Convex Latent Representation Learning with Generalized Invariance

by

Vignesh Ganapathiraman
B.E., Anna University, India, 2007
M.Sc.,Chennai Mathematical Institute, India, 2012

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2020

Chicago, Illinois

Defense Committee:
Prof. Xinhua Zhang, Chair and Advisor
Prof. Brian Ziebart
Prof. Bing Liu
Prof. Lev Reyzin, Mathematics and Computer Science, UIC
Prof. Dale Schuurmans (University of Alberta)

To Varshu and Adhwaith.

ACKNOWLEDGMENT

I have been very fortunate to have Prof. Xinhua Zhang as my mentor and advisor. It has been a privilege to witness and learn from his commitment to research at the highest standard and excellent work ethic. Prof. Zhang gave me just the right platform for exploring the world of optimization and machine learning. I'm grateful for his constant support and guidance, especially during trying times, throughout my Ph.D. study.

I thank my thesis committee members, Prof. Bing Liu, Prof. Brian Ziebart, Prof. Lev Reyzin and Prof. Dale Schuurmans for their valuable feedback and insights on my work. I would like to thank Prof. Elena Zheleva for her support and guidance. I thank my lab mates and all my friends for their collaborations and illuminating discussions.

I would like to thank my mother and sister for their love and encouragement. Finally, I would like to thank my wife Amrutha for believing in me and providing her unconditional support in all my career pursuits.

VG

TABLE OF CONTENTS

CHAPTER PAGE

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Finding representations of data that are useful for the underlying prediction task has been an important and active pursuit in machine learning. Modern-day deep learning algorithms have been hugely successful in making accurate data-driven predictions by **automatically learning** highly discriminative representations of data via multiple nonlinear transformations. In addition to learning discriminative representations, it is also useful to encode problem / task specific information (priors) directly in the representations. For instance, in image classification, it is useful to enforce the constraint that a machine learning model's prediction should not change when the image is perturbed spatially *a.k.a* translation invariance. It is well known in the computer vision community that convolutional neural networks implicitly enforce translational invariance via its parametric pooling layers. Likewise there are priors, such as group priors, that are commonly employed in vision and and successfully incorporated via parametric machine learning models.

However there are a variety of data / problem structures that do not admit a natural parameterization. Fortunately, besides parametric approaches another way to realize non-trivial structured priors in a model is via carefully designed losses and regularizers. However regularization approaches result in a hard non-convex optimization problem. Non-convex optimization problems often pose serious challenges in training and seldom results in guaranteed optimal solutions (global optima). This further creates a theoretical gap in understanding when these models are guaranteed to work. In stark contrast, convex optimization problems are globally

optimal by definition. Approximations to computationally hard convex problems are also well studied and are often endowed with convergence and approximation guarantees.

In this thesis, we address the above challenges in representation learning by: (1) exploring priors that encode non-trivial problem / data specific structures (2) designing efficient convex models and training algorithms to automatically learn these structures in a data-dependent fashion (3) providing learning and approximation guarantees wherever possible. Our first approach towards convex representation learning is to explicitly model structured priors in the latent layer of a two-layer neural network. The resulting non-convex optimization problem is then relaxed to obtain a convex model that is able to obtain all the structural regularities of the original non-convex problem. Second, we develop a new convex representation learning framework, based on semi-inner-product spaces, to model the so-called generalized invariances in an efficient and scalable manner.

CHAPTER 1

INTRODUCTION

Artificial intelligence (AI) and machine learning (ML) are in the forefront of many technological advancements in the recent years. Recently Deep Learning (DL) systems have reached superior performance in many data-driven learning tasks such as object detection [2], natural language processing [3], speech recognition [4]. Much of the success of recent ML algorithms can be attributed to the availability of large volumes of *labeled* training data, sophisticated computing systems such as Graphics Processing Units (GPU) and Tensor Processing Units (TPU) and other advanced data processing software systems. Modern-day machine learning algorithms such as deep nets are designed to automatically learn patterns in the input, by learning representations of data that are "predictive" for the given task - giving them the ability to accurately predict outcomes from previously "unseen" data.

In addition to capturing discriminative representations, it is now well known that machine learning problems naturally exhibit additional structures that can be effectively leveraged as information priors. For instance, spam filters are supposed to be stable under feature deletions, additions, and replacements, while detectors in image processing and computer vision are expected to deliver invariant response under transformations such as translation, rotation, scaling, etc. Manifold priors assume flat curvature or gradient of discriminant function in the vicinity of data samples, or in the direction of nearest-neighbor instances on a graph [5].

Incorporating problem-specific information in modeling has numerous benefits. A machine learning model that is endowed with accurate specifications of these structures sometimes enjoy improved predictive performance, better resilience to noises in data and problem specifications.

How do we learn these structured priors?

Researchers in the machine learning community have studied a variety of methods for making use of prior information. Carefully designed parametric functions have been used sometimes to model priors such as invariance. For instance convolutional neural networks have pooling layers that implicitly enforce translational invariance. But these approaches are restricted to priors that can be expressed in a parametric and decomposable manner. A lot of useful priors in machine learning don't enjoy such properties. Alternatively, in order to model these priors, a large body of existing methods first specify a given space of prediction functions and the representation of data (e.g., through the selection of kernels), and then use the priors to bias the search for the optimal predictor through loss functions and regularizers. These approaches assume the existence of a representation of data. However, modern wisdom discovers that it is effective to actually learn these representations from data (a technique employed by state-of-the-art deep models). More often, structured priors are latent in the data and can be expressed quite well using latent variable models. Latent variable models are quite flexible in practice and can model a wide variety of priors. Unfortunately latent variable models are usually hard to train as they mostly result in non-convex optimization problems.

Challenges in training deep latent variable models.

Despite their advantages in modeling and success in applications, latent models remain hard to train. The key challenge originates from the coupling of model parameter learning and latent variable inference, which in general leads to a non-convex optimization problem. Although empirical performance has been the major focus of deep learning, recently substantial progress has been made towards the analysis of global training and the structure of the optimization problem. For example, [6] and [7] showed that the lowest critical values of the random loss function are close to the global minimum, and [8] showed, under certain assumptions, that every local minimum is a global minimum for an expected loss function of a deep nonlinear neural network. Similar global trainability results have been derived for gradient descent on two-node ReLU networks [9], quadratic activations [10], and one-hidden-layer non-overlapping convolution nets [11]. The global minima in over-parameterized settings were characterized on deep and wide nets and convolutional nets [12, 13]. However most analyses are still limited, especially with assumptions on the model and data distribution that are hard to verify in practice.

Similar global trainability results have been derived for gradient descent on two-node ReLU networks [9], quadratic activations [10], and one-hidden-layer non-overlapping convolution nets [11]. The global minima in over-parametrized settings were characterized on deep and wide nets and convolutional nets [12, 13]. However most analyses are still limited, especially with assumptions on the model and data distribution that are hard to verify in practice.

Along a different line of methodology, *reformulations* of latent models have been studied which admit tractable global solutions. Examples include boosting [14], spectral methods [15, 16], kernel methods [17, 18], polynomial networks and sum-product networks [19, 20], and semidefinite relaxations [21]. Unfortunately, they either impose restrictions on the model space (*e.g.* polynomial network, recursive inverse kernels), or require tractability of underlying oracles, or rely on realizability assumptions. In a nutshell, global training of latent variable models is a hard problem. Approximation solutions are still too restrictive and limit the applicability of the model.

This thesis addresses these issues by proposing new models for learning representations of data that encode the structured priors accurately and in a manner that is amenable for optimization and training. In the first half of this thesis, we explore new latent variable models for learning predictive structured priors and address the hardness in their training and inference because of their non-convexity. In the remainder of the thesis, we present new representation learning techniques for modeling generalized invariance priors.

1.1 Organization of this thesis

The main contributions of this thesis are divided into the following 4 chapters.

- Inductive two-layer modeling with parametric bregman Transfer
- Incorporating latent structure information
- Convex representation learning for generalized invariance in semi-inner-product spaces
- Representation learning for mitigating catastrophic forgetting in deep neural networks

Inductive two-layer modeling with parametric Bregman transfer

We first begin by addressing the global trainability issues of a general two-layer neural network with a parametric activation function in its latent layer [22]. The objective function of this simple neural network is non-convex. Convex relaxations of this setup has been studied earlier via semi-definite relaxations. However, the resulting convex model is nonparametric and results in a transductive learning setting making the inference inefficient. In this work, we develop a novel convex model of the original non-convex objective without losing its inductive benefits. Additionally, we also design an efficient optimization method to solve the relaxed objective, with a constant approximation bound.

Incorporating latent structure information in two-layer neural networks

In this chapter we explore ways to incorporate structured priors in the latent layers of two-layered neural network [23]. Prior works in this setting make use of a conditional random field based auto-encoder that automatically infers latent structure information. However, besides being non-convex, this method requires a demanding inference. In this work we first propose a conditional model that captures the problem structure in its latent layers. We then develop a convex relaxation of the proposed two-layer model. The relaxed model further admits a much more efficient inference compared to earlier methods. The flexibility of the model is demonstrated using experiments on transliteration and image inpainting tasks.

Convex representation learning for generalized invariance in semi-inner-product spaces

We next focus on modeling more general representations that are useful for practical predictions [24]. Invariance (defined in a general sense) has been one of the most effective priors for representation learning. Direct factorization of parametric models is feasible only for a small range of invariances, while regularization approaches, despite improved generality, lead to non-convex optimization. In this work, we develop a convex representation learning algorithm for a variety of generalized invariances that can be modeled as semi-norms. Novel Euclidean embeddings are introduced for kernel representers in a semi-inner-product space, and approximation bounds are established. This allows invariant representations to be learned efficiently and effectively as confirmed in our experiments, along with accurate predictions.

Representation learning for mitigating catastrophic forgetting in deep neural networks

Finally, we challenge our novel general representation learning framework developed in Chapter 4 to learn in the continual learning framework. Specifically, we design new regularizers in the semi-inner-produce space to overcome the so-called catastrophic forgetting for neural networks. Experimental results confirm that the learned representations enable the downstream classifier to overcome catastrophic forgetting when used in addition to existing continual learning algorithms.

CHAPTER 2

INDUCTIVE TWO-LAYER MODELING WITH PARAMETRIC BREGMAN TRANSFER

(Parts of this chapter were previously published as "Inductive two-layer modeling with parametric bregman transfer" [22] at the International Conference on Machine Learning (ICML 2018))

2.1 Introduction

In the previous section, we discussed why globally trainable models are desirable for analysis and why it is hard to obtain globally trainable non-convex models. We introduced deep non-linear models, which are highly popular due to its large model capacity (informally, the ability of the model to capture complex relationships in data) and incredible ability to generalize to "unseen examples". However these models are highly *non-convex latent variable models*, which makes their analysis hard. We also briefly saw how convex models are globally trainable, by definition, and are naturally capable of producing solutions that are easier to analyze in general. Therefore, one natural way to address the issue of hardness in the analysis of non-convex latent variable objectives (models) is to somehow convert them into convex models and analyze them using tools from convex optimization. In this section, we show a method to develop a convex latent variable model, which is a reformulation of a non-convex *two-layer latent variable model* with a non-linear activation function. We also show an efficient optimization procedure to solve

the resulting convex objective, which comes with a constant approximation guarantee. This work was published at the *International Conference on Machine Learning (ICML 2018)* [22].

2.1.1 Problem setting.

In this work, we consider two-layer neural networks of the form $X \rightarrow H \rightarrow Y$, where X is the input layer, H is the hidden layer and Y is the output layer. Two-layer neural networks are composed of two nonlinear conditional models. The latent layer is characterized by a nonlinear transfer function $\mathbf{f} : \mathbb{R}^h \rightarrow \mathbb{R}^h$, which converts the linear transformation $W\mathbf{x}$ into $\boldsymbol{\phi} = \mathbf{f}(W\mathbf{x})$. Here $\mathbf{x} \in \mathbb{R}^n$ is the raw input feature, and $W \in \mathbb{R}^{h \times n}$ is the hidden layer weights. The resulting $\boldsymbol{\phi}$ is further multiplied with the output layer weights $U \in \mathbb{R}^{h \times m}$, and the product is measured against the given label $\mathbf{y}$ via a loss function $\ell(U'\boldsymbol{\phi}, \mathbf{y})$. Here U' is the transpose of U. Typical losses include binary hinge loss $\ell(z, y) = [1 - yz]_+$ with $m = 1$. Here $y \in \{-1, 1\}$ and $[z]_+ := \max\{0, z\}$. For multiclass problems with C classes, $\mathbf{y}$ encodes a class c with the canonical vector $\mathbf{e}_c$. Then $m = C$ and the hinge loss $\ell(\mathbf{z}, \mathbf{y}) = \max\{\mathbf{1} - \mathbf{y} + \mathbf{z} - (\mathbf{y}'\mathbf{z})\mathbf{1}\}$, where $\mathbf{1}$ is a vector of all one's. The logistic loss is $-\mathbf{z}'\mathbf{y} + \log \sum_c \exp(z_c)$.

There are several popularly used transfer functions. The simplest options are elementwise, *i.e.* $\mathbf{f}(\mathbf{z}) = (f(z_1), \ldots, f(z_h))'$, where all z_i are applied separately to the same function f: $\mathbb{R} \rightarrow \mathbb{R}$. The rectified linear function uses $f_r(z) = [z]_+$ (we use regular lowercase letters for scalar, bold lowercase letters for vector, and capital letters for matrix). Variants include the leaky rectifier which uses $f_l(z) = \max\{z, az\}$ where $a > 0$ is a small positive number, and the bounded hard tanh which uses $f_h(z) = \max\{-1, \min\{z, 1\}\}$. Transfers that are not piecewise

linear are also available, *e.g.* the sigmoid $f_s(z) = (1 + e^{-z})^{-1}$. An illustration of these transfers is shown in Figure 1.

2.1.2 Matching Loss for Transfer Functions

A major source of non-convexity in neural network is the nonlinear transfer function. Relaxations have been proposed that uses a loss function to indirectly induce the nonlinear relationship $\boldsymbol{\phi} = \mathbf{f}(\mathbf{z})$. Formally speaking, it constructs a loss $L(\boldsymbol{\phi}, \mathbf{z})$ that would (ideally) satisfy three conditions:

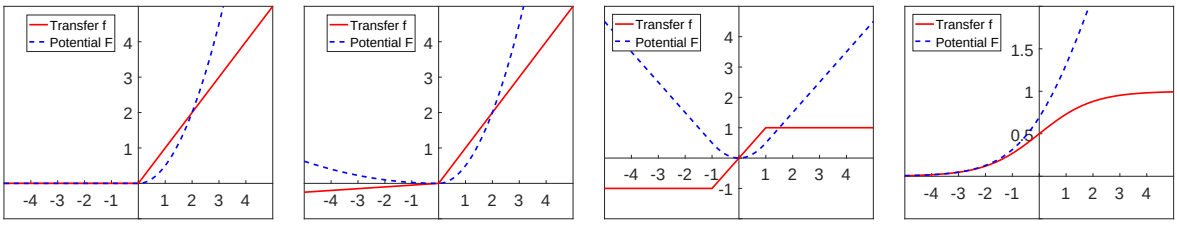
- *Unique recovery*: $\arg\min_{\boldsymbol{\phi}} L(\boldsymbol{\phi}, \mathbf{z}) = \mathbf{f}(\mathbf{z})$ for all $\mathbf{z}$, with the $\arg\min$ attained *uniquely*.
- *Joint convexity*: L is jointly convex over $\boldsymbol{\phi}$ and $\mathbf{z}$. This is required if we choose to build a jointly convex deep model by directly using L to connect the input and output of adjacent layers.
- *Grounding*: $\min_{\boldsymbol{\phi}} L(\boldsymbol{\phi}, \mathbf{z}) = 0$ for all $\mathbf{z}$, so that there is no bias towards any value of $\mathbf{z}$.

Unfortunately, it can be shown that such a loss does not exist, unless $\mathbf{f}$ is affine.

Theorem 1. *There exists a loss L that satisfies all the three conditions if, and only if, $\mathbf{f}$ is affine.*

The proof can be found in Appendix B.

This result motivates us to resort to weaker versions of loss. Interestingly, the matching loss [25] meets the first and third conditions, and satisfies a weakened version of convexity by imposing a very mild condition on $\mathbf{f}$. In particular, we assume that the transfer function is the gradient of a *strictly convex* function $F : \mathbf{f} = \nabla F$. Here F maps from $\mathbb{R}^h$ to $\mathbb{R}$. If $\mathbf{f}$



(a) Linear rectifier (ReLU) (b) Leaky rectifier $\epsilon = .05$ (c) Hard tanh (d) Sigmoid

Figure 1: Four examples of transfer function f and the corresponding potential function F

is elementwise, this just means the constituent f is continuous and strictly increasing. As a result, the inverse of $\mathbf{f}$ also exists, and it is well known that $\mathbf{f}^{-1} = \nabla F^*$, where F^* is the Fenchel conjugate of F.

Although a rectifier $f_r(z)$ is not strictly increasing in the negative half line, it can be approximated arbitrarily closely via $\max\{\epsilon z, z\}$ for infinitesimally small $\epsilon > 0$. Similar alternations can be applied to hard tanh $f_h(z)$ by allowing a tiny slope ϵ for $|z| \geq 1$. The F corresponding to the abovementioned transfers f are also shown in Figure 1.

In the case that $\mathbf{f}$ is not elementwise, this assumption of F implies: 1) $\mathbf{f}$ is strictly increasing in the vector sense: $(\mathbf{x} - \mathbf{y})'(\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y})) > 0$, and 2) The Jabobian of $\mathbf{f}$ is symmetric (as the Hessian of F): $J\mathbf{f} = (J\mathbf{f})'$, provided $\mathbf{f}$ is differentiable. In this paper, we only consider elementwise transfers. Under this assumption, we adopt the following loss function based on Bregman divergence:

$$L(\boldsymbol{\phi}, \mathbf{z}) = D_{F^*}(\boldsymbol{\phi}, \mathbf{f}(\mathbf{z})) = F^*(\boldsymbol{\phi}) + F(\mathbf{z}) - \boldsymbol{\phi}'\mathbf{z}. \tag{2.1}$$

Here D_{F^*} is the Bregman divergence induced by F^*. Obviously L meets the conditions of recovery and grounding, but is *not* jointly convex. However, the only non-convex part is the bilinear term $\boldsymbol{\phi}'\mathbf{z}$, while both F^* and F are convex. Such a decoupling of non-convex terms from the transfer functions is the key enabler for our convexification technique.

2.1.3 Convex Two-layer Modeling

Suppose we are given t training pairs $\{(\mathbf{x}_j, \mathbf{y}_j)\}_{j=1}^t$, stacked in two matrices $X = (\mathbf{x}_1, \ldots, \mathbf{x}_t) \in \mathbb{R}^{n\times t}$ and $Y = (\mathbf{y}_1, \ldots, \mathbf{y}_t) \in \mathbb{R}^{m\times t}$. The corresponding set of latent layer outputs are stacked into $\Phi = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_t) \in \mathbb{R}^{h\times t}$. The regularized risk minimization objective can be written as

$$\min_{W,U,\mathbf{b},\Phi} \sum_{j=1}^{t} D_{F^*}(\boldsymbol{\phi}_j, \mathbf{f}(W\mathbf{x}_j)) + \ell(U'\boldsymbol{\phi}_j + \mathbf{b}, \mathbf{y}_j) + \frac{1}{2}\|W\|^2 + \frac{1}{2}\|U\|^2 \tag{2.2}$$

$$= \min_{W,U,\mathbf{b},\Phi} \sum_{j=1}^{t} \{F^*(\boldsymbol{\phi}_j) - \boldsymbol{\phi}_j' W\mathbf{x}_j + F(W\mathbf{x}_j) + \ell_j(U'\boldsymbol{\phi}_j + \mathbf{b})\} + \tfrac{1}{2}\|W\|^2 + \tfrac{1}{2}\|U\|^2, \tag{2.3}$$

where $\ell_j(U'\boldsymbol{\phi}_j + \mathbf{b}) := \ell(U'\boldsymbol{\phi}_j + \mathbf{b}, \mathbf{y}_j)$. We introduced regularizations via Frobenius norm squares. The weight of both regularization terms can be tuned by any model selection method, *e.g.* cross validation, and here we put 1 to simplify the presentation. We also assume that dom ℓ_j is the entire space. To keep our notation neat we write functions on matrices, representing the sum of the function values applied to each column, *e.g.* $F^*(\Phi) = \sum_j F^*(\boldsymbol{\phi}_j)$. Now we can rewrite the objective as

$$\min_{\Phi,W,U,\mathbf{b}} F^*(\Phi) - \operatorname{tr}(\Phi' WX) + F(WX) + \ell(U'\Phi + \mathbf{b}\mathbf{1}') + \tfrac{1}{2}\|W\|^2 + \tfrac{1}{2}\|U\|^2. \tag{2.4}$$

It is bi-convex in two groups of variables $(\Phi, \mathbf{b})$ and (W, U), *i.e.* fixing one group it is convex in the other. In order to derive a jointly convex reformulation, we first note that $\ell(U'\Phi + \mathbf{b}\mathbf{1}') = \max_R \operatorname{tr}(R'(U'\Phi + \mathbf{b}\mathbf{1}')) - \ell^*(R)$, where ℓ^* is the Fenchel conjugate of ℓ, and $R \in \mathbb{R}^{m\times t}$. For binary hinge loss, $\ell^*(r) = yr$ over $r \in [\min\{0, -y\}, \max\{0, -y\}]$, and ∞ else. For multiclass hinge loss, $\ell^*(\mathbf{r}) = \mathbf{y}'\mathbf{r}$ if $\mathbf{r} + \mathbf{y} \in \Delta^m := \{\mathbf{x} \in \mathbb{R}^m_+ : \mathbf{1}'\mathbf{x} = 1\}$, and ∞ else. For multiclass logistic loss, $\ell^*(\mathbf{r}) = \sum_i (r_i + y_i)\log(r_i + y_i)$ if $\mathbf{r} + \mathbf{y} \in \Delta^m$, and ∞ else. Similarly, $F(WX) = \max_\Lambda \operatorname{tr}(\Lambda' WX) - F^*(\Lambda)$. So we can rewrite (Equation 2.3) into

$$\min_{W,U,\mathbf{b},\Phi} \max_{R,\Lambda} \quad F^*(\Phi) - \operatorname{tr}(\Phi' WX) + \operatorname{tr}(\Lambda' WX) - F^*(\Lambda) + \operatorname{tr}(R'(U'\Phi + \mathbf{b}\mathbf{1}')) - \ell^*(R) + \tfrac{\|W\|^2 + \|U\|^2}{2} \tag{2.5}$$

$$= \min_{\Phi} \max_{R,\Lambda} \min_{W,U,\mathbf{b}} \quad F^*(\Phi) - \operatorname{tr}(\Phi' WX) + \operatorname{tr}(\Lambda' WX) - F^*(\Lambda) + \operatorname{tr}(R'(U'\Phi + \mathbf{b}\mathbf{1}')) - \ell^*(R) + \tfrac{\|W\|^2 + \|U\|^2}{2} \tag{2.6}$$

$$= \min_{\Phi} \max_{R\mathbf{1}=\mathbf{0},\Lambda} \quad F^*(\Phi) - \frac{1}{2}\left\|(\Phi - \Lambda)X'\right\|^2 - \frac{1}{2}\left\|\Phi R'\right\|^2 - F^*(\Lambda) - \ell^*(R). \tag{2.7}$$

The optimal W and U for the last equality is $W = (\Phi + \Lambda)X'$ and $U = -\Phi R'$. The first equality swaps $\min_{W,U}$ with $\max_{R,\Lambda}$. Such a strong duality is indeed not trivial because the celebrated Sion's minimax lemma requires that the domain of (W, U) be compact, which is *not* assumed here. So the above "derivation" is not rigorous. However the conclusion is still correct thanks to the strong convexity of $\|\cdot\|^2$.

Theorem 2. *For any* $W, U, \mathbf{b}$*, denote* $\mathcal{L}(\Phi, R) = F^*(\Phi) - \operatorname{tr}(\Phi' W X) + \operatorname{tr}(R'(U'\Phi + \mathbf{b}\mathbf{1}')) - \ell^*(R)$*. Then*

$$\min_{\Phi} \max_{R} \mathcal{L}(\Phi, R) \;=\; \max_{R} \min_{\Phi} \mathcal{L}(\Phi, R).$$

See the proof in Appendix B.

