C}_k := \{u \;:\; u \in \{0,1\}^m, \; \mathbf{1}^\top u \leq k\},$$

and vectors v, w are as defined in (5.8):

$$\begin{aligned} v &:= (f^+ + f^-) \circ \log\Big(\frac{f^+ + f^-}{n}\Big) \\ &\quad + (n\mathbf{1} - f^+ - f^-) \circ \log\Big(\mathbf{1} - \frac{f^+ + f^-}{n}\Big) \\ w &:= w^+ + w^- \\ w^{\pm} &:= f^{\pm} \circ \log \frac{f^+}{n_{\pm}} + (n_{\pm}\mathbf{1} - f^{\pm}) \circ \log\Big(\mathbf{1} - \frac{f^{\pm}}{n_{\pm}}\Big) \end{aligned}$$

We obtain

$$p^* = \mathbf{1}^\top v + \max_{u \in \mathcal{C}_k} \; u^\top (w - v) = \mathbf{1}^\top v + s_k(w - v),$$

where $s_k(\cdot)$ denotes the sum of the k largest elements in its vector argument. Here we have exploited the fact that the map $z := w - v \geq 0$, which in turn implies that

$$s_k(z) = \max_{u \in \{0,1\}^m \,:\, \mathbf{1}^\top u = k} u^\top z = \max_{u \in \mathcal{C}_k} \; u^\top z.$$

In order to recover an optimal pair (θ_*^+, θ_*^-), we simply identify the set $\mathcal{I}$ of indices with the $m - k$ smallest elements in $w - v$, and set θ_*^+, θ_*^- according to (5.20). □

Note that the complexity of the computation (including forming the vectors $f^{\pm}$, and finding the k largest elements in the appropriate m-vector) grows as $O(mn \log(k))$. This represents a very moderate extra cost compared to the cost of the classical naive Bayes problem, which is $O(mn)$.

Multinomial Case. In the multinomial case, the sparse problem (SMNB) does not admit a closed-form solution. However, we can obtain an upper bound.

Theorem 4 (Sparse multinomial naive Bayes)**.** *Let $\phi(k)$ be the optimal value of* (SMNB)*. Then $\phi(k) \leq \psi(k)$, where $\psi(k)$ is the optimal value of the following one-dimensional convex optimization problem*

$$\psi(k) := C + \min_{\alpha \in [0,1]} s_k(h(\alpha)), \tag{USMNB}$$

where C is a constant, $s_k(\cdot)$ is the sum of the top k entries of its vector argument, and for $\alpha \in (0,1)$,

$$h(\alpha) = \tilde{C} - f^+ \log \alpha - f^- \log(1-\alpha).$$

where $\tilde{C} = f^+ \circ \log f^+ + f^- \circ \log f^- - (f^+ + f^-) \circ \log(f^+ + f^-)$. Furthermore, given an optimal dual variable α_* that solves (USMNB)*, we can reconstruct a primal feasible (sub-optimal) point (θ^+, θ^-) for* (SMNB) *as follows. For α^* optimal for* (USMNB)*, let $\mathcal{I}$ be complement of the set of indices corresponding to the top k entries of $h(\alpha_*)$; then set $B_\pm := \sum_{i \notin \mathcal{I}} f_i^\pm$, and*

$$\begin{aligned} \theta^+_{*\,i} = \theta^-_{*\,i} &= \frac{f_i^+ + f_i^-}{\mathbf{1}^\top (f^+ + f^-)}, \ \forall i \in \mathcal{I} \\ \theta^\pm_{*_i} &= \frac{B_+ + B_-}{B_\pm} \frac{f_i^\pm}{\mathbf{1}^\top (f^+ + f^-)}, \ \forall i \notin \mathcal{I} \end{aligned} \tag{5.14}$$

Proof. See Appendix 5.6. □

The key point here is that, while problem (SMNB) is nonconvex and potentially hard, the dual problem is a one-dimensional convex optimization problem which can be solved very efficiently, using bisection. The number of iterations to localize an optimal α^* with absolute accuracy ϵ grows slowly, as $O(\log(1/\epsilon))$; each step involves the evaluation of a sub-gradient of the objective function, which requires finding the k largest elements in a m-vector, and costs $O(m \log k)$. As before in the Bernoulli case, the complexity of the sparse variant in the multinomial case is $O(mn \log k)$, versus $O(mn)$ for the classical naive Bayes.

Quality estimate. The quality of the bound in the multinomial case can be analysed using bounds on the duality gap based on the Shapley-Folkman theorem.

Theorem 5 (Quality of Sparse Multinomial Naive Bayes Relaxation)**.** *Let $\phi(k)$ be the optimal value of* (SMNB) *and $\psi(k)$ that of the convex relaxation in* (USMNB)*, we have, for $k \geq 4$,*

$$\psi(k-4) \leq \phi(k) \leq \psi(k) \leq \phi(k+4). \tag{5.15}$$

Proof. See Appendix 5.7. □

While we defer details of the proof of Theorem 5 to the Appendix, we provide a high level discussion of how to bound the duality gap. The proof follows from results by [6] (see [29, 52] for a more recent discussion) which are summarized below. Given functions f_i, a vector $b \in \mathbb{R}^m$, and vector-valued functions g_i, $i \in [n]$ that take values in $\mathbb{R}^m$, we consider the following problem:

$$\mathrm{h}_P(u) := \min_x \sum_{i=1}^n f_i(x_i) \;\; : \;\; \sum_{i=1}^n g_i(x_i) \leq b + u \tag{P}$$

in the variables $x_i \in \mathbb{R}^{d_i}$, with perturbation parameter $u \in \mathbb{R}^m$. Let $\mathrm{h}_P(u)^{**}$ be the biconjugate of $\mathrm{h}_P(u)$ defined in (P), then $\mathrm{h}_P(0)^{**}$ is the optimal value of the dual to (P) [29, Lem. 2.3], and [29, Th. I.3] shows the following result.

Theorem 6. *Suppose the functions f_i, g_{ji} in problem* (P) *are proper, 1-coercive, lower semicontinuous and there exists affine minorants for $i = 1, \ldots, n$, $j = 1, \ldots, m$. Let*

$$\bar{p}_j = (m+1) \max_i \rho(g_{ji}), \quad \textit{for } j = 1, \ldots, m \tag{5.16}$$

then

$$\mathrm{h}_P(\bar{p}) \leq \mathrm{h}_P(0)^{**} + (m+1) \max_i \rho(f_i). \tag{5.17}$$

where $\rho(f) \triangleq \sup_{x \in \mathbf{dom}(f)} \{f(x) - f^{**}(x)\}$.

Hence by bounding the non-convexity of the ℓ_0 constraint, we are able to bound the overall duality gap.

The bound in Theorem 5 implies in particular

$$\psi(k-4) \leq \phi(k) \leq \psi(k-4) + \Delta(k), \text{ for } k \geq 4,$$

where $\Delta(k) := \psi(k) - \psi(k-4)$. This means that if $\psi(k)$ does not vary too fast with k, so that $\Delta(k)$ is small, then the duality gap in problem (SMNB) is itself small, bounded by $\Delta(k)$; then solving the convex problem (USMNB) will yield a good approximate solution to (SMNB). This means that when the marginal contribution of additional features, i.e. $\Delta(k)/\psi(k)$ becomes small, our bound becomes increasingly tight. The "elbow heuristic" is often used to infer the number of relevant features k^*, with $\psi(k)$ increasing fast when $k < k^*$ and much more slowly when $k \geq k^*$. In this scenario, our bound becomes tight for $k \geq k^*$.

5.4 Experiments

In this section, we compare our sparse multinomial model (SMNB) against other feature selection methods (Experiments 1-3) we empirically show the quality of our relaxation on a synthethic dataset (Experiment 4). For the former experiments, we do not use deep learning methods since we want to compare the features selected rather than the end-to-end training accuracy. For this reason, we compare (SMNB) against traditional ℓ_1 methods, recursive feature elimination (RFE) methods, and other sparsity-inducing methods.

Feature Vectors	Amazon	IMDB	Twitter	MPQA	SST2
Count Vector	31,666	103,124	273,779	6,208	16,599
tf-idf	5000	5000	5000	5000	5000
tf-idf wrd bigram	5000	5000	5000	5000	5000
tf-idf char bigram	5000	5000	5000	4838	5000
n_{TRAIN}	8000	25,000	1,600,000	8484	76,961
n_{TEST}	2000	25,000	498	2122	1821

Table 5.1: **Experiment 1 data**: Number of features for each type of feature vector for each data set. For tf-idf feature vectors, we fix the maximum number of features to 5000 for all data sets. The last two rows show the number of training and test samples.

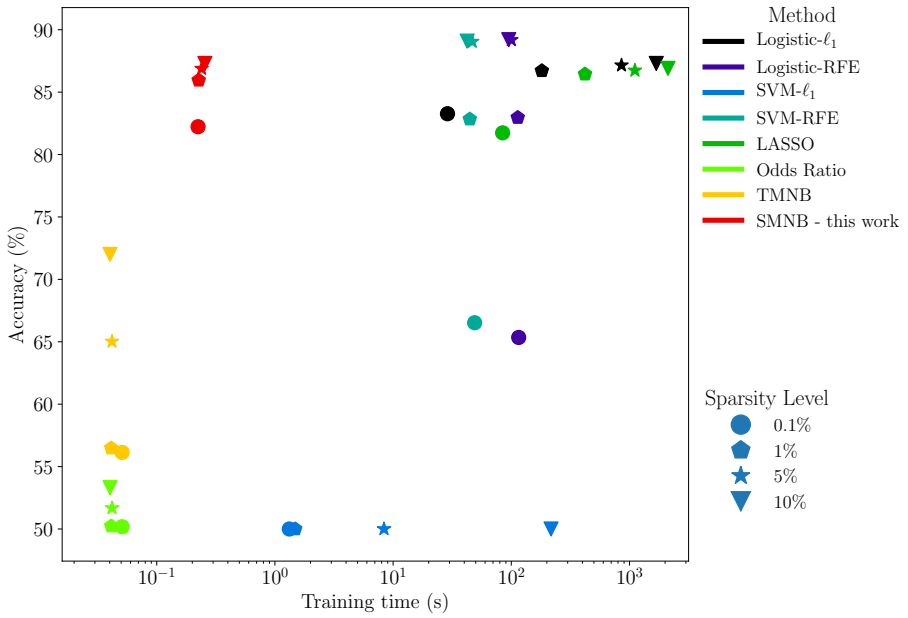

Figure 5.1: **Experiment 1:** Accuracy versus run time with the IMDB dataset/Count Vector with MNB in stage 2, showing performance on par with the best feature selection methods, at fraction of computing cost. Times *do not* include the cost of grid search to reach the target cardinality for ℓ_1-based methods. For more details on the experiment, see Appendix 5.8.

Feature Vectors	Amazon	IMDB	Twitter	MPQA	SST2
Count Vector	31,666	103,124	273,779	6,208	16,599
tf-idf	31,666	103,124	273,779	6,208	16,599
tf-idf wrd bigram	870,536	8,950,169	12,082,555	27,603	227,012
tf-idf char bigram	25,019	48,420	17,812	4838	7762

Table 5.2: **Experiment 2 data**: Number of features for each type of feature vector for each data set with no limit on the number of features for the tf-idf vectors. The train/test split is the same as in Table 5.1.

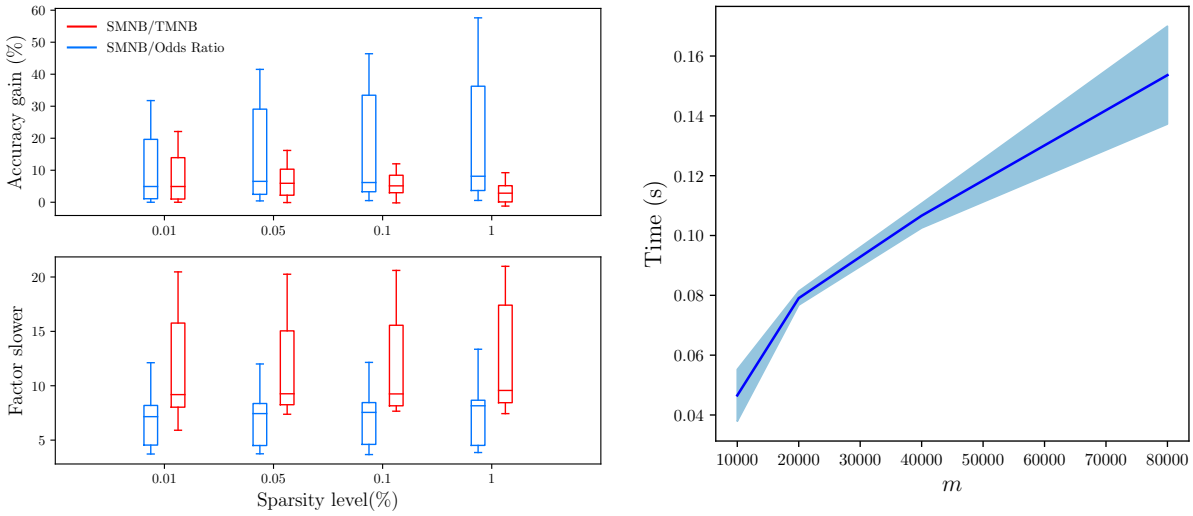

Figure 5.2: **Experiment 2 (Left):** Accuracy gain for our method (top panel) and factor slower (bottom panel) over all data sets listed in Table 5.2 with MNB in stage 2, showing substantial performance increase with a constant increase in computational cost. **Experiment 3 (Right):** Run time with IMDB dataset/tf-idf vector data set, with increasing m, k with fixed ratio k/m, empirically showing (sub-) linear complexity.

Experiment 1: Feature Selection

In the next three experiments, we compare (SMNB) with other feature selection methods for sentiment classification on five different text data sets. Some details on the data sets sizes are given in Table 5.1. More information on these data sets and how they were pre-processed are given in Appendix 5.8.

For each data set and each type of feature vector, we perform the following two-stage

procedure. In the first step, we employ a feature selection method to attain a desired sparsity level of (0.1%, 1%, 5%, 10%); in the second step, we train a classifier based on the selected features. Specifically, we use ℓ_1-regularized logistic regression, logistic regression with recursive feature elimination (RFE), ℓ_1-regularized support vector machine (SVM), SVM with RFE, LASSO, thresholded Multinomial naive Bayes (TMNB), the Odds Ratio metric described by [72] and (SMNB) in the first step. Then using the selected features, in the second step we train a logistic model, a MNB model, and a SVM. Thresholded multinomial naive Bayes (TMNB) means we train a multinomial naive Bayes model and then select the features corresponding to indices of the largest absolute value entries of the vector of classification coefficients w_m, as defined in (5.7). For each desired sparsity level and each data set in the first step, we do a grid search over the optimal Laplace smoothing parameter for MNB for each type of feature vector. We use this same parameter in (SMNB). All models were implemented using Scikit-learn [82]. Figure 5.1 shows that (SMNB) is competitive with other feature selection methods, consistently maintaining a high test set accuracy, while only taking a fraction of the time to train; for a sparsity level of 5%, a logistic regression model with ℓ_1 penalty takes more than 1000 times longer to train.

Experiment 2: large-scale feature selection

For this experiment, we consider the same data sets as before, but do not put any limit on the number of features for the tf-idf vectors. Due to the large size of the data sets, most of the feature selection methods in Experiment 1 are not feasible. We use the same two-stage procedure as before: 1) do feature selection using TMNB, the Odds Ratio method and our method (USMNB), and 2) train a MNB model using the features selected in stage 1. We tune the hyperparameters for MNB and (USMNB) the same way as in Experiment 2. In this experiment, we focus on sparsity levels of $0.01\%, 0.05\%, 0.1\%, 1\%$. Table 5.2 summarizes the data used in Experiment 2 and in Table 5.3 we display the average training time for (USMNB).

Figure 5.2 shows that, even for large datasets with millions of features and data points, our method, implemented on a standard CPU with a non-optimized solver, takes at most a few seconds, while providing a significant improvement in performance. See Appendix 5.8 for the accuracy versus sparsity plot for each data set and each type of feature vector.

Experiment 3: complexity

Using the IMDB dataset in Table 5.1, we perform the following experiment: we fix a sparsity pattern $k/m = 0.05$ and then increase k and m. Where we artificially set the number of tf-idf features to 5000 in Experiment 1, here we let the number of tf-idf features vary from $10,000$ to $80,000$. We then plot the the time it takes to train (SMNB) at a the fixed 5% sparsity level. Figure 5.2 shows that for a fixed sparsity level, the complexity of our method appears to be sub-linear.

	Amazon	IMDB	Twitter	MPQA	SST2
F_C	0.043	0.22	1.15	0.0082	0.037
F_{t_1}	0.033	0.16	0.89	0.0080	0.027
F_{t_2}	0.68	9.38	13.25	0.024	0.21
F_{t_3}	0.076	0.47	4.07	0.0084	0.082

Table 5.3: **Experiment 2 run times**: Average run time (in seconds, with a standard CPU and a non-optimized implementation) over $4 \times 30 = 120$ values for different sparsity levels and 30 randomized train/test splits per sparsity level for each data set and each type of feature vector. On the largest data set (Twitter, $\sim$ 12M features, $\sim$ 1.6M data points), the computation takes less than 15 seconds. For the full distribution of run times, see Appendix 5.8. $F_C, F_{t_1}, F_{t_2}, F_{t_3}$ refer to the count vector, tf-idf, tf-idf word bigram, and tf-idf character bigram feature vectors respectively.

Experiment 4: Duality Gap

In this experiment, we generate random synthetic data with uniform independent entries: $f^{\pm} \sim U[0,1]^m$, where $m \in \{30, 3000\}$. We then normalize $f^{\pm}$ and compute $\psi(k)$ and $\psi(k-4)$ for $4 \leq k \leq m$ and plot how this gap evolves as k increases. For each value of k, we also plot the value of the reconstructed primal feasible point, as detailed in Theorem 4. The latter serves as a lower bound on the true value $\phi(k)$, which can be used to test *a posteriori* if our bound is accurate.

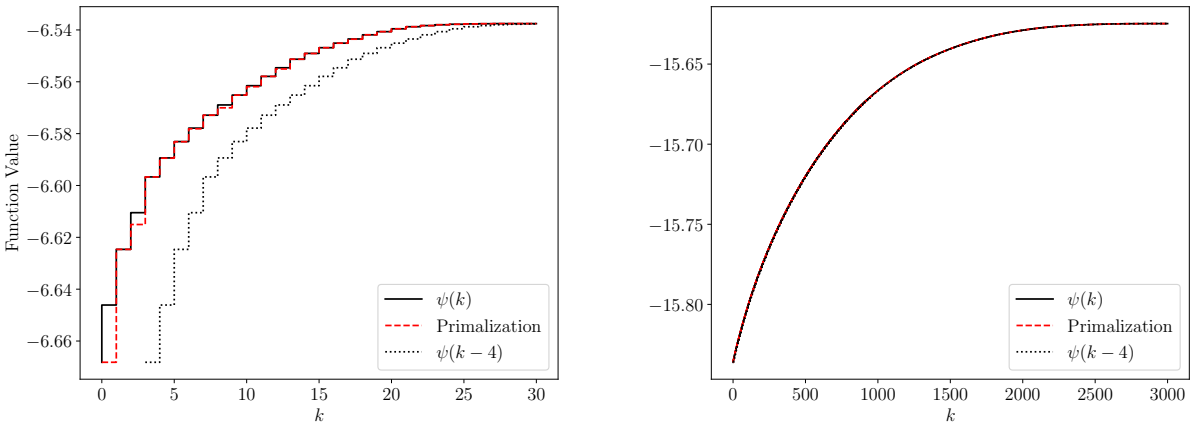

Figure 5.3: **Experiment 4:** Duality gap bound versus sparsity level for $m = 30$ (top panel) and $m = 3000$ (bottom panel), showing that the duality gap quickly closes as m or k increase.

Figure 5.3 shows that, as the number of features m or the sparsity parameter k increases, the duality gap bound decreases. Figure 5.3 also shows that the *a posteriori* gap is almost always zero, implying strong duality. In particular, as shown in Figure 5.3(b), as the number of features increases, the gap between the bounds and the primal feasible point's value becomes negligible for all values of k. This indicates that we can solve the original, non-convex problem (SBNB) by instead solving a 1-dimensional dual problem and constructing a primal feasible solution in closed form.

5.5 Proof of Theorem 3

Theorem 3 (Sparse Bernoulli naive Bayes)**.** *Consider the sparse Bernoulli naive Bayes training problem* (SBNB)*, with binary data matrix $X \in \{0,1\}^{n\times m}$. The optimal values of the variables are obtained as follows. Set*

$$v := (f^+ + f^-) \circ \log\Big(\frac{f^+ + f^-}{n}\Big) + (n\mathbf{1} - f^+ - f^-) \circ \log\Big(\mathbf{1} - \frac{f^+ + f^-}{n}\Big),$$
$$w := w^+ + w^-, \;\; w^\pm := f^\pm \circ \log\frac{f^+}{n_\pm} + (n_\pm \mathbf{1} - f^\pm) \circ \log\Big(\mathbf{1} - \frac{f^\pm}{n_\pm}\Big).$$

Then identify a set $\mathcal{I}$ of indices with the k largest elements in $w-v$, and set θ_*^+, θ_*^- according to

$$\theta^+_{*_i} = \theta^-_{*_i} = \frac{1}{n}(f_i^+ + f_i^-), \; \forall i \notin \mathcal{I}, \quad \theta^\pm_{*_i} = \frac{f_i^\pm}{n_\pm}, \; \forall i \in \mathcal{I}.$$

First note that an ℓ_0-norm constraint on a m-vector q can be reformulated as

$$\|q\|_0 \le k \Longleftrightarrow \exists\, \mathcal{I} \subseteq [m], \;\; |\mathcal{I}| \le k \;\; : \;\; \forall\, i \notin \mathcal{I}, \quad q_i = 0.$$

Hence problem (SBNB) is equivalent to

$$\max_{\theta^+,\theta^- \in [0,1]^m, \mathcal{I}} \mathcal{L}_{\rm bnb}(\theta^+, \theta^-; X) \;\; : \;\; \theta_i^+ = \theta_i^- \;\; \forall i \notin \mathcal{I}, \;\; \mathcal{I} \subseteq [m], \;\; |\mathcal{I}| \le k, \tag{5.18}$$

where the complement of the index set $\mathcal{I}$ encodes the indices where variables θ^+, θ^- agree. Then (5.18) becomes

$$\begin{aligned} p^* := \max_{\mathcal{I}\subseteq[m],\, |\mathcal{I}|\le k} \;\; & \sum_{i\notin\mathcal{I}} \Big(\max_{\theta_i\in[0,1]} (f_i^+ + f_i^-)\log\theta_i + (n - f_i^+ - f_i^-)\log(1-\theta_i) \Big) \\ & + \sum_{i\in\mathcal{I}} \Big(\max_{\theta_i^+\in[0,1]} f_i^+ \log\theta_i^+ + (n_+ - f_i^+)\log(1-\theta_i^+) \Big) \\ & + \sum_{i\in\mathcal{I}} \Big(\max_{\theta_i^-\in[0,1]} f_i^- \log\theta_i^- + (n_- - f_i^-)\log(1-\theta_i^-) \Big). \end{aligned} \tag{5.19}$$

where we use the fact that $n_+ + n_- = n$. All the sub-problems in the above can be solved in closed-form, yielding the optimal solutions

$$\theta^+_{*\,i} = \theta^-_{*\,i} = \frac{1}{n}(f_i^+ + f_i^-), \;\; \forall i \notin \mathcal{I}, \quad \text{and} \quad \theta^\pm_{*i} = \frac{f_i^\pm}{n_\pm}, \;\; \forall i \in \mathcal{I}. \tag{5.20}$$

Plugging the above inside the objective of (5.18) results in a Boolean formulation, with a Boolean vector u of cardinality $\leq k$ such that $\mathbf{1} - u$ encodes indices for which entries of θ^+, θ^- agree:

$$p^* := \max_{u \in \mathcal{C}_k} (\mathbf{1} - u)^\top v + u^\top w,$$

where, for $k \in [m]$:

$$\mathcal{C}_k := \{u \;:\; u \in \{0,1\}^m, \; \mathbf{1}^\top u \leq k\},$$

and vectors v, w are as defined in (5.8):

$$v := (f^+ + f^-) \circ \log\Big(\frac{f^+ + f^-}{n}\Big) + (n\mathbf{1} - f^+ - f^-) \circ \log\Big(\mathbf{1} - \frac{f^+ + f^-}{n}\Big),$$
$$w := w^+ + w^-, \;\; w^\pm := f^\pm \circ \log \frac{f^+}{n_\pm} + (n_\pm \mathbf{1} - f^\pm) \circ \log\Big(\mathbf{1} - \frac{f^\pm}{n_\pm}\Big).$$

We obtain

$$p^* = \mathbf{1}^\top v + \max_{u \in \mathcal{C}_k} u^\top (w - v) = \mathbf{1}^\top v + s_k(w - v),$$

where $s_k(\cdot)$ denotes the sum of the k largest elements in its vector argument. Here we have exploited the fact that the map $z := w - v \geq 0$, which in turn implies that

$$s_k(z) = \max_{u \in \{0,1\}^m \,:\, \mathbf{1}^\top u = k} u^\top z = \max_{u \in \mathcal{C}_k} u^\top z.$$

In order to recover an optimal pair (θ^+_*, θ^-_*), we simply identify the set $\mathcal{I}$ of indices with the k largest elements in $w - v$, and set θ^+_*, θ^-_* according to (5.20).