2.2 Convex relaxation

We now derive a convex relaxation for (Equation 2.7). To be concrete, consider the linear rectifier with $F_r(Z) = \frac{1}{2}\left\|[Z]_+\right\|^2$. Its Fenchel dual is $F_r^*(\Phi) = \frac{1}{2}\left\|\Phi\right\|^2$ for $\Phi \geq \mathbf{0}$ (elementwise), and $+\infty$ otherwise. Therefore (Equation 2.7) can be specialized into

$$\min_{\Phi \geq \mathbf{0}} \max_{R\mathbf{1}=\mathbf{0}, \Lambda \geq \mathbf{0}} \; \frac{1}{2}\left\|\Phi\right\|^2 - \frac{1}{2}\left\|(\Phi - \Lambda)X'\right\|^2 - \frac{1}{2}\left\|\Phi R'\right\|^2 - \frac{1}{2}\left\|\Lambda\right\|^2 - \ell^*(R). \tag{2.8}$$

Notice that both Φ and Λ are constrained to the positive orthant, and they are both sized $h \times t$. Since $t \gg h$ in general, their ranks are h and their column spaces have full rank. As a result, we may perform change of variable via $\Lambda = \Phi A$, where $A \in \mathbb{R}_+^{t \times t}$ and is not necessarily symmetric. So we can rewrite (Equation 2.8) as

$$\min_{\Phi \geq \mathbf{0}} \max_{R\mathbf{1}=\mathbf{0}, A \geq \mathbf{0}} \; \frac{1}{2}\left\|\Phi\right\|^2 - \frac{1}{2}\operatorname{tr}(\Phi'\Phi(I - A)X'X(I - A'))\frac{1}{2}\operatorname{tr}(\Phi'\Phi R'R) - \frac{1}{2}\operatorname{tr}(\Phi'\Phi AA') - \ell^*(R).$$

Although this is still not convex, all occurrences of Φ are now in the form of $\Phi'\Phi$. This leads to the natural idea of optimizing over $\Phi'\Phi$ directly. Denote $T := \Phi'\Phi \in \mathbb{R}^{t\times t}$, and then we finally arrive at

$$\min_{T\in\mathcal{T}_h} \max_{R\mathbf{1}=\mathbf{0},A\geq\mathbf{0}} \frac{1}{2}\operatorname{tr}(T) - \frac{1}{2}\operatorname{tr}(T(I-A)X'X(I-A')) - \frac{1}{2}\operatorname{tr}(TR'R) - \frac{1}{2}\operatorname{tr}(TAA') - \ell^*(R), \tag{2.9}$$

where $\mathcal{T}_h := \left\{\Phi'\Phi : \Phi \in \mathbb{R}_+^{h\times t}\right\} \subseteq \left\{T \in \mathbb{R}_+^{t\times t} : T \succeq \mathbf{0}\right\}$. $T \succeq \mathbf{0}$ means T is positive semi-definite (PSD). Now given $\mathbf{b}$ and T, the maximization over R and A is concave because $T \succeq \mathbf{0}$. Indeed A and R are decoupled, making the inner optimization efficient. The objective function is also convex in $\mathbf{b}$ and T, because maximization over linear terms gives a convex function. The only challenge left is the non-convexity of $\mathcal{T}_h$.

The set $\mathcal{T}_h$ is obviously a cone. In fact, if we relax the fixed value of h, then $\mathcal{T}_\infty$ is the well-known *completely positive* (CP) matrix cone [26]. More interestingly, it is not hard to show that $\mathcal{T}_\infty$ is the tightest convex relaxation of $\mathcal{T}_h$, *i.e.* the convex hull of $\mathcal{T}_h$ for any h. Denoting $\mathcal{T} := \mathcal{T}_\infty$, our final objective becomes.

$$\min_{T\in\mathcal{T}} \max_{R\mathbf{1}=\mathbf{0},A\geq\mathbf{0}} \frac{1}{2}\operatorname{tr}(T) - \frac{1}{2}\operatorname{tr}(T(I-A)X'X(I-A')) - \frac{1}{2}\operatorname{tr}(TR'R) - \frac{1}{2}\operatorname{tr}(TAA') - \ell^*(R). \tag{2.10}$$

It turns out that the convex relaxation does not require a pre-specified number of hidden nodes h, and is able to figure it out automatically through the rank of the optimal T. We will observe

in the sequel that the formulation does implicitly favor a low-rank solution through a gauge regularizer (Lemma 1), although a manual assignment of h can always be incorporated through truncation after optimization.

2.2.1 Generality of the convexification scheme.

We note in passing that the above technique is general. For example, when using the hard tanh transfer, we have $F_h^*(\Phi) = \frac{1}{2}\|\Phi\|^2$ if the L_∞ norm $\|\Phi\|_\infty := \max_{ij}|\Phi_{ij}| \le 1$, and ∞ otherwise. Then we get the same objective function as in (Equation 2.10), only with $\mathcal{T}_h$ changed into $\{\Phi'\Phi : \|\Phi\|_\infty \le 1\}$ and the domain of A changed into $\{A : \sum_i |A_{ij}| \le 1,\ \forall\ j\}$.

2.3 Optimization

Although the problem (Equation 2.10) is convex, the set $\mathcal{T}$ lacks a compact characterization in terms of linear/quadratic, PSD, or second-order conic constraints. Optimization over completely positive matrices is known hard [26], and even projection to $\mathcal{T}$ is NP-hard [27].[1] Therefore we resort to conditional gradient (Frank-Wolfe) methods that are free of projection [28, 29]. The key benefit of CG lies in the efficiency of optimizing a linear function over $\mathcal{T}$ (a.k.a. the polar operator), robustness in its inaccuracy [30], and the low rank of intermediate solutions due to its greedy and progressive nature (hence efficient intermediate updates).

In practice, however, CG still suffers from slow convergence, and its linearly-converging variants are typically subject to a large condition number [31]. This is partly because at each step only the weights on the existing bases are optimized, while the bases themselves are not.

---

[1] Despite "convexity", a convex function may itself be NP-hard to evaluate, and it can be NP-hard to project to a convex set, or optimize a linear function over it.

To alleviate this problem, [32] proposed the Generalized Conditional Gradient algorithm (GCG) which simultaneously optimizes the bases. Despite the lack of theoretical proof, it is much faster in practice. Furthermore, GCG is robust to inexactness in polar operators, and one of our key contributions below is to show that it can efficiently solve (Equation 2.10) with a multiplicative approximation bound of $\frac{1}{4}$.

Since GCG operates on gauge regularized objectives, our first step is to take a nontrivial path of rewriting (Equation 2.10). Recall that given a convex bounded set C containing the origin, the gauge function induced by C evaluated at T is defined as $\gamma_C(T) := \min\{\gamma \geq 0 : \gamma X = T,\ X \in C\}$. If no such (γ, X) meets the condition, then $\gamma_C(T) := \infty$. Since (Equation 2.10) does not contain a gauge function induced by a bounded set ($\mathcal{T}$ is unbounded), we first recast it into this framework.

The simplest way to add bound to $\mathcal{T}$ is via the trace norm, which is exactly $\mathrm{tr}(T)$ since $T \succeq \mathbf{0}$:

$$\mathcal{S} := \mathcal{T} \cap \{T : \mathrm{tr}(T) \leq 1\} \tag{2.11}$$

$$= \mathrm{conv}\mathcal{T}_1 \cap \{T : \mathrm{tr}(T) \leq 1\} \tag{2.12}$$

$$= \mathrm{conv}\left\{\mathbf{x}\mathbf{x}' : \mathbf{x} \in \mathbb{R}^t_+, \|\mathbf{x}\| \leq 1\right\}. \tag{2.13}$$

Our key observation is the following lemma which allows us to rewrite the problem in terms of gauge regularized objective. In particular, the domain of the gauge implicitly enforces the constraint on T.

Lemma 1. *$\mathcal{S}$ is convex, bounded, and closed. In addition*

$$\gamma_{\mathcal{S}}(T) = \begin{cases} \mathrm{tr}(T) & T \in \mathcal{T} \\ \\ +\infty & \textit{otherwise} \end{cases}. \tag{2.14}$$

The proof is relegated to Appendix B.

In fact, it is easy to show that for any convex cone C, the gauge function of its intersection with a half-space $\mathrm{tr}(A'T) \leq 1$ is exactly $\mathrm{tr}(A'T)$ over C. The significance of Lemma 1 is that it provides the cornerstone for solving the problem (Equation 2.10) by GCG. Indeed, (Equation 2.10) can be equivalently rewritten as

$$\min_T \quad J(T) := \tfrac{1}{2}\gamma_{\mathcal{S}}(T) + g(T) \quad \text{where,} \tag{2.15}$$

$$g(T) := \max_{R\mathbf{1}=\mathbf{0}, A\geq \mathbf{0}} -\tfrac{1}{2}\mathrm{tr}(T(I-A)X'X(I-A')) - \tfrac{1}{2}\mathrm{tr}(TR'R) - \tfrac{1}{2}\mathrm{tr}(TAA') - \ell^*(R).$$

This objective finally falls into the framework of GCG sketched in Algorithm 1 [29, 32].

GCG proceeds in iterations and at each step it seeks the steepest descent extreme point T^{new} (a.k.a. basis) of the set $\mathcal{S}$ with respect to the objective gradient (steps 3-4). After finding the optimal conic combination with the existing solution (step 5), it directly optimizes the underlying factor Φ, initialized by the value that corresponds to the current solution T (step 6). Although this last step is not convex (hence called "local optimization"), it offers significant practical efficiency because it allows all existing bases to be optimized along with their weights.

We next provide details on the *efficient* computational strategies for the above operations in

Algorithm 1 General GCG algorithm

Randomly sample $\Phi_1 \in [0,1]^t$, and set $T_1 = \Phi_1'\Phi_1$. **while** $k = 1, 2, \ldots$ **do**

Find $\nabla g(T_k)$ with $T_k = \Phi_k'\Phi_k$ by solving the inner maximization problem in $g(T_k)$ of (Equation 2.16).

Polar operator: find a new basis $T^{\text{new}} = \arg\max_{T\in\mathcal{S}} \langle T, -\nabla g(T_k)\rangle$.

Compute the optimal combination weight $(\alpha, \beta) := \arg\min_{\alpha\geq 0, \beta\geq 0} J(\alpha T_k + \beta T^{\text{new}})$.

Locally optimize T: $\Phi_{k+1} = \arg\min_{\Phi\geq \mathbf{0}} J(\Phi'\Phi)$ with Φ initialized by the value corresponding to $\Phi'\Phi = \alpha T_k + \beta T^{\text{new}}$ (see Section 2.4).

end

Return T_{k+1}

our problem.

2.4 Polar operator and constant multiplicative approximation guarantee

Given the negative gradient $G = -\nabla g(T_k) \in \mathbb{R}^{t\times t}$, the polar operator of $\mathcal{S}$ tries to solve the following optimization problem by using the characterization of $\mathcal{S}$ in (Equation 2.12):

$$\max_{T\in\mathcal{S}} \operatorname{tr}(G'T) \quad \Longleftrightarrow \quad \max_{\mathbf{x}\in\mathbb{R}_+^t,\ \|\mathbf{x}\|\leq 1} \operatorname{tr}(\mathbf{x}'G\mathbf{x}). \tag{2.16}$$

Unfortunately, this problem is NP-hard. If this were solvable for any G, then we could use it to answer whether $\min_{\mathbf{x}\geq\mathbf{0}} \mathbf{x}'(-G)\mathbf{x} \geq 0$. But the latter is to check the copositivity of $-G$, which is known to be co-NP-complete [33]. Usually problems like (Equation 2.16) are approached by semi-definite relaxations (SDP), and [34] showed that it can be approximately solved with a multiplicative bound of $O(\log t)$.

As one of our major contributions, we next show that when $G \succeq \mathbf{0}$, this bound can be tightened into *constant* for (Equation 2.16) with a computational procedure that is much more efficient than SDP. Furthermore, our particular problem does satisfy $G \succeq \mathbf{0}$.

Before proceeding, we first recall the definition of a multiplicative α-approximate solution.

Definition 1. *Consider an optimization problem* $\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$ *with nonnegative optimal value. Let* $\alpha \in (0, 1]$. *Then a solution* $\mathbf{x}^* \in \mathcal{X}$ *is called* α*-approximate if* $f(\mathbf{x}^*) \geq \alpha \max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$. *Similarly, for minimization problems, the condition becomes* $f(\mathbf{x}^*) \leq \frac{1}{\alpha} \min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$ *given* $\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \geq 0$.

Theorem 3. *Assume* $G \succeq \mathbf{0}$. *Then a* $\frac{1}{4}$*-approximate solution to* (Equation 2.16) *can be found in* $O(t^2)$ *time.*

Proof. Since $G \succeq \mathbf{0}$, it can be decomposed into $G = H'H$ and the problem (Equation 2.16) becomes $\max_{\mathbf{x}\in\mathbb{R}^t_+, \|\mathbf{x}\|\leq 1} \|H\mathbf{x}\|^2$. Let $\mathbf{v}$ be top eigenvector of G that corresponds to the greatest eigenvalue. Then $\mathbf{v}$ maximizes $\|H\mathbf{x}\|$ over $\|\mathbf{x}\| \leq 1$. Decompose $\mathbf{v} = \mathbf{v}_+ - \mathbf{v}_-$, where $\mathbf{v}_+ = [\mathbf{v}]_+$ collects the nonnegative components, and $\mathbf{v}_-$ collects the negative components. Apparently we have $\|\mathbf{v}_+\| \leq 1$ and $\|\mathbf{v}_-\| \leq 1$. Without loss of generality assume $\|H\mathbf{v}_+\|_2 \geq \|H\mathbf{v}_-\|_2$ and consequently let us use $\mathbf{v}_+$ as an approximate minimizer, which we demonstrate is $\frac{1}{4}$-approximate:

$$\max_{\mathbf{x}\in\mathbb{R}^t_+, \|\mathbf{x}\|\leq 1} \|H\mathbf{x}\|^2 \leq \|H\mathbf{v}\|^2 = \|H\mathbf{v}_+ - H\mathbf{v}_-\|^2 \leq 2(\|H\mathbf{v}_+\|^2 + \|H\mathbf{v}_-\|^2) \leq 4\|H\mathbf{v}_+\|^2.$$

Obviously $\frac{\mathbf{v}_+}{\|\mathbf{v}_+\|}$ is an even better solution, which can also be used as an initializer for further local optimization. The computational bottleneck lies in the top eigenvector $\mathbf{v}$ of G, which costs $O(t^2)$. □

In the case that G is not positive semi-definite, it turns out very hard to extend this technique while retaining a constant bound. However the SDP-based technique in [34] still applies, and the bound remains $1/\log t$. In hindsight, our choice of the adding Frobenius norm constraint on Φ when defining $\mathcal{S}$ in (Equation 2.11) is not arbitrary. It constitutes the most straightforward path that allows the polar operator to be approximated in a tractable fashion. Other choices, such as structured Frobenius norms, could be possible if we would like to enforce structured decompositions in the hidden representation. We leave the extension of tractable approximation for future exploration.

Finally, although our algorithm for the polar operator requires G be positive semi-definite—which is not satisfied in general—it happens to be met by our particular problem (Equation 2.15). Notice the gradient of g is simply

$$-\frac{1}{2}(I-A)X'X(I-A') - \frac{1}{2}R'R - \frac{1}{2}AA', \tag{2.17}$$

where the R and A are the optimal solution to the inner maximization. This is obviously negative semi-definite, providing the key foundation for the constant approximation bound of our approach.

2.4.1 Optimality of GCG and overall efficiency

We finally translate the bound on the polar operator to that of the original objective (Equation 2.15). As shown by Theorem 1 of [35], any α-approximate polar operator allows GCG to converge to an α-approximate solution to the original problem, and the convergence rate is $O(1/\epsilon)$. Therefore we are guaranteed to find a $\frac{1}{4}$-approximate solution to (Equation 2.15). The overall algorithm is summarized in Algorithm 2.

Algorithm 2 Solve (Equation 2.10) for T by the GCG algorithm

Randomly sample $\Phi_1 \in [0,1]^t$, and set $T_1 = \Phi_1'\Phi_1$.
while $k = 1, 2, \ldots$ **do**
  if $k = 1$ **then**
    $(U_k, \mathbf{b}_k)$ = optimal U and $\mathbf{b}$ in (Equation 2.19) for Φ_1.
    M_k = optimal M in (Equation 2.19) for Φ_1.
  end
  Recover the optimal R: $R_k = \nabla\ell(U_k'\Phi_k + \mathbf{b}_k\mathbf{1}')$.
  Recover the optimal A by (Equation 2.21).
  Compute the gradient G_k of g_μ at $T_k = \Phi_k'\Phi_k$via (Equation 2.17), with R_k and A_k serving R and A, resp

  Compute a new basis $\mathbf{x}_k$ by approximately solving $\arg\max_{\mathbf{x}\in\mathbb{R}_+^t, \|\mathbf{x}\|\le 1} \mathbf{x}'(-G_k)\mathbf{x}$ (Thm 3).
  Line search: $(\alpha, \beta) := \arg\min_{\alpha\ge 0, \beta\ge 0} J(\alpha T_k + \beta\mathbf{x}_k\mathbf{x}_k')$.
  Set $\Phi_{\text{tmp}} = (\sqrt{\alpha}\Phi_k', \sqrt{\beta}\mathbf{x}_k)'$.
  Local search: $(\Phi_{k+1}, U_{k+1}, \mathbf{b}_{k+1}, M_{k+1}) :=$ Local_Opt$(\Phi_{\text{tmp}}, U_k, \mathbf{b}_k, M_k)$ by Algorithm 3.

end
Return T_{k+1}

Algorithm 3 Local optimization used by GCG

Require $(\Phi_{\text{tmp}}, U_k, \mathbf{b}_k, M_k)$ from the current step

Initialize : $\Phi = \Phi_{\text{tmp}}$, $U = U_k$, $\mathbf{b} = \mathbf{b}_k$, $M = M_k$.

for $t = 1, 2, \ldots$ **do** // till change is small

$(U, \mathbf{b}) = \arg\min_{U,\mathbf{b}} \{\ell(U'\Phi + \mathbf{b}\mathbf{1}') + \frac{1}{2}\|U\|^2\}$.

$M = \arg\min_{M \geq \mathbf{0}} h(M, \Phi)$.

$\Phi = \arg\min_{\Phi \geq \mathbf{0}} \{\ell(U'\Phi + \mathbf{b}\mathbf{1}') + h(M, \Phi)\}$.

end

Return $(\Phi, U, \mathbf{b}, M)$.

2.5 Accelerating local optimization by converting min-max into min-min

The computational bottleneck of applying GCG to our problem (Equation 2.15) is the step of local optimization: $\min_\Phi J(\Phi'\Phi)$ over $\Phi \in \mathbb{R}_+^{h \times t}$. Owing to the $\Phi'\Phi$ term, this objective is not convex. However, it is often observed in practice that the overall optimization can be much accelerated if we solve it *just locally* (*e.g.* by BFGS), with Φ initialized based on the value of the convex optimization variable T (step 6 of Algorithm 1 or step 11 of Algorithm 2).

Unfortunately, since g defined in (Equation 2.16) employs a *nested maximization*, we are now faced with a *min-max* problem. Different from *min-min* optimizations like $\min_x \min_y f(x, y)$ which can be solved very efficiently by alternating between optimizing x and y, a min-max problem $\min_x \max_y f(x, y)$ cannot be solved by alternating: fixing x solve y, and fixing y solve x. Instead, one needs to treat the objective as a function of x, and for each x solve the inner maximization in y *exactly*, before obtaining a gradient in x that is supplied to standard solvers such as BFGS. This is often much slower than alternating.

To enable an efficient solution by alternating, we next develop a novel reformulation of g as a minimization, such that minimizing g becomes a min-min problem:

$$
\begin{aligned}
g(\Phi'\Phi) &= \max_{R\mathbf{1}=\mathbf{0}} \left\{ -\frac{1}{2}\left\|\Phi R'\right\|^2 - \ell^*(R) \right\} + \max_{A\geq\mathbf{0}} \left\{ -\frac{1}{2}\left\|\Phi(I-A)X'\right\|^2 - \frac{1}{2}\left\|\Phi A\right\|^2 \right\} \\
&= \max_{R} \min_{\mathbf{b}} \left\{ \mathbf{b}'R\mathbf{1} - \ell^*(R) - \max_{U} -\operatorname{tr}(U'\Phi R') - \tfrac{\|U\|^2}{2} \right\} \\
&+ \max_{A} \min_{M\geq\mathbf{0}} \left\{ -\tfrac{\|\Phi(I-A)X'\|^2}{2} - \tfrac{\|\Phi A\|^2}{2} + \operatorname{tr}(M'A) \right\} \qquad (2.18) \\
&= \min_{U,\mathbf{b}} \left\{ \ell(U'\Phi + \mathbf{b}\mathbf{1}') + \frac{1}{2}\|U\|^2 \right\} + \min_{M\geq\mathbf{0}} h(M,\Phi), \qquad (2.19) \\
\text{where} \quad & h(M,\Phi) := \max_{A} \Big\{ -\frac{1}{2}\left\|\Phi(I-A)X'\right\|^2 - \frac{1}{2}\left\|\Phi A\right\|^2 + \operatorname{tr}(M'A) \Big\}. \qquad (2.20)
\end{aligned}
$$

As the key advantage achieved here, the local optimization $\min_{\Phi\geq\mathbf{0}} J(\Phi'\Phi) = \min_{\Phi\geq\mathbf{0}} \frac{1}{2}\|\Phi\|^2 + g(\Phi'\Phi)$ can now be solved by alternating between $(U,\mathbf{b})$, M, and Φ. The details are shown in Algorithm 3. The optimization over $(U,\mathbf{b})$ is the standard supervised learning. However, the optimization over M and Φ is trickier because they require evaluating h which in turn involves a nested optimization on A. Fortunately h is quadratic in A, which allows us to design an efficient closed-form scheme by leveraging the celebrated Woodbury formula [36].

Given (M,Φ), the optimal A can be found by setting its gradient to zero: $\Phi'\Phi A(X'X+I) = M + \Phi'\Phi X'X$. Unfortunately, the rank of $\Phi'\Phi A$ (hence the left-hand side) is at most $h < t$. So no A can satisfy the equality if the rank of the right-hand side is greater than h, and hence $h(M,\Phi)$ is finite only if the column space of $(M+\Phi'\Phi X'X)(X'X+I)^{-1}$ is contained in the column space of Φ'. Such an implicit constraint between variables precludes the application of alternating.

To address this problem, we introduce a small strongly convex regularizer on A in the definition of $h(M, \Phi)$ in (Equation 2.20), akin to the commonly used smoothing technique:

$$h_\mu(M, \Phi) := \max_A \Big\{ -\frac{1}{2} \left\|\Phi(I - A)X'\right\|^2 - \frac{1}{2} \left\|\Phi A\right\|^2 + \operatorname{tr}(M'A) - \frac{\mu}{2} \operatorname{tr}(A(X'X + I)A') \Big\},$$

where $\mu > 0$ is small. The new term $\frac{\mu}{2} \operatorname{tr}(A(X'X + I)A')$ also needs to be added to the definition of g in (Equation 2.16), which we will denote as g_μ. Then the optimal A can be found by setting the gradient to 0:

$$A = (\Phi'\Phi + \mu I)^{-1}(M + \Phi'\Phi X'X)(X'X + I)^{-1}. \tag{2.21}$$

To efficiently compute A, we make use of the Woodbury formula:

$$\mu A = (M + \Phi'\Phi X'X)(X'X + I)^{-1} - \Phi'(\mu I + \Phi\Phi')^{-1}\Phi(M + \Phi'\Phi X'X)(X'X+I)^{-1}.$$

Here $(\mu I+\Phi\Phi')^{-1} \in \mathbb{R}^{h\times h}$ can be computed efficiently as h is not large (it is exactly the iteration index k in GCG Algorithm 2). Then the second term (after the minus) can be computed in $O(ht^2)$ time as we can pre-compute $(X'X + I)^{-1}$. So the only challenge in computing A is the term $M(X'X + I)^{-1}$, which costs $O(t^3)$ time. However, if $n \ll t$, then we may again save computations by applying the Woodbury formula: $M(X'X + I)^{-1} = M - MX'(I + XX')^{-1}X$, which costs $O(nt^2)$ time. $(I + XX')^{-1}X$ can be pre-computed.