5.6 Proof of Theorem 4

Theorem 4 (Sparse Multinomial Naive Bayes)**.** *Let $\phi(k)$ be the optimal value of* (SMNB)*. Then $\phi(k) \leq \psi(k)$, where $\psi(k)$ is the optimal value of the following one-dimensional convex optimization problem*

$$\psi(k) := C + \min_{\alpha \in [0,1]} \; s_k(h(\alpha)), \tag{USMNB}$$

where C is a constant, $s_k(\cdot)$ is the sum of the top k entries of its vector argument, and for $\alpha \in (0,1)$

$$h(\alpha) := f_+ \circ \log f_+ + f_- \circ \log f_- - (f_+ + f_-) \circ \log(f_+ + f_-) - f_+ \log \alpha - f_- \log(1 - \alpha).$$

Further, given an optimal dual variable α_* that solves (USMNB)*, we can reconstruct a primal feasible (sub-optimal) point (θ^+, θ^-) for* (SMNB) *as follows. For α^* optimal for* (USMNB)*, let $\mathcal{I}$ be complement of the set of indices corresponding to the top k entries of $h(\alpha_*)$; then set $B_\pm := \sum_{i \notin \mathcal{I}} f_i^\pm$, and*

$$\theta^+_{*\,i} = \theta^-_{*\,i} = \frac{f_i^+ + f_i^-}{\mathbf{1}^\top(f^+ + f^-)},\ \forall i \in \mathcal{I}, \quad \theta^\pm_{*_i} = \frac{B_+ + B_-}{B_\pm}\frac{f_i^\pm}{\mathbf{1}^\top(f^+ + f^-)},\ \forall i \notin \mathcal{I}. \tag{5.21}$$

Proof. We begin by deriving the expression for the upper bound $\psi(k)$.

Duality bound. We first derive the bound stated in the theorem. Problem (SMNB) is written

$$(\theta^+_*, \theta^-_*) = \arg \max_{\theta^+, \theta^- \in [0,1]^m} f^{+\top} \log \theta^+ + f^{-\top} \log \theta^- \ : \ \begin{array}{l} \mathbf{1}^\top \theta^+ = \mathbf{1}^\top \theta^- = 1, \\ \|\theta^+ - \theta^-\|_0 \leq k. \end{array} \tag{SMNB}$$

By weak duality we have $\phi(k) \leq \psi(k)$ where

$$\psi(k) := \min_{\substack{\mu^+, \mu^- \\ \lambda \geq 0}} \max_{\theta^+, \theta^- \in [0,1]^m} f^{+\top} \log \theta^+ + f^{-\top} \log \theta^- + \mu^+(1 - \mathbf{1}^\top \theta^+) + \mu^-(1 - \mathbf{1}^\top \theta^-) \\ + \lambda(k - \|\theta^+ - \theta^-\|_0).$$

The inner maximization is separable across the components of θ^+, θ^- since $\|\theta^+ - \theta^-\|_0 = \sum_{i=1}^m \mathbf{1}_{\{\theta_i^+ \neq \theta_i^-\}}$. To solve it, we thus only need to consider one dimensional problems written

$$\max_{q,r \in [0,1]} f_i^+ \log q + f_i^- \log r - \mu^+ q - \mu^- r - \lambda \mathbb{1}_{\{q \neq r\}}, \tag{5.22}$$

where $f_i^+, f_i^- > 0$ and $\mu^\pm > 0$ are given. We can split the max into two cases; one case in which $q = r$ and another when $q \neq r$, then compare the objective values of both solutions and take the larger one. Hence (5.22) becomes

$$\max\Big(\max_{u \in [0,1]} (f_i^+ + f_i^-) \log u - (\mu^+ + \mu^-)u, \max_{q,r \in [0,1]} f_i^+ \log q + f_i^- \log r - \mu^+ q - \mu^- r - \lambda \Big).$$

Each of the individual maximizations can be solved in closed form, with optimal point

$$u^* = \frac{(f_i^+ + f_i^-)}{\mu^+ + \mu^-}, \quad q^* = \frac{f_i^+}{\mu^+}, \quad r^* = \frac{f_i^-}{\mu^-}. \tag{5.23}$$

Note that none of u^*, q^*, r^* can be equal to either 0 or 1, which implies $\mu^+, \mu^- > 0$. Hence (5.22) reduces to

$$\max\Big((f_i^+ + f_i^-) \log\Big(\frac{(f_i^+ + f_i^-)}{\mu^+ + \mu^-}\Big), f_i^+ \log\Big(\frac{f_i^+}{\mu^+}\Big) + f_i^- \log\Big(\frac{f_i^-}{\mu^-}\Big) - \lambda \Big) - (f_i^+ + f_i^-). \tag{5.24}$$

We obtain, with $S := \mathbf{1}^\top(f^+ + f^-)$,

$$\psi(k) = -S + \min_{\substack{\mu^+,\mu^->0 \\ \lambda\geq 0}} \ \mu^+ + \mu^- + \lambda k + \sum_{i=1}^m \max(v_i(\mu), w_i(\mu) - \lambda). \tag{5.25}$$

where, for given $\mu = (\mu^+, \mu^-) > 0$,

$$v(\mu) := (f^+ + f^-) \circ \log\Big(\frac{f^+ + f^-}{\mu^+ + \mu^-}\Big), \ \ w(\mu) := f^+ \circ \log\Big(\frac{f^+}{\mu^+}\Big) + f^- \circ \log\Big(\frac{f^-}{\mu^-}\Big).$$

Recall the variational form of $s_k(z)$. For a given vector $z \geq 0$, Lemma 6 shows

$$s_k(z) = \min_{\lambda\geq 0} \ \lambda k + \sum_{i=1}^m \max(0, z_i - \lambda).$$

Problem (5.25) can thus be written

$$\begin{aligned}
\psi(k) &= -S + \min_{\substack{\mu>0 \\ \lambda\geq 0}} \ \mu^+ + \mu^- + \lambda k + \mathbf{1}^\top v(\mu) + \sum_{i=1}^m \max(0, w_i(\mu) - v_i(\mu) - \lambda) \\
&= -S + \min_{\mu>0} \ \mu^+ + \mu^- + \mathbf{1}^\top v(\mu) + s_k(w(\mu) - v(\mu)),
\end{aligned}$$

where the last equality follows from $w(\mu) \ \geq v(\mu)$, valid for any $\mu > 0$. To prove this, observe that the negative entropy function $x \to x \log x$ is convex, implying that its perspective P also is. The latter is the function with domain $\mathbb{R}_+ \times \mathbb{R}_{++}$, and values for $x \geq 0$, $t > 0$ given by $P(x,t) = x \log(x/t)$. Since P is homogeneous and convex (hence subadditive), we have, for any pair z_+, z_- in the domain of P: $P(z_+ + z_-) \leq P(z_+) + P(z_-)$. Applying this to $z_\pm := (f_i^\pm, \mu_i^+)$ for given $i \in [m]$ results in $w_i(\mu) \geq v_i(\mu)$, as claimed.

We further notice that the map $\mu \to w(\mu) - v(\mu)$ is homogeneous, which motivates the change of variables $\mu_\pm = t\, p_\pm$, where $t = \mu_+ + \mu_- > 0$ and $p_\pm > 0$, $p_+ + p_- = 1$. The problem reads

$$\begin{aligned}
\psi(k) &= -S + (f^+ + f^-)^\top \log(f^+ + f^-) + \min_{\substack{t>0,\, p>0, \\ p_+ + p_- = 1}} \{t - S \log t + s_k(H(p))\} \\
&= C + \min_{p>0,\, p_+ + p_- = 1} \ s_k(H(p)),
\end{aligned}$$

where $C := (f^+ + f^-)^\top \log(f^+ + f^-) - S \log S$, because $t = S$ at the optimum, and

$$H(p) := v - f^+ \circ \log p_+ - f^- \circ \log p_-,$$

with

$$v = f^+ \circ \log f^+ + f^- \circ \log f^- - (f^+ + f^-) \circ \log(f^+ + f^-).$$

Solving for $\psi(k)$ thus reduces to a 1D bisection

$$\psi(k) = C + \min_{\alpha \in [0,1]} \; s_k(h(\alpha)),$$

where

$$h(\alpha) := H(\alpha, 1-\alpha) = v - f^+ \log \alpha - f^- \log(1-\alpha).$$

This establishes the first part of the theorem. Note that it is straightforward to check that with $k = n$, the bound is exact: $\phi(n) = \psi(n)$.

Primalization. Next we focus on recovering a primal feasible (sub-optimal) point $(\theta^{+\text{sub}}, \theta^{-\text{sub}})$ from the dual bound obtained before. Assume that α_* is optimal for the dual problem (USMNB). We sort the vector $h(\alpha_*)$ and find the indices corresponding to the top k entries. Denote the complement of this set of indices by $\mathcal{I}$. These indices are then the candidates for which $\theta_i^+ = \theta_i^-$ for $i \in \mathcal{I}$ in the primal problem to eliminate the cardinality constraint. Hence we are left with solving

$$\begin{aligned} (\theta^{+\text{sub}}, \theta^{-\text{sub}}) = \arg \max_{\theta^+, \theta^- \in [0,1]^m} \; & f^{+\top} \log \theta^+ + f^{-\top} \log \theta^- \\ \text{s.t.} & \mathbf{1}^\top \theta^+ = \mathbf{1}^\top \theta^- = 1, \\ & \theta_i^+ = \theta_i^-, \;\; i \in \mathcal{I} \end{aligned} \tag{5.26}$$

or, equivalently

$$\begin{aligned} \max_{\theta, \theta^+, \theta^-, s \in [0,1]} & \sum_{i \in \mathcal{I}} (f_i^+ + f_i^-) \log \theta_i + \sum_{i \notin \mathcal{I}} (f_i^+ \log \theta_i^+ + f_i^- \log \theta_i^-) \\ \text{s.t. } & \mathbf{1}^\top \theta^+ = \mathbf{1}^\top \theta^- = 1 - s, \;\; \mathbf{1}^\top \theta = s. \end{aligned} \tag{5.27}$$

For given $\kappa \in [0,1]$, and $f \in \mathbb{R}^m_{++}$, we have

$$\max_{u \,:\, \mathbf{1}^\top u = \kappa} \; f^\top \log(u) = f^\top \log f - (\mathbf{1}^\top f) \log(\mathbf{1}^\top f) + (\mathbf{1}^\top f) \log \kappa,$$

with optimal point given by $u^* = (\kappa / (\mathbf{1}^\top f)) f$. Applying this to problem (5.27), we obtain that the optimal value of s is given by

$$s^* = \arg \max_{s \in (0,1)} \; \{ A \log s + B \log(1-s) \} = \frac{A}{A+B},$$

where

$$A := \sum_{i \in \mathcal{I}} (f_i^+ + f_i^-), \;\; B_\pm := \sum_{i \notin \mathcal{I}} f_i^\pm, \;\; B := B_+ + B_- = \mathbf{1}^\top (f^+ + f^-) - A.$$

We obtain

$$\theta_i^{+\text{sub}} = \theta_i^{-\text{sub}} = \frac{s^*}{A} (f_i^+ + f_i^-), \;\; i \in \mathcal{I}, \;\; \theta_i^{\pm\text{sub}} = \frac{(1-s^*)}{B_\pm (A+B)} f_i^\pm, \;\; i \notin \mathcal{I},$$

which further reduces to the expression stated in the theorem. □

5.7 Proof of Theorem 5

The proof follows from results by [6] (see also [29, 52] for a more recent discussion) which are briefly summarized below for the sake of completeness. Given functions f_i, a vector $b \in \mathbb{R}^m$, and vector-valued functions g_i, $i \in [n]$ that take values in $\mathbb{R}^m$, we consider the following problem:

$$\mathrm{h}_P(u) := \min_x \sum_{i=1}^n f_i(x_i) \;\; : \;\; \sum_{i=1}^n g_i(x_i) \leq b + u \tag{P}$$

in the variables $x_i \in \mathbb{R}^{d_i}$, with perturbation parameter $u \in \mathbb{R}^m$. We first recall some basic results about conjugate functions and convex envelopes.

Biconjugate and convex envelope. Given a function f, not identically $+\infty$, minorized by an affine function, we write

$$f^*(y) \triangleq \inf_{x \in \mathbf{dom}\, f} \{y^\top x - f(x)\}$$

the conjugate of f, and $f^{**}(y)$ its biconjugate. The biconjugate of f (aka the convex envelope of f) is the pointwise supremum of all affine functions majorized by f (see e.g. [86, Th. 12.1] or [46, Th. X.1.3.5]), a corollary then shows that $\mathbf{epi}(f^{**}) = \overline{\mathbf{Co}(\mathbf{epi}(f))}$. For simplicity, we write $S^{**} = \overline{\mathbf{Co}(S)}$ for any set S in what follows. We will make the following technical assumptions on the functions f_i and g_i in our problem.

Assumption 5. *The functions $f_i : \mathbb{R}^{d_i} \to \mathbb{R}$ are proper, 1-coercive, lower semicontinuous and there exists an affine function minorizing them.*

Note that coercivity trivially holds if $\mathbf{dom}(f_i)$ is compact (since f can be set to $+\infty$ outside w.l.o.g.). When Assumption 5 holds, $\mathbf{epi}(f^{**})$, f_i^{**} and hence $\sum_{i=1}^n f_i^{**}(x_i)$ are closed [46, Lem. X.1.5.3]. Also, as in e.g. [29], we define the lack of convexity of a function as follows.

Definition 6. *Let $f : \mathbb{R}^d \to \mathbb{R}$, we let*

$$\rho(f) \triangleq \sup_{x \in \mathbf{dom}(f)} \{f(x) - f^{**}(x)\} \tag{5.28}$$

Many other quantities measure lack of convexity (see e.g. [6, 10] for further examples). In particular, the nonconvexity measure $\rho(f)$ can be rewritten as

$$\rho(f) = \sup_{\substack{x_i \in \mathbf{dom}(f) \\ \mu \in \mathbb{R}^{d+1}}} \left\{ f\left(\sum_{i=1}^{d+1} \mu_i x_i\right) - \sum_{i=1}^{d+1} \mu_i f(x_i) : \mathbf{1}^\top \mu = 1, \mu \geq 0 \right\} \tag{5.29}$$

when f satisfies Assumption 5 (see [46, Th. X.1.5.4]).

Bounds on the duality gap and the Shapley-Folkman Theorem Let $\mathrm{h}_P(u)^{**}$ be the biconjugate of $\mathrm{h}_P(u)$ defined in (P), then $\mathrm{h}_P(0)^{**}$ is the optimal value of the dual to (P) [29, Lem. 2.3], and [29, Th. I.3] shows the following result.

Theorem 7. *Suppose the functions f_i, g_{ji} in problem* (P) *satisfy Assumption 5 for $i = 1, \ldots, n$, $j = 1, \ldots, m$. Let*

$$\bar{p}_j = (m+1) \max_i \rho(g_{ji}), \quad \textit{for } j = 1, \ldots, m \tag{5.30}$$

then

$$\mathrm{h}_P(\bar{p}) \leq \mathrm{h}_P(0)^{**} + (m+1) \max_i \rho(f_i). \tag{5.31}$$

where $\rho(\cdot)$ is defined in Def. 6.

We are now ready to prove Theorem 5, whose proof follows from Theorem 7 above.

Theorem 8 (Quality of Sparse Multinomial Naive Bayes Relaxation)**.** *Let $\phi(k)$ be the optimal value of* (SMNB) *and $\psi(k)$ that of the convex relaxation in* (USMNB)*, we have for $k \geq 4$,*

$$\psi(k-4) \leq \phi(k) \leq \psi(k) \leq \phi(k+4).$$

for $k \geq 4$.

Proof. Problem (SMNB) is *separable* and can be written in perturbation form as in the result by [29, Th. I.3] recalled in Theorem 7, to get

$$\begin{array}{rlll} \mathrm{h}_P(u) = & \min_{q,r} & -f^{+\top} \log q - f^{-\top} \log r \\ & \text{subject to} & \mathbf{1}^\top q = 1 + u_1, \\ & & \mathbf{1}^\top r = 1 + u_2, \\ & & \sum_{i=1}^m \mathbf{1}_{q_i \neq r_i} \leq k + u_3 \end{array} \tag{5.32}$$

in the variables $q, r \in [0,1]^m$, where $u \in \mathbb{R}^3$ is a perturbation vector. By construction, we have $\phi(k) = -\mathrm{h}_P(0)$ and $\phi(k+l) = -\mathrm{h}_P((0,0,l))$. Note that the functions $\mathbf{1}_{q_i \neq r_i}$ are lower semicontinuous and, because the domain of problem (SMNB) is compact, the functions

$$f_i^+ \log q_i + q_i + f_i^- \log r_i + r_i + \mathbf{1}_{q_i \neq r_i}$$

are 1-coercive for $i = 1, \ldots, m$ on the domain and satisfy Assumption 5 above.

Now, because $q, r \geq 0$ with $\mathbf{1}^\top q = \mathbf{1}^\top r = 1$, we have $q - r \in [-1,1]^m$ and the convex envelope of $\mathbf{1}_{q_i \neq r_i}$ on $q, r \in [0,1]^m$ is $|q_i - r_i|$, hence the *lack of convexity* (5.29) of $\mathbf{1}_{q_i \neq r_i}$ on $[0,1]^2$ is bounded by one, because

$$\rho(\mathbf{1}_{x \neq y}) := \sup_{x,y \in [0,1]} \{\mathbf{1}_{y \neq x} - |x - y|\} = 1$$

which means that $\max_{i=1,\dots,n} \rho(g_{3i}) = 1$ in the statement of Theorem 7. The fact that the first two constraints in problem (5.32) are convex means that $\max_{i=1,\dots,n} \rho(g_{ji}) = 0$ for $j = 1, 2$, and the perturbation vector in (5.30) is given by $\bar{p} = (0, 0, 4)$, because there are three constraints in problem (5.32) so $m = 3$ in (5.30), hence

$$\mathrm{h}_P(\bar{p}) = \mathrm{h}_P((0,0,4)) = -\phi(k+4).$$

The objective function being convex separable, we have $\max_{i=1,\dots,n} \rho(f_i) = 0$. Theorem 7 then states that

$$\mathrm{h}_P(\bar{p}) = \mathrm{h}_P((0,0,4)) = -\phi(k+4) \leq \mathrm{h}_P(0)^{**} + 0 = -\psi(k)$$

because $-\mathrm{h}_P(0)^{**}$ is the optimal value of the dual to $\phi(k)$ which is here $\psi(k)$ defined in Theorem 4. The other bound in (5.15), namely $\phi(k) \leq \psi(k)$, follows directly from weak duality. □

Primalization. We first derive the second dual of problem (P), i.e. the dual of problem (USMNB), which will be used to extract good primal solutions.

Proposition 9. *A dual of problem* (USMNB) *is written*

$$\begin{aligned} \textit{max.} \quad & z^\top (g \circ \log(g)) + x^\top (f^+ \circ \log(f^+) + f^- \circ \log(f^-)) + (x^\top g)\log(x^\top g) - (x^\top g) \\ & -(\mathbf{1}^\top g)\log(\mathbf{1}^\top g) - (x^\top f^+)\log(x^\top f^+) - (x^\top f^-)\log(x^\top f^-) \qquad \text{(D)} \\ \textit{s.t.} \quad & x + z = \mathbf{1}, \;\; \mathbf{1}^\top x \leq k, \;\; x \geq 0, \;\; z \geq 0 \end{aligned}$$

in the variables $x, z \in \mathbb{R}^n$. *Furthermore, strong duality holds between the dual* (USMNB) *and its dual* (D).

Proof. The dual optimum value $\psi(k)$ in (USMNB) can be written as in (5.25),

$$\psi(k) = -S + \min_{\substack{\mu^+,\mu^->0 \\ \lambda \geq 0}} \; \mu^+ + \mu^- + \lambda k + \sum_{i=1}^m \max(v_i(\mu), w_i(\mu) - \lambda).$$

with $S := \mathbf{1}^\top (f^+ + f^-)$, and

$$v(\mu) := (f^+ + f^-) \circ \log\Big(\frac{f^+ + f^-}{\mu^+ + \mu^-}\Big), \;\; w(\mu) := f^+ \circ \log\Big(\frac{f^+}{\mu^+}\Big) + f^- \circ \log\Big(\frac{f^-}{\mu^-}\Big).$$

for given $\mu = (\mu^+, \mu^-) > 0$. This can be rewritten

$$\min_{\substack{\mu^+,\mu^->0 \\ \lambda \geq 0}} \; \max_{\substack{x+z=1 \\ x,z \geq 0}} \; \mu^+ + \mu^- - S + \lambda(k - \mathbf{1}^\top x) + z^\top v(\mu) + x^\top w(\mu)$$

using additional variables $x, z \in \mathbb{R}^n$, or again

$$\begin{array}{lll} \min\limits_{\substack{\mu^+,\mu^->0\\ \lambda\geq 0}} & \max\limits_{\substack{x+z=1\\ x,z\geq 0}} & \lambda(k-\mathbf{1}^\top x)-(x+z)^\top g-(z^\top g)\log(\mu^++\mu^-)+z^\top(g\circ\log(g)) \\ & & -(x^\top f^+)\log(\mu^+)-(x^\top f^-)\log(\mu^-) \\ & & +x^\top(f^+\circ\log(f^+)+f^-\circ\log(f^-))+\mu^++\mu^- \end{array} \tag{5.33}$$

calling $g = f^+ + f^-$. Strong duality holds in this min max problem so we can switch the min and the max. Writing $\mu_\pm = t\,p_\pm$, where $t = \mu_+ + \mu_-$ and $p_\pm > 0$, $p^+ + p^- = 1$ the Lagrangian becomes

$$\begin{array}{rcl} L(p_+,p_-,t,\lambda,x,z,\alpha) & = & \mathbf{1}^\top\nu - z^\top\nu - x^\top\nu + \lambda k - \lambda\mathbf{1}^\top x - \mathbf{1}^\top g - (z^\top g)\log(t) \\ & & -(x^\top f^+)\log(t\,p_+) - (x^\top f^-)\log(t\,p_-) + t \\ & & +z^\top(g\circ\log(g)) + x^\top(f^+\circ\log(f^+) + f^-\circ\log(f^-)) \\ & & +\alpha(p_+ + p_- - 1), \end{array}$$

where α is the dual variable associated with the constraint $p_+ + p_- = 1$. The dual of problem (USMNB) is then written

$$\sup_{\{x\geq 0, z\geq 0, \alpha\}} \ \inf_{\substack{p_+\geq 0, p_-\geq 0,\\ t\geq 0, \lambda\geq 0}} L(p_+,p_-,t,\mu^-,\lambda,x,z,\alpha)$$

The inner infimum will be $-\infty$ unless $\mathbf{1}^\top x \leq k$, so the dual becomes

$$\begin{array}{lll} \sup\limits_{\substack{x+z=\mathbf{1},\mathbf{1}^\top x\leq k,\\ x\geq 0, z\geq 0,\alpha}} & \inf\limits_{\substack{p_+\geq 0, p_-\geq 0,\\ t\geq 0}} & z^\top(g\circ\log(g)) + x^\top(f^+\circ\log(f^+) + f^-\circ\log(f^-)) \\ & & -(x^\top f^+)(\log t + \log(p_+)) - (x^\top f^-)(\log t + \log(p_-)) \\ & & +t - \mathbf{1}^\top g - (z^\top g)\log(t) + \alpha(p_+ + p_- - 1) \end{array}$$

and the first order optimality conditions in t, p_+, p_- yield

$$\begin{array}{rcl} t & = & \mathbf{1}^\top g \\ p_+ & = & (x^\top f^+)/\alpha \\ p_- & = & (x^\top f^-)/\alpha \end{array} \tag{5.34}$$

which means the above problem reduces to

$$\begin{array}{ll} \sup\limits_{\substack{x+z=\mathbf{1},\mathbf{1}^\top x\leq k,\\ x\geq 0, z\geq 0,\alpha}} & z^\top(g\circ\log(g)) + x^\top(f^+\circ\log(f^+) + f^-\circ\log(f^-)) \\ & -(\mathbf{1}^\top g)\log(\mathbf{1}^\top g) - (x^\top f^+)\log(x^\top f^+) - (x^\top f^-)\log(x^\top f^-) \\ & +(x^\top g)\log\alpha - \alpha \end{array}$$

and setting in $\alpha = x^\top g$ leads to the dual in (D). □

We now use this last result to better characterize scenarios where the bound produced by problem (USMNB) is tight and recovers an optimal solution to problem (SMNB).