2.6 Experiment

We evaluated the proposed inductive training of convexified two-layer model (CVX-IN) by comparing the generalization accuracy with 4 other baselines: FFNN: a two-layer feedforward neural network; Ker-CVX: the kernel-based convex model proposed by [37]; LOCAL: a model obtained by alternative minimization of the two-layer objective (Equation 2.4); and CVX-TR: our model learned transductively (see below). SVM was not included since it was already shown inferior to Ker-CVX by [37].

2.6.1 Inductive learning.

A key advantage of our method is the purely inductive setting, which obviates any retraining during test time, as opposed to a transductive setting. After completing the GCG optimization, CVX-IN directly obtains the optimal U and $\mathbf{b}$ thanks to the local minimization in Algorithm 3. The optimal W can be recovered by solving (Equation 2.4) with fixed $(\Phi, U, \mathbf{b})$, and it is a simple convex problem. With this initialization, we finely tuned all parameters by backpropagation.

2.6.2 Transductive learning.

As Ker-CVX is transductive, we also considered the following transductive variant of CVX-IN. The objective (Equation 2.15) was first trained with X being the combination of (X_{train}, X_{test}), and accordingly the intermediate representation Φ (along with the corresponding T) also consisted of the combination of $(\Phi_{train}, \Phi_{test})$. Since only Y_{train} was available for training, the loss function $\ell(U'\Phi + \mathbf{b}\mathbf{1}')$ was applied only to the training data. As a result, Φ_{test} was learned largely from the matching loss in the latent layer given by (Equation 2.20). After recovering the optimal U and $\mathbf{b}$ by local minimization (same as in CVX-IN), test data were labeled by

$\hat{Y}_{test} = U'\Phi_{test} + \mathbf{b1}'$. Although CVX-TR bypasses the recovery of W, optimization has to be redone from scratch when new test data arrives.

2.6.3 Comparison on smaller datasets.

To enable comparison with Ker-CVX which is highly expensive in computation, we first used smaller datasets including a synthetic XOR dataset and three "real world" datasets for binary classification: Letter [38], CIFAR-SM, a binary classification dataset from [39] based on CIFAR-100 [40], and G241N [41].

All methods were applied to two different sizes of training and test data (X_{train} and X_{test}): 100/100 and 200/200, and the resulting test error, averaged over 10 trials, were presented in Table Table I and Table II respectively. CVX-IN outperforms FFNN on G241N, Letter, and CIFAR-SM, and they both delivered perfect classification on XOR. This corroborates the advantage of convex models, suggesting that predictive structures are preserved by the relaxation. CVX-IN also marginally outperforms or is comparable to CVX-TR on all the datasets, confirming that inductive learning saves computation at test time without sacrificing the accuracy. Consistently poor performance is observed on the LOCAL method (used in a transductive fashion), and it does not work even for XOR. This implies that it does suffer seriously from local optimality. Ker-CVX (transductive only) performs competitively on 200 examples especially on the Letter dataset, but its error on 100 examples is significantly higher than CVX-IN and CVX-TR. It ran into computational issues on G241N, hence marked by N/A.

On the CIFAR-SM dataset all methods produced a slightly higher error with 200 training examples than 100 examples, probably due to the small size of training set and high variance. However the comparative results between algorithms remain similar to other datasets.

2.6.4 Comparison on larger datasets.

Thanks to the fast local optimization enabled by the new min-min alternating (§2.5), our model enjoys significant speedup compared with [37, 39]. To demonstrate this, we applied CVX-IN to Letter, XOR, and CIFAR-10 [40] with 1000/1000 and 2000/2000 train/test examples, and to G241N with 1000/500 examples (the entire dataset only has 1500 examples). Details on data pre-processing are available in Appendix B.2.

As Table Table III and Table IV show, CVX-IN again achieves significantly lower test error on these larger datasets over FFNN, CVX-TR, and LOCAL. The training time of CVX-IN is summarized in Table Table V, and it took 2.5 hours on CIFAR-10 with 2000 examples and 256 features. Although still expensive, it is substantially faster than Ker-CVX which is completely incapable of scaling here (hence omitted). In contrast, the run time of FFNN and LOCAL is much lower (not shown). Overall CVX-IN scales quadratically in #examples (t), which is consistent with our analysis in §2.5.

2.6.5 Intermediate representation.

One of the key merits of our two-layer model is that the relaxation retains the necessary structure in the input data to make accurate predictions. To test this feature, we tried to visualize the latent representation learned by our CVX-IN. Figure Figure 2 demonstrates the original features in the input data X_{train} and the learned intermediate representation Φ_{train},

	Letter	G241N	XOR	CIFAR-SM
CVX-IN	4.8±0.8	24.2±1.6	0	21.2±1.2
CVX-TR	4.9±1.3	23.1±0.7	0	22.4±0.8
FFNN	7.9±0.8	31.9±0.9	0	31.0±1.1
LOCAL	8.0±1.2	34.0±0.9	27.0±1.5	25.0±0.8
Ker-CVX	5.7±2.9	N/A	0	27.7±5.5

TABLE I: Mean test error for 100 training and 100 test examples

	Letter	G241N	XOR	CIFAR-SM
CVX-IN	5.1±1.3	21.6±0.9	0	22.6±1.5
CVX-TR	5.3±0.8	22.0±0.8	0	23.4±1.5
FFNN	5.5±0.8	29.9±0.4	0	32.9±1.0
LOCAL	10.5±0.8	33.0±0.6	25.0±1.2	29.5±0.5
Ker-CVX	4.5±0.9	N/A	0	23.3±3.5

TABLE II: Mean test error for 200 training and 200 test examples

for two datasets Box and XOR which both employ a rich latent structure. Clearly, the convex relaxation was able to separate the two classes and preserve sufficient structures that allows it to outperform single-layer models.

2.7 Discussion and Conclusion

In this section, we showed how to obtain a convex relaxation of a simple two-layered neural network objective with activation functions such as ReLU, based on *matching loss*. We also developed an efficient optimization procedure based on Generalized Conditional Gradient (GCG) algorithm with a constant approximation guarantee.

There are however two main drawbacks of this current approach, which are potential directions for future research. First, the relaxation procedure that is described in this work is hard to extend to networks beyond 2 layers. Even adding an additional layer might require a

	Letter	G241N	XOR	CIFAR-10
CVX-IN	2.7±0.8	13.0±0.8	0	27.6±1.4
CVX-TR	2.7±0.9	15.1±0.9	0	27.9±2.3
FFNN	3.5±0.7	24.5±1.0	0	30.4±0.9
LOCAL	5.8±0.7	21.4±1.1	26.7±0.5	32.3±0.7

TABLE III: Mean test error for 1000 training and 1000 test examples

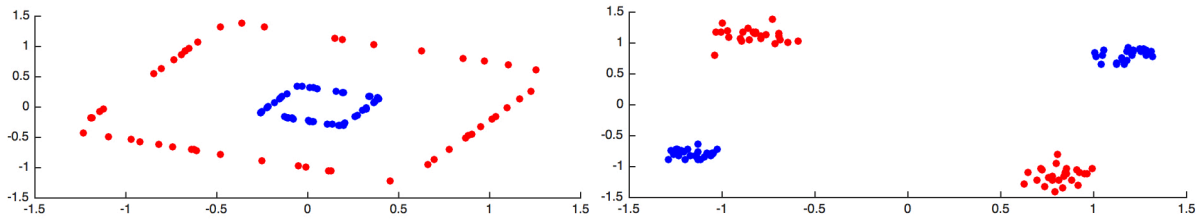
	Letter	XOR	CIFAR-10
CVX-IN	1.0±0.5	0	26.8±1.6
CVX-TR	1.2±0.7	0	27.0±1.9
FFNN	1.7±0.3	0	30.0±1.8
LOCAL	2.3±0.4	27.2±0.3	33.0±1.5

TABLE IV: Mean test error for 2000 training and 2000 test examples

	100	200	1000	2000
Letter	0.45	1.1	17.1	90.6
G241N	0.68	1.5	27.3	N/A
XOR	0.45	1.0	42.0	144.2
CIFAR-10	0.63	1.5	50.6	153.6

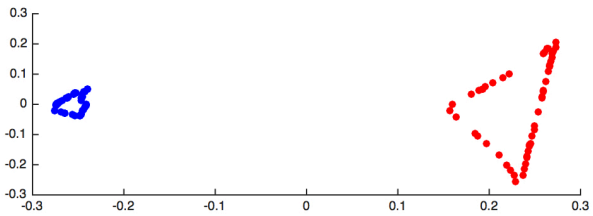
TABLE V: Training times (in minutes) for CVX-IN on 100, 200, 1000, and 2000 training examples

careful change in the relaxation and optimization strategies. Hence, a more general approach to modeling and optimization is in order. Second, there are no *non-trivial* recovery procedures known for SDP and CP relaxations. Our results are currently purely empirical and it would be interesting to establish more rigorous theoretical guarantees in approximation. As another possible research exercise, it would also be interesting to extend our results to other non-linear activation functions that satisfy our framework.

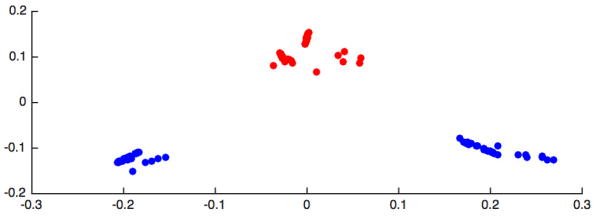


(a) BOX

(b) XOR



(c) h_{BOX}



(d) h_{XOR}

Figure 2: BOX and XOR datasets (subplots a and b) and their intermediate representations (h_{dataset} in subplots c and d). The representations were reduced to 2-D by using the standard PCA.

CHAPTER 3

INCORPORATING LATENT STRUCTURE INFORMATION

(Parts of this chapter were previously published as "Convex two-layer modeling with latent structure" [23] at the Neural Information Processing Systems (NIPS 2016))

3.1 Introduction

In this section, we consider the problem of modeling (structured) relationships in data that are not directly apparent (latent). Several real-life machine learning tasks (two of which we introduce in this section) can benefit from modeling of latent relationships between entities of interest. To this end, we explore the problem of finding a convex formulation of a two-layer model, where we want to explicitly learn a *structured latent hidden representation*, in an *unsupervised manner*, that subsequently helps in learning highly discriminative model parameters that facilitate accurate predictions. We will make this clear with the help of the following application of such modeling - the problem of **transliteration** between two languages.

3.2 Transliteration via graph matching

Transliteration is the problem of accurately matching word pairs (w_1, w_2) written in two different languages, that are phonetically identical to each other. Identifying this mapping is not easy, as most writing systems do not perfectly align phonetically and orthographically; rather, this mapping can be context-dependent and ambiguous [42].

Suppose we are given a word from the English language $\mathbf{e}$ with m letters, and another word from Hebrew $\mathbf{h}$ with n letters. The task is to predict whether $\mathbf{e}$ and $\mathbf{h}$ are phonetically similar to each other. If we represent $\mathbf{e}$ and $\mathbf{h}$, by a joint feature vector $\mathbf{x}$, then this is a binary classification problem with $z \in \{-1, 1\}$. Here, we would like to capture some additional structural information $\mathbf{y}$ regarding w_1 and w_2 that would help us with this classification task . One such structure could be the letter-wise *matching* between $\mathbf{e}$ and $\mathbf{h}$ [42, 43]. We would also like to note that, there are no class labels provided for the latent $\mathbf{y}$, hence this is an *unsupervised latent structured learning* problem.

3.3 Problem setup and related works

In this section, we set up the problem formally. We are interested in two-layer conditional models of the form $X \to Y \to Z$, where X is the input layer, Y is the latent hidden layer and Z is the output layer. So far, most deep models for structured output are designed for supervised learning where structured labels are available. Recently an extension has been made to the *unsupervised* learning. [44] proposed a conditional random field auto-encoder (CRF-AE), where given the observed data $\mathbf{x}$, the latent structure $\mathbf{y}$ is first generated based on $p(\mathbf{y}|\mathbf{x})$ and then applied to reconstruct the observations using $p(\mathbf{x}|\mathbf{y})$. The motivation is to find the predictive and latent structure in the data. Along similar lines other discriminative unsupervised learning methods are also available.

Extending auto-encoders $X \to Y \to X$ to general two-layer setting $X \to Y \to Z$ is not hard. The transliteration problem introduced in the previous section was addressed by [42, 43], where Z is the binary output and Y is the *unobserved* latent structure, representing the letter-wise

matching between words in the context of transliteration. In essence, their model optimizes $p(\mathbf{z}|\arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}))$, uncovering the latent $\mathbf{y}$ via its mode under the first layer model. This is commonly known as **bi-level** optimization, since the optimal value(arg max) of the inner optimization is used in learning the optimal $\mathbf{z}$.

3.3.1 Challenges in inference.

All the methods described above, will need some form of tractability in inference in estimating latent structure, in computation. CRF-AE leverages *marginal inference* on $p(\mathbf{y}|\mathbf{x})$ for Expectation Maximization(EM). Contrastive divergence, instead samples from $p(y|x)$ [45]. However, for some structures such as graph matching, neither of them is tractable [46, 47]. For *single-layer* models, this challenge has been resolved by the so-called *max-margin* estimation, which only relies on the MAP of $p(\mathbf{y}|\mathbf{x})$, which is a point estimate. This oracle is much less demanding than sampling or normalization, as finding a point estimate can be much easier than summarizing over all possible values of $\mathbf{y}$. For example MAP for graph matching can b solved by max-flow.

Extending the max-margin inference directly to the two-layered(our) setting, is non-trivial and is immediately faced with the following problem. The max-margin inference in our setting would require us to solve $\max_y p(\mathbf{y}|\mathbf{x})p(\mathbf{z}|\mathbf{y})$. In general, $p(\mathbf{z}|\mathbf{y}))$ depends on $\mathbf{y}$ in a highly non-linear form, making this *augmented inference* intractable. This seems to leave the aforementioned bi-level optimization the only option that retains the sole dependency on MAP.

3.3.2 Preliminaries

We consider a two-layer latent conditional model $X \to Y \to Z$, where X is the input, Z is the output, and Y is a latent layer composed of h random variables: $\{Y_i\}_{i=1}^h$. Instead of assuming no interdependency between Y_i as in [37], our major goal here is to model the structure in the latent layer Y. Specifically, we assume a conditional model for the first layer based on an exponential family

$$p(\mathbf{y}|\mathbf{x}) = q_0(\mathbf{y}) \exp(-\mathbf{y}'U\mathbf{x} - \Omega(U\mathbf{x})), \quad \text{where} \quad q_0(\mathbf{y}) = [\![\mathbf{y} \in \mathcal{Y}]\!]\,. \tag{3.1}$$

Here U is the first layer weight matrix, and Ω is the log-partition function. $q_0(\mathbf{y})$ is the base measure, with $[\![x]\!] = 1$ if x is true, and 0 otherwise. The correlation among Y_i is instilled by the support set $\mathcal{Y}$, which plays a central role here. For example, when $\mathcal{Y}$ consists of all h-dimensional canonical vectors, $p(\mathbf{y}|\mathbf{x})$ recovers the logistic multiclass model. In general, to achieve a tradeoff between computational efficiency and representational flexibility, we make the following assumptions on $\mathcal{Y}$:

Postulate 1 (PO-tractable)**.** *We assume $\mathcal{Y}$ is bounded, and admits an efficient* polar operator*. That is, for any vector $\mathbf{d} \in \mathbb{R}^h$, $\min_{\mathbf{y}\in\mathcal{Y}} \mathbf{d}'\mathbf{y}$ is efficiently solvable.*

Note the support set $\mathcal{Y}$ (hence the base measure q_0) is fixed and does *not* contain any more parameter. PO-tractability is available in a variety of applications, and we give two examples here.

3.3.2.1 Graph matching.

In a bipartite graph with two sets of vertices $\{a_i\}_{i=1}^n$ and $\{b_j\}_{j=1}^n$, each edge between a_i and b_j has a weight T_{ij}. The task is to find a one-to-one mapping (can be extended) between $\{a_i\}$ and $\{b_j\}$, such that the sum of weights on the edges is maximized. Denote the matching by $Y \in \{0,1\}^{n\times n}$ where $Y_{ij} = 1$ iff the edge (a_i, b_j) is selected. So the optimal matching is the mode of $p(Y) \propto [\![Y \in \mathcal{Y}]\!] \exp(\mathrm{tr}(Y'T))$ where the support is $\mathcal{Y} = \{Y \in \{0,1\}^{n\times n} : Y\mathbf{1} = Y'\mathbf{1} = \mathbf{1}\}$.

3.3.2.2 Graphical models.

For simplicity, consider a linear chain model $V_1 - V_2 - \cdots - V_p$. Here each V_i can take one of C possible values, which we encode using the C-dimensional canonical basis $\mathbf{v}_i$. Suppose there is a node potential $\mathbf{m}_i \in \mathbb{R}^C$ for each V_i, and each edge (V_i, V_{i+1}) has an edge potential $M_i \in \mathbb{R}^{C\times C}$. Then we could directly define a distribution on $\{V_i\}$. Unfortunately, it will involve quadratic terms such as $\mathbf{v}_i' M_i \mathbf{v}_{i+1}$, and so a different parameterization is in order. Let $Y_i \in \{0,1\}^{C\times C}$ encode the values of (V_i, V_{i+1}) via row and column indices of Y_i respectively. Then the distribution on $\{V_i\}$ can be equivalently represented by a distribution on $\{Y_i\}$:

$$p(\{Y_i\}) \propto [\![\{Y_i\} \in \mathcal{Y}]\!] \exp\left(\sum_{i=1}^{p} \mathbf{m}_i' Y_i \mathbf{1} + \sum_{i=1}^{p-1} \mathrm{tr}(M_i' Y_i) \right), \tag{3.2}$$

$$\text{where } \mathcal{Y} = \left\{\{Y_i\} : Y_i \in \{0,1\}^{C\times C}\right\} \cap \mathcal{H}, \quad \text{with } \mathcal{H} := \left\{\{Y_i\} : \mathbf{1}'Y_i\mathbf{1} = 1,\ Y_i'\mathbf{1} = Y_{i+1}\mathbf{1}\right\}. \tag{3.3}$$

The constraints in $\mathcal{H}$ encode the obvious consistency constraints between overlapping edges. This model ultimately falls into our framework in (Equation 3.1).

In both examples, the constraints in $\mathcal{Y}$ are totally unimodular (TUM), and therefore the polar operator can be computed by solving a linear programming (LP), with the $\{0,1\}$ constraints relaxed to $[0,1]$. In §3.6.1 and 3.6.2 we will generalize $\mathbf{y}'U\mathbf{x}$ to $\mathbf{y}'\mathbf{d}(U\mathbf{x})$, where $\mathbf{d}$ is an affine function of $U\mathbf{x}$ that allows for homogeneity in temporal models. For clarity, we first develop a general framework using $\mathbf{y}'U\mathbf{x}$.

3.3.2.3 Output layer

As for the output layer, we assume a conditional model from an exponential family

$$p(\mathbf{z}|\mathbf{y}) = \exp(\mathbf{z}'R'\mathbf{y} - G(R'\mathbf{y}))q_1(\mathbf{z}) = \exp(-D_{G^*}(\mathbf{z}||\nabla G(R'\mathbf{y})) + G^*(\mathbf{z}))q_1(\mathbf{z}), \tag{3.4}$$

where G is a smooth and strictly convex function, and D_{G^*} is the Bregman divergence induced by the Fenchel dual G^*. Such a parameterization is justified by the equivalence between regular Bregman divergence and regular exponential family [48]. Thanks to the convexity of G, it is trivial to extend $p(\mathbf{z}|\mathbf{y})$ to $\mathbf{y} \in \text{conv}\mathcal{Y}$ (the convex hull of $\mathcal{Y}$), and $G(R'\mathbf{y})$ will still be convex over $\text{conv}\mathcal{Y}$ (fixing R).

Finally we highlight the assumptions we make and do not make. First we only assume PO-tractability of $\mathcal{Y}$, hence tractability in MAP inference of $p(\mathbf{y}|\mathbf{x})$. We do *not* assume it is tractable to compute the normalizer Ω or its gradient (marginal distributions). We also do not assume that unbiased samples of $\mathbf{y}$ can be drawn efficiently from $p(\mathbf{y}|\mathbf{x})$. In general, PO-tractability is a weaker assumption. For example, in graph matching MAP inference is tractable while marginalization is NP-hard [49] and sampling requires MCMC [50]. Finally, we

do *not* assume tractability of any sort for $p(\mathbf{y}|\mathbf{x})p(\mathbf{z}|\mathbf{y})$ (in $\mathbf{y}$), and so it may be hard to solve $\min_{\mathbf{y}\in\mathcal{Y}}\{\mathbf{d}'\mathbf{y} + G(R'\mathbf{y}) - \mathbf{z}'R'\mathbf{y}\}$, as G is generally not affine.

3.4 Training principles

At training time, we are provided with a set of feature-label pairs $(\mathbf{x}, \mathbf{z})\sim\tilde{p}$, where $\tilde{p}$ is the empirical distribution. In the special case of auto-encoder, $\mathbf{z}$ is tied with $\mathbf{x}$. The "bootstrapping" style estimation [51] optimizes the joint likelihood with the latent $\mathbf{y}$ imputed in an optimistic fashion

$$\min_{U,R} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{z})\sim\tilde{p}} \left[\min_{\mathbf{y}\in\mathcal{Y}} -\log p(\mathbf{y}|\mathbf{x})p(\mathbf{z}|\mathbf{y})\right] =$$
$$\min_{U,R} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{z})\sim\tilde{p}} \left[\min_{\mathbf{y}\in\mathcal{Y}} \mathbf{y}'U\mathbf{x} + \omega(U\mathbf{x}) - \mathbf{z}'R'\mathbf{y} + G(R'\mathbf{y})\right].$$

This results in a hard EM estimation, and a soft version can be achieved by adding entropic regularizers on $\mathbf{y}$. Regularization can be imposed on U and R which we will make explicit later (*e.g.* bounding the L_2 norm). Since the log-partition function ω in $p(\mathbf{y}|\mathbf{x})$ is hard to compute, the max-margin approach is introduced which replaces $\omega(U\mathbf{x})$ by an upper bound $\max_{\hat{\mathbf{y}}\in\mathcal{Y}} -\hat{\mathbf{y}}'U\mathbf{x}$, leading to a surrogate loss

$$\min_{U,R} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{z})\sim\tilde{p}} \left[\min_{\mathbf{y}\in\mathcal{Y}} \left\{-\mathbf{z}'R'\mathbf{y} + G(R'\mathbf{y}) + \mathbf{y}'U\mathbf{x} - \min_{\hat{\mathbf{y}}\in\mathcal{Y}} \hat{\mathbf{y}}'U\mathbf{x}\right\}\right]. \tag{3.5}$$

However, the key disadvantage of this method is the augmented inference on $\mathbf{y}$, because we have only assumed the tractability of $\min_{\mathbf{y}\in\mathcal{Y}} \mathbf{d}'\mathbf{y}$ for all $\mathbf{d}$, *not* $\min_{\mathbf{y}\in\mathcal{Y}}\{\mathbf{y}'\mathbf{d} + G(R'\mathbf{y}) - \mathbf{z}'R'\mathbf{y}\}$.

In addition, this principle intrinsically determines the latent $\mathbf{y}$ as a function of *both* the input *and* the output, while at test time the output itself is unknown and is the subject of prediction. The common practice therefore requires a joint optimization over $\mathbf{y}$ and $\mathbf{z}$ at *test* time, which is costly in computation.

The goal of this paper is to design a convex formulation in which the latent $\mathbf{y}$ is completely determined by the input $\mathbf{x}$, and both the prediction and estimation rely only on the polar operator: $\arg\min_{\mathbf{y}\in\mathcal{Y}} \mathbf{y}'U\mathbf{x}$. As a consequence of this goal, it is natural to postulate that the $\mathbf{y}$ found this way renders an accurate prediction of $\mathbf{z}$, or a faithful recovery of $\mathbf{x}$ in auto-encoders. This idea, which has been employed by [52, 53], leads to the following bi-level optimization problem

$$\max_{U,R} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{z})\sim\tilde{p}} \left[\log p\left(\mathbf{z}\middle|\arg\max_{\mathbf{y}\in\mathcal{Y}} p(\mathbf{y}|\mathbf{x})\right)\right] \quad \leftrightarrow \quad \max_{U,R} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{z})\sim\tilde{p}} \left[\log p\left(\mathbf{z}\middle|\arg\min_{\mathbf{y}\in\mathcal{Y}} \mathbf{y}'U\mathbf{x}\right)\right] \tag{3.6}$$

$$\leftrightarrow \quad \min_{U,R} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{z})\sim\tilde{p}} \left[-\mathbf{z}'R'\mathbf{y}^*_{\mathbf{x}} + G(R'\mathbf{y}^*_{\mathbf{x}})\right], \tag{3.7}$$

$$\text{where } \mathbf{y}^*_{\mathbf{x}} = \arg\min_{\mathbf{y}\in\mathcal{Y}} \mathbf{y}'U\mathbf{x}.$$

Directly solving this optimization problem is challenging, because the optimal $\mathbf{y}^*_{\mathbf{x}}$ is almost surely invariant to small perturbations of U (*e.g.* when $\mathcal{Y}$ is discrete). So a zero valued gradient is witnessed almost everywhere. Therefore a more carefully designed optimization algorithm is in demand.

3.5 A General Framework of Convexification

We propose addressing this bi-level optimization by convex relaxation, and it is built upon the first-order optimality conditions of the inner-level optimization. First notice that the set $\mathcal{Y}$ participates in the problem (Equation 3.7) only via the polar operator at $U\mathbf{x}$: $\arg\min_{\mathbf{y}\in\mathcal{Y}} \mathbf{y}'U\mathbf{x}$. If $\mathcal{Y}$ is discrete, this problem is equivalent to optimizing over $S := \text{conv}\mathcal{Y}$, because a linear function on a convex set is always optimized on its extreme points. Clearly, S is convex, bounded, closed, and is PO-tractable. It is important to note that the origin is *not* necessarily contained in S. To remove the potential non-uniqueness of the minimizer in (Equation 3.7), we next add a small proximal term to the polar operator problem (σ is a small positive number):

$$\min_{\mathbf{w}\in S} \mathbf{w}'U\mathbf{x} + \frac{\sigma}{2}\|\mathbf{w}\|^2 . \tag{3.8}$$

This leads to a small change in the problem and makes sure that the minimizer is unique.[1] Adding strongly convex terms to the primal and dual objectives is a commonly used technique for accelerated optimization [54], and has been used in graphical model inference [55]. We intentionally changed the symbol $\mathbf{y}$ into $\mathbf{w}$, because here the optimal $\mathbf{w}$ is not necessarily in $\mathcal{Y}$.

---

[1] If $p(\mathbf{y}|\mathbf{x}) \propto p_0(\mathbf{y}) \exp(-\mathbf{y}'U\mathbf{x} - \frac{\sigma}{2}\|\mathbf{y}\|^2)$ (for any $\sigma > 0$), then there is no need to add this $\frac{\sigma}{2}\|\mathbf{w}\|^2$ term. In this case, all our subsequent developments apply directly. Therefore our approach applies to a broader setting where L_2 projection to S is tractable, but here we focus on PO-tractability just for the clarity of presentation.

By the convexity of the problem (Equation 3.8) and noting that the gradient of the objective is $U\mathbf{x}+\sigma\mathbf{w}$, $\mathbf{w}$ is optimal *if and only if*

$$\mathbf{w}\in S, \quad \text{and} \quad (U\mathbf{x}+\sigma\mathbf{w})'(\hat{\boldsymbol{\theta}}-\mathbf{w})\geq 0, \quad \forall\hat{\boldsymbol{\theta}}\in S. \tag{3.9}$$

These optimality conditions can be plugged into the bi-level optimization problem (Equation 3.7). Introducing "Lagrange multipliers" $(\gamma,\hat{\boldsymbol{\theta}})$ to enforce the latter condition via a mini-max formulation, we obtain

$$\min_{\|U\|\leq 1}\min_{\|R\|\leq 1}\mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{z})\sim\tilde{p}}\Big[\min_{\mathbf{w}}\max_{\gamma\geq 0,\hat{\boldsymbol{\theta}}\in S}\max_{\mathbf{v}} -\mathbf{z}'R'\mathbf{w}+\mathbf{v}'R'\mathbf{w}-G^*(\mathbf{v}) \tag{3.10}$$

$$+\iota_S(\mathbf{w})+\gamma(U\mathbf{x}+\sigma\mathbf{w})'(\mathbf{w}-\hat{\boldsymbol{\theta}})\Big], \tag{3.11}$$

where ι_S is the $\{0,\infty\}$-valued indicator function of the set S. Here we dualized G via $G(R'\mathbf{w})=\max_{\mathbf{v}}\mathbf{v}'R'\mathbf{w}-G^*(\mathbf{v})$, and made explicit the Frobenius norm constraints ($\|\cdot\|$) on U and R.[1] Applying change of variable $\boldsymbol{\theta}=\gamma\hat{\boldsymbol{\theta}}$, the constraints $\gamma\geq 0$ and $\hat{\boldsymbol{\theta}}\in S$ (a convex set) become

$$(\boldsymbol{\theta},\gamma)\in\mathcal{N}:=\text{cone}\{(\hat{\boldsymbol{\theta}},1):\hat{\boldsymbol{\theta}}\in S\},$$

[1]To simplify the presentation, we bound the radius by 1 while in practice it is a hyperparameter to be tuned.

where cone stands for the conic hull (convex). Similarly we can dualize $\iota_S(\mathbf{w}) = \max_{\boldsymbol{\pi}} \boldsymbol{\pi}'\mathbf{w} - \sigma_S(\boldsymbol{\pi})$, where $\sigma_S(\boldsymbol{\pi}) := \max_{\mathbf{w}\in S} \boldsymbol{\pi}'\mathbf{w}$ is the support function on S. Now swap $\min_{\mathbf{w}}$ with all the subsequent max (strong duality), we arrive at a form where $\mathbf{w}$ can be minimized out analytically

$$\min_{\|U\|\le 1} \min_{\|R\|\le 1} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{z})\sim\tilde{p}} \Big[\max_{\boldsymbol{\pi}} \max_{(\boldsymbol{\theta},\gamma)\in\mathcal{N}} \max_{\mathbf{v}} \min_{\mathbf{w}} -\mathbf{z}'R'\mathbf{w} + \mathbf{v}'R'\mathbf{w} - G^*(\mathbf{v}) \tag{3.12}$$

$$+ \boldsymbol{\pi}'\mathbf{w} - \sigma_S(\boldsymbol{\pi}) + (U\mathbf{x} + \sigma\mathbf{w})'(\gamma\mathbf{w} - \boldsymbol{\theta}) \Big] \tag{3.13}$$

$$= \min_{\|U\|\le 1} \min_{\|R\|\le 1} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{z})\sim\tilde{p}} \Big[\max_{\boldsymbol{\pi}} \max_{(\boldsymbol{\theta},\gamma)\in\mathcal{N}} \max_{\mathbf{v}} -G^*(\mathbf{v}) - \sigma_S(\boldsymbol{\pi}) - \boldsymbol{\theta}'U\mathbf{x} \tag{3.14}$$

$$- \tfrac{1}{4\sigma\gamma} \left\| R(\mathbf{v}-\mathbf{z}) + \gamma U\mathbf{x} + \boldsymbol{\pi} - \sigma\boldsymbol{\theta} \right\|^2 \Big]. \tag{3.15}$$

Given (U, R), the optimal $(\mathbf{v}, \boldsymbol{\pi}, \boldsymbol{\theta}, \gamma)$ can be efficiently solved through a concave maximization. However the overall objective is *not* convex in (U, R) because the quadratic term in (Equation 3.15) is subtracted. Fortunately it turns out not hard to tackle this issue by using semi-definite programming (SDP) relaxation which linearizes the quadratic terms. In particular, let I be the identity matrix, and define

$$M := M(U,R) := \begin{pmatrix} I \\ U' \\ R' \end{pmatrix} \Big(I, U, R \Big) = \begin{pmatrix} I & U & R \\ U' & U'U & U'R \\ R' & R'U & R'R \end{pmatrix} =: \begin{pmatrix} M_1 & M_u & M_r \\ M_u' & M_{u,u} & M_{r,u}' \\ M_r' & M_{r,u} & M_{r,r} \end{pmatrix}. \quad (3.16)$$

Then $\boldsymbol{\theta}' U \mathbf{x}$ can be replaced by $\boldsymbol{\theta}' M_u \mathbf{x}$ and the quadratic term in (Equation 3.15) can be expanded as

$$f(M, \boldsymbol{\pi}, \boldsymbol{\theta}, \gamma, \mathbf{v}; \mathbf{x}, \mathbf{z}) := \operatorname{tr}(M_{r,r}(\mathbf{v}-\mathbf{z})(\mathbf{v}-\mathbf{z})') + \gamma^2 \operatorname{tr}(M_{u,u}\mathbf{x}\mathbf{x}') + 2\gamma \operatorname{tr}(M_{r,u}\mathbf{x}(\mathbf{v}-\mathbf{z})') + 2(\boldsymbol{\pi} - \sigma\boldsymbol{\theta})'(M_r(\mathbf{v}-\mathbf{z}) + \gamma M_u \mathbf{x}) + \|\boldsymbol{\pi} - \sigma\boldsymbol{\theta}\|^2 . \quad (3.17)$$

Since given $(\boldsymbol{\pi}, \boldsymbol{\theta}, \gamma, \mathbf{v})$ the objective function becomes linear in M, so after maximizing out these variables the overall objective is convex in M. Although this change of variable turns the objective into convex, it indeed shifts the intractability into the feasible region of M:

$$\mathcal{M}_0 := \underbrace{\{M \succeq \mathbf{0} : M_1 = I, \operatorname{tr}(M_{u,u}) \le 1, \operatorname{tr}(M_{r,r}) \le 1\}}_{=:\mathcal{M}_1} \cap \{M : \operatorname{rank}(M) = h\} . \quad (3.18)$$

Here $M \succeq \mathbf{0}$ means M is real symmetric and positive semi-definite. Due to the rank constraint, $\mathcal{M}_0$ is *not convex*. So a natural relaxation—the only relaxation we introduce besides

the proximal term in (Equation 3.8)—is to drop this rank constraint and optimize with the resulting convex set $\mathcal{M}_1$. This leads to the final convex formulation:

$$\min_{M\in\mathcal{M}_1}\mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{z})\sim\tilde{p}}\Big[\max_{\boldsymbol{\pi}}\max_{(\boldsymbol{\theta},\gamma)\in\mathcal{N}}\max_{\mathbf{v}} -G^*(\mathbf{v}) \; - \sigma_S(\boldsymbol{\pi}) - \boldsymbol{\theta}' M_u \mathbf{x} - \frac{1}{4\sigma\gamma} f(M,\boldsymbol{\pi},\boldsymbol{\theta},\gamma,\mathbf{v};\mathbf{x},\mathbf{z})\Big]. \tag{3.19}$$

To summarize, we have achieved a convex model for two-layer conditional models in which the latent structured representation is determined by a polar operator. Instead of bypassing this bi-level optimization via the normal loss based approach [37, 56], we addressed it directly by leveraging the optimality conditions of the inner optimization. A convex relaxation is then achieved via SDP.

3.5.1 Inducing low rank solutions

Although it is generally hard to provide a theoretical guarantee for nonlinear SDP relaxations, it is interesting to note that the constraint set $\mathcal{M}_1$ effectively encourages low-rank solutions (hence tighter relaxations). As a key technical result, we next show that all extreme points of $\mathcal{M}_1$ are rank h (the number of hidden nodes) for all $h \geq 2$. Recall that in sparse coding, the atomic norm framework [57] induces low-complexity solutions by setting up the optimization over the convex hull of atoms, or penalize via its gauge function. Therefore the characterization of the extreme points of $\mathcal{M}_1$ might open up the possibility of analyzing our relaxation by leveraging the results in sparse coding.

Lemma 2. *Let A_i be symmetric matrices. Consider the set of*

$$\mathcal{R} := \{X : X \succeq \mathbf{0},\ \mathrm{tr}(A_i X) \lesseqgtr b_i,\ i = 1, \ldots, m\}, \tag{3.20}$$

where m is the number of linear (in)equality constraints. $\lesseqgtr$ means it can be any one of $\leq$, $=$, or $\geq$. Then the rank r of all extreme points of $\mathcal{R}$ is upper bounded by

$$r \leq \left\lfloor \tfrac{1}{2}(\sqrt{8m+1} - 1) \right\rfloor. \tag{3.21}$$

This result extends [58] by accommodating inequalities in (Equation 3.20), and its proof is given in Appendix B.3. Now we show that the feasible region $\mathcal{M}_1$ as defined by (Equation 3.18) has all extreme points with rank h.

Theorem 4. *If $h \geq 2$, then all extreme points of $\mathcal{M}_1$ have rank h, and $\mathcal{M}_1$ is the convex hull of $\mathcal{M}_0$.*

Proof. Let M be an extreme point of $\mathcal{M}_1$. Noting that $M \succeq \mathbf{0}$ already encodes the symmetry of M, the linear constraints for $\mathcal{M}_1$ in (Equation 3.18) can be written as $\frac{1}{2}h(h+1)$ linear equality constraints and two linear inequality constraints. In total $m = \frac{1}{2}h(h+1) + 2$. Plug it into (Equation 3.21) in the above lemma

$$\mathrm{rank}(M) \leq \left\lfloor \tfrac{1}{2}(\sqrt{8m+1} - 1) \right\rfloor = \left\lfloor \tfrac{1}{2}(\sqrt{4h(h+1)+17} - 1) \right\rfloor = h + [\![h = 1]\!]. \tag{3.22}$$

Finally, the identity matrix in the top-left corner of M forces $\text{rank}(M) \geq h$. So $\text{rank}(M) = h$ for all $h \geq 2$. It then follows that $\mathcal{M}_1 = \text{conv}\mathcal{M}_0$. □

3.6 Application in Machine Learning Problems

The framework developed above is generic. For example, when $\mathcal{Y}$ represents classification for h classes by canonical vectors, $S = \text{conv}\mathcal{Y}$ is the h dimensional probability simplex (sum up to 1). Clearly $\sigma_S(\boldsymbol{\pi}) = \max_i \pi_i$, and $\mathcal{N} = \{(\mathbf{x}, t) \in \mathbb{R}_+^{h+1} : \mathbf{1}'\mathbf{x} = t\}$. In many applications, $\mathcal{Y}$ can be characterized as $\{\mathbf{y} \in \{0,1\}^h : A\mathbf{y} \leq \mathbf{c}\}$, where A is TUM and all entries of $\mathbf{c}$ are in $\{-1, 1, 0\}$.[1] In this case, the convex hull S has all extreme points being integral, and S employs an explicit form:

$$\mathcal{Y} = \{\mathbf{y} \in \{0,1\}^h : A\mathbf{y} \leq \mathbf{c}\} \quad \rightarrow \quad S = \text{conv}\mathcal{Y} = \{\mathbf{w} \in [0,1]^h : A\mathbf{w} \leq \mathbf{c}\}, \tag{3.23}$$

replacing all binary constraints $\{0, 1\}$ by intervals $[0, 1]$. Clearly TUM is a sufficient condition for PO-tractability, because $\min_{\mathbf{y} \in \mathcal{Y}} \mathbf{d}'\mathbf{y}$ is equivalent to $\min_{\mathbf{w} \in S} \mathbf{d}'\mathbf{w}$, an LP. Examples include the above graph matching and linear chain model. We will refer to $A\mathbf{w} \leq \mathbf{c}$ as the non-box constraint.

3.6.1 Graph matching

As the first concrete example, we consider convex relaxation for latent graph matching. One task in natural language is transliteration [42,59]. Suppose we are given an English word $\mathbf{e}$ with

---

[1] For simplicity, we write equality constraints (handled separately in practice) using two inequality constraints.

m letters, and a corresponding Hebrew word $\mathbf{h}$ with n letters. The goal is to predict whether $\mathbf{e}$ and $\mathbf{h}$ are phonetically similar, a binary classification problem with $z \in \{-1, 1\}$. However it obviously helps to find, as an intermediate step, the letter-wise matching between $\mathbf{e}$ and $\mathbf{h}$. The underlying assumption is that each letter corresponds to *at most* one letter in the word of the other language. So if we augment both $\mathbf{e}$ and $\mathbf{h}$ with a sink symbol * at the end (hence making their length $\hat{m} := m + 1$ and $\hat{n} := n + 1$ respectively), we would like to find a matching $\mathbf{y} \in \{0, 1\}^{\hat{m}\hat{n}}$ that minimizes the following cost

$$\begin{aligned} &\min_{Y \in \mathcal{Y}} \sum_{i=1}^{\hat{m}} \sum_{j=1}^{\hat{n}} Y_{ij} \mathbf{u}' \boldsymbol{\phi}_{ij}, \\ \text{where } \mathcal{Y} = \{0,1\}^{\hat{m} \times \hat{n}} \cap &\underbrace{\{Y : Y_{i,:}\mathbf{1} = 1, \forall i \le m,\ \mathbf{1}' Y_{:,j} = 1, \forall j \le n\}}_{=:\mathcal{G}}. \end{aligned} \tag{3.24}$$

Here $Y_{i,:}$ is the i-th row of Y. $\boldsymbol{\phi}_{ij} \in \mathbb{R}^p$ is a feature vector associated with the pair of i-th letter in $\mathbf{e}$ and j-th letter in $\mathbf{h}$, including the dummy *. Our notation omits its dependency on $\mathbf{e}$ and $\mathbf{h}$. $\mathbf{u}$ is a discriminative weight vector that will be learned from data. After finding the optimal Y^*, [42] uses the maximal objective value of (Equation 3.24) to make the final binary prediction: $-\operatorname{sign}(\sum_{ij} Y^*_{ij} \mathbf{u}' \boldsymbol{\phi}_{ij})$.

To pose the problem in our framework, we first notice that the non-box constraints $\mathcal{G}$ in (Equation 3.24) are TUM. Therefore, S is simply $[0, 1]^{\hat{m} \times \hat{n}} \cap \mathcal{G}$. Given the decoded $\mathbf{w}$, the output labeling principle above essentially duplicates $\mathbf{u}$ as the output layer weight. A key advantage of our method is to allow the weights of the two layers to be *decoupled*. By using a weight vector $\mathbf{r} \in \mathbb{R}^p$, we define the output score as $\mathbf{r}' \phi \mathbf{w}$, where ϕ is a p-by-$\hat{m}\hat{n}$ matrix whose (i, j)-th column

is $\boldsymbol{\phi}_{ij}$. So ϕ depends on $\mathbf{e}$ and $\mathbf{h}$. Overall, our model follows by instantiating (Equation 3.12) as:

$$\min_{\|U\|\leq 1} \min_{\|R\|\leq 1} \mathop{\mathbb{E}}_{(\mathbf{e},\mathbf{h},z)\sim\tilde{p}} \Big[\max_{\boldsymbol{\pi}} \max_{(\boldsymbol{\theta},\gamma)\in\mathcal{N}} \max_{v\in\mathbb{R}} \min_{\mathbf{w}} -z\mathbf{r}'\phi\mathbf{w} + v\mathbf{r}'\phi\mathbf{w} - G^*(v) + \boldsymbol{\pi}'\mathbf{w} \\ -\sigma_S(\boldsymbol{\pi}) + \sum\nolimits_{ij} (\mathbf{u}'\boldsymbol{\phi}_{ij} + \sigma w_{ij})(\gamma w_{ij} - \theta_{ij}) \Big]. \tag{3.25}$$

Once more we can minimize out $\mathbf{w}$, which gives rise to a quadratic,

$$(v-z)\phi'\mathbf{r} + \gamma\phi'\mathbf{u} + \boldsymbol{\pi} - \sigma\boldsymbol{\theta}^2.$$

It is again amenable to SDP relaxation, where $(M_{u,u}, M_{r,u}, M_{r,r})$ correspond to $(\mathbf{u}\mathbf{u}', \mathbf{r}\mathbf{u}', \mathbf{r}\mathbf{r}')$ resp.

3.6.2 Homogeneous temporal models

A variety of structured output problems are formulated with graphical models. We highlight the gist of our technique by using a concrete example: unsupervised structured learning for inpainting. Suppose we are given images of handwritten words, each segmented into p letters, and the latent representation is the corresponding letters. Since letters are correlated in their appearance in words, the recognition problem has long been addressed using linear chain conditional random fields. However imagine no ground truth letter label is available, and instead of predicting labels, we are given images in which a random small patch is occluded. So our goal will be inpainting the patches.

To cast the problem in our two-layer latent structure model, let each letter image in the word be denoted as a vector $\mathbf{x}_i \in \mathbb{R}^n$, and the reconstructed image be $\mathbf{z}_i \in \mathbb{R}^m$ ($m = n$ here). Let $Y_i \in \{0,1\}^{h \times h}$ ($h = 26$) encode the labels of the letter pair at position i and $i+1$ (as rows and columns of Y_i respectively). Let $U_v \in \mathbb{R}^{h \times n}$ be the letter-wise discriminative weights, and $U_e \in \mathbb{R}^{h \times h}$ be the pairwise weights. Then by (Equation 3.2), the MAP inference can be reformulated as (ref. definition of $\mathcal{H}$ in (Equation 3.3))

$$\min_{\{Y_i\} \in \mathcal{Y}} \sum_{i=1}^{p} \mathbf{1}' Y_i' U_v \mathbf{x}_i + \sum_{i=1}^{p-1} \operatorname{tr}(U_e' Y_i) \quad \text{where} \quad \mathcal{Y} = \left\{ \{Y_i\} : Y_i \in \{0,1\}^{C \times C} \right\} \cap \mathcal{H}. \tag{3.26}$$

Since the non-box constraints in $\mathcal{H}$ are TUM, the problem can be cast in our framework with $S = \operatorname{conv}\mathcal{Y} = \left\{ \{Y_i\} : Y_i \in [0,1]^{C \times C} \right\} \cap \mathcal{H}$. Finally to reconstruct the image for each letter, we assume that each letter j has a basis vector $\mathbf{r}_j \in \mathbb{R}^m$. So given W_i, the output of reconstruction is $R' W_i \mathbf{1}$, where $R = (\mathbf{r}_1, \ldots, \mathbf{r}_h)'$. To summarize, our model can be instantiated from (Equation 3.12) as

$$\begin{aligned} \min_{\|U\| \le 1} \min_{\|R\| \le 1} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{z}) \sim \tilde{p}} \Big[\max_{\pi} \max_{(\theta,\gamma) \in \mathcal{N}} \max_{\mathbf{v}} \min_{W} \sum_{i=1}^{p} (\mathbf{v}_i - \mathbf{z}_i)' R' W_i \mathbf{1} - G^*(\mathbf{v}_i) \\ + \operatorname{tr}(\pi' W) - \sigma_S(\pi) + \sum_{i=1}^{p} \operatorname{tr}((U_v \mathbf{x}_i \mathbf{1}' + [\![i \neq p]\!] U_e + \sigma W_i)'(\gamma W_i - \theta_i)) \Big]. \end{aligned} \tag{3.27}$$

Here $\mathbf{z}_i$ is the inpainted images in the training set. If no training image is occluded, then just set $\mathbf{z}_i$ to $\mathbf{x}_i$. The constraints on U and R can be refined, e.g. bounding $\|U_v\|$, $\|U_e\|$, and $\|\mathbf{r}_j\|$ separately. As before, we can derive a quadratic term,

$$R(\mathbf{v}_i - \mathbf{z}_i)\mathbf{1}' + \gamma U_v \mathbf{x}_i \mathbf{1}' + \gamma U_e + \pi_i - \sigma {\theta_i}^2,$$

by minimizing out W_i, which again leads to SDP relaxations. Even further, we may allow each letter to employ a *set* of principal components, whose combination yields the reconstruction (Appendix B.4).