Proposition 10. *Given $k > 0$, let $\phi(k)$ be the optimal value of* (SMNB)*. Given an optimal solution (x, z) of problem* (D)*, let $J = \{i : x_i \notin \{0, 1\}\}$ be the set of indices where x_i, z_i are not binary in $\{0, 1\}$. There is a feasible point $\bar{\theta}, \bar{\theta}^+, \bar{\theta}^-$ of problem* (SMNB) *for $\bar{k} = k + |J|$, with objective value OPT such that*

$$\phi(k) \leq OPT \leq \phi(k + |J|).$$

Proof. Using the fact that

$$\max_x \; a \log(x) - bx = a \log\left(\frac{a}{b}\right) - a$$

the max min problem in (5.33) can be rewritten as

$$\begin{array}{lll} \max\limits_{\substack{x+z=1\\ x,z\geq 0}} & \min\limits_{\substack{\mu^+,\mu^->0\\ \lambda\geq 0}} & \max\limits_{\theta,\theta^+,\theta^-} \quad \lambda(k - \mathbf{1}^\top x) + z^\top(g \circ \log\theta) \\ & & \quad +x^\top(f^+ \circ \log\theta^+) + x^\top(f^- \circ \log\theta^-) \\ & & \quad +\mu^+(1 - z^\top\theta - x^\top\theta^+) + \mu^-(1 - z^\top\theta - x^\top\theta^-) \end{array} \tag{5.35}$$

in the additional variables $\theta, \theta^+, \theta^- \in \mathbb{R}^n$, with (5.23) showing that

$$\theta_i = \frac{(f_i^+ + f_i^-)}{\mu^+ + \mu^-}, \quad \theta_i^+ = \frac{f_i^+}{\mu^+}, \quad \theta_i^- = \frac{f_i^-}{\mu^-}.$$

at the optimum. Strong duality holds in the inner min max, which means we can also rewrite problem (D) as

$$\max_{\substack{x+z=1\\ x,z\geq 0}} \; \max_{\substack{z^\top\theta + x^\top\theta^+ \leq 1\\ z^\top\theta + x^\top\theta^- \leq 1\\ x^\top\mathbf{1}\leq k}} \; z^\top(g \circ \log\theta) + x^\top(f^+ \circ \log\theta^+ + f^- \circ \log\theta^-) \tag{5.36}$$

or again, in epigraph form

$$\begin{array}{ll} \text{max.} & r \\ \text{s.t.} & \begin{pmatrix} r \\ 1 \\ 1 \\ k \end{pmatrix} \in \begin{pmatrix} 0 \\ \mathbb{R}_+ \\ \mathbb{R}_+ \\ \mathbb{R}_+ \end{pmatrix} + \sum_{i=1}^n \left\{ z_i \begin{pmatrix} g_i \log\theta_i \\ \theta_i \\ \theta_i \\ 0 \end{pmatrix} + x_i \begin{pmatrix} f_i^+ \log\theta_i^+ + f_i^- \log\theta_i^- \\ \theta_i^+ \\ \theta_i^- \\ 1 \end{pmatrix} \right\} \end{array} \tag{5.37}$$

Suppose the optimal solutions $x^\star, z^\star$ of problem (D) are binary in $\{0, 1\}^n$ and let $\mathcal{I} = \{i : z_i = 0\}$, then problem (hence problem (D)) reads

$$\begin{aligned} (\theta^{+\text{sub}}, \theta^{-\text{sub}}) = \arg \max_{\theta^+,\theta^-\in[0,1]^m} \; & f^{+\top}\log\theta^+ + f^{-\top}\log\theta^- \\ \text{s.t.} \; & \mathbf{1}^\top\theta^+ = \mathbf{1}^\top\theta^- = 1, \\ & \theta_i^+ = \theta_i^-, \;\; i \in \mathcal{I}. \end{aligned} \tag{5.38}$$

which is exactly (5.38). This means that the optimal values of problem (5.38) and (D) are equal, so that the relaxation is tight and $\theta_i^+ = \theta_i^-$ for $i \in \mathcal{I}$. Suppose now that some coefficients x_i are not binary. Let us call J the set $J = \{i : x_i \notin \{0, 1\}\}$. As in [29, Th. I.3], we define new solutions $\bar\theta, \bar\theta^+, \bar\theta^-$ and $\bar x, \bar z$ as follows,

$$\begin{cases} \bar\theta_i = \theta_i,\ \bar\theta_i^+ = \theta_i^+,\ \bar\theta_i^- = \theta_i^- \text{ and } \bar z_i = z_i,\ \bar x_i = x_i & \text{if } i \notin J \\ \bar\theta_i = 0,\ \bar\theta_i^+ = z_i\theta + x_i\theta_i^+,\ \bar\theta_i^- = z_i\theta + x_i\theta_i^- \text{ and } \bar z_i = 0,\ \bar x_i = 1 & \text{if } i \in J \end{cases}$$

By construction, the points $\bar\theta, \bar\theta^+, \bar\theta^-$ and $\bar z, \bar x$ satisfy the constraints $\bar z^\top\bar\theta + \bar x^\top\bar\theta^+ \le 1$, $\bar z^\top\bar\theta + \bar x^\top\bar\theta^- \le 1$ and $\bar x^\top\mathbf{1} \le k$. We also have $\bar x^\top \le k + |J|$ and

$$\begin{aligned} & z^\top((f^+ + f^-) \circ \log\theta) + x^\top(f^+ \circ \log\theta^+ + f^- \circ \log\theta^-) \\ \le\ & \bar z^\top((f^+ + f^-) \circ \log\bar\theta) + \bar x^\top(f^+ \circ \log\bar\theta^+ + f^- \circ \log\bar\theta^-) \end{aligned}$$

by concavity of the objective, hence the last inequality. □

We will now use the Shapley-Folkman theorem to bound the number of nonbinary coefficients in Proposition 9 and construct a solution to (D) satisfying the bound in Theorem 5.

Proposition 11. *There is a solution to problem* (D) *with at most four nonbinary pairs* (x_i, z_i).

Proof. Suppose $(x^\star, z^\star, r^\star)$ and $(\theta, \theta_i^+, \theta_i^-)$ solve problem (D) written as in (5.7), we get

$$\begin{pmatrix} r^\star \\ 1 - s_1 \\ 1 - s_2 \\ k - s_3 \end{pmatrix} = \sum_{i=1}^n \left\{ z_i \begin{pmatrix} g_i \log\theta_i \\ \theta_i \\ \theta_i \\ 0 \end{pmatrix} + x_i \begin{pmatrix} f_i^+ \log\theta_i^+ + f_i^- \log\theta_i^- \\ \theta_i^+ \\ \theta_i^- \\ 1 \end{pmatrix} \right\} \tag{5.39}$$

where $s_1, s_2, s_3 \ge 0$. This means that the point $(r^\star, 1 - s_1, 1 - s_1, k - s_3)$ belongs to a Minkowski sum of segments, with

$$\begin{pmatrix} r^\star \\ 1 - s_1 \\ 1 - s_2 \\ k - s_3 \end{pmatrix} \in \sum_{i=1}^n \mathbf{Co}\left(\left\{ \begin{pmatrix} g_i \log\theta_i \\ \theta_i \\ \theta_i \\ 0 \end{pmatrix}, \begin{pmatrix} f_i^+ \log\theta_i^+ + f_i^- \log\theta_i^- \\ \theta_i^+ \\ \theta_i^- \\ 1 \end{pmatrix} \right\} \right) \tag{5.40}$$

The Shapley-Folkman theorem [93] then shows that

$$\begin{aligned} \begin{pmatrix} r^\star \\ 1 - s_1 \\ 1 - s_2 \\ k - s_3 \end{pmatrix} \in & \sum_{[1,n]\setminus\mathcal{S}} \left\{ \begin{pmatrix} g_i \log\theta_i \\ \theta_i \\ \theta_i \\ 0 \end{pmatrix}, \begin{pmatrix} f_i^+ \log\theta_i^+ + f_i^- \log\theta_i^- \\ \theta_i^+ \\ \theta_i^- \\ 1 \end{pmatrix} \right\} \\ & + \sum_{\mathcal{S}} \mathbf{Co}\left(\left\{ \begin{pmatrix} g_i \log\theta_i \\ \theta_i \\ \theta_i \\ 0 \end{pmatrix}, \begin{pmatrix} f_i^+ \log\theta_i^+ + f_i^- \log\theta_i^- \\ \theta_i^+ \\ \theta_i^- \\ 1 \end{pmatrix} \right\} \right) \end{aligned}$$

where $|\mathcal{S}| \leq 4$, which means that there exists a solution to (D) with at most four nonbinary pairs (x_i, z_i) with indices $i \in \mathcal{S}$. □

In our case, since the Minkowski sum in (5.40) is a polytope (as a Minkowski sum of segments), the Shapley-Folkman result reduces to a direct application of the fundamental theorem of linear programming, which allows us to reconstruct the solution of Proposition 11 by solving a linear program.

Proposition 12. *Given $(x^\star, z^\star, r^\star)$ and $(\theta, \theta_i^+, \theta_i^-)$ solving problem (D), we can reconstruct a solution (x, z) solving problem (9), such that at most four pairs (x_i, z_i) are nonbinary, by solving*

$$\begin{array}{ll} \textit{min.} & c^\top x \\ \textit{s.t.} & \sum_{i=1}^n (1-x_i) g_i \log \theta_i + x_i (f_i^+ \log \theta_i^+ + f_i^- \log \theta_i^-) = r^\star \\ & \sum_{i=1}^n (1-x_i)\theta_i + x_i \theta_i^+ \leq 1 \\ & \sum_{i=1}^n (1-x_i)\theta_i + x_i \theta_i^- \leq 1 \\ & \sum_{i=1}^n x_i \leq k \\ & 0 \leq x \leq 1 \end{array} \tag{5.41}$$

which is a linear program in the variable $x \in \mathbb{R}^n$ where $c \in \mathbb{R}^n$ is e.g. a i.i.d. Gaussian vector.

Proof. Given $(x^\star, z^\star, r^\star)$ and $(\theta, \theta_i^+, \theta_i^-)$ solving problem (D), we can reconstruct a solution (x, z) solving problem (9), by solving (5.41) which is a linear program in the variable $x \in \mathbb{R}^n$ where $c \in \mathbb{R}^n$ is e.g. a i.i.d. Gaussian vector. This program has $2n + 4$ constraints, at least n of which will be saturated at the optimum. In particular, at least $n - 4$ constraints in $0 \leq x \leq 1$ will be saturated so at least $n - 4$ coefficients x_i will be binary at the optimum, idem for the corresponding coefficients $z_i = 1 - x_i$. □

Proposition 12 shows that solving the linear program in (5.41) as a postprocessing step will produce a solution to problem (D) with at most $n-4$ nonbinary coefficient pairs (x_i, z_i). Proposition 10 then shows that this solution satisfies

$$\phi(k) \leq OPT \leq \phi(k+4).$$

which is the bound in Theorem (5).

Finally, we show a technical lemma linking the dual solution (x, z) in (D) above and the support of the k largest coefficients in the computation of $s_k(h(\alpha))$ in theorem 4.

Lemma 6. *Given $c \in \mathbb{R}_+^n$, we have*

$$s_k(c) = \min_{\lambda \geq 0} \ \lambda k + \sum_{i=1}^n \max(0, c_i - \lambda) \tag{5.42}$$

and given k, $\lambda \in [c_{[k+1]}, c_{[k]}]$ at the optimum, where $c_{[1]} \geq \ldots \geq c_{[n]}$. Its dual is written

$$\begin{array}{ll} \textit{max.} & x^\top c \\ \textit{s.t.} & \mathbf{1}^\top x \leq k \\ & x + z = 1 \\ & 0 \leq z, x \end{array} \tag{5.43}$$

When all coefficients c_i are distinct, the optimum solutions x, z of the dual have at most one nonbinary coefficient each, i.e. $x_i, z_i \in (0, 1)$ for a single $i \in [1, n]$. If in addition $c_{[k]} > 0$, the solution to (5.43) *is binary.*

Proof. Problem (5.42) can be written

$$\begin{array}{ll} \text{min.} & \lambda k + \mathbf{1}^\top t \\ \text{s.t.} & c - \lambda \mathbf{1} \leq t \\ & 0 \leq t \end{array}$$

and its Lagrangian is then

$$L(\lambda, t, z, x) = \lambda k + \mathbf{1}^\top t + x^\top (c - \lambda \mathbf{1} - t) + z^\top t.$$

The dual to the minimization problem (5.42) reads

$$\begin{array}{ll} \text{max.} & x^\top c \\ \text{s.t.} & \mathbf{1}^\top x \leq k \\ & x + z = 1 \\ & 0 \leq z, x \end{array}$$

in the variable $w \in \mathbb{R}^n$, its optimum value is $s_k(z)$. By construction, given $\lambda \in [c_{[k+1]}, c_{[k]}]$, only the k largest terms in $\sum_{i=1}^m \max(0, c_i - \lambda)$ are nonzero, and they sum to $s_k(c) - k\lambda$. The KKT optimality conditions impose

$$x_i(c_i - \lambda - t_i) = 0 \quad \text{and} \quad z_i t_i = 0, \quad i = 1, \ldots, n$$

at the optimum. This, together with $x + z = 1$ and $t, x, z \geq 0$, means in particular that

$$\begin{cases} x_i = 0, z_i = 1, & \text{if } c_i - \lambda < 0 \\ x_i = 0, z_i = 1, \text{ or } x_i = 1, z_i = 0 & \text{if } c_i - \lambda > 0 \end{cases} \tag{5.44}$$

the result of the second line comes from the fact that if $c_i - \lambda > 0$ and $t_i = c_i - \lambda$ then $z_i = 0$ hence $x_i = 1$, if on the other hand $t_i \neq c_i - \lambda$, then $x_i = 0$ hence $z_i = 1$. When the coefficients c_i are all distinct, $c_i - \lambda = 0$ for at most a single index i and (5.44) yields the desired result. When $c_{[k]} > 0$ and the c_i are all distinct, then the only way to enforce zero gap, i.e.

$$x^\top c = s_k(c)$$

is to set the corresponding coefficients of x_i to one. □

5.8 Details on Datasets

This section details the data sets used in our experiments.

Downloading data sets.

1. AMZN The complete Amazon reviews data set was collected from here; only a subset of this data was used which can be found here. This data set was randomly split into 80/20 train/test.
2. IMDB The large movie review (or IMDB) data set was collected from here and was already split 50/50 into train/test.
3. TWTR The Twitter Sentiment140 data set was downloaded from here and was pre-processed according to the method highlighted here.
4. MPQA The MPQA opinion corpus can be found here and was pre-processed using the code found here.
5. SST2 The Stanford Sentiment Treebank data set was downloaded from here and the pre-processing code can be found here.

Creating feature vectors. After all data sets were downloaded and pre-processed, the diffeent types of feature vectors were constructed using `CounterVectorizer` and `TfidfVectorizer` from Sklearn [82]. Counter vector, tf-idf, and tf-idf word bigrams use the `analyzer = 'word'` specification while the tf-idf char bigrams use `analyzer = 'char'`.

Two-stage procedures. For experiments 2 and 3, all standard models were trained in Sklearn [82]. In particular, the following settings were used in stage 2 for each model

1. `LogisticRegression(penalty='l2', solver='lbfgs', C =1e4, max_iter=1e2)`
2. `LinearSVC(C = 1e4)`
3. `MultinomialNB(alpha=a)`

In the first stage of the two stage procedures, the following settings were used for each of the different feature selection methods

1. `LogisticRegression(random_state=0, C =` λ_1`,penalty='l1',solver='saga', max_iter=1e2)`
2. `clf = LogisticRegression(C = 1e4, penalty='l2', solver = 'lbfgs', max_iter = 1e2).fit(train_x,train_y)`
   `selector_log = RFE(clf,` k`), step=0.3)`
3. `Lasso(alpha =` λ_2`, selection='cyclic', tol = 1e-5)`

4. `LinearSVC(C =`λ_3`, penalty='l1',dual=False)`
5. `clf = LinearSVC(C = 1e4, penalty='l2',dual=False).fit(train_x,train_y)`
   `selector_svm = RFE(clf,`k`, step=0.3)`
6. `MultinomialNB(alpha=a)`

where λ_i are hyper-parameters used by the ℓ_1 methods to achieve a desired sparsity level k. a is a hyper-parameter for the different MNB models which we compute using cross validation (explained below).

Hyper-parameters. For each of the ℓ_1 methods we manually do a grid search over all hyper-parameters to achieve an approximate desired sparsity pattern. For determining the hyper-parameter for the MNB models, we employ 10-fold cross validation on each data set for each type of feature vector and determine the best value of a. In total, this is $16+20 = 36$ values of a – 16 for experiment 2 and 20 for experiment 3. In experiment 2, we do not use the twitter data set since computing the λ_i's to achieve a desired sparsity pattern for the ℓ_1 based feature selection methods was computationally intractable.

Experiment 2 and 3: full results. Here we show the results of experiments 2 and 3 for all the data sets. All error bars represents 10 separate simulations where each simulation is a different appropriately-sized train-test split (as per Table 5.1). As seen in Figure 5.1, the SVM-ℓ_1 model was unable to converge and hence has an accuracy of 50%. This was in spite of manually adjusting `max_iter=1e7` and using the liblinear solver which is default for LinearSVC in sci-kit learn.

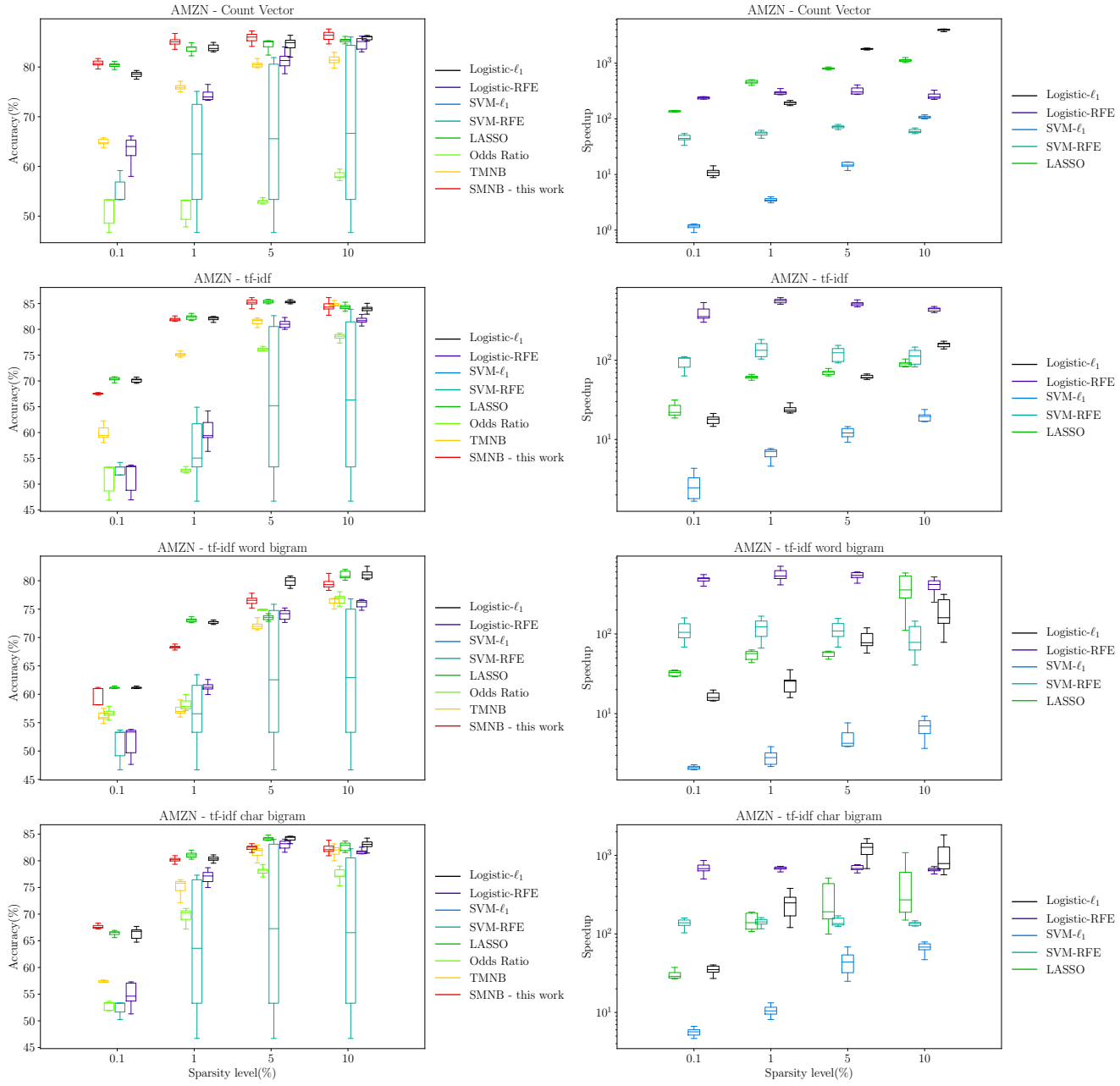

Figure 5.4: Experiment 2: AMZN - Stage 2 Logistic

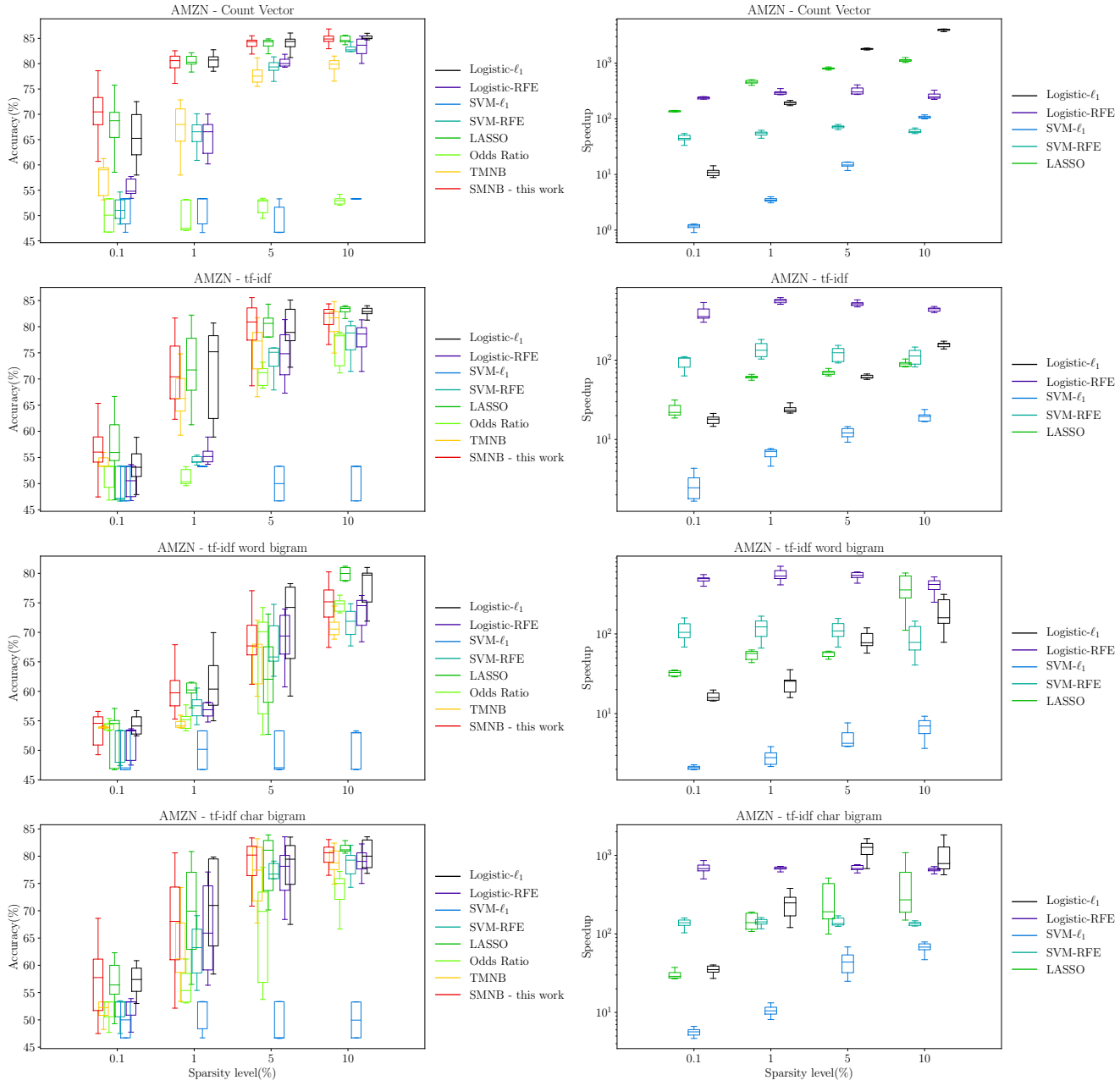

Figure 5.5: Experiment 2: AMZN - Stage 2 SVM

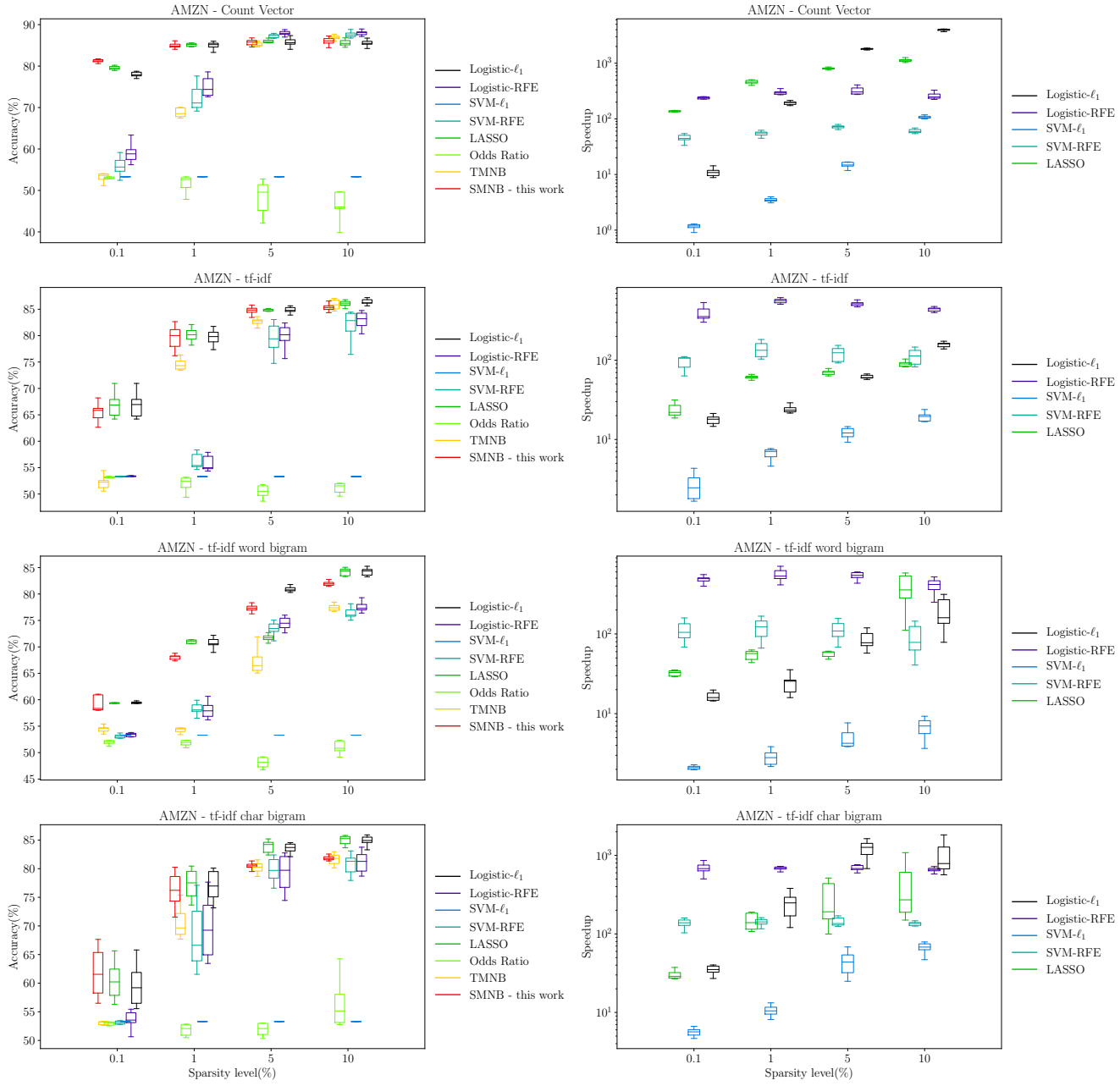

Figure 5.6: Experiment 2: AMZN - Stage 2 MNB

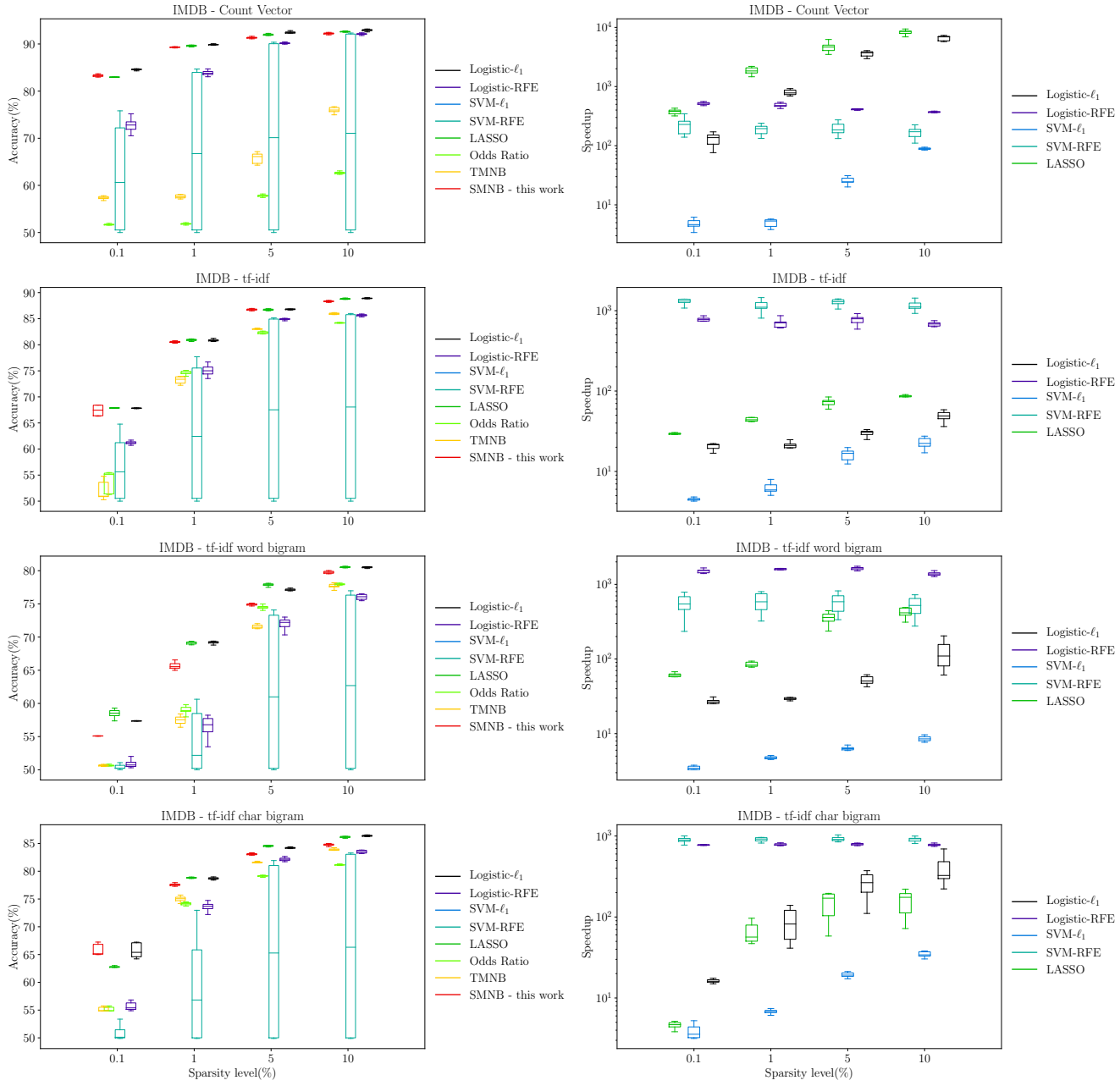

Figure 5.7: Experiment 2: IMDB - Stage 2 Logistic

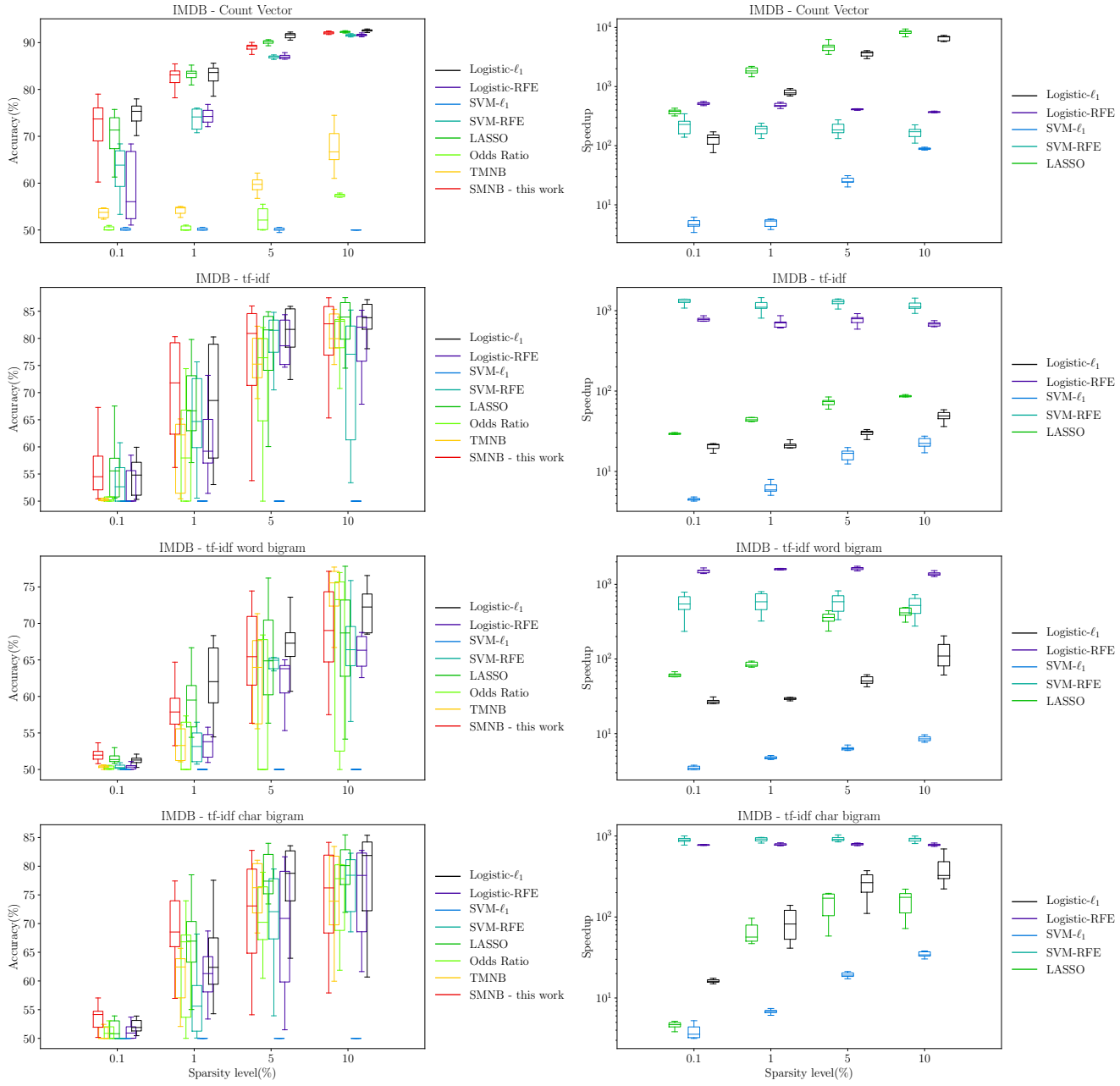

Figure 5.8: Experiment 2: IMDB - Stage 2 SVM

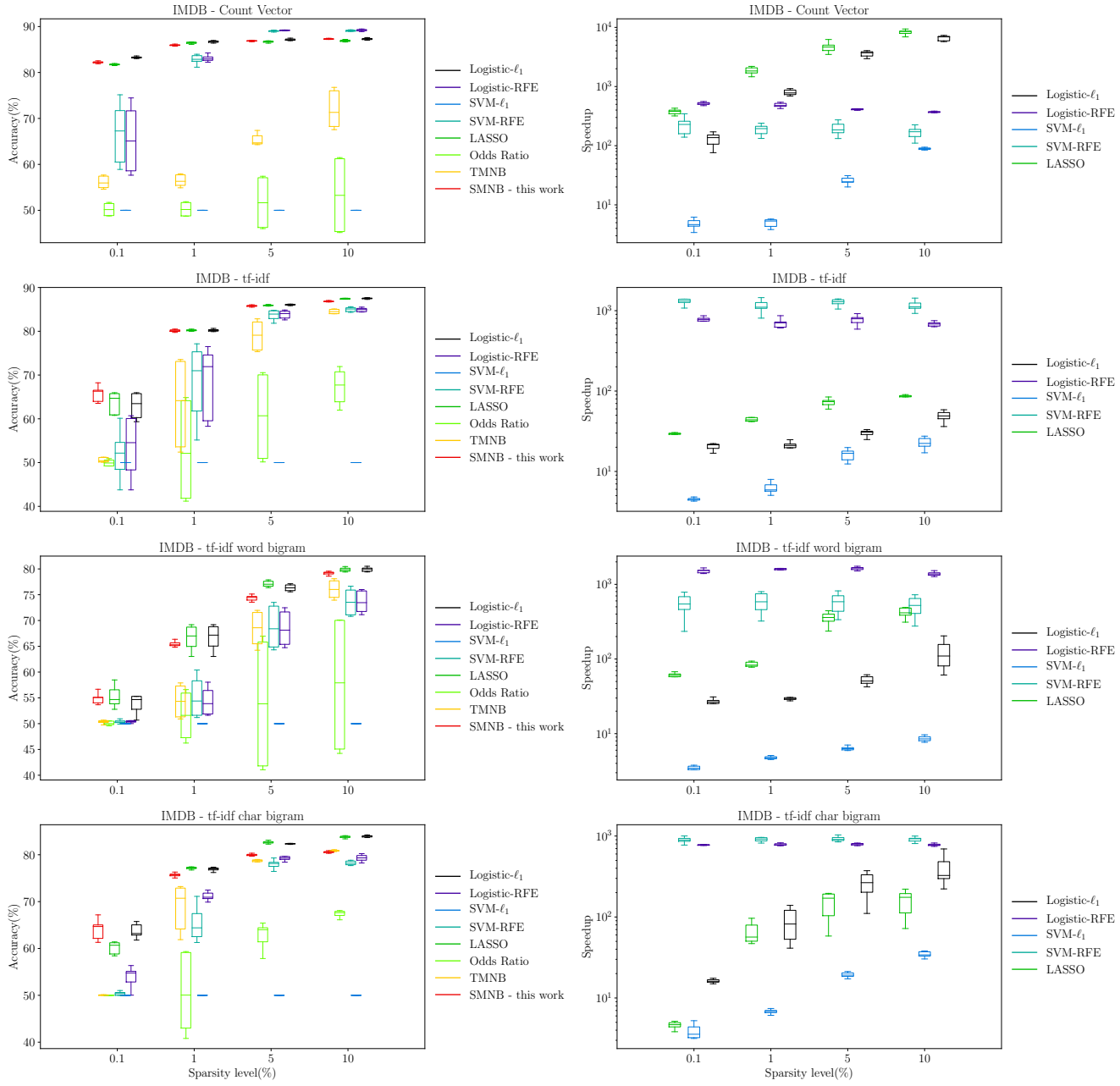

Figure 5.9: Experiment 2: IMDB - Stage 2 MNB

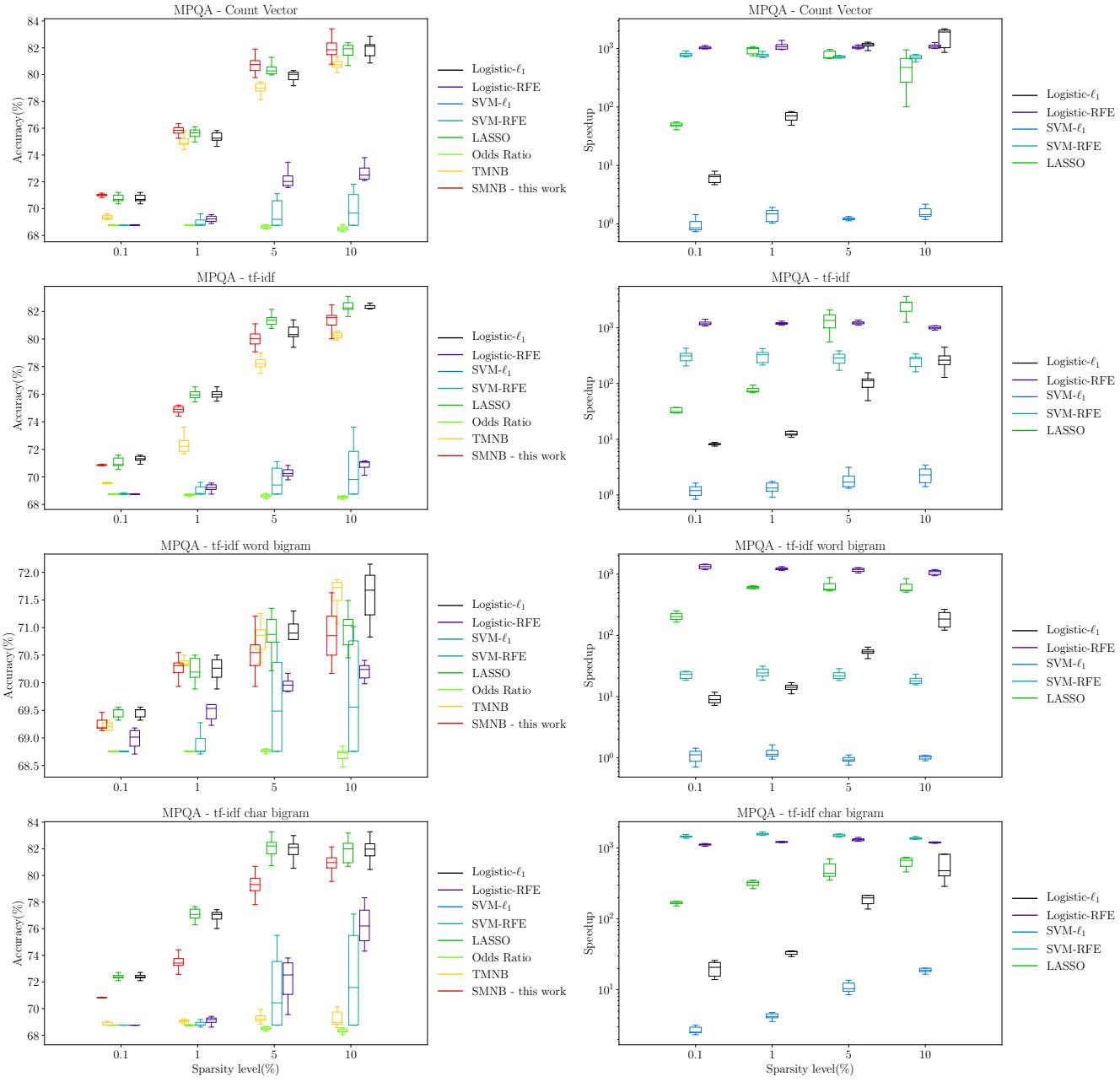

Figure 5.10: Experiment 2: MPQA - Stage 2 Logistic

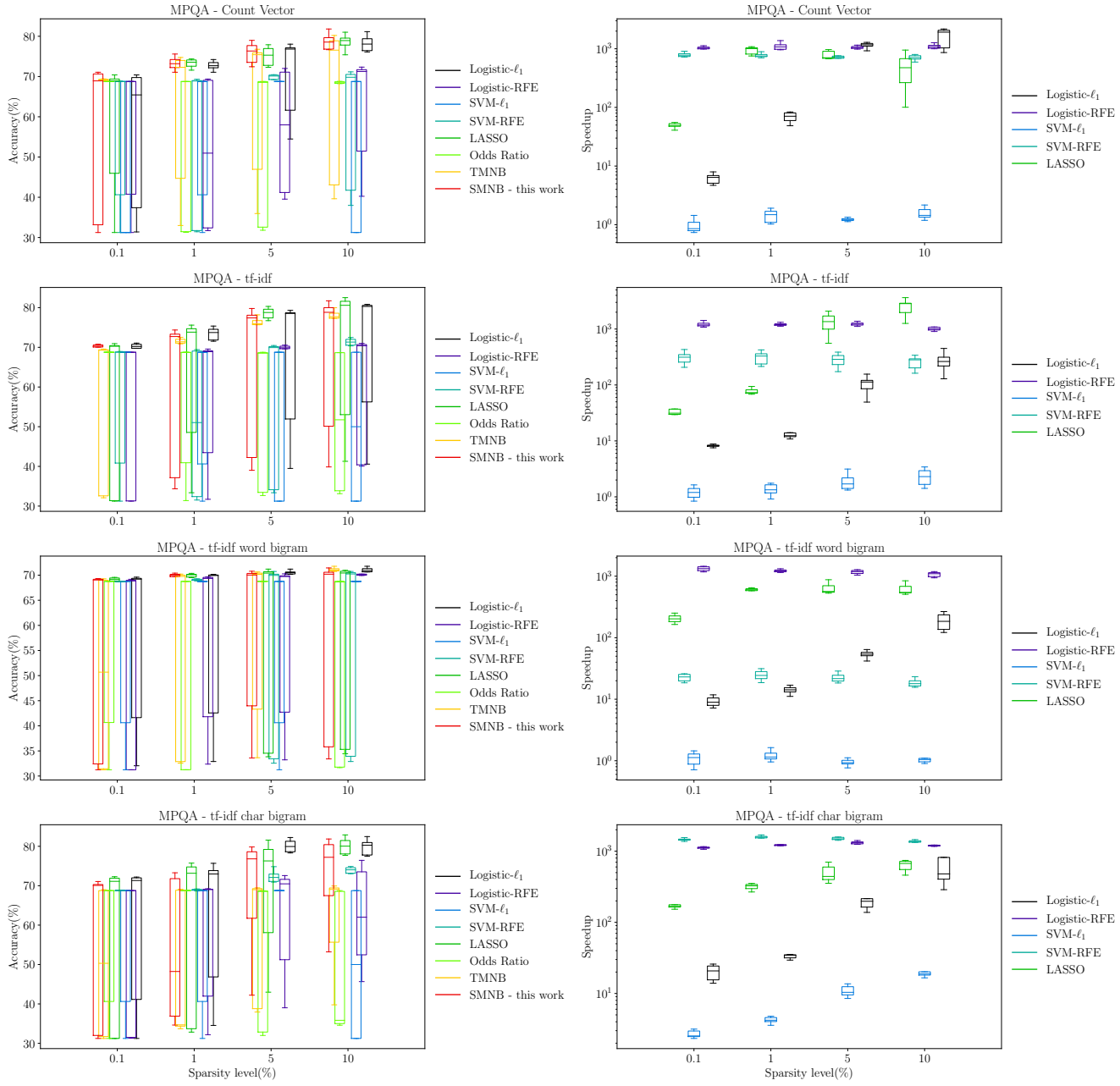

Figure 5.11: Experiment 2: MPQA - Stage 2 SVM

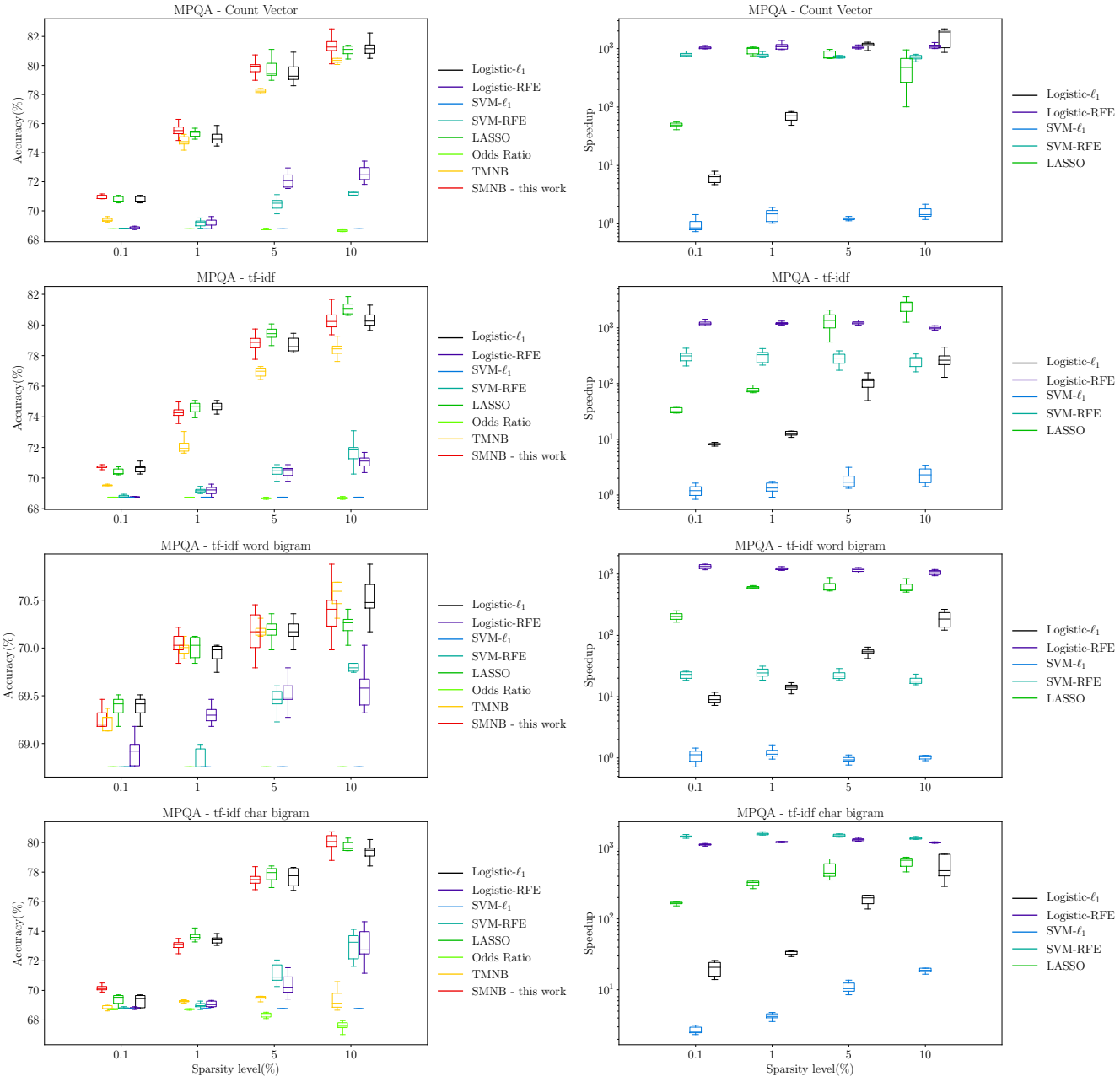

Figure 5.12: Experiment 2: MPQA - Stage 2 MNB

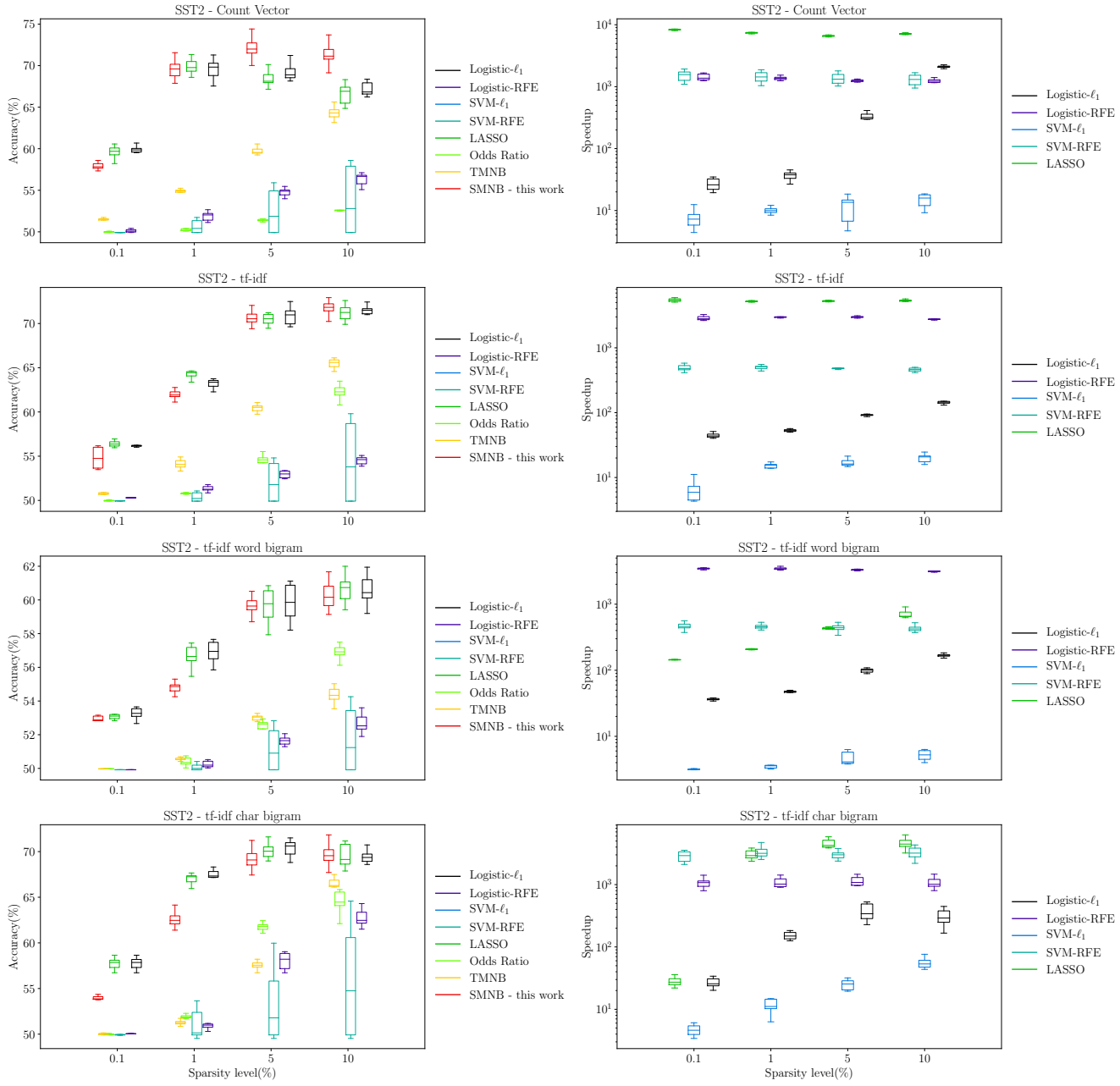