Besides modeling flexibility, our method also accommodates problem-specific simplification. For example, the dimension of $\mathbf{w}$ is often much higher than the number of non-box constraints. Appendix B.5 shows that for linear chain, the dimension of $\mathbf{w}$ can be reduced from C^2 to C via partial Lagrangian.

3.7 Experiments

To empirically evaluate our convex method (henceforth referred to as CVX), we compared it with the state-of-the-art methods on two prediction problems with latent structure.

3.7.1 Transliteration

The first experiment is based on the English-Hebrew corpus [60]. It consists of 250 positive transliteration pairs for training, and 300 pairs for testing. On average there are 6 characters per word in each of the languages. All these pairs are considered "positive examples", and for negative examples we followed [42] and randomly sampled $t_- \in \{50, 75, 100\}$ pairs from

$250^2 - 250$ mismatched pairings (which are 20%, 30%, and 40% of 250, resp). We did not use many negative examples because, as per [42], our test performance measure will depend mainly on the highest few discriminative values, which are learned largely from the positive examples.

Given a pair of words $(\mathbf{e}, \mathbf{h})$, the feature representation $\boldsymbol{\phi}_{ij}$ for the i-th letter in $\mathbf{e}$ and j-th letter in $\mathbf{h}$ is defined as the unigram feature: an n-dimensional vector with all 0's except a single one in the (e_i, h_j)-th coordinate. In this dataset, there are $n = 655$ possible letter pairs (* included). Since our primary objective is to determine whether the convex relaxation of a two-layer model with latent structure can outperform locally trained models, we adopted this simple but effective feature representation (rather than delving into heuristic feature engineering).

Our test evaluation measurement is the Mean Reciprocal Rank (MRR), which is the average of the reciprocal of the rank of the correct answer. In particular, for each English word $\mathbf{e}$, we calculated the discriminative score of respective methods when $\mathbf{e}$ is paired with each Hebrew word in the test set, and then found the rank of the correct word (1 for the highest). The reciprocal of the rank is averaged over all test pairs, giving the MRR. So a higher value is preferred, and 50% means on average the true Hebrew word is the runner-up. For our method, the discriminative score is simply $f := \mathbf{r}'\Phi\mathbf{w}$ (using the symbols in (Equation 3.25)), and that for [42] is $f := \max_{Y \in \mathcal{Y}} \mathbf{u}'\Phi\text{vec}(Y)$ (vectorization of Y).

We compared our method (with $\sigma = 0.1$) against the state-of-the-art approach in [42]. It is a special case of our model with the second-layer weight $\mathbf{r}$ tied with the first-layer weight $\mathbf{u}$. They trained it using a local optimization method, and we will refer to it as Local. Both methods employ an output loss function $\max\{0, yf\}^2$ with $y \in \{+1, -1\}$, and both contain only

one parameter—the bound on $\|\mathbf{u}\|$ (and $\|\mathbf{r}\|$). We simply tuned it to optimize the performance of Local. The test MRR is shown in Figure 3, where the number of negative examples was varied in 50, 75, and 100. Local was trained with random initialization, and we repeated the random selection of the negative examples for 10 times, yielding 10 dots in each scatter plot. It is clear that CVX in general delivers significantly higher MRR than Local, with the dots lying above or close to the diagonal. Since this dataset is not big,

the randomness of the negative set leads to notable variations in the performance (for both methods).

3.7.2 Inpainting for occluded image

Our second experiment used structured latent model to inpaint images. We generated 200 sequences of images for training, each with $p \in \{4, 6, 8\}$ digits. In order to introduce structure, each sequence can be either odd (*i.e.* all digits are either 1 or 3) or even (all digits are 2 or 4). So $C = 4$. Given the digit label, the corresponding image ($\mathbf{x} \in [0, 1]^{196}$) was sampled from the MNIST dataset, downsampled to 14-by-14. 200 test sequences were also generated.

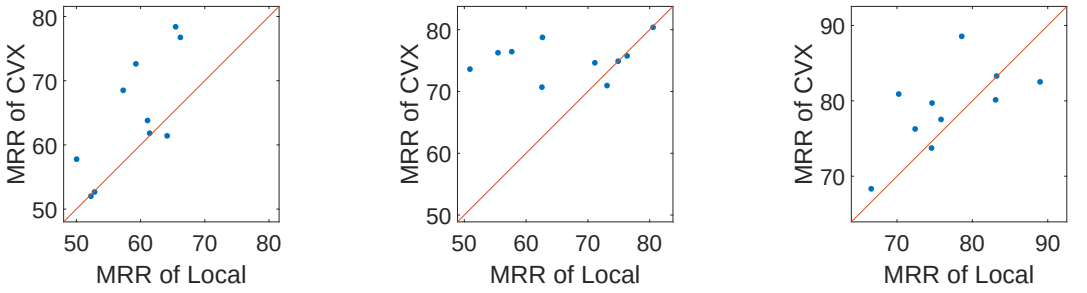
In the test data, we randomly set a $k \times k$ patch of each image to 0 as occluded ($k \in \{2, 3, 4\}$), and the task is to inpaint it. This setting is entirely unsupervised, with no digit label available for training. It falls in the framework of $X \to Y \to Z$, where X is the occluded input, Y is the latent digit sequence, and Z is the recovered image. In our convex method, we tied U_v with R and so we still have a 3-by-3 block matrix M, corresponding to I, U_v and U_e. We set σ to 10^{-1} and $G(\cdot) = \frac{1}{2}\|\cdot\|^2$ (Gaussian). Y was predicted using the polar operator, based on which Z was predicted with the Gaussian mean.

For comparison, we used CRF-AE, which was proposed very recently by [44]. Although it ties X and Z, extension to our setting is trivial by computing the expected value of Z given X. Here $P(Z|Y)$ is assumed a Gaussian whose mean is learned by maximizing $P(Z = \mathbf{x}|X = \mathbf{x})$, and we initialized all model parameters by unit Gaussian. For the ease of comparison, we introduced regularization by constraining model parameters to L_2 norm balls rather than penalizing the squared L_2 norm. For both methods, the radius bound was simply chosen as the maximum L_2 norm of the images, which produced consistently good results. We did not use higher k because the images are sized 14-by-14.

The error of inpainting given by the two methods is shown in Table Table VI where we varied the size of the occluded patch with p fixed to 6, and in Table Table VII where the length of the sequence p was varied while k was fixed to 4. Each number is the sum of squared error in the occluded patch, averaged over 5 random generations of training and test data (hence producing the mean and standard deviation). Here we can see that CVX gives lower error than CRF-AE. With no surprise, the error grows almost quadratically in k. When the length of sequence grows, the error of both CVX and CRF-AE fluctuates nonmonotonically. This is probably because with more images in each node, the total error is summed over more images, but the error per image decays thanks to the structure.

3.8 Conclusion and discussion

In this section, we presented another example of a two-layer neural network setting, where we wanted to explicitly incoporate latent structure information, while maintaining a jointly convex training objective. To this end, we started with a bi-level optimization problem based



(a) 50 negative examples (b) 75 negative examples (c) 100 negative examples

Figure 3: MRR of Local versus CVX over 50, 75, and 100 negative examples.

	Size of occluded patch ($k \times k$)		
	$k = 2$	$k = 3$	$k = 4$
CRF-AE	$0.29 \pm$ 0.01	$0.80 \pm$ 0.01	$1.31 \pm$ 0.02
CVX	$0.27 \pm$ 0.01	$0.79 \pm$ 0.01	$1.28 \pm$ 0.02

TABLE VI: Total inpainting error as a function of the size of occluded patch ($p = 8$).

	Length of sequence		
	$p = 4$	$p = 6$	$p = 8$
CRF-AE	$1.33 \pm$ 0.04	$1.30 \pm$ 0.02	$1.31 \pm$ 0.03
CVX	$1.29 \pm$ 0.04	$1.27 \pm$ 0.02	$1.28 \pm$ 0.03

TABLE VII: Total inpainting error as a function of the length of sequences ($k = 4$).

on the framework of total unimodularity (TUM), and we proposed a novel convex model using the first order optimality conditions of the inner level optimization, which resulted in a semi-definite program (SDP). The effectiveness of our model was demonstrated by the superior empirical performance over local training, along with low-rank characterization of the extreme points of the feasible region. A simple but interesting extension of this framework would be

to analyze the model when the latent layer employs submodularity, with its base polytope mirroring the support set S.

CHAPTER 4

CONVEX REPRESENTATION LEARNING FOR GENERALIZED INVARIANCE IN SEMI-INNER-PRODUCT SPACES

(Parts of this chapter were previously published as "Convex Representation Learning for Generalized Invariance in Semi-Inner-Product Space" [24] at the International Conference on Machine Learning (ICML 2020))

4.1 Introduction

Effective modeling of structural priors has been the workhorse of a variety of machine learning algorithms. Such priors are available in a rich supply, including invariance [61, 62], equivariance [63, 64], disentanglement [65, 66], homophily/heterophily [67], fairness [68], correlations in multiple views and modalities [69, 70], etc.

In this paper we focus on "generalized invariance", where certain relationship holds irrespective of certain changes in data. This extends traditional settings that are limited to, *e.g.*, transformation and permutation. For instance, in multilabel classification there are semantic or logical relationships between classes which hold for any input. Common examples include mutual exclusion and implication [71, 72]. In mixup [73], a convex interpolation of a pair of examples is postulated to yield the same interpolation of output labels.

While conventional wisdom learns models whose prediction accords with these structures, recent developments show that it can be more effective to learn structure-encoding represen-

tations. Towards this goal, the most straightforward approach is to directly parameterize the model. For example, deep sets model permutation invariance via an additive decomposition [74], convolutional networks use sparse connection and parameter sharing to model translational invariance, and a similar approach has been developed for equivariance [75]. Although they simplify the model and can enforce invariance over the *entire* space, their applicability is very restricted, because most useful structures do not admit a readily decomposable parameterization. As a result, most invariance/equivariance models are restricted to permutations and group based diffeomorphism.

In order to achieve significantly improved generality and flexibility, the regularization approach can be leveraged, which penalizes the violation of pre-specified structures. For example, [76] penalizes the norm of the Jacobian matrix to enforce contractivity, conceivably a generalized type of invariance. [77] proposed using a max-margin loss over all transformations [78]. However, for most structures, regularization leads to a nonconvex problem. Despite the recent progress in optimization for deep learning, the process still requires a lot of trial and error. Therefore a convex learning procedure will be desirable, because besides the convenience in optimization, it also offers the profound advantage of decoupling parameter optimization from problem specification: poor learning performance can only be ascribed to a poor model architecture, not to poor local minima.

Indeed convex invariant representation learning has been studied, but in limited settings. Tangent distance kernels [79] and Haar integration kernels are engineered to be invariant to a group of transformations [80–82], but it relies on sampling for tractable computation and the

sample complexity is $O(d/\epsilon^2)$ where d is the dimension of the underlying space. [83] treated all perturbations within an ellipsoid neighborhood as invariances, and it led to an expensive second order cone program (SOCP). Other distributionally robust formulations also lead to SOCP/SDPs [84]. The most related work is [85], which warped a reproducing kernel Hilbert space (RKHS) by linear functionals that encode the invariances. However, in order to keep the warped space an RKHS, their applicability is restricted to *quadratic* losses on linear functionals.

In practice, however, there are many invariances that cannot be modeled by quadratic penalties. For example, the logical relationships between classes impose an ordering in the discriminative output [71], and this can hardly be captured by quadratic forms. Similarly, when a large or infinite number of invariances are available, measuring the maximum violation makes more sense than their sum, and it is indeed the principle underlying adversarial learning [86]. Again this is not amenable to quadratic forms.

Our goal, therefore, is to develop a *convex* representation learning approach that efficiently incorporates generalized invariances as semi-norm functionals. Our first contribution is to show that compared with linear functionals, semi-norm functionals encompass a much broader range of invariance (Sections 4.5 and 4.6).

Our key tool is the semi-inner-product space [87], into which an RKHS can be warped by augmenting the RKHS norm with semi-norm functionals. A specific example of s.i.p. space is the reproducing kernel Banach space [88], which has been used for ℓ_p regularization in, *e.g.*, kernel SVMs, and suffers from high computational cost [89–94]. A s.i.p. space extends RKHS by relaxing the underlying inner product into a semi-inner-product, while retaining the most

important construct: *kernel function*. To the best of our knowledge, s.i.p. space has yet been applied to representation learning.

Secondly, we developed efficient computation algorithms for solving the regularized risk minimization (RRM) with the new s.i.p. norm (Section 4.3). Although [88] established the representer theorem from a pure mathematical perspective, no practical algorithm was provided and ours is the first to fill this gap.

However, even with this progress, RRMs still do not provide invariant representations of data instances; it simply learns a discriminant function by leveraging the representer theorem (which does hold in the applications we consider). So our third contribution, as presented in Section 4.4, is to learn and extract representations by embedding s.i.p. kernel representers in Euclidean spaces. This is accomplished in a *convex* and efficient fashion, constituting a secondary advantage over RRMs which is not convex in the dual coefficients. Different from Nyström or Fourier linearization of kernels in RKHS, the kernel representers in a s.i.p. space carry interestingly different meanings and expressions in primal and dual spaces. Finally, our experiments demonstrate that the new s.i.p.-based algorithm learns more predictive representations than strong baselines.

4.2 Preliminaries

Suppose we have an RKHS $\mathcal{H} = (\mathcal{F}, \langle \cdot, \cdot \rangle_{\mathcal{H}}, k)$ with $\mathcal{F} \subseteq \mathbb{R}^{\mathcal{X}}$, inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ and kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Our goal is to renorm $\mathcal{H}$ hence *warp the distance metric* by adding a functional R that induces desired structures.

4.2.1 Existing works on invariance modeling by RKHS

[95] and [96] proposed modeling invariances by bounded linear functionals in RKHS. Given a function f, the graph Laplacian is $\sum_{ij} w_{ij}(f(x_i) - f(x_j))^2$, and obviously $f(x_i) - f(x_j)$ is bounded and linear. Transformation invariance can be characterized by $\frac{\partial}{\partial \alpha}|_{\alpha=0} f(I(\alpha))$, where $I(\alpha)$ stands for the image after applying an α amount of rotation, translation, etc. It is again bounded and linear. By Riesz representation theorem, a bounded linear functional can be written as $\langle z_i, f\rangle_{\mathcal{H}}$ for some $z_i \in \mathcal{H}$.

Based on this view, [85] took a step towards representation learning. By adding $R(f)^2 := \sum_i \langle z_i, f\rangle_{\mathcal{H}}^2$ to the RKHS norm square, the space is warped to favor f that respects invariance, *i.e.*, small magnitude of $\langle z_i, f\rangle$. They showed that it leads to a new RKHS with a kernel

$$k^\circ(x_1, x_2) = k(x_1, x_2) - z(x_1)^\top (I + K_z)^{-1} z(x_2), \tag{4.1}$$

where $z(x) = (z_1(x), \ldots, z_m(x))^\top$ and $K_z = (\langle z_i, z_j\rangle)_{i,j}$.

Although the kernel representer of k° offers a new invariance aware representation, the requirement that the resulting space remains an RKHS forces the penalties in R to be quadratic on $\langle z_i, f\rangle$, significantly limiting its applicability to a broader range of invariances such as total variation $\int_x |f'(x)| \mathrm{d}x$. Our goal is to relax this restriction by enabling *semi-norm* regularizers with new tools in functional analysis, and illustrate its applications in Sections 4.5 and 4.6.

4.2.2 Semi-inner-product spaces

We first specify the range of regularizer R considered here.

Postulate 2. *We assume that $R : \mathcal{F} \to \mathbb{R}$ is a semi-norm. Equivalently, $R : \mathcal{F} \to \mathbb{R}$ is convex and $R(\alpha f) = |\alpha| R(f)$ for all $f \in \mathcal{F}$ and $\alpha \in \mathbb{R}$ (absolute homogeneity). Furthermore, we assume R is closed (i.e., lower semicontinuous)* w.r.t. *the topology in $\mathcal{H}$.*

Since R is closed convex and its domain is the entire Hilbert space $\mathcal{H}$, R must be continuous. By exempting R from being induced by an inner product, we enjoy substantially improved flexibility in modeling various regularities.

For most learning tasks addressed below, it will be convenient to directly construct R from the specific regularity. However, in some context it will also be convenient to constructively explicate R in terms of support functions.

Proposition 1. *$R(f)$ satisfies Assumption 2 if, and only if, $R(f) = \sup_{g \in S} \langle f, g \rangle_{\mathcal{H}}$, where $S \subseteq \mathcal{H}$ is bounded in the RKHS norm and is symmetric ($g \in S \Leftrightarrow -g \in S$).*

The proof is in Appendix C.1. Using R, we arrive at a new norm defined by

$$\|f\|_{\mathcal{B}} := \sqrt{\|f\|_{\mathcal{H}}^2 + R(f)^2}, \tag{4.2}$$

thanks to Assumption 2. It is immediately clear from Proposition 1 that $\|f\|_{\mathcal{H}} \leq \|f\|_{\mathcal{B}} \leq C \|f\|_{\mathcal{H}}$, for some constant $C > 0$ that bounds the norm of S. In other words, the two norms $\|\cdot\|_{\mathcal{H}}$ and $\|\cdot\|_{\mathcal{B}}$ are equivalent, hence in particular the norm $\|\cdot\|_{\mathcal{B}}$ is complete. We thus arrive at a Banach space $\mathcal{B} = (\mathcal{F}, \|\cdot\|_{\mathcal{B}})$. Note that both $\mathcal{H}$ and $\mathcal{B}$ have the same underlying vector space $\mathcal{F}$—the difference is in the norm or distance metric.

To proceed, we need to endow more structures on $\mathcal{B}$.

Definition 2 (Strict convexity)**.** *A normed vector space* $(\mathcal{F}, \|\cdot\|)$ *is strictly convex if for all* $\mathbf{0} \neq f, g \in \mathcal{F}$*,*

$$\|f + g\| = \|f\| + \|g\| \tag{4.3}$$

implies $g = \alpha f$ *for some* $\alpha \geq 0$*. Equivalently, if the unit ball* $\mathsf{B} := \{f \in \mathcal{F} : \|f\| \leq 1\}$ *is strictly convex.*

Using the parallelogram law it is clear that the Hilbert norm $\|\cdot\|_{\mathcal{H}}$ is strictly convex. Moreover, since summation preserves strict convexity, it follows that the new norm $\|\cdot\|_{\mathcal{B}}$ is strictly convex as well.

Definition 3 (Gâteaux differentiability)**.** *A normed vector space* $(\mathcal{F}, \|\cdot\|)$ *is Gâteaux differentiable if for all* $\mathbf{0} \neq f, g \in \mathcal{F}$*, there exists the directional derivative*

$$\lim_{t \in \mathbb{R}, t \to 0} \tfrac{1}{t}(\|f + tg\| - \|f\|). \tag{4.4}$$

We remark that both strict convexity and Gâteaux differentiability are algebraic but not topological properties of the norm. In other words, two equivalent (in terms of topology) norms may not be strictly convex or Gâteaux differentiable at the same time. For instance, the ℓ_2-norm on $\mathbb{R}^d$ is both strictly convex and Gâteaux differentiable, while the equivalent ℓ_1-norm is not.

Recall that $\mathcal{B}^*$ is the dual space of $\mathcal{B}$, consisting of all continuous linear functionals on $\mathcal{B}$ and equipped with the dual norm $\|F\|_{\mathcal{B}^*} = \sup_{\|f\|_{\mathcal{B}} \leq 1} F(f)$. The dual space of a normed (reflexive) space is Banach (reflexive).

Definition 4. *A Banach space $\mathcal{B}$ is reflexive if the canonical map $\jmath : \mathcal{B} \to \mathcal{B}^{**}$, $f \mapsto \langle \cdot ; \jmath f \rangle := \langle f ; \cdot \rangle$ is onto, where $\langle f ; F \rangle$ is the (bilinear) duality pairing between dual spaces. Here $\cdot$ is any element in $\mathcal{B}^*$.*

Note that reflexivity is a topological property. In particular, equivalent norms are all reflexive if any one of them is. As any Hilbert space $\mathcal{H}$ is reflexive, so is the equivalent norm $\|\cdot\|_{\mathcal{B}}$ in (Equation 4.2).

Theorem 5 ([97]). *A Banach space $\mathcal{B}$ is strictly convex (Gâteaux differentiable) if its dual space $\mathcal{B}^*$ is Gâteaux differentiable (strictly convex). The converse is also true if $\mathcal{B}$ is reflexive.*

Combining Proposition 1 and Theorem 5, we see that R, hence $\|\cdot\|_{\mathcal{B}}$, is Gâteaux differentiable if (the closed convex hull of) the set S in Proposition 1 is strictly convex.

We are now ready to define a semi-inner-product (s.i.p.) on a normed space $(\mathcal{F}, \|\cdot\|)$. We call a bivariate mapping $[\cdot, \cdot] : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$ a s.i.p. if for all $f, g, h \in \mathcal{F}$ and $\lambda \in \mathbb{R}$,

- additivity: $[f + g, h] = [f, h] + [g, h]$
- homogeneity: $[\lambda f, g] = [f, \lambda g] = \lambda [f, g]$,
- norm-inducing: $[f, f] = \|f\|^2$,
- Cauchy-Schwarz: $[f, g] \leq \|f\| \cdot \|g\|$.

We note that an s.i.p. is additive in its second argument iff it is an inner product (by simply verifying the parellelogram law). [87] proved that s.i.p. does exist on every normed space. Indeed, let the subdifferential $J = \partial \frac{1}{2}\|\cdot\|_{\mathcal{B}}^2 : \mathcal{B} \rightrightarrows \mathcal{B}^*$ be the (multi-valued) duality mapping. Then, any selection $j : \mathcal{B} \to \mathcal{B}^*, f \mapsto j(f) \in J(f)$ with the convention that $j(\mathbf{0}) = \mathbf{0}$ leads to a s.i.p.:

$$[f, g] := \langle f; j(g) \rangle \,. \tag{4.5}$$

Indeed, from definition, for any $f \neq \mathbf{0}$, $j(f) = \|f\|F$, where $\|F\|^* = 1$ and $\langle f; F\rangle = \|f\|$. A celebrated result due to [98] revealed the uniqueness of s.i.p. if the norm $\|\cdot\|$ is Gâteaux differentiable, and later [99] proved that the (unique) mapping j is onto iff $\mathcal{B}$ is reflexive. Moreover, j is 1-1 if $\mathcal{B}$ is strictly convex (like in (Equation 4.2)), as was shown originally in [98].

Let us summarize the above results.

Definition 5. *A Banach space $\mathcal{B}$ is called a s.i.p. space iff it is reflexive, strictly convex, and Gâteaux differentiable. Clearly, the dual $\mathcal{B}^*$ of a s.i.p. space is s.i.p. too.*

Theorem 6 (Riesz representation)**.** *Let $\mathcal{B}$ be a s.i.p. space. Then, for any continuous linear functional $f^* \in \mathcal{B}^*$, there exists a unique $f \in \mathcal{B}$ such that*

$$f^* = [\cdot, f] = j(f), \quad \textit{and} \quad \|f\| = \|f^*\|_{\mathcal{B}^*}. \tag{4.6}$$

From now on, we identify the duality mapping j with the star operator $f^* := j(f)$. Thus, we have a unique way to represent all continuous functionals on a s.i.p. space. Conveniently, the unique s.i.p. on the dual space follows from (Equation 4.5): for all $f^*, g^* \in \mathcal{B}^*$,

$$[f^*, g^*] := [g, f] = \langle g; f^* \rangle \,, \tag{4.7}$$

from which one easily verifies all properties of an s.i.p. Some literature writes $[f^*, g^*]_{\mathcal{B}^*}$, $[g, f]_{\mathcal{B}}$, $\langle g; f^* \rangle_{\mathcal{B}^*}$, and $\langle f; g^* \rangle_{\mathcal{B}}$ to make it explicit where the operations take place. We simplify these notations by omitting subscripts when the context is clear, but still write $\|f\|_{\mathcal{B}}$ and $\|f^*\|_{\mathcal{B}^*}$.