Figure 5.13: Experiment 2: SST2 - Stage 2 Logistic

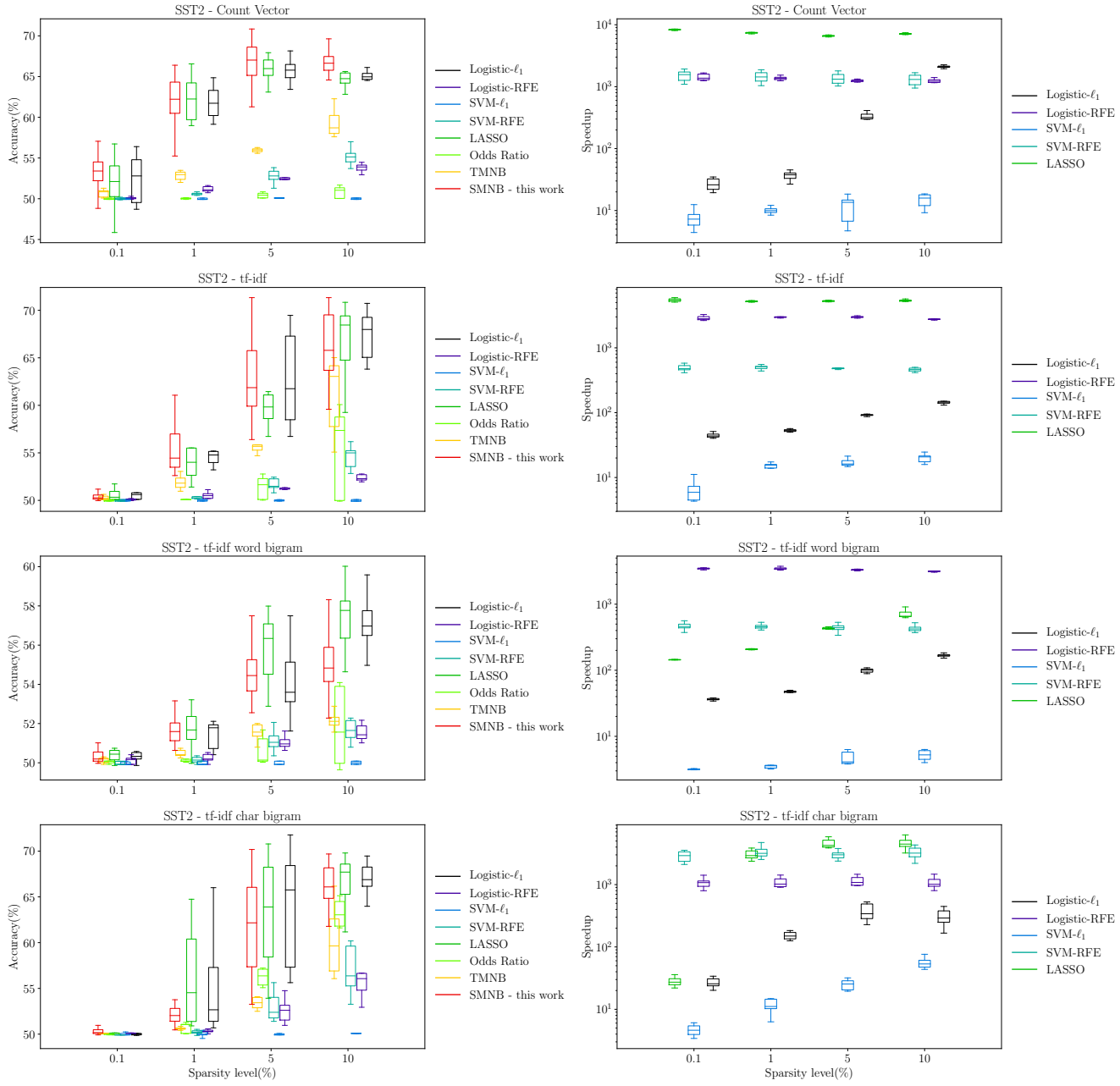

Figure 5.14: Experiment 2: SST2 - Stage 2 SVM

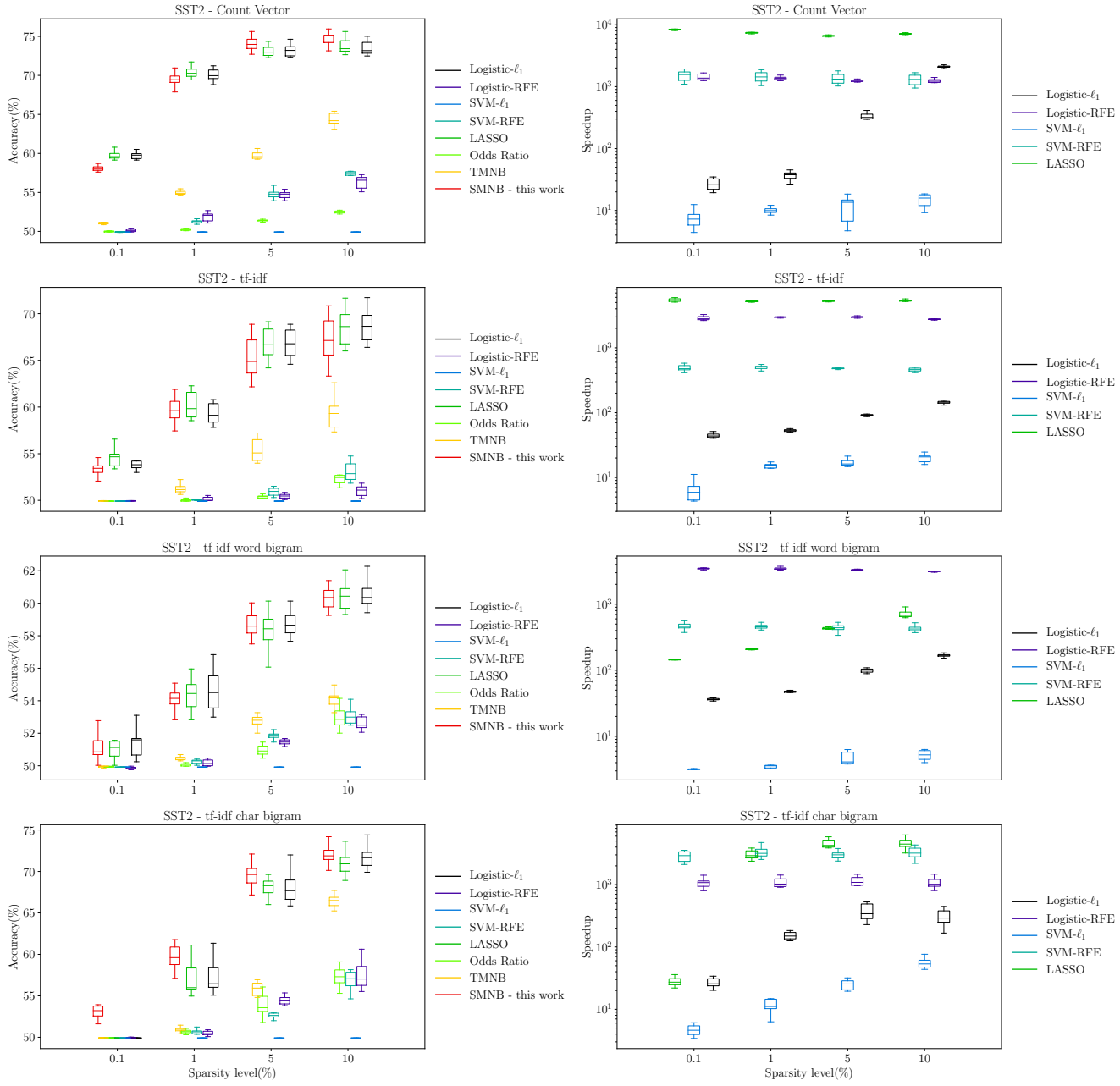

Figure 5.15: Experiment 2: SST2 - Stage 2 MNB

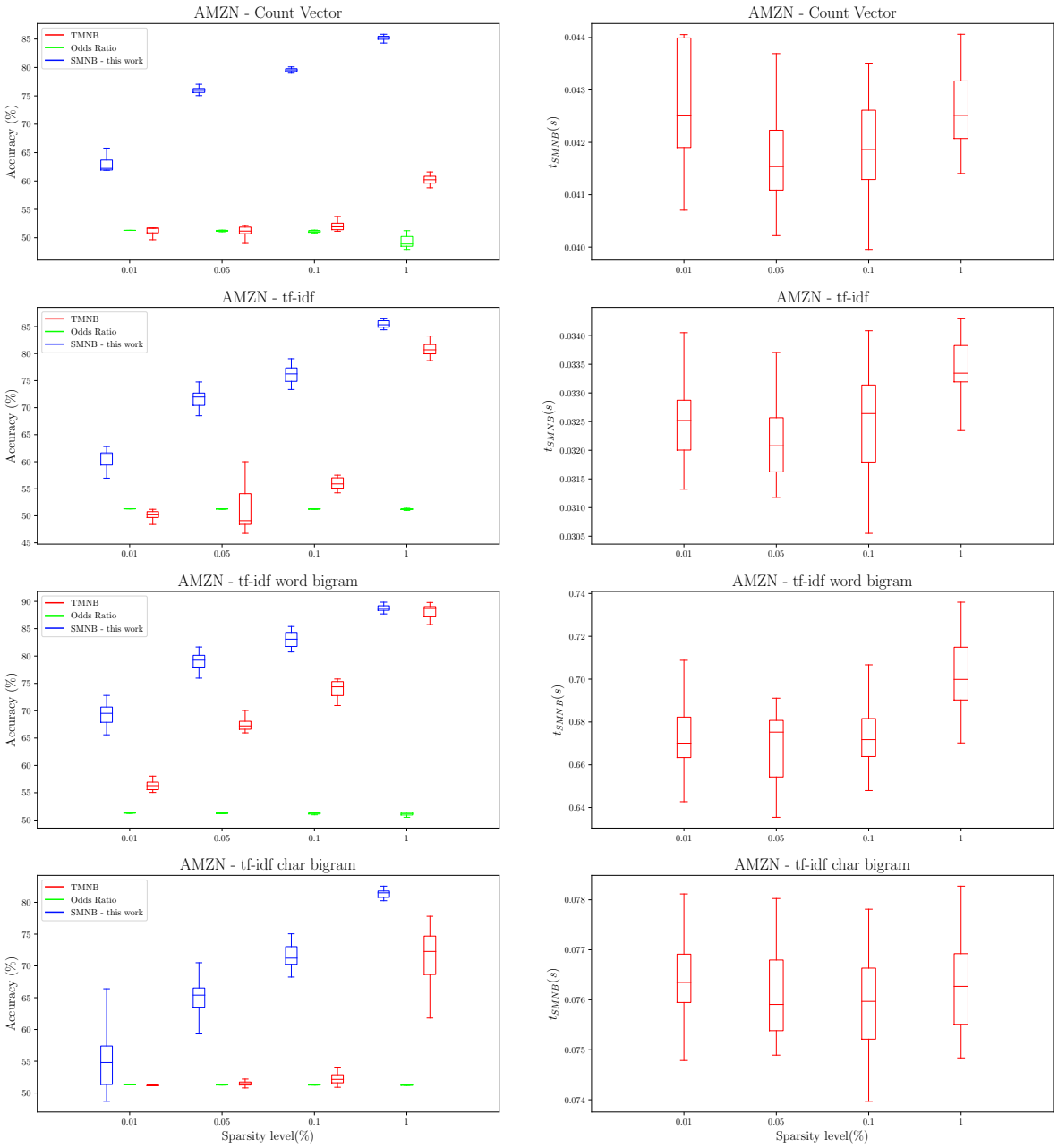

Figure 5.16: Experiment 3: AMZN - Stage 2 MNB

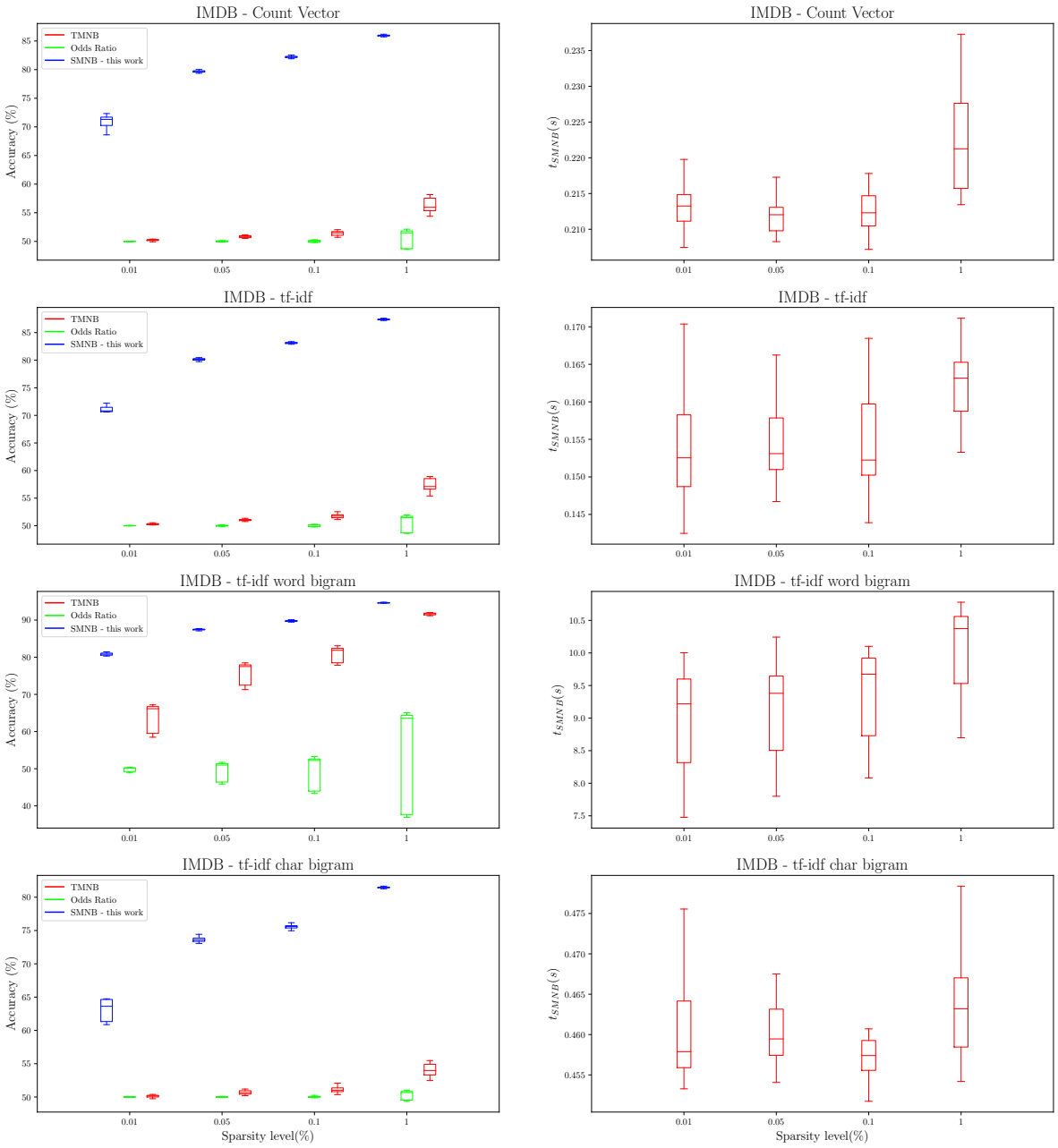

Figure 5.17: Experiment 3: IMDB - Stage 2 MNB

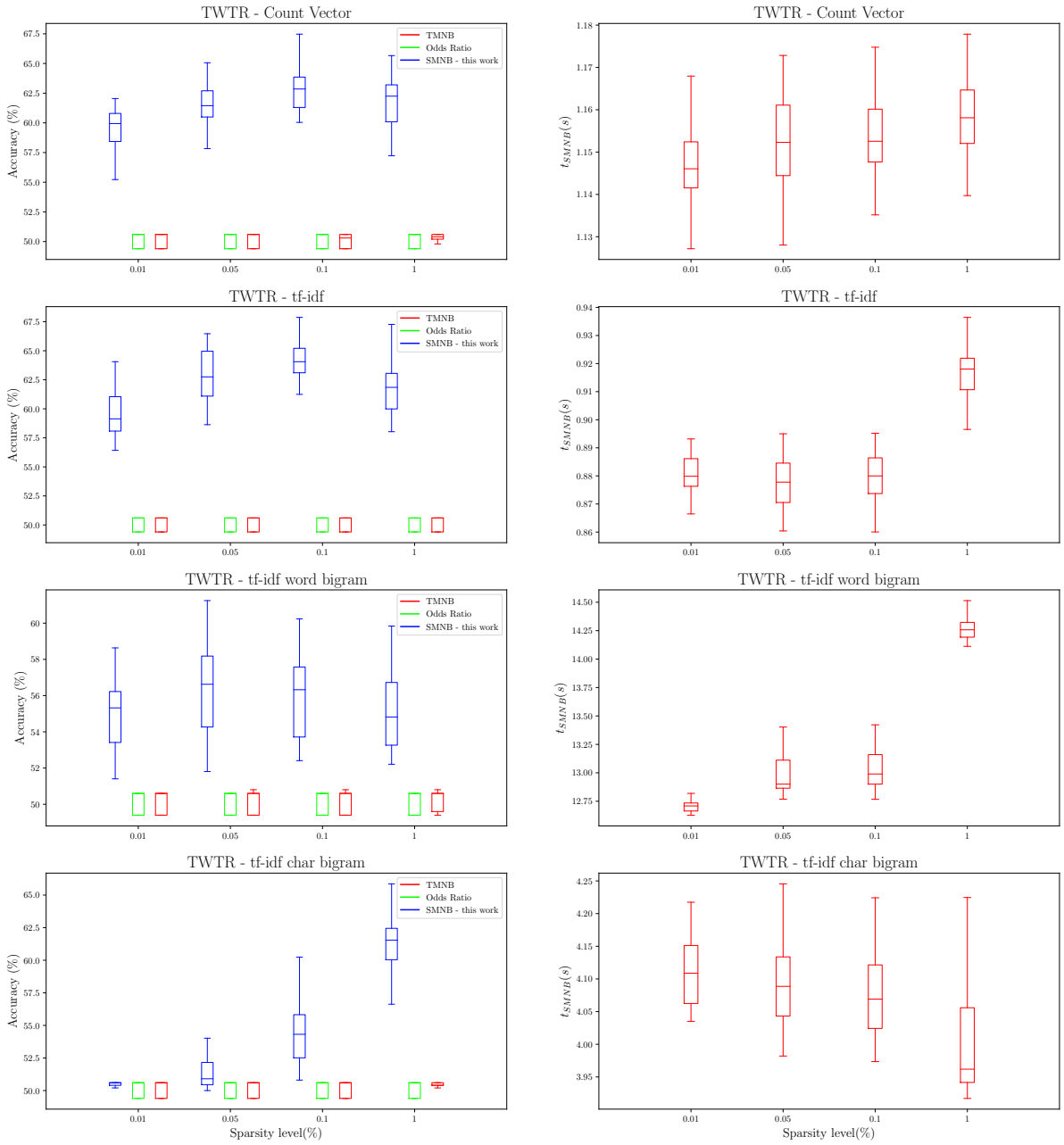

Figure 5.18: Experiment 3: TWTR - Stage 2 MNB

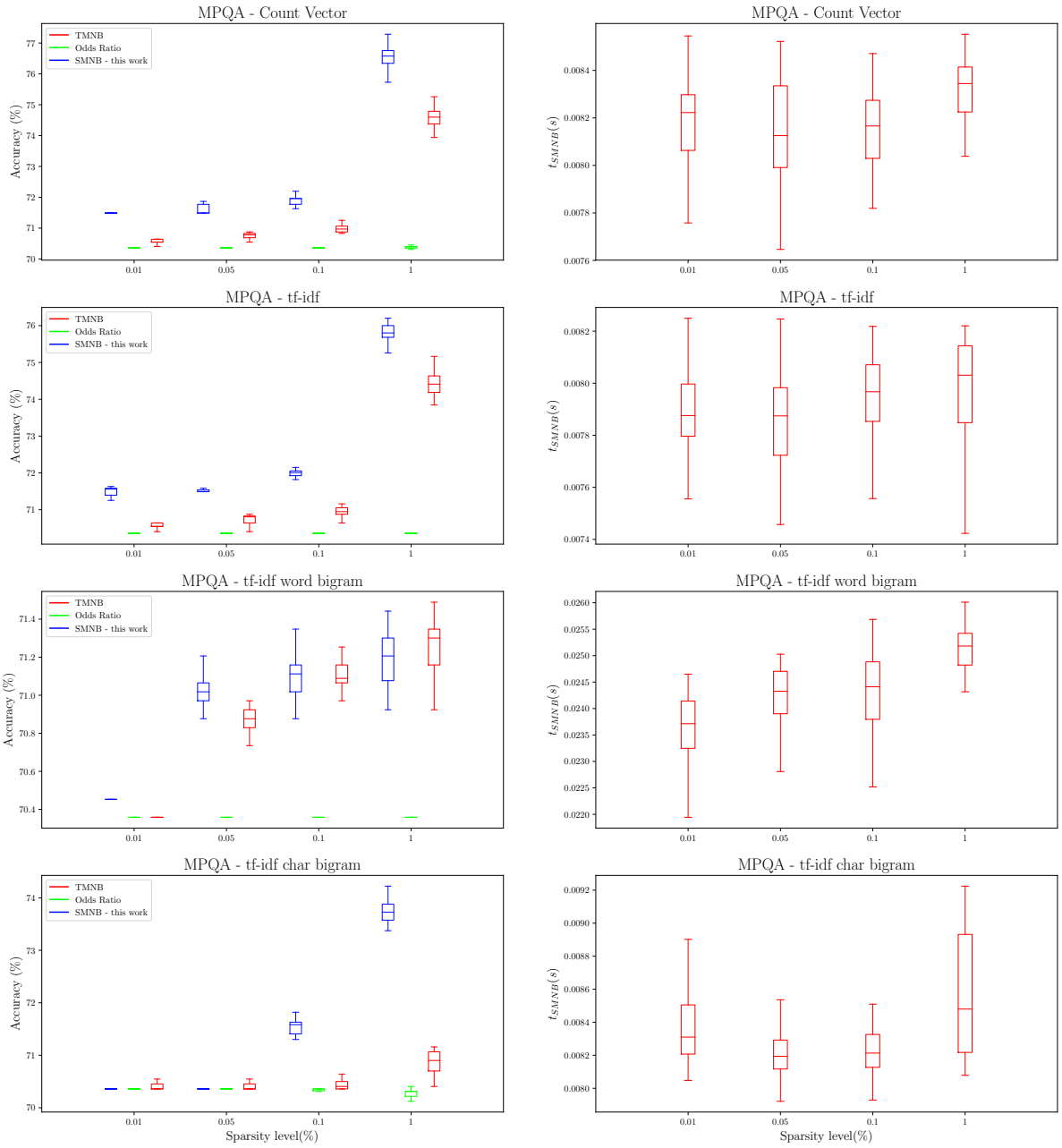

Figure 5.19: Experiment 3: MPQA - Stage 2 MNB

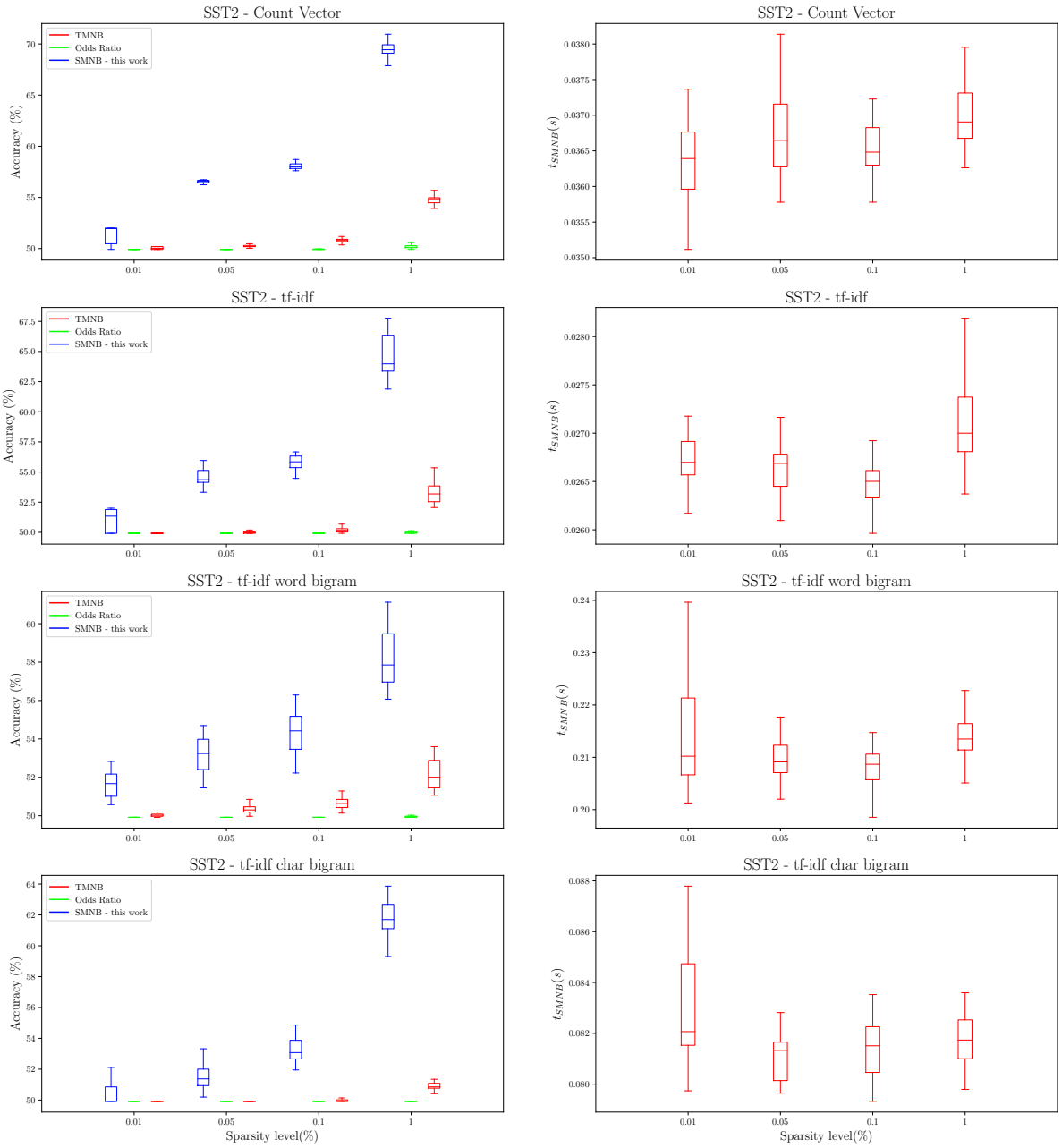

Figure 5.20: Experiment 3: SST2 - Stage 2 MNB

Chapter 6

Approximation Bounds for Sparse Programs

6.1 Introduction

We study optimization problems with low-rank data and sparsity constraints, written

$$p_{\text{con}}(k) \triangleq \min_{\|w\|_0 \leq k} \; f(Xw; y) + \frac{\gamma}{2}\|w\|_2^2, \tag{P-CON}$$

in the variable $w \in \mathbb{R}^m$, where $X \in \mathbb{R}^{n\times m}$ is assumed low-rank, $y \in \mathbb{R}^n, \gamma > 0$, and $k \geq 0$. Here, $\|\cdot\|_0$ stands for the l_0-"norm" (cardinality) of its vector argument. We also study a penalized formulation of this problem written

$$p_{\text{pen}}(\lambda) \triangleq \min_{w} \; f(Xw; y) + \frac{\gamma}{2}\|w\|_2^2 + \lambda\|w\|_0 \tag{P-PEN}$$

in the variable $w \in \mathbb{R}^m$, where $\lambda > 0$. We provide explicit upper and lower bounds on $p_{\text{con}}(k)$ and $p_{\text{pen}}(\lambda)$ that are a function of the bidual problem and the rank of X. We also provide a tractable procedure to compute primal feasible points w that satisfy the aforementioned bounds. We first begin with the case where $f(\cdot)$ is convex and show how to extend the results to the case when $f(\cdot)$ is non convex.

Related literature In a general setting, (P-CON) and (P-PEN) are NP-hard [74]. A very significant amount of research has been focused on producing tractable approximations and on proving recovery under certain conditions. This is the case in compressed sensing , for example, where work stemming from [27, 17] shows that ℓ_1-like penalties recover sparse solutions under various conditions enforcing independence among sparse subsets of variables of cardinality at most k.

The convex quadratic case (i.e., $f(Xw; y) = \|Xw - y\|_2^2 = w^\top Qw + 2y^\top w + y^\top y$ with $X^\top X = Q$) has been heavily studied. In [80], the authors relax (P-CON) to a non convex

quadratically constrained quadratic program for which they invoke the S-procedure to arrive at a convex problem; they also draw a connection between their semidefinite relaxation and a probabilistic interpretation to construct a simple randomized algorithm. In [85], the authors obtain a semidefinite programming (SDP) relaxation of the problem. They also consider the cardinality-penalized version of (P-CON) and use a convex relaxation that is connected with the reverse Huber penalty. In [91], the authors compute the biconjugate of the cardinality-penalized objective in one dimension and in the case when Q is an identity matrix and compare the minimum of their problem using a penalty term inspired from the derivation of the biconjugate. In [4, 104, 5], the authors take advantage ofthe explicit structure of Q (e.g., when Q is rank one) to arrive at tighter relaxations of (P-CON) by considering convex hulls of perspective relaxations of the problem. They additionally study the case when there is a quadratic penalty on consecutive observations for smoothness considerations. In [109], the authors show the equivalence between many of the formulations derived in the above papers and provide scalable algorithms for solving the convex relaxations of (P-CON). In [38], the authors take a different approach by looking at the Lagrangian dual of the problem and decoupling the ellipsoidal level sets by considering separable outer approximations of the quadratic program defining the portfolio selection problem. The non convex quadratic case has also been studied. Namely, it is a well-known fact that a quadratic optimization with one quadratic constraint has zero duality gap and can be solved exactly via SDP even when the quadratic forms are non convex (see, e.g., [14, Appendix B]).

The Shapley Folkman theorem, used to construct our bounds, was derived by Shapley and Folkman and first published in [93]. In [6], the authors used the theorem to derive a priori bounds on the duality gap in separable optimization problems and showcased applications such as the unit commitment problem. Extreme points of the set of solutions of a convex relaxation are then used to produce good approximations, and [101] describes a randomized purification procedure to find such points with probability one.

Contributions While the works listed above do produce tractable relaxations of problems (P-CON) and (P-PEN), they do not yield a priori guarantees on the quality of these solutions (outside of the sparse recovery results mentioned above) and do not handle the generic low-rank case. Our bounds are expressed in terms of the value of the bidual, the desired sparsity level, and the rank of X, which is often low in practice.

Here, we use the Shapley Folkman theorem to produce a priori bounds on the duality gap of problems (P-CON) and (P-PEN). Our convex relaxations, which are essentially interval relaxations of a discrete reformulation of the sparsity constraint and penalty, produce both upper and lower approximation bounds on the optima of problems (P-CON) and (P-PEN). These relaxations come with primalization procedures, that is, tractable schemes to construct feasible points satisfying these approximation bounds. Furthermore, these error bounds are proportional to the rate of growth of the objective with the target cardinality k, which means, in feature selection problems, for instance, that the relaxations are nearly tight as soon as k is large enough so that only uninformative features are added.

Notation and preliminaries

For a vector $u \in \mathbb{R}^m$, let $D(u) = \mathbf{diag}(u_1, \ldots, u_m)$. Let $M^\dagger$ denote the pseudoinverse of the matrix M. For a closed function $f(x)$, let $f^*(y) \triangleq \max_x x^\top y - f(x)$ denote the Fenchel conjugate, and let $f^{**}(x)$ be the biconjguate (the conjugate of $f^*(x)$). Throughout the paper, we will assume f is closed. If we additionally assume f is convex, then $f^{**} = f$ (see, e.g., [46, Proposition. 6.1.1]). For simplicity, we will drop the explicit dependence of y in our objective and simply write $f(Xw)$ instead.

Our results fundamentally rely on the Shapely Folkman theorem [93], which we now state (without proof) for completeness. Let $Q_n,\ n = 1, \ldots, N$, be non empty (possibly non convex) subsets of $\mathbb{R}^p$. Let $x \in \mathbf{Co}\Big(\sum_{n=1}^N Q_n\Big)$, where $\mathbf{Co}(\cdot)$ denotes the convex hull of a set, and here the sum over sets is understood as the Minkowski sum. In words, the Shapley Folkman theorem states that x can be decomposed as the sum of some vectors belonging in $\mathbf{Co}(Q_n)$ and the other vectors belonging to Q_n. Mathematically, the theorem states that there exists an index set $S \subseteq \{1, \ldots, N\}$ with $|S| \leq p$ such that

$$x = \sum_{n \in S} x_{\mathbf{Co}(q_n)} + \sum_{i \in S^c} x_{q_n},$$

where $x_{\mathbf{Co}(q_n)}$ represents a point that belongs in $\mathbf{Co}(Q_n)$ and x_{q_n} represents a point that belongs in Q_n. More succinctly,

$$\mathbf{Co}\Big(\sum_{n=1}^N Q_n\Big) \subseteq \bigcup_{S \subseteq 1,\ldots,N : |S| \leq p} \sum_{n \in S} \mathbf{Co}(Q_n) + \sum_{n \in S^c} Q_n.$$

6.2 Bounds on the duality gap of the constrained problem

We derive upper and lower bounds on the constrained case (P-CON) in this section. The penalized case will follow from similar arguments in Section 6.3. In both sections, we assume $f(\cdot)$ is convex and show in Section 6.3 how the results change when $f(\cdot)$ is non convex. We begin by forming the dual problem.

Dual problem

Note that the constrained problem is equivalent to

$$p_{\text{con}}(k) = \min_{v \in \mathbb{R}^m, u \in \{0,1\}^m} \ f(XD(u)v) + \frac{\gamma}{2} v^\top D(u) v \ : \ \mathbf{1}^\top u \leq k \tag{6.1}$$

in the variables $u \in \mathbb{R}^m$ and $u \in \{0,1\}^m$, where $D(u) = \mathbf{diag}(u_1, \ldots, u_m)$, using the fact $D(u)^2 = D(u)$. Rewriting $f(\cdot)$ using its Fenchel conjugate and swapping the outer min with the inner max to get a dual, we have $d_{\text{con}}(k) \leq p_{\text{con}}(k)$ by weak duality, with

$$d_{\text{con}}(k) = \max_z -f^*(z) + \min_{v,u\in\{0,1\}^m} \frac{\gamma}{2} v^\top D(u) v + z^\top X D(u) v \ : \ \mathbf{1}^\top u \leq k$$

in the variable $z \in \mathbb{R}^n$. Solving the inner minimum over v, we have $v^* = -\frac{1}{\gamma} D(u)^\dagger D(u) X^\top z$. Plugging this back into our problem, we get

$$d_{\text{con}}(k) = \max_z -f^*(z) + \min_{u\in\{0,1\}^m} -\frac{1}{2\gamma} z^\top X D(u) D(u)^\dagger D(u) X^\top z \ : \ \mathbf{1}^\top u \leq k.$$

Noting that $D(u)D(u)^\dagger D(u) = D(u)$ and that $z^\top X D(u) D(u)^\dagger D(u) X^\top z$ is increasing with u, we have

$$d_{\text{con}}(k) = \max_{z,\zeta} -f^*(z) - \frac{1}{2\gamma} s_k(\zeta \circ \zeta) \ : \ \zeta = X^\top z,$$

where $s_k(\cdot)$ denotes the sum of top k entries of its vector argument (all nonnegative here).

Bidual problem

Rewriting $-s_k(x) = \min_{u\in[0,1]^m} -u^\top x$, we have

$$p^{**}_{\text{con}}(k) = d_{\text{con}}(k) = \max_z \min_{u\in[0,1]^m} -f^*(z) - \frac{1}{2\gamma} z^\top X D(u) D(u)^\dagger D(u) X^\top z \ : \ \mathbf{1}^\top u \leq k.$$

Note this is equivalent to realizing that the inner minimization in u in the previous section could be computed over the convex hull of the feasible set since the objective is in fact linear in u. Using convexity of the u variables (recall $D(u)D(u)^\dagger D(u) = D(u)$, and hence the objective is linear in u) and the concavity of the z variables, Sion's minimax theorem [88] allows us exchange the inner min and max to arrive at

$$p^{**}_{\text{con}}(k) = \min_{u\in[0,1]^m} \max_z -f^*(z) - \frac{1}{2\gamma} z^\top X D(u) D(u)^\dagger D(u) X^\top z \ : \ \mathbf{1}^\top u \leq k.$$

Since $D(u)D(u)^\dagger D(u) \succeq 0$ for all feasible u, we have using conjugacy on the quadratic form

$$p^{**}_{\text{con}}(k) = \min_{u\in[0,1]^m} \max_z \min_v -f^*(z) + \frac{\gamma}{2} v^\top D(u) v + z^\top X D(u) v \ : \ \mathbf{1}^\top u \leq k.$$

Switching the inner min and max again, using the definition of the biconjugate of $f(\cdot)$ and the relation that $f = f^{**}$ since $f(\cdot)$ is closed and convex, we get

$$p^{**}_{\text{con}}(k) = \min_{v\in\mathbb{R}^m, u\in[0,1]^m} f(XD(u)v) + \frac{\gamma}{2} v^\top D(u) v \ : \ \mathbf{1}^\top u \leq k. \qquad \text{(BD-CON)}$$

While (BD-CON) is non convex, setting $\tilde{v} = D(u)v$ means it is equivalent to the following convex program

$$p^{**}_{\text{con}}(k) = \min_{\tilde{v}\in\mathbb{R}^m, u\in[0,1]^m} f(X\tilde{v}) + \frac{\gamma}{2}\tilde{v}D(u)^{\dagger}\tilde{v} \;:\; \mathbf{1}^\top u \leq k \tag{6.2}$$

in the variables $\tilde{v}, u \in \mathbb{R}^m$, where $\tilde{v}^\top D(u)^{\dagger}\tilde{v}$ is jointly convex in $(\tilde{v}, u)$ since it can be rewritten as a second-order cone constraint. This substitution is without loss of generality since we can take $v_i = 0$ when $u_i = 0$ and vice versa. To compute (u^*, v^*), we solve the above problem and set $v^* = D(u^*)^{\dagger}\tilde{v}^*$. Note also that (BD-CON) is simply the interval relaxation of the (P-CON). We could have arrived at (BD-CON) by taking the interval relaxation of (6.1). However, by working through the dual, we have shown that the interval relaxation is equivalent to deriving a dual of the problem, implicitly relaxing the binary constraint on u by reformulating $s_k(\cdot)$, and then inverting the steps taken to arrive at the dual. In fact, in the analysis that follows, we only rely on (BD-CON) and not the dual.

Duality gap bounds and primalization

After deriving the bidual, we are now ready to derive our main result, which is explicit upper and lower bounds on the optimum of (P-CON) as a function of the rank of the data matrix X and $p^{**}_{\text{con}}(k)$ in (6.2). We will also detail a procedure to compute a primal feasible solution that satisfies the bounds. An equivalent analysis will follow for the penalized case. Later, in Section 6.5, we will make weaker assumptions on the rank of X and prove a more general result.

Theorem 13. *Suppose $X = U_r\Sigma_r V_r^\top$ is a compact, rank-r SVD decomposition of X. From a solution (v^*, u^*) of* (BD-CON) *with objective value t^*, with probability one, we can construct a point with at most $k + r + 2$ nonzero coefficients and objective value* OPT *satisfying*

$$p_{con}(k + r + 2) \leq \text{OPT} \leq p^{**}_{con}(k) \leq p_{con}(k) \tag{Gap-Bound}$$

by solving a linear program written

$$\begin{array}{ll} \textit{minimize} & c^\top u \\ \textit{subject to} & f(U_r z^*) + \sum_{i=1}^m u_i \frac{\gamma}{2} {v_i^*}^2 = t^* \\ & \sum_{i=1}^m u_i \leq k \\ & \sum_{i=1}^m u_i \ell_i v_i^* = z^* \\ & u \in [0,1]^m \end{array} \tag{6.3}$$

in the variable $u \in \mathbb{R}^m$, where $c \sim \mathcal{N}(0, I_m)$, $z^* = \Sigma_r V_r^\top D(u^*)v^*$.

Proof. Making the variable substitution $\Sigma_r V_r^\top D(u)v = z$, (BD-CON) can be rewritten as

$$p^{**}_{\text{con}}(k) = \min_{v,u\in[0,1]^m} \; f(U_r z) + \frac{\gamma}{2} v^\top D(u)v \;\;:\;\; \mathbf{1}^\top u \leq k,\; \Sigma_r V_r^\top D(u)v = z$$

and in epigraph form as

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \begin{bmatrix} t \\ k \\ z \end{bmatrix} \in \left(\begin{bmatrix} f(U_r z) \\ \mathbb{R}^+ \\ 0 \end{bmatrix} + \sum_{i=1}^m u_i \begin{bmatrix} \frac{\gamma}{2} v_i^2 \\ 1 \\ \ell_i v_i \end{bmatrix} \right) \\ & u \in [0,1]^m \end{array}$$

in the variables $t \in \mathbb{R}$, $z \in \mathbb{R}^r$, and $v, u \in \mathbb{R}^m$, where ℓ_i is the ith column of $\Sigma_r V_r^\top$. Note the above is equivalent to

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \begin{bmatrix} t \\ k \\ z \end{bmatrix} \in \left(\begin{bmatrix} f(U_r z) \\ \mathbb{R}^+ \\ 0 \end{bmatrix} + \sum_{i=1}^m \mathbf{Co} \left\{ 0, \begin{bmatrix} \frac{\gamma}{2} v_i^2 \\ 1 \\ \ell_i v_i \end{bmatrix} \right\} \right) \end{array}$$

in the variables $t \in \mathbb{R}$, $z \in \mathbb{R}^r$, and $v \in \mathbb{R}^m$. Recall $\mathbf{Co}$ denotes the convex hull of a set (in particular $\mathbf{Co}(0, x) = (1-t)x$ for $t \in [0,1]$ and $x \in \mathbb{R}^p$). The Shapley Folkman theorem [93] shows that for any

$$x \in \left(\sum_{i=1}^m \mathbf{Co} \left\{ 0, \begin{bmatrix} \frac{\gamma}{2} v_i^2 \\ 1 \\ \ell_i v_i \end{bmatrix} \right\} \right),$$

there exists some $\bar{u} \in [0,1]^m$ such that

$$x = \sum_{i \in S} \bar{u}_i \begin{bmatrix} \frac{\gamma}{2} v_i^2 \\ 1 \\ \ell_i v_i \end{bmatrix} + \sum_{i \in S^c} \bar{u}_i \begin{bmatrix} \frac{\gamma}{2} v_i^2 \\ 1 \\ \ell_i v_i \end{bmatrix},$$

where $S = \{i \mid \bar{u}_i \neq \{0,1\}\}$ and $|S| \leq r+2$. Let (t^*, z^*, v^*, u^*) be OPTimal for (BD-CON). Then there exists $s_1 \geq 0$ such that

$$\begin{bmatrix} t^* \\ k - s_1 \\ z^* \end{bmatrix} = \begin{bmatrix} f(U_r z^*) \\ 0 \\ 0 \end{bmatrix} + \sum_{i=1}^m \mathbf{Co} \left\{ 0, \begin{bmatrix} \frac{\gamma}{2} {v_i^*}^2 \\ 1 \\ \ell_i v_i^* \end{bmatrix} \right\}.$$

From the above, we know there exists $\bar{u}_i$ that satisfies these equality constraints, with at most $r+2$ non binary entries. In fact, we can compute this $\bar{u}$ by solving a linear program. To see this, given optimal (t^*, z^*, v^*, u^*) for the epigraph reformulation of (BD-CON), consider the linear program

$$\begin{array}{ll} \text{minimize} & c^\top u \\ \text{subject to} & f(U_r z^*) + \sum_{i=1}^m u_i \frac{\gamma}{2} {v_i^*}^2 = t^* \\ & \sum_{i=1}^m u_i \leq k \\ & \sum_{i=1}^m u_i \ell_i v_i^* = z^* \\ & u \in [0,1]^m \end{array} \tag{6.4}$$

in the variable $u \in \mathbb{R}^m$, where $c \sim \mathcal{N}(0, I_m)$. The problem is feasible since u^* is feasible. This is a linear program with $2m + r + 2$ constraints, of which m will be saturated at a non degenerate basic feasible solution. This implies that at least $m - r - 2$ constraints in $0 \leq u \leq 1$ are saturated with probability one, so at least $m - r - 2$ coefficients of u_i will be binary at the optimum.

Now we primalize as follows: given $(t^*, z^*, v^*, \bar{u})$ where $\bar{u}$ is a non degenerate basic feasible solution of the LP in (6.3), let $S \equiv \{i \mid \bar{u}_i \notin \{0, 1\}\}$, and define

$$\begin{cases} \tilde{v}_i = \bar{u}_i v_i^*, \;\; \tilde{u}_i = 1, & i \in S, \\ \tilde{v}_i = v_i^*, \;\; \tilde{u}_i = \bar{u}_i, & i \in S^c. \end{cases}$$

We now claim that $(z^*, \tilde{v}, \tilde{u})$ is feasible for the primal problem $p_{\text{con}}(k+r+2)$ and has objective value smaller than $p_{\text{con}}^{**}(k)$. By construction, $\tilde{u} \in \{0, 1\}^m$ and $\mathbf{1}^\top \tilde{u} = \|\tilde{u}\|_0 \leq k + r + 2$. Furthermore, we have

$$\begin{aligned} z^* &= \sum_{i=1}^{m} \bar{u}_i \ell_i v_i^* \\ &= \sum_{i \in S} \bar{u}_i \ell_i v_i^* + \sum_{i \in S^c} \bar{u}_i \ell_i v_i^* \\ &= \sum_{i \in S} \tilde{u}_i \ell_i \tilde{v}_i + \sum_{i \in S^c} \tilde{u}_i \ell_i \tilde{v}_i. \end{aligned}$$

Hence, $(z^*, \tilde{v}, \tilde{u})$ is feasible for $p_{\text{con}}(k + r + 2)$ in (6.2) and reaches an objective value OPT satisfying

$$\begin{aligned} t^* &= f(U_r z^*) + \tfrac{\gamma}{2} \left(\sum_{i \in S}^{m} \bar{u}_i v_i^{*2} + \sum_{i \in S^c}^{m} \bar{u}_i v_i^{*2} \right) \\ &\geq f(U_r z^*) + \tfrac{\gamma}{2} \left(\sum_{i \in S}^{m} \bar{u}_i^2 v_i^{*2} + \sum_{i \in S^c}^{m} \bar{u}_i v_i^{*2} \right) \\ &= f(U_r z^*) + \tfrac{\gamma}{2} \left(\sum_{i \in S}^{m} \tilde{u}_i \tilde{v}_i^{\,2} + \sum_{i \in S^c}^{m} \tilde{u}_i \tilde{v}_i^2 \right) \\ &\equiv \text{OPT}. \end{aligned}$$

Since $(z^*, \tilde{v}, \tilde{u})$ is feasible for $p_{\text{con}}(k + r + 2)$, we have $p_{\text{con}}(k + r + 2) \leq \text{OPT}$, and the result follows. □

This means that the primalization procedure will always reconstruct a point with at most $k + r + 2$ nonzero coefficients, with objective value at most $p_{\text{con}}(k) - p_{\text{con}}(k + r + 2)$ away from the optimal value $p_{\text{con}}(k)$. Note that this bound does not explicitly depend on the value of $\gamma > 0$, which could be arbitrarily small and could simply be treated as a technical regularization term.

6.3 Bounds on the duality gap of the penalized problem

The analysis for the penalized case is very similar to that of the constrained case. We start with deriving the dual problem.

Dual problem

The penalized problem is equivalent to

$$p_{\text{pen}}(\lambda) = \min_{v\in\mathbb{R}^m, u\in\{0,1\}^m} f(XD(u)v) + \frac{\gamma}{2} v^\top D(u)v + \lambda \mathbf{1}^\top u \tag{6.5}$$

in the variables $u, v \in \mathbb{R}^m$. Rewriting f using its Fenchel conjugate, switching the min and max, and solving the minimization over v, we have

$$d_{\text{pen}}(\lambda) = \max_z -f^*(z) + \min_{u\in\{0,1\}^m} -\frac{1}{2\gamma} z^\top XD(u)D(u)^\dagger D(u)X^\top z + \lambda \mathbf{1}^\top u.$$

Using

$$\min_{u\in\{0,1\}^m} -\frac{1}{2\gamma} z^\top XD(u)D(u)^\dagger D(u)X^\top z + \lambda \mathbf{1}^\top u = \sum_{i=1}^m \min\left(0, \lambda - \tfrac{1}{2\gamma}(X^\top z)_i^2\right),$$

the dual problem then becomes

$$d_{\text{pen}}(\lambda) = \max_z -f^*(z) + \sum_{i=1}^m \min\left(0, \lambda - \tfrac{1}{2\gamma}(X^\top z)_i^2\right)$$

with $d_{\text{pen}}(\lambda) \leq p_{\text{pen}}(\lambda)$.

Bidual problem

Rewriting the second term of our objective in variational form, we have

$$p_{\text{pen}}^{**}(\lambda) = d^*(\lambda) = \max_z \min_{u\in[0,1]^m} -f^*(z) - \frac{1}{2\gamma} z^\top XD(u)D(u)^\dagger D(u)X^\top z + \lambda \mathbf{1}^\top u.$$

Performing the same analysis as for the constrained case (cf. Section 6.2), we get

$$p_{\text{pen}}^{**}(\lambda) = \min_{v,u\in[0,1]^m} f(XD(u)v) + \frac{\gamma}{2} v^\top D(u)v + \lambda \mathbf{1}^\top u \tag{BD-PEN}$$

in the variables $u, v \in \mathbb{R}^m$, which can be recast as a convex program as in (6.2).