Finally, fix $x \in \mathcal{X}$ and consider the evaluation (linear) functional $\mathsf{ev}_x : \mathcal{B} \to \mathbb{R}$, $f \mapsto f(x)$. When ev_x is continuous (which indeed holds for our norm (Equation 4.2)), Theorem 6 implies the existence of a unique $G_x \in \mathcal{B}$ such that

$$f(x) = \mathsf{ev}_x(f) = [f, G_x] = [G_x^*, f^*]. \tag{4.8}$$

Varying $x \in \mathcal{X}$ we obtain a unique *s.i.p. kernel* $G : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that $G_x := G(\cdot, x) \in \mathcal{B}$. Thus, using s.i.p. we obtain the reproducing property:

$$f(x) = [f, G(\cdot, x)], \;\; G(x, y) = [G(\cdot, y), G(\cdot, x)]. \tag{4.9}$$

Different from a reproducing kernel in RKHS, G is not necessarily symmetric or positive semi-definite.

4.3 Regularized Risk Minimization

In this section we aim to provide a computational device for the following regularized risk minimization problem:

$$\min_{f\in\mathcal{H}} \quad \ell(f) + \|f\|_{\mathcal{H}}^2 + R(f)^2. \tag{4.10}$$

where $\ell(f)$ is the empirical risk depending on discriminant function values $\{f(x_j)\}_{j=1}^n$ for training examples $\{x_j\}$. Clearly, this objective is equivalent to

$$\min_{f\in\mathcal{B}} \quad \ell(f) + \|f\|_{\mathcal{B}}^2 . \tag{4.11}$$

Remark 1. *Unlike the usual treatment in reproducing kernel Banach spaces (RKBS) [88], we only require $\mathcal{B}$ to be reflexive, strictly convex and Gâteaux differentiable, instead of the much more demanding uniform convexity and smoothness. This more general condition not only suffices for our subsequent results but also simplifies the presentation. A similar definition like ours was termed pre-RKBS in [100].*

[88] established the representer theorem for RKBS: the optimal f for (Equation 4.11) has its dual form

$$f^* = \sum\nolimits_j c_j G_{x_j}^*, \tag{4.12}$$

where $\{c_j\}$ are real coefficients. To optimize $\{c_j\}$, we need to substitute (Equation 4.12) into (Equation 4.11), which in turn requires evaluating i) $\|f\|_{\mathcal{B}}^2$, which equals $\|f^*\|_{\mathcal{B}^*}^2$; ii) $f(x)$, which, can be computed through inverting the star operator as follows:

$$\|f^*\|_{\mathcal{B}^*} = \max_{\|h\|_{\mathcal{B}}\leq 1} \langle h; f^* \rangle = \max_{\|h\|_{\mathcal{B}}\leq 1} \sum\nolimits_j c_j \left\langle h; G_{x_j}^* \right\rangle = \max_{h:\|h\|_{\mathcal{H}}^2 + R(h)^2 \leq 1} \sum\nolimits_j c_j h(x_j),$$

where the last equality is due to (Equation 4.8) and (Equation 4.5). The last maximization step operates in the RKHS $\mathcal{H}$, and thanks to the strict convexity of $\|\cdot\|_{\mathcal{B}}$, it admits the unique solution

$$h = f/\|f\|_{\mathcal{B}} = f/\|f^*\|_{\mathcal{B}^*}, \tag{4.13}$$

because $\langle f; f^* \rangle = \|f\|_{\mathcal{B}}\|f^*\|_{\mathcal{B}^*}$, and $\mathcal{B}$ is a s.i.p. space.

We summarize this computational inverse below:

Theorem 7. *If $f^* = \sum_j c_j G_{x_j}^*$, then $f = \|f\|_{\mathcal{B}} f^\circ$, where*

$$f^\circ := \arg \max_{h:\|h\|_{\mathcal{H}}^2 + R(h)^2 \leq 1} \sum_j c_j h(x_j), \tag{4.14}$$

$$\|f\|_{\mathcal{B}} = \sum\nolimits_j c_j f^\circ(x_j) = \left\langle f^\circ, \sum\nolimits_j c_j k(x_j, \cdot) \right\rangle_{\mathcal{H}}. \tag{4.15}$$

In addition, the argmax in (Equation 4.14) *is attained uniquely.*

In practice, we first compute f° by solving (Equation 4.14), and then f can be evaluated at different x without redoing any optimization. As a special case, setting $f^* = G_x^*$ allows us to evaluate the kernel $G_x = G_x^\circ(x)G_x^\circ$.

Specialization to RKHS.

When $R(f)^2 = \sum_i \langle z_i, f\rangle_{\mathcal{H}}^2$, $\|\cdot\|_{\mathcal{B}}$ is induced by an inner product, making $\mathcal{B}$ an RKHS. Now we can easily recover (Equation 4.1) by applying Theorem 7, because the optimization in (Equation 4.14) with $f^* = G_x^*$ is

$$\max_{h\in\mathcal{H}} h(x), \quad \text{s.t.} \quad \|h\|_{\mathcal{H}}^2 + \sum\nolimits_k \langle z_k, h\rangle_{\mathcal{H}}^2 \le 1, \tag{4.16}$$

and its unique solution can be easily found in closed form:

$$G_x^\circ = \frac{k(\cdot,x) - (z_1,\ldots,z_m)(I+K_z)^{-1}z(x)}{(k(x,x) - z(x)^\top(I+K_z)^{-1}z(x))^{1/2}}. \tag{4.17}$$

Plugging into $G_x = G_x^\circ(x)G_x^\circ$, we recover (Equation 4.1).

Overall, the optimization of (Equation 4.11) may no longer be convex in $\{c_j\}$, because $f(x)$ is generally not linear in $\{c_j\}$ even though f^* is (since the star operator is not linear). In practice, we can initialize $\{c_j\}$ by training without $R(f)$ (*i.e.*, setting $R(f)$ to 0). Despite the nonconvexity, we have achieved a new solution technique for a broad class of inverse problems, where the regularizer is a semi-norm.

4.4 Convex Representation Learning by Euclidean Embedding

Interestingly, our framework—which so far only learns a predictive model—can be directly extended to learn structured *representations* in a *convex* fashion. In representation learning, one identifies an "object" for each example x, which, in our case, can be a function in $\mathcal{F}$ or a vector in Euclidean space. Such a representation is supposed to have incorporated the prior invariances in R, and can be directly used for other (new) tasks such as supervised learning without further regularizing by R. This is different from the regularized risk minimization in Section 4.3, which, although still enjoys the representer theorem in the applications we consider, only seeks a discriminant function f without providing a new representation for each example.

Our approach to convex representation learning is based on Euclidean embeddings (a.k.a. finite approximation or linearization) of the kernel representers in a s.i.p. space, which is analogous to the use of RKHS in extracting useful features. However, different from RKHS, G_x and G_x^* play different roles in a s.i.p. space, hence require *different* embeddings in $\mathbb{R}^d$. For any $f \in \mathcal{B}$ and $g^* \in \mathcal{B}^*$, we will seek their Euclidean embeddings $\iota(f)$ and $\iota^*(g^*)$, respectively. Note ι^* is just a notation, not to be interpreted as "the adjoint of ι."

We start by identifying the properties that a reasonable Euclidean embedding should satisfy intuitively. Motivated by the bilinearity of $\langle \cdot ; \cdot \rangle_{\mathcal{B}}$, it is natural to require

$$\langle f ; g^* \rangle_{\mathcal{B}} \approx \langle \iota(f), \iota^*(g^*) \rangle \,, \quad \forall f \in \mathcal{B}, g^* \in \mathcal{B}^*, \tag{4.18}$$

where $\langle \cdot, \cdot \rangle$ stands for Euclidean inner product. As $\langle \cdot ; \cdot \rangle_{\mathcal{B}}$ is bilinear, ι and ι^* should be linear on $\mathcal{B}$ and $\mathcal{B}^*$ respectively. Also note $\iota^*((f+g)^*) \neq \iota^*(f^*) + \iota^*(g^*)$ in general.

Similar to the linearization of RKHS kernels, we can apply invertible transformations to ι and ι^*. For example, doubling ι while halving ι^* makes no difference. We will just choose one representation out of them. It is also noteworthy that in general, $\|\iota(f)\|$ (Euclidean norm) approximates $\|f\|_{\mathcal{H}}$ instead of $\|f\|_{\mathcal{B}}$. (Equation 4.18) is the only property that our Euclidean embedding needs to satisfy.

We start by embedding the unit ball $\mathsf{B} := \{f \in \mathcal{F} : \|f\|_{\mathcal{B}} \leq 1\}$. Characterizing R by support functions as in Proposition 1, a natural Euclidean approximation of $\|\cdot\|_{\mathcal{B}}$ is

$$\|v\|_{\tilde{\mathcal{B}}}^2 := \|v\|^2 + \max_{g \in S} \langle v, \tilde{g} \rangle^2, \quad \forall\ v \in \mathbb{R}^d, \tag{4.19}$$

where $\tilde{g}$ is the Euclidean embedding of g in the original RKHS, designed to satisfy that $\langle \tilde{f}, \tilde{g} \rangle \approx \langle f, g \rangle_{\mathcal{H}}$ for all $f, g \in \mathcal{H}$ (or a subset of interest). Commonly used embeddings include Fourier [101], hash [102], Nyström [103], etc. For example, given landmarks $\{z_i\}_{i=1}^n$ sampled from $\mathcal{X}$, the Nyström approximation for a function $f \in \mathcal{H}$ is

$$\tilde{f} = K_z^{-1/2}(f(z_1), \ldots, f(z_n))^\top, \qquad \text{where} \quad K_z := [k(z_i, z_j)]_{i,j} \in \mathbb{R}^{n \times n}. \tag{4.20}$$

Naturally, the dual norm of $\|\cdot\|_{\tilde{\mathcal{B}}}$ is

$$\|u\|_{\tilde{\mathcal{B}}^*} := \max_{v : \|v\|_{\tilde{\mathcal{B}}} \leq 1} \langle u, v \rangle, \quad \forall\ u \in \mathbb{R}^d. \tag{4.21}$$

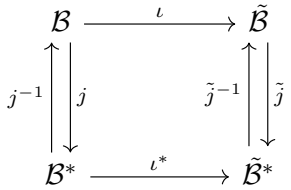


Figure 4: The commutative diagram for our embeddings.

Clearly the unit ball of $\|\cdot\|_{\tilde{\mathcal{B}}}$ and $\|\cdot\|_{\tilde{\mathcal{B}}^*}$ are also symmetric, and we denote them as $\tilde{\mathsf{B}}$ and $\tilde{\mathsf{B}}^*$, respectively.

As shown in Figure Figure 4, we have the following commutative diagram. Let $j : \mathcal{B} \to \mathcal{B}^*$ be the star operator and j^{-1} its inverse, and similarly for $\tilde{j} : \tilde{\mathcal{B}} \to \tilde{\mathcal{B}}^*$ and its inverse $\tilde{j}^{-1}$. Then, it is natural to require

$$\iota = \tilde{j}^{-1} \circ \iota^* \circ j, \tag{4.22}$$

where $\tilde{j}^{-1}$ can be computed for any $u := \iota^*(f^*)$ via a Euclidean counterpart of Theorem 7:

$$\tilde{j}^{-1}(u) := \|u\|_{\tilde{\mathcal{B}}^*} \cdot \arg\max_{v \in \tilde{\mathsf{B}}} \langle v, u \rangle . \tag{4.23}$$

The argmax is unique because $\|\cdot\|_{\tilde{\mathcal{B}}}$ is strictly convex.

At last, how can we get $\iota^*(f^*)$ in the first place? We start from the simpler case where f^* has a kernel expansion as in (Equation 4.12).[1] Here, by the linearity of ι^*, it will suffice to compute $\iota^*(G_x^*)$. By Theorem 7,

$$G_x = G_x^\circ(x) G_x^\circ, \quad \text{where} \quad G_x^\circ := \arg\max_{h \in \mathsf{B}} h(x)$$

is uniquely attained. Denoting $k_y := k(\cdot, y)$, it follows

$$\left\langle G_x; G_y^* \right\rangle_{\mathcal{B}} \overset{\text{by (Equation 4.8)}}{=} G(x,y) = \langle G_x, k_y \rangle_{\mathcal{H}} = \langle G_x^\circ(x) G_x^\circ, k_y \rangle_{\mathcal{H}} .$$

So by comparing with (Equation 4.18), it is natural to introduce

$$\iota^*(G_y^*) := \tilde{k}_y, \tag{4.24}$$

$$\iota(G_x) := G_x^\circ(x) \tilde{G}_x^\circ \approx \left\langle \tilde{G}_x^\circ, \tilde{k}_x \right\rangle \tilde{G}_x^\circ, \tag{4.25}$$

$$\text{where} \quad \tilde{G}_x^\circ := \arg\max_{v \in \tilde{\mathsf{B}}} \left\langle v, \tilde{k}_x \right\rangle . \tag{4.26}$$

---

[1] We stress that although the kernel expansion (Equation 4.12) is leveraged to *motivate* the design of ι^*, the underlying foundation is that the span of $\{G_x^* : x \in \mathcal{X}\}$ is dense in $\mathcal{B}^*$ (Theorem 8). The representer theorem [88], which showed that the solution to (Equation 4.11) must be in the form of (Equation 4.12), is *not* relevant to our construction.

The last optimization is *convex* and can be solved very efficiently because, thanks to the positive homogeneity of R, it is equivalent to

$$\min_{v} \left\{ \|v\|^2 + \max_{g \in S} \langle v, \tilde{g} \rangle^2 \right\} \quad s.t. \quad v^\top \tilde{k}_x = 1. \tag{4.27}$$

Detailed derivation and proof are relegated to Appendix C.3. To solve (Equation 4.27), LBFGS with projection to a hyperplane (which has a trivial closed-form solution) turned out to be very efficient in our experiment. Overall, the construction of $\iota(f)$ and $\iota^*(f^*)$ for f^* from (Equation 4.12) proceeds as follows:

1. Define $\iota^*(G_x^*) = \tilde{k}_x$;
2. Define $\iota^*(f^*) = \sum_i \alpha_i \tilde{k}_{x_i}$ for $f^* = \sum_i \alpha_i G_{x_i}^*$;
3. Define $\iota(f)$ based on $\iota^*(f^*)$ by using (Equation 4.22).

In the next subsection, we will show that these definitions are sound, and both ι and ι^* are linear. However, the procedure may still be inconvenient in computation, because f needs to be first dualized to f^*, which in turn needs to be expanded into the form of (Equation 4.12). Fortunately, our representation learning only needs to compute the embedding of G_x, bypassing all these computational challenges.

4.4.1 Analysis of Euclidean Embeddings

The previous derivations are based on the necessary conditions for (Equation 4.18) to hold. We now show that ι and ι^* are well-defined, and are linear. To start with, denote the base

Euclidean embedding on $\mathcal{H}$ by $T : \mathcal{H} \to \mathbb{R}^d$, where $T(f) = \tilde{f}$. Then by assumption, T is linear and $\tilde{k}_x = T(k(\cdot, x))$.

Theorem 8. *$\iota^*(f^*)$ is well defined for all $f^* \in \mathcal{B}^*$, and $\iota^* : \mathcal{B}^* \to \mathbb{R}^d$ is linear. That is,*

a) If $f^* = \sum_i \alpha_i G^*_{x_i} = \sum_j \beta_j G^*_{z_j}$ are two different expansions of f^*, then $\sum_i \alpha_i \tilde{k}_{x_i} = \sum_j \beta_j \tilde{k}_{z_j}$.

b) The linear span of $\{G^*_x : x \in \mathcal{X}\}$ is dense in $\mathcal{B}^*$. So extending the above to the whole $\mathcal{B}^*$ is straightforward thanks to the linearity of T.

We next analyze the linearity of ι. To start with, we make two assumptions on the Euclidean embedding of $\mathcal{H}$.

Postulate 3 (surjectivity)**.** *For all $v \in \mathbb{R}^d$, there exists a $g_v \in \mathcal{H}$ such that $\tilde{g}_v = v$.*

Assumption 3 does not cost any generality, because it is satisfied whenever the d coordinates of the embedding are linearly independent. Otherwise, this can still be enforced easily by projecting to an orthonormal basis of $\{\tilde{g} : g \in \mathcal{H}\}$.

Postulate 4 (lossless)**.** $\left\langle \tilde{f}, \tilde{g} \right\rangle = \langle f, g \rangle_{\mathcal{H}}$ *for all $f, g \in \mathcal{H}$. This is possible when,* e.g.*, $\mathcal{H}$ is finite dimensional.*

Theorem 9. *$\iota : \mathcal{B} \to \mathbb{R}^d$ is linear under Assumptions 3 & 4.*

Although Theorems 8 and 9 appear intuitive, the proof for the latter is rather nontrivial and is deferred to Appendix C.1. Some lemmas there under Assumptions 3 and 4 may be of interest too, hence highlighted here.

1. $\langle \iota(f), \iota^*(g^*) \rangle = \langle f; g^* \rangle$, $\forall\ f \in \mathcal{B}, g^* \in \mathcal{B}^*$.

2. $\|g\|_{\mathcal{B}} = \|g^*\|_{\mathcal{B}^*} = \|\iota^*(g^*)\|_{\tilde{\mathcal{B}}^*} = \|\iota(g)\|_{\tilde{\mathcal{B}}}$, $\forall\ g \in \mathcal{B}$.

3. $\tilde{\mathsf{B}} = \iota(\mathsf{B}) := \{\iota(f) : \|f\|_{\mathcal{B}} \leq 1\}$.

4. $\tilde{\mathsf{B}}^* = \iota^*(\mathsf{B}^*) := \{\iota^*(g^*) : \|g^*\|_{\mathcal{B}^*} \leq 1\}$.

5. $\max_{v \in \tilde{\mathsf{B}}} \langle v, \iota^*(g^*) \rangle = \max_{f \in \mathsf{B}} \langle f; g^* \rangle$, $\quad \forall g^* \in \mathcal{B}^*$.

4.4.2 Analysis under Inexact Euclidean Embedding

When Assumption 4 is unavailable, Theorem 8 still holds, but the linearity of ι has to be relaxed to an approximate sense. To analyze it, we first rigorously quantify the inexactness of the Euclidean embedding T. Consider a subspace based embedding, such as Nyström approximation. Here T satisfies that there exists a countable set of orthonormal bases $\{e_i\}_{i=1}^{\infty}$ of $\mathcal{H}$, such that

1. $Te_k = 0$ for all $k > d$,

2. $\langle Tf, Tg \rangle = \langle f, g \rangle_{\mathcal{H}}$, $\quad \forall f, g \in V := \text{span}\{e_1, \ldots, e_d\}$.

Clearly the Nyström approximation in (Equation 4.20) satisfies these conditions, where $d = n$, and $\{e_1, \ldots, e_d\}$ is any orthornormal basis of $\{k_{z_1}, \ldots, k_{z_d}\}$ (assuming d is no more than the dimensionality of $\mathcal{H}$).

Definition 6. *$f \in \mathcal{H}$ is called ϵ-approximable by T if*

$$\left\| f - \sum_{i=1}^{d} \langle f, e_i \rangle_{\mathcal{H}}\, e_i \right\|_{\mathcal{H}} \leq \epsilon. \tag{4.28}$$

In other words, the component of f in $V^\perp$ is at most ϵ.

Theorem 10. *Let $f, g \in \mathcal{F}$ and $\alpha \in \mathbb{R}$. Then $\iota(\alpha f_1) = \alpha\iota(f_1)$. If f, g, and all elements in S are ϵ-approximable by T, then*

$$|\langle \iota(f), \iota^*(g^*)\rangle - \langle f; g^*\rangle| = O(\sqrt{\epsilon}) \tag{4.29}$$

$$\|\iota(f+g) - \iota(f) - \iota(g)\| = O(\sqrt{\epsilon}). \tag{4.30}$$

The proof is in Appendix C.2.

To summarize, the primal embedding $\iota(G_x)$ as defined in (Equation 4.25) provides a new feature representation that incorporates structures in the data. Based on it, a simple linear model can be trained to achieve the desired regularities in prediction. We now demonstrate its flexibility and effectiveness on two example applications.

4.5 Application 1: Mixup

Mixup is a data augmentation technique [73], where a pair of training examples x_i and x_j are randomly selected, and their convex interpolation is postulated to yield the same interpolation of output labels. In particular, when $y_i \in \{0,1\}^m$ is the one-hot vector encoding the class that x_i belongs to, the loss for the pair is

$$\mathbb{E}_\lambda[\ell(f(\underbrace{\lambda x_i + (1-\lambda)x_j}_{=:\,\tilde{x}_\lambda}), \underbrace{\lambda y_i + (1-\lambda)y_j}_{=:\,\tilde{y}_\lambda})]. \tag{4.31}$$

Existing literature relies on stochastic optimization, with a probability pre-specified on λ. This is somewhat artificial. Changing expectation to maximization appears more appealing, but no longer amenable to stochastic optimization.

To address this issue and to learn representations that incorporate mixup prior while also accommodating classification with multiclass or even structured output, we resort to a joint kernel $k((x, y), (x', y'))$, whose simplest form is decomposed as $k^x(x, x')k^y(y, y')$. Here k^x and k^y are separate kernels on input and output respectively. Now a function $f(x, y)$ learned from the corresponding RKHS quantifies the "compatibility" between x and y, and the prediction can be made by $\arg\max_y f(x, y)$. In this setting, the $R(f)$ for mixup regularization can leverage the ℓ_p norm of $g_{ij}(\lambda) := \frac{\partial}{\partial\lambda} f(\tilde{x}_\lambda, \tilde{y}_\lambda)$ over $\lambda \in [0, 1]$, effectively accounting for an infinite number of invariances.

Theorem 11. *$R_{ij}(f) := \|g_{ij}(\lambda)\|_p$ satisfies Assumption 2 for all $p \in (1, \infty)$. The proof is in Appendix C.1.*

Clearly taking *expectation* or *maximization* over all pairs of n training examples still satisfies Assumption 2. In our experiment, we will use the ℓ_∞ norm, which despite not being covered by Theorem 11, is directly amenable to the embedding algorithm. More specifically, for each pair (x, y) we need to embed $k((\cdot, \cdot), (x, y))$ as a $d \times m$ matrix. This is different from the conventional setting where each example x employs one feature representation shared for all classes; here

the representation changes for different classes y. To this end, we need to first embed each invariance $g_{ij}(\lambda)$ by

$$Z_\lambda^{ij} := \frac{\partial}{\partial\lambda}\left(\tilde{k}_{\hat{x}_\lambda}\tilde{y}_\lambda^\top\right) = \left(\frac{\partial}{\partial\lambda}\tilde{k}_{\hat{x}_\lambda}\right)\tilde{y}_\lambda^\top + \tilde{k}_{\hat{x}_\lambda}(y_i - y_j)^\top.$$

Letting $\langle A, B\rangle = \mathrm{tr}(A^\top B)$ and $\|V\|_{\mathsf{F}}^2 = \langle V, V\rangle$, the Euclidean embedding $\iota(G_{x,y})$ can be derived by solving (Equation 4.27):

$$\min_{V\in\mathbb{R}^{d\times m}}\left\{\alpha\|V\|_{\mathsf{F}}^2 + \frac{1}{n^2}\sum_{ij}\max_{\lambda\in[0,1]}\left\langle V, Z_\lambda^{ij}\right\rangle^2\right\} \tag{4.32}$$

$$s.t.\ \left\langle V, \tilde{k}_x y^\top\right\rangle = 1. \tag{4.33}$$

Although the maximization over λ in (Equation 4.32) is not concave, it is one dimensional and grid search can solve it globally with $O(1/\epsilon)$ complexity. In practice, a local solver like L-BFGS almost always found its global optimum in 10 iterations.

4.6 Application 2: Embedding Inference for Structured Multilabel Prediction

In output space, there is often prior knowledge about pairwise or multi-way relationships between labels/classes. For example, if an image represents a cat, then it must represent an animal, but not a dog (assuming there is at most one object in an image). Such logic relationships of implication and exclusion can be highly useful priors for learning [71, 72]. One way to leverage it is to perform inference at test time so that the predicted multilabel conforms to these logic. However, this can be computation intensive at test time, and it will be ideal if the

predictor has already accounted for these logic, and at test time, one just needs to make binary decisions (relevant/irrelevant) for each individual category separately. We aim to achieve this by learning a representation that embeds this structured prior.

To this end, it is natural to employ the joint kernel framework. We model the implication relationship of $y_1 \to y_2$ by enforcing $f(x, y_2) \geq f(x, y_1)$, which translates to a penalty on the amount by which $f(x, y_1)$ is above $f(x, y_2)$

$$[f(x, y_1) - f(x, y_2)]_+, \quad \text{where } [z]_+ = \max\{0, z\}. \tag{4.34}$$

To model the mutual exclusion relationship of $y_1 \nleftrightarrow y_2$, intuitively we can encourage that $f(x, y_1) + f(x, y_2) \leq 0$, *i.e.*, a higher likelihood of being a cat demotes the likelihood of being a dog. It also allows both y_1 and y_2 to be irrelevant, *i.e.*, both $f(x, y_1)$ and $f(x, y_2)$ are negative. This amounts to another sublinear penalty on f:

$$[f(x, y_1) + f(x, y_2)]_+. \tag{4.35}$$

To summarize, letting $\tilde{p}$ be the empirical distribution, we can define $R(f)$ by

$$R(f)^2 := \mathop{\mathbb{E}}_{x \sim \tilde{p}} \left[\max_{y_1 \to y_2} [f(x, y_1) - f(x, y_2)]_+^2 + \max_{y_1 \nleftrightarrow y_2} [f(x, y_1) + f(x, y_2)]_+^2 \right]. \tag{4.36}$$

It is noteworthy that although $R(f)$ is positively homogeneous and convex (hence sublinear), it is no longer absolutely homogeneous and therefore not satisfying Assumption 2. However,

the embedding algorithm is still applicable without change. It will be interesting to study the presence of kernel function G in spaces "normed" by sublinear functions. We leave it for future work.

4.7 Experiments

Here we highlight the major results and experiment setup. Details on data preprocessing, experiment setting, optimization, and additional results are given in Appendix C.5.

4.7.1 Sanity check for s.i.p. based methods

Our first experiment tries to test the effectiveness of optimizing the regularized risk (Equation 4.11) with respect to the dual coefficients $\{c_j\}$ in (Equation 4.12). We compared 4 algorithms: SVM with Gaussian kernel; Warping which incorporates transformation invariance by kernel warping as described in [85]; Dual which trains the dual coefficients $\{c_j\}$ by LBFGS to minimize empirical risk as in (Equation 4.11); Embed which finds the Euclidean embeddings by convex optimization as in (Equation 4.27), followed by a linear classifier. The detailed derivation of the gradient in $\{c_j\}$ for Dual is relegated to Appendix C.4.

Four transformation invariances were considered, including rotation, scaling, and shifts to the left and upwards. Warping summed up the square of $\frac{\partial}{\partial \alpha}|_{\alpha=0} f(I(\alpha))$ over the four transformations, while Dual and Embed took their max as the $R(f)^2$. To ease the computation of derivative, we resorted to finite difference for all methods, with two pixels for shifting, 10 degrees for rotation, and 0.1 unit for scaling. No data augmentation was applied.

All algorithms were evaluated on two binary classification tasks: 4 v.s. 9 and 2 v.s. 3, both sampling 1000 training and 1000 test examples from the MNIST dataset. Since the square

TABLE VIII: Test accuracy of minimizing empirical risk on binary classification tasks.

	SVM	Warping	Dual	Embed
4 v.s. 9	97.1	98.0	97.6	97.8
2 v.s. 3	98.4	99.1	98.7	98.9

loss on the invariances used by Warping makes good sense, the purpose of this experiment is *not* to show that the s.i.p. based methods are better in this setting. Instead we aim to perform a sanity check on a) good solutions can be found for the nonconvex optimization over the dual variables in Dual, b) the Euclidean embedding of s.i.p. representers performs competitively. As Table Table VIII shows, both of the checks turned out affirmative, with both Dual and Embed delivering similar accuracy as Warping. In addition, Embed achieved higher accuracy than dual optimization, suggesting that the learned representations have well captured the invariances and possess better predictive power.

4.7.2 Mixup

We next investigated the performance of Embed on mixup.

4.7.2.0.1 Datasets.

We experimented with three image datasets: MNIST, USPS, and Fashion MNIST, each containing 10 classes. From each dataset, we drew n example for training and n examples for testing, with n varied in 500 and 1000. Based on the training data, p number of pairs were drawn from it.

Both Vanilla and Embed used Gaussian RKHS, along with Nyström approximation whose landmark points consisted of the entire training set. The vanilla mixup optimizes the objective (Equation 4.31) averaged over all sampled pairs. Following [73], The λ was generated from a Beta distribution, whose parameter was tuned to optimize the performance. Again, Embed was trained with a linear classifier.

Algorithms.

We first ran mixup with stochastic optimization where pairs were drawn on the fly. Then we switched to batch training of mixup (denoted as Vanilla), with the number of sampled pair increased from $p = n$, $2n$, up to $5n$. It turned out when $p = 4n$, the performance already matches the best test accuracy of the online stochastic version, which generally witnesses much more pairs. Therefore we also varied p in $\{n, 2n, 4n\}$ when training Embed. each setting was evaluated 10 times with randomly sampled training and test data. The mean and standard deviation are reported in Table Table IX.

Results.

As Table Table IX shows, Embed achieves higher accuracy than Vanilla on almost all datasets and combinations of n and p. The margin tends to be higher when the training set size (n and p) is smaller. Besides, Vanilla achieves the highest accuracy at $p = 4n$.

4.7.3 Structured multilabel prediction

Finally, we validate the performance of Embed on *structured multilabel prediction* as described in Section 4.6, showing that it is able to capture the structured relationships between the class labels (implication and exclusion) in a hierarchical multilabel prediction task.

Datasets.

We conducted experiments on three multilabel datasets where additional information is available about the hierarchy in its class labels [104]: Enron [105], WIPO [106], Reuters [107]. Implication constraints were trivially derived from the hierarchy, and we took siblings (of the same parent) as *exclusion* constraints. For each dataset, we experimented with $100/100, 200/200, 500/500$ randomly drawn train/test examples.

Algorithms.

We compared Embed with two baseline algorithms for multilabel classification: a multilabel SVM with RBF kernel (ML-SVM), and an SVM that incorporates the hierarchical label constraints (HR-SVM) [108]. No inference is conducted at test time, such as removing violations of implications or exclusions known a priori.

Results.

Table X reports the accuracy on the three train/test splits for each of the datasets. Clearly, Embed outperforms both the baselines in most of the cases.

4.8 Conclusions and Future Work

In this paper, we introduced a new framework of representation learning where an RKHS is turned into a semi-inner-product space via a semi-norm regularizer, broadening the applicability of kernel warping to *generalized* invariances, *i.e.*, relationships that hold irrespective of certain changes in data. For example, the mixup regularizer enforces smooth variation irrespective of the interpolation parameter λ, and the structured multilabel regularizer enforces logic relationships between labels regardless of input features. Neither of them can be modeled convexly

TABLE IX: Test accuracy on mixup classification task based on 10 random runs.

Dataset		$n = 500$			$n = 1000$		
	p	n	$2n$	$4n$	n	$2n$	$4n$
MNIST	Vanilla	90.16±1.40	90.93±1.01	91.40±1.04	91.00±1.17	92.01±1.21	92.48±1.03
	Embed	**91.36**±1.41	**91.90**±1.08	**92.11**±1.01	**92.51**±1.01	**92.79**±0.98	**93.03**±1.00
USPS	Vanilla	90.54±1.28	91.76±1.14	92.40±1.25	93.87±1.19	94.72±1.12	95.32±1.13
	Embed	**92.46**±1.24	**93.02**±1.12	**93.21**±1.14	**94.74**±0.97	**95.11**±0.94	**95.67**±0.96
Fashion MNIST	Vanilla	79.37±3.11	81.15±2.08	81.72±1.96	82.53±1.49	83.13±1.36	83.69±1.31
	Embed	**81.56**±2.27	**82.16**±1.56	**82.52**±1.49	**83.28**±1.48	**84.07**±1.32	**84.34**±1.31

TABLE X: Test accuracy on multilabel prediction with logic relationship

Dataset	Embed			ML-SVM			HR-SVM		
	100	200	500	100	200	500	100	200	500
Enron	**96.2**±0.3	**95.7**±0.2	**94.7**±0.2	92.7±0.4	91.8±0.4	91.0±0.3	93.1±0.3	92.5±0.3	92.0±0.2
Reuters	**95.7**±1.4	**97.2**±1.2	**98.0**±0.4	94.2±1.4	95.1±1.3	95.2±1.2	95.1±1.2	**97.3**±1.3	97.7±1.3
WIPO	**98.6**±0.1	**98.4**±0.1	98.4±0.1	98.1±0.3	98.2±0.2	98.3±0.1	98.3±0.1	**98.5**±0.1	**98.7**±0.2

by conventional methods in transformation invariance, and the framework can also be directly applied to non-parametric transformations [109]. An efficient Euclidean embedding algorithm was designed and its theoretical properties are analyzed. Favorable experimental results were demonstrated for the above two applications.

This new framework has considerable potential of being applied to other invariances and learning scenarios. For example, it can be directly used in maximum mean discrepancy and the Hilbert–Schmidt independence criterion, providing efficient algorithms that complement the mathematical analysis in [110]. It can also be applied to convex deep neural networks [22, 23], which convexify multi-layer neural networks through kernel matrices of the hidden layer outputs.

Other examples of generalized invariance include *convex* learning of: a) node representations in large networks that are robust to topological perturbations [111]. The exponential number of perturbation necessitates max instead of sum; b) equivariance based on the largest deviation under swapped transformations over the input domain [75]; and c) co-embedding multiway relations that preserve co-occurrence and affinity between groups [112].

CHAPTER 5

REPRESENTATION LEARNING FOR MINIMIZING CATASTROPHIC FORGETTING IN DEEP NEURAL NETWORKS

5.1 Introduction

In order for real world artificial intelligence systems to get closer to human level intelligence, it is imperative for machine learning models to learn continuously, as and when the data becomes available. For instance, a multi-object detection system can be enriched by training on new objects incrementally. In this case, the model has to learn new images and new class labels in order to learn new objects. In some cases, a machine learning model might have to learn what it already knows, but just better. For example, an image classifier which was previously trained on images taken from the front view of a camera, can benefit from training with images captured from the top view or other angles later on. Here, the label set (of the training data) remains the same, but the model acquires richer details about already seen objects. All of the above mentioned examples are use cases of continual learning, where a model is provided with a sequence of tasks (data) and is expected to acquire new information on top of the existing knowledge, by learning from the sequence. Formally, we are given a finite sequence of tasks $\mathcal{T} = \{T_i\}_{i=1}^{n}$, where each $T_i = \{\mathbf{x}_{ij}, y_{ij}\}$ is a labelled dataset, and $\mathbf{x}_{ij}$ is the j^{th} example of the i^{th} task. In the continual learning setting, a machine learning model is expected to learn

incrementally on tasks from $\mathcal{T}$ and after learning on all the tasks, the model is expected to perform well on the all the tasks in $\mathcal{T}$.

Recently, the machine learning scientific community has been working on problems such as (1) life long machine learning [113], where the knowledge accumulated by learning in the past is used to solve new problems in the future, (2) transfer learning, where the gained knowledge is re-purposed on a related, but different task, (3) multi-task learning, where a model is expected to learn multiple different tasks (often a subset of a huge main task) in expectation that the model will perform better on the main task after learning several sub tasks. All these above mentioned popular and important machine learning problems are continual learning problems.

Indeed, in a continual learning setup, training each T_i is a standard supervised learning task and can be handled by modern neural network models quite effectively. Unfortunately, successively training a modern neural network model on $\mathcal{T}$ leads to a well known problem, where the model exhibits degradation of performance on the old tasks gradually, as it gets trained on the newer tasks. This phenomenon is called *catastrophic forgetting* [114]. Trivially, the ML model can be retrained with the old and the new tasks jointly to mitigate catastrophic forgetting. But with increase in size or the number of tasks (model update with new data), such joint training approaches can become extremely inefficient.

Algorithmic solutions to deal with catastrophic forgetting for continual learning have been recently proposed. One of the simplest approaches proposed, was to add a regularizer to the parameters of the model to reduce catastrophic forgetting [115]. [116, 117] retains a part or the whole of the existing network untouched and expands the network while adding new tasks.

They share parameters between tasks aiming to retain the outputs of the old tasks. A few other works such as EWC (Elastic Weight Consolidation) [114], while not strictly enforcing network retention, try to minimally change the network parameters, as new tasks are learned. In order to perform well on new tasks, they adapt the model parameters for the new tasks while trying to perform well on the old tasks. Authors of GEM [118] replay examples from earlier tasks when training on new tasks. Works such as generative replay (GR) [119] and [1], while not storing the past examples directly, learn a generative model of already seen examples so that they can generate them for replay. Incremental moment matching (IMM) [120] explicitly matches the network parameters of the old and new tasks to constrain the learned parameters to be closer between tasks.

In a nutshell, most of the above mentioned solutions attempt to match tasks or predictions on tasks, in one way or the other, in order to mitigate catastrophic forgetting and have shown to be effective in doing so. However, all these methods assume the existence of strong correlations between tasks, which may or may not exist. In the absence of such correlations, continual learning algorithms could suffer significant loss of performance and forgetting due to the so-called *negative transfer* phenomenon. On the other hand, accounting for such regularities could in fact result in the opposite (and a more beneficial) effect called positive transfer, where learning on a new task improves performance on an older task. The question then naturally is, how to minimize catastrophic forgetting and aid positive transfer?

To that end, in this work we develop new regularizers in the framework of semi-inner-product (s.i.p) space [85] that promotes learning representations that distill salient information

from previously learned tasks while also simultaneously improving classification accuracies on newer tasks. The learned representations when used as input features to an existing continual learning algorithm are shown to improve its performance significantly, especially when individual learning tasks are distributed differently.

5.2 Preliminaries

5.2.1 Short introduction to the existing solution

Our solution to address catastrophic forgetting for continual learning extends the work of [1]. The authors of [1] propose a new model architecture along with a specialized training procedure that is aimed to address forgetting in every stage of the training process. Their model consists of the following main components.

- Data Generator DG
- Dynamic Parameter Generator (DPG) $f(.)$
- Solver / Classifier S

S is a neural network parameterized by θ. The parameters θ are further divided into 2 sets; (1) θ_0 - that learns features that are common to all the datasets, (2) p_i - parameters specific to a given dataset. p_i are generated dynamically by the DPG network for each task.

$f(.)$ is another neural network that takes as input a feature embedding z_{ij} (generated by DG) and predicts a parameter set p_i. p_i is then used by S (in addition to θ_0) to make predictions.

DG is typically an auto-encoder (Wasserstein Auto Encoder (WAE) in this case) that has the following uses.

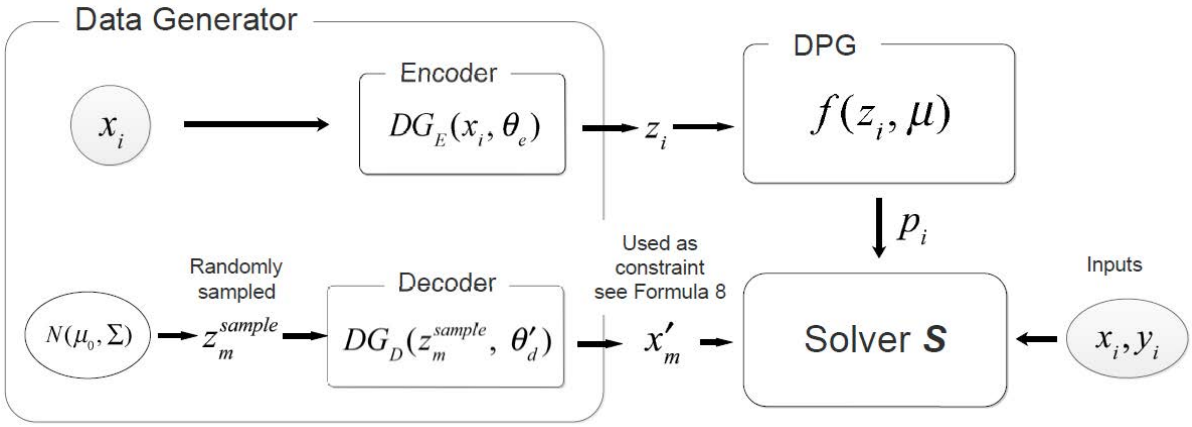


Figure 5: Training pipeline of [1]. (Image credit Hu et al. [1]).

1. Encode the raw input x_i to a lower dimensional subspace z_i - to be consumed by DPG.
2. Replay examples from an already "seen" dataset, which is used to ensure that the information learned in the previous training iteration is intact.

The overall architecture is given in Figure 5. [1] addresses the catastrophic forgetting problem by training the various neural network modules in their pipeline to explicitly remember information from previously trained datasets using specially designed loss functions. Additionally, the dynamic component of the solver S, which is generated for every incoming task ensures that accurate predictions are made on new datasets.

Loss functions employed by [1].

The authors of [1] use several loss functions in order to efficiently optimize the various components (S, f, and DPG) mentioned in the previous section to minimize forgetting. This

is of course in addition to the standard classification loss (for discriminative training of S) and reconstruction loss (for training the auto encoder).

We are particularly interested in the mechanism for mitigating catastrophic forgetting in S. Here [1] uses an extension of the so-called distillation loss, which constrains the outputs of two successive sets of learned parameters to be close to each other. Specifically, if θ_i and θ_{i-1} are the learned parameters of S in successive iterations $i, i-1$, then the proposed loss function minimizes the following objective.

$$\begin{aligned} &\min_{\mu,\theta_0} \mathcal{L}_{ce}(S(x_i, \theta_i), y_i) \\ &\text{s.t.} \sum_{m=1}^{M} ||\mathcal{R}(x'_m, \theta_i) - \mathcal{R}(x'_m, \theta_{i-1})|| < \epsilon_r \end{aligned} \tag{5.1}$$

where $\mathcal{R}(.)$ denotes the output of a layer of the solver S and $\mathcal{L}_{ce}$ is the standard cross-entropy loss. Simply put, the loss function constraints the predictions of S to be closer to each other. In addition to Equation 5.1, [1] has other losses to address forgetting in the autoencoder.

5.2.2 Unsupervised Domain Adaptation

In this section, we will describe a different but related problem in machine learning called domain adaptation. We are specifically interested in a special case of domain adaptation called unsupervised domain adaptation (UDA) where data comes from two domains: source and target [70]. In both domains, the input takes value in $\mathcal{X}$, and the label space is $\mathcal{Y}$. The source data has distribution P_s while the target data has distribution P_t. In the covariate shift setting, $P_t(X)$ differs from $P_s(X)$ while the concept $P(Y|X)$ does not change, *i.e.*, the

conditional distribution of label given the input is invariant across domains. Our goal is then to learn accurate predictors for the target domain using labeled examples in the source domain along with *unlabeled* examples in the target domain.

Domain alignment techniques attempt to match the feature distribution across domains. Ideally, it would seek a measurable feature mapping $\tilde{g}$ such that the class *conditional* distributions are aligned: $\tilde{g}\#P_s(\cdot|y) \approx \tilde{g}\#P_t(\cdot|y)$ for all $y \in \mathcal{Y}$, where $\tilde{g}\#P$ is the push-forward of a distribution P under $\tilde{g}$. However, since no label is available in the target domain, most approaches resort to aligning $\tilde{g}\#P_s(x)$ and $\tilde{g}\#P_t(x)$, which unfortunately does not guarantee the alignment of $\tilde{g}\#P_s(\cdot|y)$ and $\tilde{g}\#P_t(\cdot|y)$.

To address this issue, [70] proposed co-regularization for unsupervised domain adaptation (Co-DA). In particular, it learns two feature extractors $\tilde{g}_1$ and $\tilde{g}_2$, followed by predictors $\tilde{h}_1$ and $\tilde{h}_2$ respectively. In addition to aligning $\tilde{g}_1\#P_s$ and $\tilde{g}_1\#P_t$ (likewise for $\tilde{g}_2\#P_s$ and $\tilde{g}_2\#P_t$), it further promotes that $(\tilde{h}_1 \circ \tilde{g}_1)(x)$ and $(\tilde{h}_2 \circ \tilde{g}_2)(x)$ yield similar predictions on unlabeled target examples x from P_t. This is not only feasible in computation as no target example label is required, it also improves generalization by trimming the hypothesis space of $\tilde{g}_i$ using an additional prior: some $\tilde{g}_1$ and $\tilde{g}_2$ pairs may not admit any $\tilde{h}_2$ for a given $\tilde{h}_1$ such that $\tilde{h}_1 \circ \tilde{g}_1$ agrees with $\tilde{h}_2 \circ \tilde{g}_2$ on target domain examples.

Despite the improved empirical performance of Co-DA, it follows a regularization approach which suffers from two limitations. Firstly, the weights in the model must cater for both risk minimization and regularization simultaneously, a contention that generally exacerbates sample complexity and complicates optimization—the output of hidden nodes (the object under

regularization) is not itself the optimization variable. In contrast, it will be much more efficient if the structures are directly encoded in the representation. For example, deep sets incorporate invariance and equivariance by factorization in parameterization [74], which is far more efficient than penalizing the violation of invariance. Furthermore, the desired regularities in latent representation are only enforced on training data, and whether they are exhibited on test data relies solely on generalization. It would be ideal if such structures can be explicitly enforced and upheld in test data.

Towards these goals, we will first develop a new representation learning framework in semi-inner-product spaces (s.i.p) which captures all the essential structures in Co-DA [70] for learning under the UDA setting. We will then intuitively motivate how UDA is closely related to our original continual learning setting. Finally, we will develop a training procedure, based on [1] that improves the model's performance on several artificial continual learning tasks.

5.3 Kernel warping for unsupervised domain adaptation

In [121], coregularization was proposed in a kernel warping framework that aims to make use of unlabeled data. Here two RKHSs $\mathcal{H}_1$ and $\mathcal{H}_2$ are defined over different "views" of the data with kernels k_1 and k_2, and discriminant functions l and r are sought from $\mathcal{H}_1$ and $\mathcal{H}_2$

respectively, such that they agree on unlabeled data $U := \{x_i\}$, *i.e.*, $l(x) \approx r(x)$ for all $x \in U$.[1] This leads to a new norm on the sum space $\mathcal{H}_1 \oplus \mathcal{H}_2 := \{l+r : l \in \mathcal{H}_1,\ r \in \mathcal{H}_2\}$: for $f \in \mathcal{H}_1 \oplus \mathcal{H}_2$,

$$\|f\|^2 = \min_{l \in \mathcal{H}_1, r \in \mathcal{H}_2 : l+r=f} R(l,r)^2, \quad \text{where} \quad R(l,r)^2 := \|l\|^2_{\mathcal{H}_1} + \|r\|^2_{\mathcal{H}_2} + \sum\nolimits_{x \in U} (l(x) - r(x))^2.$$

Intuitively, it renorms the functions in $\mathcal{H}_1 \oplus \mathcal{H}_2$, favoring those that can be decomposed into $\mathcal{H}_1$ and $\mathcal{H}_2$ with small discrepancy on unlabeled data $\{x_i\}$. In this warped RKHS, the kernel representer for a data instance z can be computed by [24]

$$\arg \max_{f \in \mathcal{H}_1 \oplus \mathcal{H}_2 : \|f\| \le 1} f(z) = l + r, \quad \text{where} \quad (l,r) = \arg \max_{l \in \mathcal{H}_1, r \in \mathcal{H}_2 : R(l,r) \le 1} l(z) + r(z). \tag{5.2}$$

[121] then proved that warping the norm on $\mathcal{H}_1 \oplus \mathcal{H}_2$ as above can reduce the Rademacher complexity. Our goal is to develop new regularizers R that can model a variety of desirable structures. The above R was a Hilbert norm (*i.e.*, corresponds to an inner product) and $\mathcal{H}_1 \oplus \mathcal{H}_2$ an RKHS. This is a severe restriction, and we will extend R to semi-norms.