Corollary 1. *Suppose $X = U_r \Sigma_r V_r^\top$ is a compact, rank-r SVD decomposition of X. From a solution (v^*, u^*) of* (BD-PEN) *with objective value t^*, with probability one, we can construct a point with objective value* OPT *satisfying*

$$p^{**}_{pen}(\lambda) \leq p_{pen}(\lambda) \leq \text{OPT} \leq p^{**}_{pen}(\lambda) + \lambda(r+1) \qquad \text{(Gap-Bound-Pen)}$$

by solving a linear program written

$$\begin{array}{ll} \textit{minimize} & c^\top u \\ \textit{subject to} & f(U_r z^*) + \sum_{i=1}^m u_i \frac{\gamma}{2} {v_i^*}^2 + \lambda u_i = t^* \\ & \sum_{i=1}^m u_i \ell_i v_i^* = z^* \\ & u \in [0,1]^m \end{array} \tag{6.6}$$

in the variable $u \in \mathbb{R}^m$, where $c \sim \mathcal{N}(0, I_m)$ and $z^* = \Sigma_r V_r^\top D(u^*) v^*$.

Proof. The primalization procedure is analogous to the constrained case, the only difference being the linear program becoming (6.6). We then get the chain of inequalities in (Gap-Bound-Pen), which means that starting from an optimal point of (BD-PEN), the primalization procedure will generate a feasible point with objective value at most $\lambda(r+1)$ larger than that of the original problem (P-PEN). □

Connections with other relaxations

We first draw the connection between the penalty term in the bidual and the reverse Huber penalty. The reverse Huber function is defined as

$$\begin{aligned} B(\zeta) &= \frac{1}{2} \min_{0 \leq \nu \leq 1} \nu + \frac{\zeta^2}{\nu} \\ &= \begin{cases} |\zeta| & \text{if } |\zeta| \leq 1 \\ \dfrac{\zeta^2 + 1}{2} & \text{otherwise.} \end{cases} \end{aligned}$$

We have

$$\min_{\substack{u \in [0,1]^m \\ \mathbf{1}^\top u \leq k}} x^\top D(u)^{-1} x = \max_{t > 0} \sum_{i=1}^n tB\Big(\frac{|x_i|}{\sqrt{t}}\Big) - \frac{1}{2}tk.$$

There is a direct connection between the second representation of (BD-CON) (based on the variable substitution $\tilde{v} = D(u)v$) and the well-known perspective-based relaxation [35] (a similar argument can also be made for (BD-PEN)). Note that (P-CON) is equivalent to

$$\min_{x,u,v} f(x) + \mathbf{1}^\top v \ : \ u \in \{0,1\}^m, \ \mathbf{1}^\top u \leq k, \ u_i v_i \geq x_i^2, \ i = 1, \ldots, m.$$

To see this, assume that x is optimal for (P-CON). If u encodes the sparsity pattern of x, we simply set $v_i = x_i^2$, so we have $\mathbf{1}^\top v = x^\top x$, and that triplet (x, u, v) is feasible for the above problem. Similarly, if (x, u, v) are optimal for the above representation, then $x_i = 0$ if $u_i = 0$ and $x_i^2 = v_i$ otherwise. Similarly, $\mathbf{1}^\top v = x^\top x$, and x is feasible for (P-CON). Relaxing $u \in \{u \mid u \in [0,1]^m,\ \mathbf{1}^\top u \leq k\}$ and replacing f^{**} with f results in the perspective relaxation of the problem, which is equivalent to (BD-CON).

Extension to non convex setting

The gap bounds derived above can be extended to the case when f is non convex. Starting from (P-CON) and following the structure of (BD-CON), consider the relaxation

$$p^{**}_{\text{con}}(k) = \min_{v,u \in [0,1]^m} f^{**}(XD(u)v) + \frac{\gamma}{2} v^\top D(u) v \ :\ \mathbf{1}^\top u \leq k,$$

where $f(\cdot)$ in (BD-CON) has been replaced by its convex envelope $f^{**}(\cdot)$ (i.e., the largest convex lower bound on f). By construction, this constitutes a lower bound on (P-CON). The analysis follows the same steps as in the proof of Theorem 13, replacing f with f^{**} everywhere. The only bound that changes is $p_{\text{con}}(k + r + 2) \leq \text{OPT}$ since the objective defining OPT uses f^{**} while that defining $p_{\text{con}}(k+r+2)$ uses f. For a non convex function, we can define the lack of convexity $\rho(f) = \sup_w f(Xw) - f^{**}(Xw)$ with $\rho(f) \geq 0$. We then have $-\rho(f) \leq f^{**}(U_r z^*) - f(U_r z^*)$ and then the chain of inequalities in (Gap-Bound) becomes

$$p_{\text{con}}(k + r + 2) - \rho(f) \leq \text{OPT} \leq p^{**}_{\text{con}}(k) \leq p_{\text{con}}(k).$$

The exact same analysis and reasoning can be applied to the penalized case to arrive at

$$p^{**}_{\text{pen}}(\lambda) - \rho(f) \leq p_{\text{pen}}(\lambda) - \rho(f) \leq \text{OPT} \leq p^{**}_{\text{pen}}(\lambda) + \lambda(r+1).$$

6.4 Quadratically constrained sparse problems

In this section, we consider a version of (P-CON) where the ℓ_2 penalty is replaced by a hard constraint. The explicit ℓ_2 constraint proves useful to get tractable bounds when solving approximate versions of (P-CON) where X is approximately low rank (see Section 6.5). We follow the same analysis as before and derive similar duality gap bounds and primalization procedures. We omit some steps of the analysis for brevity and refer the reader to Sections 6.2 and 6.3 for more details. We assume f is convex and can extend the analysis to the non convex setting using the same arguments in Section 6.3 (for brevity, we omit this). We wish to point out that there is nothing enlightening about the proofs in this section, and on a first pass, the reader can skip directly to Section 6.5.

ℓ_2-ℓ_0 constrained optimization

As before, we first derive dual and bidual problems in the quadratically constrained case.

Dual problem

Note that the ℓ_2-constrained problem is equivalent to

$$p^*_{\text{con}}(k) = \min_{v,u\in\{0,1\}^m} f(XD(u)v) \ : \ \mathbf{1}^\top u \leq k, \ v^\top D(u)v \leq \gamma,$$

where $D(u) = \mathbf{diag}(u_1, \ldots, u_m)$ and we use the fact $D(u)^2 = D(u)$. Rewriting f using its Fenchel conjugate, introducing a dual variable η for the ℓ_2 constraint, swapping the outer min with the inner max via weak duality, and solving the minimum over v, we have

$$d^*_{\text{con}}(k) = \max_{z,\eta\geq 0} -f^*(z) - \frac{\eta\gamma}{2} + \min_{u\in\{0,1\}^m} -\frac{1}{2\eta} z^\top XD(u)D(u)^\dagger D(u)X^\top z \ : \ \mathbf{1}^\top u \leq k,$$

where $d^*_{\text{con}}(k) \leq p^*_{\text{con}}(k)$. This further reduces to

$$d^*_{\text{con}}(k) = \max_{z,\eta\geq 0} -f^*(z) - \frac{\eta\gamma}{2} - \frac{1}{2\eta} s_k(\zeta \circ \zeta) \ : \ \zeta = X^\top z,$$

where $s_k(\cdot)$ denotes the sum of top k entries of its vector argument. Note the problem is convex since the latter term is the perspective function of $s_k(\zeta \circ \zeta)$.

Bidual problem

Rewriting $s_k(\cdot)$ in variational form, we have that

$$p^{**}_{\text{con}}(k) = d^*_{\text{con}}(k) = \max_{z,\eta\geq 0} \min_{u\in[0,1]^m} -f^*(z) - \frac{\eta\gamma}{2} - \frac{1}{2\eta} z^\top XD(u)D(u)^\dagger D(u)X^\top z \ : \ \mathbf{1}^\top u \leq k.$$

Swapping the min and max and using the Fenchel conjugate of the quadratic form, we have

$$p^{**}_{\text{con}}(k) = \min_{u\in[0,1]^m} \max_{z,\eta\geq 0} \min_v -f^*(z) - \frac{\eta\gamma}{2} + \frac{\eta}{2} v^\top D(u)v + z^\top XD(u)v \ : \ \mathbf{1}^\top u \leq k.$$

Switching the inner min and max again, using the definition of the biconjugate conjugate of $f(\cdot)$, and computing the maximum over η, we arrive at

$$p^{**}_{\text{con}}(k) = \min_{v,u\in[0,1]^m} f(XD(u)v) \ : \ \mathbf{1}^\top u \leq k, \ \ v^\top D(u)v \leq \gamma, \tag{6.7}$$

which can be rewritten as a convex program (cf. Section 6.2).

Corollary 2. *Suppose $X = U_r\Sigma_r V_r^\top$ is a compact, rank-r SVD decomposition of X. From a solution (v^*, u^*) of (6.7) with objective value t^*, with probability one, we can construct a point with objective value* OPT *satisfying*

$$p^*_{con}(k+r+2) \leq \text{OPT} \leq p^{**}_{con}(k) \leq p^*_{con}(k) \tag{Gap-Bound2}$$

by solving a linear program written

$$\begin{array}{ll} \textit{minimize} & c^\top u \\ \textit{subject to} & \sum_{i=1}^m u_i \leq k \\ & \sum_{i=1}^m u_i {v_i^*}^2 \leq \gamma \\ & \sum_{i=1}^m u_i \ell_i v_i^* = z^* \\ & u \in [0,1]^m \end{array} \tag{6.8}$$

in the variable $u \in \mathbb{R}^m$, where $c \sim \mathcal{N}(0, I_m)$ and (t^*, v^*) are optimal for the bidual, with $z^* = \Sigma_r V_r^\top D(u^*)v^*$.

Proof. Following the analysis in Section 6.2, let $X = U_r\Sigma_r V_r^\top$ be a compact, rank-r SVD decomposition of X. Making the variable substitution $\Sigma_r V_r^\top D(u)v = z$, our bidual can be rewritten in epigraph form as

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \begin{bmatrix} t \\ k \\ \gamma \\ z \end{bmatrix} \in \begin{bmatrix} f(U_r z) \\ \mathbb{R}^+ \\ \mathbb{R}^+ \\ 0 \end{bmatrix} + \sum_{i=1}^m \mathbf{Co}\left\{ 0, \begin{bmatrix} 0 \\ 1 \\ v_i^2 \\ \ell_i v_i \end{bmatrix} \right\} \\ & u \in [0,1]^m \end{array}$$

in the variables $t \in \mathbb{R}$, $z \in \mathbb{R}^n$, and $v, u \in \mathbb{R}^m$, where ℓ_i is the ith column of $\Sigma_r V_r^\top$. Note that from the Shapley Folkman lemma [93], there exists some $\bar{u} \in [0,1]^m$ such that

$$x = \sum_{i \in S} \bar{u}_i \begin{bmatrix} 0 \\ 1 \\ v_i^2 \\ \ell_i v_i \end{bmatrix} + \sum_{i \in S^c} \bar{u}_i \begin{bmatrix} 0 \\ 1 \\ v_i^2 \\ \ell_i v_i \end{bmatrix},$$

where $S = \{i \mid \bar{u}_i \neq \{0,1\}\}$ and $|S| \leq r+2$ (note we disregard the first entry of the vector, and hence it is $r+2$ and not $r+3$). Now let (t^*, z^*, v^*, u^*) be optimal for the bidual. That means there exists $s_1, s_2 \geq 0$ such that

$$\begin{bmatrix} t^* \\ k - s_1 \\ \gamma - s_2 \\ z^* \end{bmatrix} = \begin{bmatrix} f(U_r z^*) \\ 0 \\ 0 \end{bmatrix} + \sum_{i=1}^m \mathbf{Co}\left\{ 0, \begin{bmatrix} 0 \\ 1 \\ {v_i^*}^2 \\ \ell_i v_i^* \end{bmatrix} \right\}.$$

From the above, we know there exists $\bar{u}_i$ that satisfies the above vector equality with at most $r+2$ non binary entries. We can compute this $\bar{u}$ via the linear program in (6.8). We then primalize precisely as before to arrive at the chain of inequalities

$$p^*_{\text{con}}(k+r+2) \leq \text{OPT} \leq p^{**}_{\text{con}}(k) \leq p^*_{\text{con}}(k),$$

which is the desired result. □

ℓ_2-constrained, ℓ_0-penalized optimization

The analysis for the penalized case is very similar to that of Section 6.3.

Dual problem

The penalized problem is equivalent to

$$p^*_{\text{pen}}(\lambda) = \min_{v,u\in\{0,1\}^m} f(XD(u)v) + \lambda \mathbf{1}^\top u \ : \ v^\top D(u) v \leq \gamma. \tag{6.9}$$

Using the Fenchel conjugate of f, introducing a dual variable η for the ℓ_2 constrain, using weak duality, and computing the minimization over v, we have

$$d^*_{\text{pen}}(\lambda) = \max_{z,\eta\geq 0} -f^*(z) - \frac{\eta\gamma}{2} + \min_{u\in\{0,1\}^m} -\frac{1}{2\eta} z^\top XD(u)D(u)^\dagger D(u)X^\top z + \lambda \mathbf{1}^\top u.$$

Using the fact that

$$\min_{u\in\{0,1\}^m} -\frac{1}{2\eta} z^\top XD(u)D(u)^\dagger D(u)X^\top z + \lambda \mathbf{1}^\top u = \sum_{i=1}^m \min\left(0, \lambda - \tfrac{1}{2\eta}(X^\top z)_i^2\right),$$

the dual problem becomes

$$d^*_{\text{pen}}(\lambda) = \max_{z} -f^*(z) - \frac{\eta\gamma}{2} + \sum_{i=1}^m \min\left(0, \lambda - \tfrac{1}{2\eta}(X^\top z)_i^2\right)$$

with $d^*_{\text{pen}}(\lambda) \leq p^*_{\text{pen}}(\lambda)$. The term $\frac{1}{2\eta}(X^\top z)_i^2$ is jointly convex since it can be recast as a second-order cone constraint using the fact that $z_i^2/\eta \leq t \iff \left\| \begin{bmatrix} z_i \\ t-\eta \end{bmatrix} \right\|_2 \leq \frac{1}{2}(t+\eta)$.

Bidual problem

Rewriting the second term of our objective in variational form, we have

$$p^{**}_{\text{pen}}(\lambda) = d^*(\lambda) = \max_{z,\eta\geq 0} \min_{u\in[0,1]^m} -f^*(z) - \frac{\eta\gamma}{2} - \frac{1}{2\gamma} z^\top XD(u)D(u)^\dagger D(u)X^\top z + \lambda \mathbf{1}^\top u.$$

Performing the same analysis as for the constrained case, we have that

$$p^{**}_{\text{pen}}(\lambda) = \min_{v,u\in[0,1]^m} f(XD(u)v) + \lambda \mathbf{1}^\top u \ : \ v^\top D(u) v \leq \gamma, \tag{6.10}$$

which can be recast as a convex program (cf. Section 6.2).

Corollary 3. *Suppose $X = U_r \Sigma_r V_r^\top$ is a compact, rank-r SVD decomposition of X. From a solution (v^*, u^*) of* (BD-PEN) *with objective value t^*, with probability one, we can construct a point with objective value* OPT *satisfying*

$$p^{**}_{pen}(\lambda) \leq p_{pen}(\lambda) \leq \text{OPT} \leq p^{**}_{pen}(\lambda) + \lambda(r+1) \tag{Gap-Bound-Pen-l2}$$

by solving a linear program written

$$\begin{array}{ll} \textit{minimize} & c^\top u \\ \textit{subject to} & f^{**}(U_r z^*) + \lambda u_i = t^* \\ & \sum_{i=1}^m u_i v_i^{*2} \leq \gamma \\ & \sum_{i=1}^m u_i \ell_i v_i^* = z^* \\ & u \in [0,1]^m \end{array} \tag{6.11}$$

in the variable $u \in \mathbb{R}^m$ with $z^* = \Sigma_r V_r^\top D(u^*) v^*$.

Proof. The primalization procedure is analogous to the constrained case with the only difference being the linear program becoming (6.11). Performing the same analysis as for the penalized case, we have the chain of inequalities in (Gap-Bound-Pen-l2). □

6.5 Tighter bounds for approximately low-rank matrices

The duality gap bounds detailed above depend on r, the rank of the matrix X, which is an unstable quantity. In other words, a very marginal change in X can have a significant impact on the quality of the bounds. In what follows, we will see how to improve these bounds when the matrix X is approximately low-rank, which will allow us to bound the duality gap.

Starting from the ℓ_2-ℓ_0 constrained formulation, we formulate a perturbed version

$$\begin{array}{rll} p^*_{\text{con}}(k, X, \delta) = & \text{minimize} & f(z; y) \\ & \text{subject to} & Xw = z + \delta, \\ & & \|w\|_0 \leq k \\ & & \|w\|_2^2 \leq \gamma \end{array} \tag{6.12}$$

in the variables $w \in \mathbb{R}^m$ and $z \in \mathbb{R}^n$, where $\delta \in \mathbb{R}^n$ is a perturbation parameter. Let

$$X = X_r + \Delta X, \quad \mathbf{Rank}\, X_r = r,$$

be a decomposition of the matrix X. For notational convenience, we set $p^*_{\text{con}}(k, X) = p^*_{\text{con}}(k, X, 0)$. We have the following result.

Proposition 14. *Let $w_r^\star$ be the optimal solution of $p^*_{con}(k, X_r, 0)$ and ν_r^* the dual optimal variable corresponding to the equality constraint, and write $(w^\star, \nu^\star)$ the corresponding solutions for $p^*_{con}(k, X, 0)$. We have*

$$p^*_{con}(k, X, 0) - \nu^{*T}\Delta X w_r^\star \leq p^*_{con}(k, X_r, 0) \leq p^*_{con}(k, X, 0) - \nu_r^{*T}\Delta X w^\star \tag{6.13}$$

and the exact same bound when we start with the ℓ_2-constrained, ℓ_0-penalized formulation.

Proof. Suppose $w_r^\star$ is an optimal solution of problem $p^*_{\text{con}}(k, X_r, 0)$. Then $w_r^\star$ is also a feasible point of problem $p^*_{\text{con}}(k, X, \Delta X w_r^\star)$ because

$$(X_r + \Delta X)w_r^\star = z + \Delta X w_r^\star$$

by construction. Since the two problems share the same objective function, this means $p^*_{\text{con}}(k, X_r, \Delta X w_r^\star) \leq p^*_{\text{con}}(k, X_r, 0)$. Now weak duality yields

$$p^*_{\text{con}}(k, X, \Delta X w_r^\star) \geq p^*_{\text{con}}(k, X, 0) - \nu^{\star T}\Delta X w_r^\star$$

and

$$p^*_{\text{con}}(k, X_r, 0) \leq p^*_{\text{con}}(k, X_r, -\Delta X w^\star) - \nu_r^{\star T}\Delta X w^\star.$$

We conclude using as above the fact that if $w^\star$ is an optimal solution of problem $p^*_{\text{con}}(k, X, 0)$, then $w^\star$ is also a feasible point of problem $p^*_{\text{con}}(k, X_r, \Delta X w^\star)$ because $X_r w^\star = z - \Delta X w^\star$, which yields $p^*_{\text{con}}(k, X_r, -\Delta X w^\star) \leq p^*_{\text{con}}(k, X, 0)$ and the desired result. In the proof, we only used weak duality and the equality constraint in $p^*_{\text{con}}(k, X, \delta)$ to arrive at the result. Consequently, the exact same proof and bounds hold for $p^*_{\text{pen}}(\lambda, X, \delta)$. □

We are now ready to combine the bound in Proposition 14 with the bounds derived in Section 6.4.

Proposition 15. *Let $w_r^\star$ be the optimal solution of $p^*_{con}(k, X_r)$ and ν_r^* the dual optimal variable corresponding to the equality constraint, and write $(w^\star, \nu^\star)$ the corresponding solutions for $p^*_{con}(k, X)$. Furthermore, let $\zeta_r = \sqrt{\gamma}\|\Delta X^\top \nu_r^*\|_2$ and $\zeta = \sqrt{\gamma}\|\Delta X^\top \nu^*\|_2$. We have*

$$-\zeta_r - \zeta + p^{**}_{con}(k + r + 2, X_r) \leq \text{OPT} \leq p_{con}(k, X) - \zeta \leq p^{**}_{con}(k, X_r). \tag{6.14}$$

Similarly, for $p^*_{pen}(\lambda, X_r)$, we have

$$-\zeta_r - \zeta + p^{**}_{pen}(\lambda, X_r) \leq p_{pen}(\lambda, X) - \zeta \leq \text{OPT} \leq p^{**}_{pen}(\lambda, X_r) + \lambda(r + 1). \tag{6.15}$$

Proof. Starting from $p_{\text{con}}(k, X) = p_{\text{con}}(k, X) - p_{\text{con}}(k, X_r) + p_{\text{con}}(k, X_r)$, upper and lower bounding $p_{\text{con}}(k, X) - p_{\text{con}}(k, X_r)$ using Proposition (14) and the Cauchy Schwarz inequality, and using the bounds derived in Section 6.4, the result follows. The proof for the penalized case is identical. □

6.6 Experiments

Experiment 1: Duality gap bounds

In this experiment, we generate synthetic data to illustrate the duality gap bounds derived in Sections 6.2 and 6.3. We plot these bounds for the $f(Xw; y) = \frac{1}{2n}\|Xw - y\|_2^2$ (linear regression) and $f(Xw; y) = \frac{1}{n}\sum_{i=1}^{n} \log(1 + \exp(-y_i(x_i^\top w)))$ (logistic regression). Note that both functions are convex and closed, hence, $f^{**} = f$ and $\rho(f) = 0$. Specifically, we generate samples $X \in \mathbb{R}^{1000\times 100}$ with $\text{rank}(X) = 10$ by first generating $X_{ij} \sim \mathcal{N}(0, 1)$ and then taking a rank-10 SVD. We generate $\beta \in \mathbb{R}^{100}$ with $\beta_i \sim \mathcal{N}(0, 25)$ and $\|\beta\|_0 = 10$. In the case of ℓ_2 loss, we set $y = X\beta + \epsilon$, and for the logistic loss, we set $y = 2\text{Round}(\text{Sigmoid}(X\beta + \epsilon)) - 1 \in \{-1, 1\}^n$, where $\epsilon_i \sim \mathcal{N}(0, 1)$. For both models, we add a ridge penalty $\frac{\gamma}{2}\|w\|_2^2$ with $\gamma = 0.01$. For the regression task, we use a ℓ_0-penalty, while for the classification task, we use a ℓ_0-constraint. Figure 6.1 shows the primalized optimal values as well as the upper and lower bounds derived earlier. When running the primalization procedure, we pick 20 random linear objectives and show the standard deviation in the value OPT.

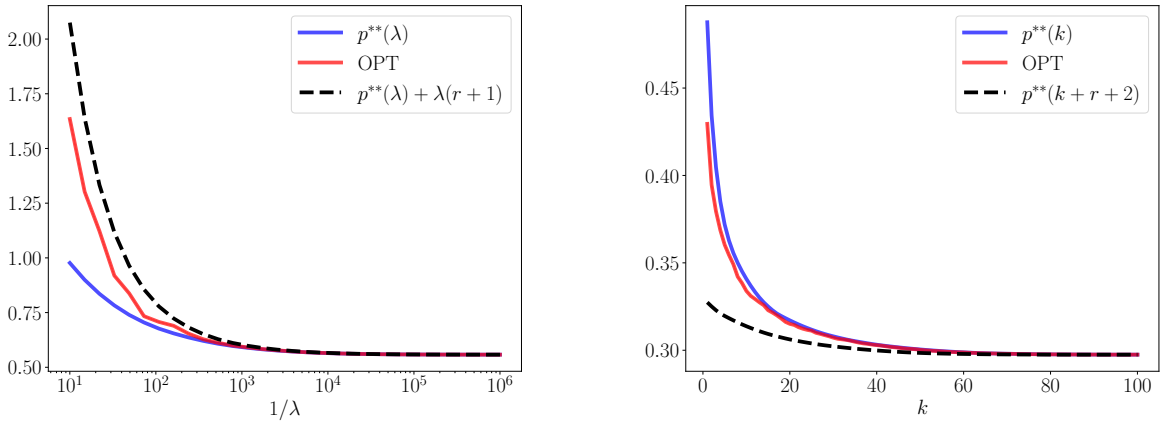

Figure 6.1: Experiment 1. *(Left)* Linear regression with a ℓ_0-penalty. *(Right)* Logistic regression with a ℓ_0-constraint.

Note that there are no error bars around OPT despite having solved the primalization linear program with 20 different random linear objectives for each value of the regularization parameters (λ or k). This strongly indicates that our feasible set for the linear program is actually a singleton (which was verified by changing the linear objective to arbitrary convex objectives and noting that the arg min was identical each time). In this case, the solution is identical to the solution that can be inferred from the bidual (since we know the linear program is feasible since the solution of the bidual satisfies the constraints). As a result, primalization simply reduces to rounding the bidual solution to make it primal feasible. Furthermore, note that in the left plot of Figure 6.1, we know that the true value $p^*_{\text{pen}}(\lambda)$

must lie somewhere between OPT (red line) and p^{**} (blue line) and that this gap decreases as λ decreases. This is also apparent in the right plot of Figure 6.1 as the marginal importance of the features decreases as k increases.

Experiment 2: Numerical rank bounds

In this experiment, we plot the bounds outlined by Proposition 15 that combine Shapley Folkman with numerical rank bounds. We generate $X \in \mathbb{R}^{1000\times 100}$ with bell-shaped singular values using the `make_low_rank_matrix` function in scikit-learn [83] to get a numerical rank of 10. We then generate β and y as in Experiment 1 for the ℓ_2 loss. As was used to derive the numerical rank bounds, we use a constraint $\|w\|_2^2 \leq \gamma$ with $\gamma = 30$ instead of a ridge penalty. We consider the ℓ_0-penalized case and fix three values of $\lambda : 10^{-4}, 10^{-3}, 10^{-2}$. In Figure 6.2, we show how the bounds change as we vary the rank of our approximation X_r from 1 to 100. While running the primalization procedure, we pick a random linear objective 20 times and show the standard deviation in the value of OPT.

From Proposition 15, we know that $p_{\text{pen}}(\lambda, X)$ lies between the red and blue lines. For small values of λ (e.g. $\lambda = 10^{-4}$), we see that as the rank increases, this gap is essentially zero. This means that in the case of $\lambda = 10^{-4}$, taking a rank 20 approximation of the data matrix or a rank 100 matrix and doing the procedure highlighted in Section 6.4 results in two different solutions that are both essentially optimal. Both plots at the bottom of Figure 6.2 highlight a trade-off in choosing the numerical rank; a lower rank improves the duality gap, while it coarsens the objective function approximation and vice versa. This is further illustrated in the experiment below.

Experiment 3: Numerical Rank Bounds on Natural Data Sets

In this experiment, we generate the same plot as in Experiment 2 but now with real data, which we expect to be low rank [102]. Specifically, we use the leukemia data [25] with $n = 72$ binary responses and $m = 3751$ features. We scale the data matrix X and then plot the difference between the upper and lower bounds (duality gap) and the difference between the primalized upper bound and lower bounds (primalized gap) in Figure 6.3 under a logistic loss with $\lambda = 0.1$ and $\gamma = 50$.

In Figure 6.3, we see that both gaps drop sharply until a rank of about 10 (which explains around 50% of the variance) and then begin to taper off. We also see that the primalization gap is approaching zero, which shows that as the approximation quality increases, we are converging closer to the non convex solution via the primalization procedure. However, our duality gap bound does not approach zero because of the $\lambda(r + 1)$ term. The difference between the two gaps in Figure 6.3 highlights that the primalized upper bound and lower bound are approaching one another, while the dual upper bound and lower bound are not.

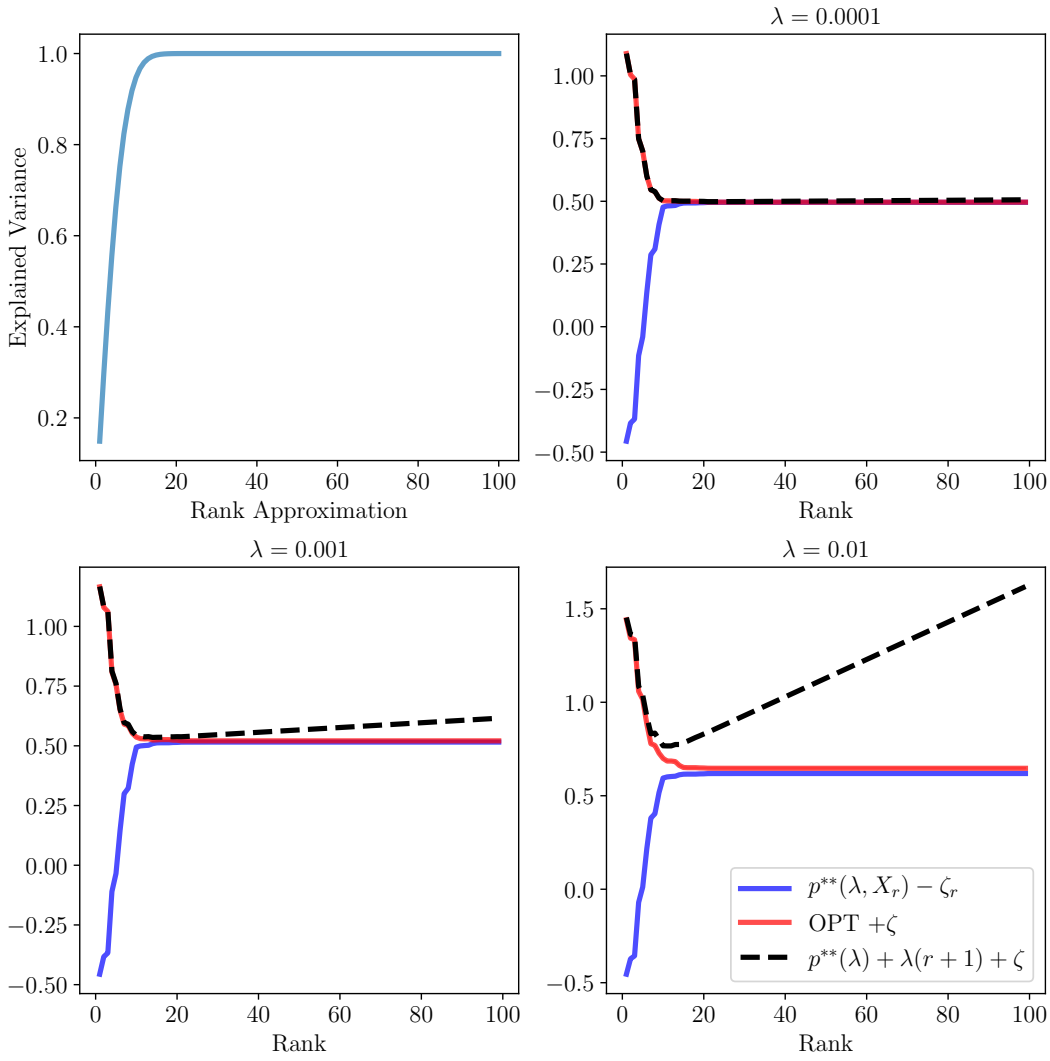

Figure 6.2: Experiment 2. *(Top left)* Explained variance plot of successive rank-r SVD approximations of X. Note that rank 10 explains most of the variance. *(Top right, bottom)* Plot of upper and lower bounds on $p_{\text{pen}}(\lambda, X)$ for various values of λ as the rank-r approximation of X changes.

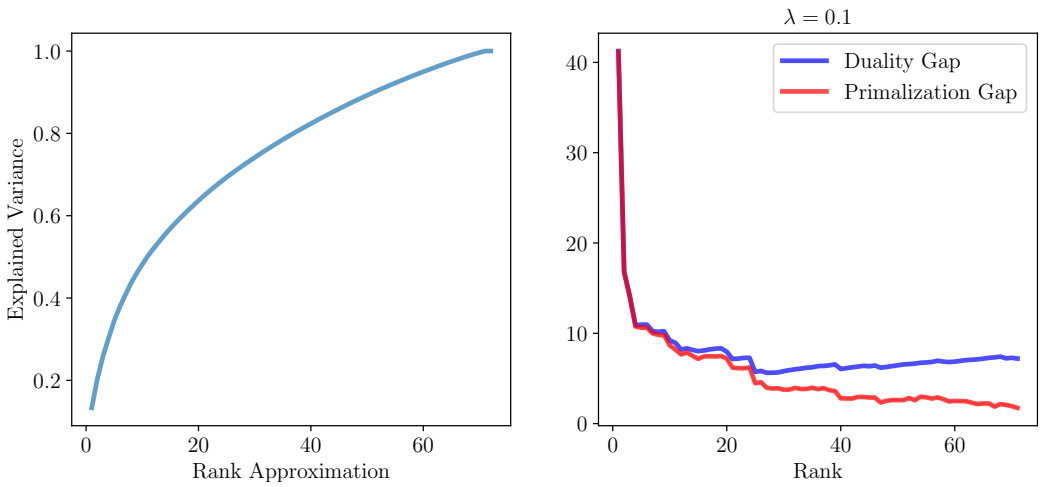

Figure 6.3: Experiment 3. *(Left)* Explained variance plot of successive rank-r SVD approximations of X. *(Right)* Difference between upper bounds and primalized value with lower bound as in Proposition 15.

Bibliography

[1] Alexandre Abraham et al. "Machine learning for neuroimaging with scikit-learn". In: *Frontiers in Neuroinformatics* 8 (2014), p. 14. ISSN: 1662-5196. DOI: 10.3389/fninf.2014.00014. URL: https://www.frontiersin.org/article/10.3389/fninf.2014.00014.

[2] Martin Andersen et al. "Interior-point methods for large-scale cone programming". In: *Optimization for machine learning* 5583 (2011).

[3] Armin Askari et al. "Lifted Neural Networks". In: *arXiv preprint arXiv:1805.01532* (2018).

[4] Alper Atamturk and Andres Gomez. "Rank-One Convexification for Sparse Regression". In: *arXiv preprint arXiv:1901.10334* (2019).

[5] Alper Atamturk, Andres Gomez, and Shaoning Han. "Sparse and Smooth Signal Estimation: Convexification of l0 Formulations". In: *arXiv preprint arXiv:1811.02655* (2018).

[6] Jean-Pierre Aubin and Ivar Ekeland. "Estimates of the duality gap in nonconvex optimization". In: *Mathematics of Operations Research* 1.3 (1976), pp. 225–245.

[7] Jushan Bai. "Inferential theory for factor models of large dimensions". In: *Econometrica* 71.1 (2003), pp. 135–171.

[8] O. Banerjee, A. d'Aspremont, and L. El Ghaoui. "Sparse Covariance Selection via Robust Maximum Likelihood Estimation". In: *ArXiv: cs.CE/0506023* (2005).

[9] Leonard Berrada, Andrew Zisserman, and M. Pawan Kumar. "Trusting SVM for Piecewise Linear CNNs". In: *CoRR* abs/1611.02185 (2016).

[10] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[11] Léon Bottou, Frank E Curtis, and Jorge Nocedal. "Optimization methods for large-scale machine learning". In: *Siam Review* 60.2 (2018), pp. 223–311.

[12] Marc Boullé. "Compression-based averaging of selective naive Bayes classifiers". In: *Journal of Machine Learning Research* 8.Jul (2007), pp. 1659–1685.

[13] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[14] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2002.

[15] Sébastien Bubeck. "Convex Optimization: Algorithms and Complexity". In: *Found. Trends Mach. Learn.* (2015). DOI: 10.1561/2200000050. URL: http://dx.doi.org/10.1561/2200000050.

[16] Giuseppe C. Calafiore and Giulia Fracastoro. "Sparse ℓ_1 and ℓ_2 Center Classifiers". In: *arXiv e-prints*, arXiv:1911.07320 (Nov. 2019), arXiv:1911.07320. arXiv: 1911.07320 [cs.LG].

[17] E. J. Candès and T. Tao. "Decoding by linear programming". In: *IEEE Transactions on Information Theory* 51.12 (2005), pp. 4203–4215.

[18] Miguel Carreira-Perpinan and Weiran Wang. "Distributed optimization of deeply nested systems". In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*. Ed. by Samuel Kaski and Jukka Corander. Vol. 33. Proceedings of Machine Learning Research. Reykjavik, Iceland: PMLR, 22–25 Apr 2014, pp. 10–19. URL: http://proceedings.mlr.press/v33/carreira-perpinan14.html.

[19] Miguel Carreira-Perpinan and Weiran Wang. "Distributed optimization of deeply nested systems". In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*. Ed. by Samuel Kaski and Jukka Corander. Vol. 33. Proceedings of Machine Learning Research. Reykjavik, Iceland: PMLR, 22–25 Apr 2014, pp. 10–19. URL: http://proceedings.mlr.press/v33/carreira-perpinan14.html.

[20] Miguel A Carreira-Perpinán and Mehdi Alizadeh. "ParMAC: distributed optimisation of nested functions, with application to learning binary autoencoders". In: *arXiv preprint arXiv:1605.09114* (2016).

[21] Pratik Chaudhari et al. "Entropy-sgd: Biasing gradient descent into wide valleys". In: *arXiv preprint arXiv:1611.01838* (2016).

[22] A. d'Aspremont, O. Banerjee, and L. El Ghaoui. "First-order methods for sparse covariance selection". In: *SIAM Journal on Matrix Analysis and Applications* 30.1 (2006), pp. 56–66.

[23] Yann N Dauphin et al. "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization". In: *Advances in neural information processing systems*. 2014, pp. 2933–2941.

[24] Yann N Dauphin et al. "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization". In: *Advances in neural information processing systems* 27 (2014).

[25] Marcel Dettling. "BagBoosting for tumor classification with gene expression data". In: *Bioinformatics* 20.18 (2004), pp. 3583–3593.

[26] Steven Diamond and Stephen Boyd. "CVXPY: A Python-embedded modeling language for convex optimization". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 2909–2913.

[27] D. L. Donoho. "Neighborly Polytopes and Sparse Solution of Underdetermined Linear Equations". In: *Stanford dept. of statistics working paper* (2004).

[28] John C. Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12 (2011). URL: http://dl.acm.org/citation.cfm?id=2021068.

[29] Ivar Ekeland and Roger Temam. *Convex analysis and variational problems*. SIAM, 1999.

[30] Tolga Ergen and Mert Pilanci. "Revealing the structure of deep neural networks via convex duality". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 3004–3014.

[31] D.C. Essen et al. "The Human Connectome Project: A data acquisition perspective". In: *NeuroImage* 62 (Feb. 2012), pp. 2222–31. DOI: 10.1016/j.neuroimage.2012.02.018.

[32] Rong-En Fan et al. "LIBLINEAR: A library for large linear classification". In: *Journal of machine learning research* 9.Aug (2008), pp. 1871–1874.

[33] François Fleuret. "Fast binary feature selection with conditional mutual information". In: *Journal of Machine learning research* 5.Nov (2004), pp. 1531–1555.

[34] Rina Foygel Barber and Emmanuel J Candès. "Controlling the false discovery rate via knockoffs". In: *The Annals of Statistics* 43.5 (Oct. 2015), pp. 2055–2085.

[35] Antonio Frangioni and Claudio Gentile. "Perspective cuts for a class of convex 0–1 mixed integer programs". In: *Mathematical Programming* 106.2 (2006), pp. 225–236.

[36] Eibe Frank, Mark Hall, and Bernhard Pfahringer. "Locally weighted naive Bayes". In: *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc. 2002, pp. 249–256.

[37] Nir Friedman, Dan Geiger, and Moises Goldszmidt. "Bayesian network classifiers". In: *Machine learning* 29.2-3 (1997), pp. 131–163.

[38] Jianjun Gao and Duan Li. "Optimal cardinality constrained portfolio selection". In: *Operations research* 61.3 (2013), pp. 745–761.

[39] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.

[40] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*. 2010.

[41] G.H. Golub and C.F. Van Loan. "Matrix Computation". In: *North Oxford Academic* (1990).

[42] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions". In: *SIAM review* 53.2 (2011), pp. 217–288.

[43] Ibrahim Abaker Targio Hashem et al. "The rise of "big data" on cloud computing: Review and open research issues". In: *Information systems* 47 (2015), pp. 98–115.

[44] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *CoRR* abs/1502.01852 (2015). URL: `http://arxiv.org/abs/1502.01852`.

[45] C. Helmberg et al. "An interior–point method for semidefinite programming". In: *SIAM Journal on Optimization* 6 (1996), pp. 342–361.

[46] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex Analysis and Minimization Algorithms*. Springer, 1993.

[47] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[48] Martin Jaggi. "Revisiting Frank-Wolfe: Projection-free sparse convex optimization". In: *International Conference on Machine Learning*. PMLR. 2013, pp. 427–435.

[49] Prateek Jain, Purushottam Kar, et al. "Non-convex optimization for machine learning". In: *Foundations and Trends® in Machine Learning* 10.3-4 (2017), pp. 142–363.

[50] Liangxiao Jiang et al. "Survey of improving naive Bayes for classification". In: *International Conference on Advanced Data Mining and Applications*. Springer. 2007, pp. 134–145.

[51] Sally C. Johnson. "Hierarchical clustering schemes". In: *Psychometrika* 32 (1967), pp. 241–254.

[52] Thomas Kerdreux, Igor Colin, and Alexandre d'Aspremont. "An Approximate Shapley-Folkman Theorem". In: *arXiv preprint arXiv:1712.08559* (2017).

[53] Dongmin Kim, Suvrit Sra, and Inderjit S Dhillon. "Fast Newton-type methods for the least squares nonnegative matrix approximation problem". In: *Proceedings of the 2007 SIAM international conference on data mining*. SIAM. 2007, pp. 343–354.

[54] Jingu Kim, Yunlong He, and Haesun Park. "Algorithms for nonnegative matrix and tensor factorizations: A unified view based on block coordinate descent framework". In: *Journal of Global Optimization* 58.2 (2014), pp. 285–319.

[55] Sang-Bum Kim et al. "Some effective techniques for naive Bayes text classification". In: *IEEE transactions on knowledge and data engineering* 18.11 (2006), pp. 1457–1466.

[56] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[57] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *International Conference on Learning Representations* (2015).

[58] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference for Learning Representations (ICLR)*. 2015.

[59] Anders Krogh and John A Hertz. "A simple weight decay can improve generalization". In: *Advances in neural information processing systems*. 1992, pp. 950–957.

[60] Guanghui Lan. *First-order and stochastic optimization methods for machine learning*. Springer, 2020.

[61] Tim Tsz-Kit Lau et al. "A Proximal Block Coordinate Descent Algorithm for Deep Neural Network Training". In: *Workshop track - International Conference on Learning Representations* (2018).

[62] Thomas Laurent and James Brecht. "Deep linear networks with arbitrary loss: All local minima are global". In: *International conference on machine learning*. PMLR. 2018, pp. 2902–2907.

[63] Yann LeCun and Corinna Cortes. "MNIST handwritten digit database". In: (2010). URL: http://yann.lecun.com/exdb/mnist/.

[64] Yann LeCun et al. "Effiicient BackProp". In: *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*. London, UK, UK: Springer-Verlag, 1998, pp. 9–50. ISBN: 3-540-65311-2. URL: http://dl.acm.org/citation.cfm?id=645754.668382.

[65] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[66] Olivier Ledoit and Michael Wolf. "A well-conditioned estimator for large-dimensional covariance matrices". In: *Journal of Multivariate Analysis* 88.2 (2004), pp. 365–411. ISSN: 0047-259X. DOI: https://doi.org/10.1016/S0047-259X(03)00096-4. URL: http://www.sciencedirect.com/science/article/pii/S0047259X03000964.

[67] Guohao Li et al. "Training graph neural networks with 1000 layers". In: *International conference on machine learning*. PMLR. 2021, pp. 6437–6449.

[68] Jia Li, Cong Fang, and Zhouchen Lin. "Lifted proximal operator machines". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 4181–4188.

[69] Yuanzhi Li, Colin Wei, and Tengyu Ma. "Towards explaining the regularization effect of initial large learning rate in training neural networks". In: *Advances in Neural Information Processing Systems* 32 (2019).

[70] Richard Maclin and Jude W. Shavlik. "Combining the Predictions of Multiple Classifiers: Using Competitive Learning to Initialize Neural Networks". In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*. IJCAI'95. Montreal, Quebec, Canada: Morgan Kaufmann Publishers Inc., 1995. ISBN: 1-55860-363-8, 978-1-558-60363-9. URL: `http://dl.acm.org/citation.cfm?id=1625855.1625924`.

[71] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[72] Dunja Mladenic and Marko Grobelnik. "Feature selection for unbalanced class distribution and naive Bayes". In: *ICML*. Vol. 99. 1999, pp. 258–267.

[73] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[74] B. K. Natarajan. "Sparse Approximate Solutions to Linear Systems". In: *SIAM J. Comput.* 24.2 (1995), pp. 227–234.

[75] Geoffrey Negiar, Armin Askari, and Laurent El Ghaoui. "OPTML 2017 : Lifted Neural Networks for Weight Initialization". In: (2017).

[76] Y. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming*. Philadelphia: Society for Industrial and Applied Mathematics, 1994.

[77] Behnam Neyshabur, Russ R Salakhutdinov, and Nati Srebro. "Path-sgd: Path-normalized optimization in deep neural networks". In: *Advances in neural information processing systems* 28 (2015).

[78] Chao Ning and Fengqi You. "Optimization under uncertainty in the era of big data and deep learning: When machine learning meets mathematical programming". In: *Computers & Chemical Engineering* 125 (2019), pp. 434–448.

[79] Brendan O'donoghue et al. "Conic optimization via operator splitting and homogeneous self-dual embedding". In: *Journal of Optimization Theory and Applications* 169.3 (2016), pp. 1042–1068.

[80] Jaehyun Park and Stephen Boyd. "A semidefinite programming method for integer convex quadratic minimization". In: *Optimization Letters* 12.3 (2018), pp. 499–518.

[81] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[82] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *Journal of machine learning research* 12.Oct (2011), pp. 2825–2830.

[83] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *the Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[84] Mert Pilanci and Martin J Wainwright. "Iterative Hessian sketch: Fast and accurate solution approximation for constrained least-squares". In: *The Journal of Machine Learning Research* 17.1 (2016).

[85] Mert Pilanci, Martin J Wainwright, and Laurent El Ghaoui. "Sparse learning via boolean relaxations". In: *Mathematical Programming* 151.1 (2015), pp. 63–87.

[86] R. T. Rockafellar. *Convex Analysis*. Princeton.: Princeton University Press., 1970.

[87] Mathias Seuret et al. "PCA-Initialized Deep Neural Networks Applied To Document Image Analysis". In: *CoRR* abs/1702.00177 (2017). URL: http://arxiv.org/abs/1702.00177.

[88] Maurice Sion. "On general minimax theorems." In: *Pacific Journal of mathematics* 8.1 (1958), pp. 171–176.

[89] Leslie N Smith. "Cyclical learning rates for training neural networks". In: *2017 IEEE winter conference on applications of computer vision (WACV)*. IEEE. 2017, pp. 464–472.

[90] Alex J Smola and SVN Vishwanathan. *LDL Factorization for Rank-K Modifications of Diagonal Matrices*. 2004.

[91] Emmanuel Soubies, Laure Blanc-Féraud, and Gilles Aubert. "A Continous Exact l0 penalty (CEL0) for least squares regularized problem". In: *SIAM J. Imaging Sci* 8 (2015), pp. 1574–1606.

[92] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[93] Ross M Starr. "Quasi-equilibria in markets with non-convex preferences". In: *Econometrica: journal of the Econometric Society* (1969), pp. 25–38.

[94] Ilya Sutskever et al. "On the Importance of Initialization and Momentum in Deep Learning". In: *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*. ICML'13. 2013.

[95] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. 1st. Cambridge, MA, USA: MIT Press, 1998. ISBN: 0262193981.

[96] Gavin Taylor et al. "Training Neural Networks Without Gradients: A Scalable ADMM Approach". In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. New York, NY, USA: JMLR.org, 2016, pp. 2722–2731. URL: http://dl.acm.org/citation.cfm?id=3045390.3045677.

[97] Gavin Taylor et al. "Training neural networks without gradients: A scalable admm approach". In: *International Conference on Machine Learning*. 2016, pp. 2722–2731.

[98] Bertrand Thirion et al. "Which fMRI clustering gives good brain parcellations?" In: *Frontiers in neuroscience* 8 (July 2014), p. 167. DOI: 10.3389/fnins.2014.00167.

[99] Robert Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[100] Shashanka Ubaru, Jie Chen, and Yousef Saad. "Fast Estimation of $tr(f(A))$ via Stochastic Lanczos Quadrature". In: *SIAM Journal on Matrix Analysis and Applications* 38 (Jan. 2017), pp. 1075–1099. DOI: 10.1137/16M1104974.

[101] Madeleine Udell and Stephen Boyd. "Bounding duality gap for separable problems with linear constraints". In: *Computational Optimization and Applications* 64.2 (2016), pp. 355–378.

[102] Madeleine Udell and Alex Townsend. "Why are big data matrices approximately low rank?" In: *SIAM Journal on Mathematics of Data Science* 1.1 (2019), pp. 144–160.

[103] Geoffrey I Webb, Janice R Boughton, and Zhihai Wang. "Not so naive Bayes: aggregating one-dependence estimators". In: *Machine learning* 58.1 (2005), pp. 5–24.

[104] Linchuan Wei, Andrés Gómez, and Simge Küçükyavuz. "On the convexification of constrained quadratic optimization problems with indicator variables". In: *International Conference on Integer Programming and Combinatorial Optimization*. Springer. 2020, pp. 433–447.

[105] Z. Wen et al. *Row by row methods for semidefinite programming*. Tech. rep. Technical report, Department of IEOR, Columbia University, 2009.

[106] Zaiwen Wen, Donald Goldfarb, and Katya Scheinberg. "Block coordinate descent methods for semidefinite programming". In: *Handbook on semidefinite, conic and polynomial optimization*. Springer, 2012, pp. 533–564.

[107] Ashia C. Wilson et al. "The Marginal Value of Adaptive Gradient Methods in Machine Learning". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 2017, pp. 4151–4161.

[108] David P Woodruff et al. "Sketching as a tool for numerical linear algebra". In: *Foundations and Trends® in Theoretical Computer Science* 10.1–2 (2014), pp. 1–157.

[109] Weijun Xie and Xinwei Deng. "Scalable Algorithms for the Sparse Ridge Regression". In: *arXiv preprint arXiv:1806.03756* (2018).

[110] E. L. Yip. "A Note on the Stability of Solving a Rank-p Modification of a Linear System by the Sherman–Morrison–Woodbury Formula". In: *SIAM Journal on Scientific and Statistical Computing* 7.2 (1986), pp. 507–513. DOI: 10.1137/0907034.

[111] Alp Yurtsever et al. "Sketchy decisions: Convex low-rank matrix optimization with optimal storage". In: *arXiv preprint arXiv:1702.06838* (2017).

[112] Nayyar A Zaidi et al. "Alleviating naive Bayes attribute independence assumption by attribute weighting". In: *The Journal of Machine Learning Research* 14.1 (2013), pp. 1947–1988.

[113] Yuchen Zhang, Percy Liang, and Martin J Wainwright. "Convexified convolutional neural networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 4044–4053.

[114] Ziming Zhang and Matthew Brand. "Convergent block coordinate descent for training tikhonov regularized deep neural networks". In: *Advances in Neural Information Processing Systems*. 2017, pp. 1721–1730.

[115] Ziming Zhang, Yuting Chen, and Venkatesh Saligrama. "Efficient training of very deep neural networks for supervised hashing". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 1487–1495.

[116] Zijian Zheng and Geoffrey I Webb. "Lazy learning of Bayesian rules". In: *Machine Learning* 41.1 (2000), pp. 53–84.