Since mini-batch based training has been commonly used in machine learning, we can extend kernel warping to a mini-batch $\{z_i\}_{i=1}^m$ by using $(\mathcal{H}_1 \oplus \mathcal{H}_2)^m$, and define a mini-batch based

---

[1] It is tempting to use f_1 for $\mathcal{H}_1$ and f_2 for $\mathcal{H}_2$. However, as can be seen in the sequel, this will complicate the notation. So to keep the notation intuitive, we use l for "left" view ($\mathcal{H}_1$) and r for "right" view ($\mathcal{H}_2$).

norm as $R_b(\{l_i, r_i\}) := \max_i R(l_i, r_i)$. Then the kernel representation for $\{z_i\}_{i=1}^m$ is $\{l_i, r_i\}_{i=1}^m$, where

$$\{l_i, r_i\}_{i=1}^m = \arg \max_{R_b(\{l_i, r_i\}) \leq 1} \sum\nolimits_i l_i(z_i) + r_i(z_i). \tag{5.3}$$

It is easy to see that the optimal (l_i, r_i) is the same as applying (Equation 5.2) to each z_i separately. However, formulating the kernel warping in a mini-batch setting will allow us to model more refined structures that can only be captured in the context of mini-batches, *e.g.*, correlations.

It is also noteworthy that R_b is no longer a Hilbert norm. So the definition in (Equation 5.3) in fact uses the semi-inner-product space from [24]. Besides, some examples in a mini-batch are from the source domain, and some are from the target domain. Let us call them as S and T respectively, so that $S \cup T = [m]$, $S \subseteq \mathcal{S}$ (original source dataset), $T \subseteq \mathcal{T}$ (original target dataset). We will use the shorthand of $\mathbb{E}_{j \sim \tilde{P}_S}[l_j] := \frac{1}{|S|} \sum_{j \in S} l_j$. It is then natural to enforce coregularization on T (since it is unlabeled):

$$\begin{aligned} V_1 &:= \text{mean}_{i \in [m]} \left\{ \mathbb{E}_{j \sim \tilde{P}_T}[(l_i(x_j) - r_i(x_j))^2] \right\} \quad \text{or} \quad \max_{i \in [m]} \left\{ \mathbb{E}_{j \sim \tilde{P}_T}[(l_i(x_j) - r_i(x_j))^2] \right\}. \\ &= \frac{1}{m|T|} \sum_{i=1}^m \sum_{j \in T} (\tilde{k}_{x_j,1}^\top \tilde{l}_i - \tilde{k}_{x_j,2}^\top \tilde{r}_i)^2 \end{aligned} \tag{5.4}$$

V_1 is convex and 2-homogeneous. When computing the Euclidean embedding $\tilde{l}_i$, we simply replace $l_i(x_j)$ by $\tilde{k}_{x_j}^\top \tilde{l}_i$.

5.3.1 Alignment

The original $\Delta\mathcal{H}\Delta$-distance for domain adaptation [122], when applied to the left view, is

$$d_{\Delta\mathcal{H}\Delta} := \max_{h,h'\in H} \left|\Pr_{x\sim P_s}[h(x) \neq h'(x)] - \Pr_{x\sim P_t}[h(x) \neq h'(x)]\right| \tag{5.5}$$

$$= \max_{h,h'\in H} \left|\mathbb{E}_{x\sim P_s}[\delta(h(x) \neq h'(x))] - \mathop{\mathbb{E}}_{x\sim P_t}[\delta(h(x) \neq h'(x))]\right|, \tag{5.6}$$

where δ is the indicator function valued in $\{0, 1\}$. Here H can represent the unit ball of $\mathcal{H}_1$, and we only assume that H is symmetric. Obviously the $\Delta\mathcal{H}\Delta$-distance can also be computed for the "right" view. Since the indicator function is not continuous, we can approximate it by square loss:

$$d_{\Delta\mathcal{H}\Delta} \approx \max_{h,h'\in H} \left|\mathbb{E}_{x\sim P_s}[(h(x) - h'(x))^2] - \mathbb{E}_{x\sim P_t}[(h(x) - h'(x))^2]\right|, \tag{5.7}$$

$$= \max_{h\in 2H} \left|\mathbb{E}_{x\sim P_s}[h(x)^2] - \mathbb{E}_{x\sim P_t}[h(x)^2]\right| \quad \text{(since only } h - h' \text{ matters)} \tag{5.8}$$

$$= \max_{h\in 2H} \left|\left\langle \mu, h^2 \right\rangle_{\mathcal{H}_1}\right| \text{ (if } h^2 \in \mathcal{H}_1), \quad \text{where } \mu = \mathbb{E}_{x\sim P_s}[k_1(x, \cdot)] - \mathbb{E}_{x\sim P_t}[k_1(x, \cdot)]. \tag{5.9}$$

Since h^2 is often hard to characterize (or not even in $\mathcal{H}_1$), the most prevalent approach is to penalize by the RKHS norm of μ. This immediately leads to a counterpart in our model based on mini-batch

$$V_{2,l}^{ez} = \left\| \mathbb{E}_{i\sim\tilde{P}_S}[l_i] - \mathbb{E}_{i\sim\tilde{P}_T}[l_i] \right\|_{\mathcal{H}_1}^2. \tag{5.10}$$

However, it turns out we can obtain a finer control of $d_{\Delta\mathcal{H}\Delta}$ than $\|\mu\|_{\mathcal{H}_1}$. Note $h(x)$ in (Equation 5.8) stands for the discriminant value of x. In our framework, we first compute G_x and then multiply with a weight $v \in \mathcal{H}_1$ with unit RKHS norm. So $h(x_i)$ is replaced by $\langle v, l_i \rangle$, and (Equation 5.8) naturally motivates us to define

$$V_{2,l} = \max_{\|v\|_{\mathcal{H}_1} \leq 1} \left| \mathbb{E}_{i \sim \tilde{P}_S} [\langle v, l_i \rangle^2_{\mathcal{H}_1}] - \mathbb{E}_{i \sim \tilde{P}_T} [\langle v, l_i \rangle^2_{\mathcal{H}_1}] \right|. \tag{5.11}$$

Clearly $V_{2,l}$ is 2-homogeneous, but not convex in $\{l_i\}$. Its Euclidean embedding version is equal to the largest magnitude eigenvalue of $\mathbb{E}_{i \sim \tilde{P}_S}[\tilde{l}_i \tilde{l}_i^\top] - \mathbb{E}_{i \sim \tilde{P}_T}[\tilde{l}_i \tilde{l}_i^\top]$.

In practice, we find that $V_{2,l}$ is indeed quite likely to be convex as long as the Nyström dimension is greater than m. We randomly generated two sets of $\{\tilde{l}_i, \tilde{r}_i\}$, took their mean, and checked if its $V_{3,l}$ value is lower than the mean of the $V_{2,l}$ values at these two sets. This holds true almost all the time.

Putting it all together

Eventually, our regularizer on $\mathcal{M} := \{l_i, r_i\}_{i=1}^m$ is

$$R(\mathcal{M}) = \max_{i \in [m]} \left(\|l_i\|^2_{\mathcal{H}_1} + \|r_i\|^2_{\mathcal{H}_2} \right) + V_1(\mathcal{M}) + V_{2,l}(\{l_i\}) + V_{2,r}(\{r_i\}). \tag{5.12}$$

Here we omitted the weights for V_1 and $V_{2,\cdot}$, and $V_{2,\cdot}$ are not convex. As long as their weights are not large, the strongly convex terms in V_1 and RKHS norm square will keep the overall R convex. R must be 2-homogeneous.

5.4 Preventing catastrophic interference via kernel warping

Earlier works on continual learning [1, 120] have shown that aligning successive datasets in some form has helped mitigate catastrophic forgetting. This has been achieved by several means such as, parameter matching [120], parameter sharing [1] and network prediction matching [1]. While alignment helps in minimizing forgetting, some prior work also use techniques to improve supervised learning performance in the overall continual learning task. For instance, [1] uses a mechanism to dynamically adapt the model parameters to new datasets.

In hindsight, we observe that the above mentioned properties are exactly the modeling objectives of the regularizers we developed earlier for the UDA problem. The feature alignment regularizer V_2 (Equation 5.11) encourages the pairs of input features to be probabilistically aligned, in effect reinforcing salient features that are common to both of the datasets. On the other hand, the regularizer V_1 (Equation 5.4) has been shown to improve classification accuracies on new (unlabelled) examples by making predictions of any classifier consistent across multiple views [70] of the data.

The key idea behind our proposed solution is to incorporate these regularizers to obtain representations that facilitate continual learning in the framework of kernel warping. In our experiments, we use the learned representations as inputs to the solver S and DPG of the PGMA architecture [1].

5.4.1 Training details

When we observe samples from a new dataset $T_i = \{x_{ij}, y_{ij}\}$ for supervised learning, we first apply kernel warping with T_i as the source dataset and (memorized) samples from the

previous dataset T_{i-1} as the target dataset as in (Equation 5.12). Recall that the regularizers in the kernel warping only require labelled examples from the source examples. Therefore we only need the input samples from the target dataset and not the labels. The warped source samples are then used for training S and DPG using the distillation loss (Equation 5.1) and the standard classification loss. Likewise, when we observe T_{i+1} we warp it with T_i (as the target). However by doing so, we overwrite the previously learned representations for T_i. This makes the representations of T_i and T_{i-1} obsolete and can lead to performance drop at inference time. To work around this problem, after warping T_{i+1} with T_i, we recursively warp all the previous T_j with its immediate predecessor in the continual learning setup, T_{j-1}. T_1 is warped only with T_2 after T_2 is observed.

In our setup, we also remove the autoencoder from [1]'s architecture. The autoencoder was primarily meant for replaying examples from earlier training. We use a memory bank of a subset of previously trained samples (again only input samples and not labels) for learning new representations for the current and future tasks. The complete training algorithm is presented in Algorithm 4 and an illustration of the training pipeline is given in Figure 6.

5.5 Experiments

In order to test the effectiveness of the proposed algorithm, we ran two sets of experiments and compared it with state-of-the-art baselines.

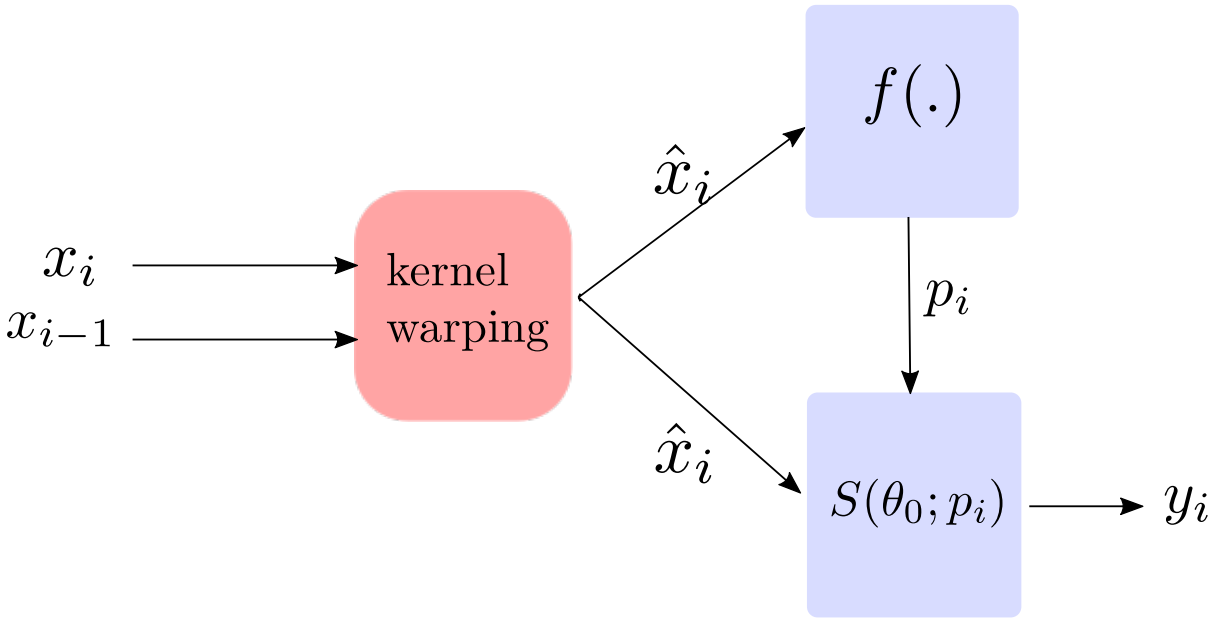


Figure 6: Training pipeline with kernel warping to minimize catastrophic forgetting. Here $\hat{x}_i$ represents the warped embeddings.

5.5.1 Experiment setup

(a) Shuffle and disjoint continual learning.

First, we followed [1] and created two continual learning setups - *shuffle* and *disjoint* on standard image classification datasets MNIST [123] and CIFAR10 [40].

- In the disjoint setup, a task is first divided into disjoint subsets based on its label set. Each subset is then treated as a new dataset for continual learning. For instance consider the MNIST digit dataset which has 10 classes $\{1 \dots 10\}$. In one of the setups, the dataset is divided into two disjoint groups with examples corresponding to classes $\{1 \dots 5\}$ as the first dataset and examples with labels $\{6 \dots 10\}$ as the second.

Algorithm 4 Training of our proposed solution with kernel warping.

Input: $T = \{T_i\}_{i=1}^{N}$, where $T_i = \{x_{ij}, y_{ij}\}$
Initial: Randomly initialize $f(.), S$;
// Learning the first task.
for all $n = 0 \ldots$ until convergence **do**
  Sample a mini-batch from $\hat{T}_1$
  // Training f and S
  Compute p_n
  Construct S_n using p_n and θ_0
  Minimize $\mathcal{L}_{ce}$ (Equation 5.1) and update S and f
end for
// Learning subsequent tasks
for all $i = 2;\ i \leq N;\ i++$ **do**
  for all $j = i;\ j \geq 1;\ j--$ **do**
    Apply Kernel Warping with $X_s = T_j$ and $X_t = T_{j-1}$ to obtain new representations $\hat{T}_j$ and $\hat{T}_{j-1}$
    Update the representations // $T_j \leftarrow \hat{T}_j$; $T_{j-1} \leftarrow \hat{T}_{j-1}$
  end for
  for all $n = 0, \ldots,$ until convergence **do**
    Sample a mini-batch from T_i
    Compute p_n
    Construct S_n using p_n and θ_0
    Minimize $\mathcal{L}_{ce}$ and update f and S
  end for
end for

- In the shuffle setup, new datasets are created by randomly shuffling the pixels in the input images of the original dataset. The label set is kept intact for all the datasets.

(b) Continual learning under different marginal input distributions.

Second, we created a continual learning (we call this task `MARGDIFF`) setup where the label space remains the same for all the datasets in the continual learning setup. However, the marginal distributions of the input images between any two pairs of datasets are significantly different. This setup simulates a commonly faced problem while accruing training data for supervised training of machine learning models, where the task, say image classification, is fixed and new images of already "seen" objects are being constantly collected to improve the model performance. In this case, joint (re)training of the newly collected data along with the

existing dataset is often expensive and difficult to maintain. Therefore treating the model update as a continual learning task is often useful in practise.

In order to simulate the experiment mentioned above, we considered three digit datasets: MNIST, USPS, Fashion-MNIST. We created a two-task continual learning problem, by pairing up the above mentioned tasks. The results of this experiment is given in Table XII

5.5.2 Baselines

The setup for the first experiment (shuffle and disjoint datasets) is exactly the same as [1] and we perform experiments on all the image datasets reported by [1]. So for this task, we compare ourselves with [1] and all their baselines, namely EWC [114], IMM [120], and `NO-CL`. [124]. For the second task (`MARGDIFF`), we use `NO-CL ADA` and `WARP-QUAD` as the baselines. `WARP-QUAD` here refers to our representation learning algorithm but only containing quadratic penalties (especially $V^{ez}_{2,l}$ instead of $V_{2,l}$). This baseline is meant to showcase the power of non-quadratic penalties that the s.i.p based representations that our framework can capture.

5.5.3 Results and Discussion

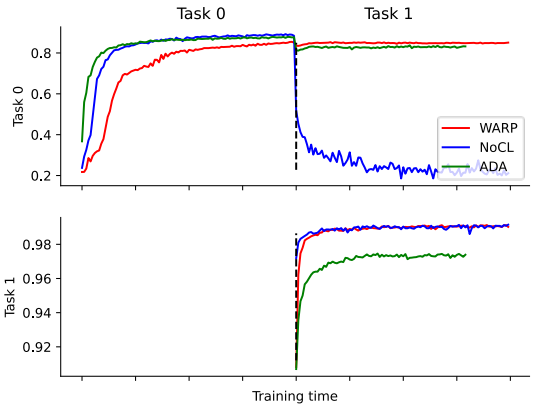
Table XI compares `WARP` with the baselines mentioned in the previous section on shuffle and disjoint experiments on the MNIST dataset. `WARP` performs competitively on all the tasks with a slight improvement over `ADA` on disjoint tasks.

As mentioned earlier, the `MARGDIFF` task tests the algorithms for forgetting when the individual tasks in the continual learning setup are distributed differently. Table XII records the average test accuracies on a two-task experiment on `MARGDIFF` comparing against `ADA` and `NO-CL`, showing significant performance gains by `WARP` over its baselines. To understand the re-

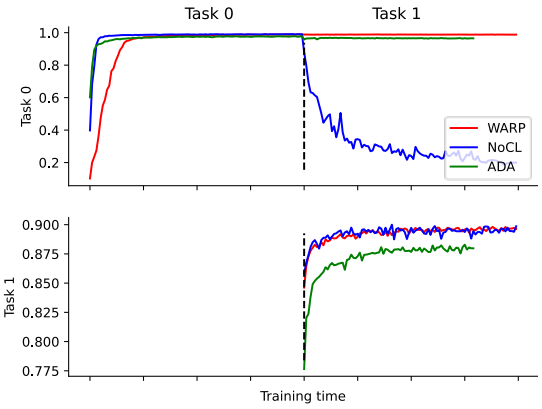
sult on this experiment, we further plotted the test accuracies for every 100 training iterations, similar to [1]. Here we found that in some cases with the warped representations, the solver S is able to get improved performance on Task 0. This can be attributed to the "positive transfer" effect on S on training with a new dataset, enabled by our algorithm. The effect can be visibly seen in subplots (d) and (e) of Figure 7. Also, just to highlight the hardness of transfer caused by forgetting, we plotted the result of `NO-CL` (No continual learning solution).

5.6 Conclusion

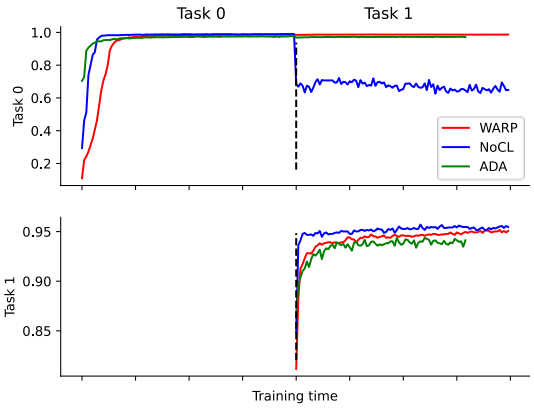
In this work, we developed a new representation learning algorithm for overcoming catastrophic forgetting for continual learning in neural networks. Experimental results show a marked increase in classification performance on continual learning tasks when trained with the learned representations as input. In future, we would like to explore new constraints that can address catastrophic forgetting in neural networks that can be modeled in our representation learning framework.



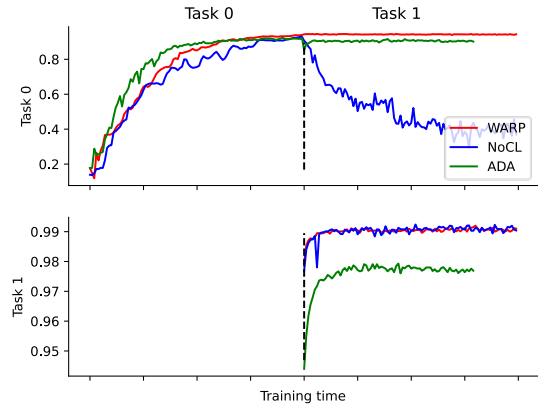
(a) Fashion-MNIST, MNIST



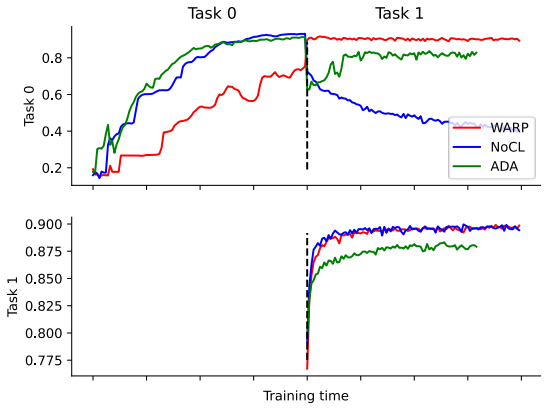
(b) MNIST, Fashion-MNIST



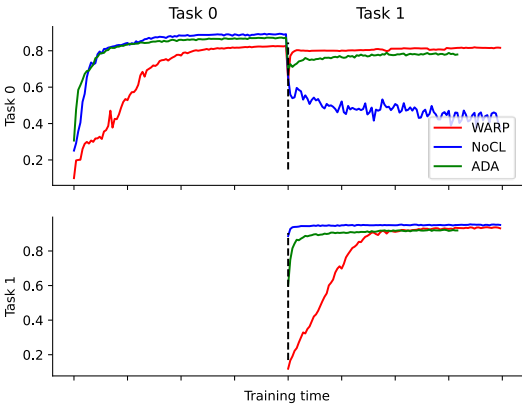
(c) MNIST,USPS



(d) USPS,MNIST



(e) USPS,Fashion-MNIST



(f) Fashion-MNIST,USPS

Figure 7: Test accuracies of Task 0 and Task 1 after training every 100 batches on the `MARGDIFF` task.

Method	shuffled MNIST (3 tasks)	disjoint MNIST (2 tasks)	disjoint CIFAR10 (2 tasks)	shuffled MNIST (5 tasks)	disjoint MNSIT (5 tasks)
WARP	97.88	**96.7**	69.47	96.23	**82.1**
ADA	**98.14**	96.53	**69.51**	**96.77**	81.7
NO-CL	91.46	48.64	41.99	-	-
EWC	96.7	48.96	37.75	-	-
IMM	97.92	96.53	69.51	96.09	67.25
GR	97.57	89.96	65.11	94.54	75.47

TABLE XI: Performance of WARP against baselines on *shuffle* and *disjoint* continual learning tasks. The baseline numbers were taken from the results reported in [1]. Some results were not reported for NO-CL and EWC and they are marked with a '-'.

Dataset	Method	Avg. Accuracy
MNIST - USPS	WARP	**97.1**
	ADA	94.3
	NO-CL	80.1
	WARP-QUAD	95.9
USPS - MNIST	WARP	**95.2**
	ADA	89.3
	NO-CL	68.9
	WARP-QUAD	93.1
MNIST - Fashion	WARP	**93.8**
	ADA	91.9
	NO-CL	54.8
	WARP-QUAD	93.2
Fashion - MNIST	WARP	**92.4**
	ADA	89.3
	NO-CL	60.0
	WARP-QUAD	91.8
Fashion - USPS	WARP	**87.3**
	ADA	84.9
	NO-CL	87.11
USPS - Fashion	WARP	**89.6**
	ADA	85.3
	NO-CL	65.0
	WARP-QUAD	86.7

TABLE XII: Results on the MARGDIFF task. Here, our kernel warping based solution WARP significantly outperforms the baselines.

CHAPTER 6

CONCLUSION AND FUTURE WORKS

6.1 Conclusion

In this thesis we studied the problem of modeling structured priors for machine learning using convex representation learning. In the framework of convex neural networks, we first showed how to incorporate useful structures in the latent hidden layers of a two-layer neural network and then solve the resulting optimization problem in a convex fashion using convex relaxations. We then developed a novel representation learning framework in kernel banach spaces that enabled us to express more general structures that are useful in popular machine learning applications. We proposed different regularizers that can effectively capture priors in prediction problems such as mixup and multi-label classification. We then applied this framework to address the catastrophic forgetting problem in neural networks in the continual learning setting and demonstrated the usefulness of our representation learning framework.

6.2 Future works

Delaying convexity.

In [22], we also observed that the favorable properties of convex neural models are often offset by difficulties in scaling. State of the art non-convex deep learning models, on the other hand, are highly touted for their scalability. An interesting middle ground is to find models that are convex near the optimal solution, a line of work explored by [16, 125, 126] and others.

However, incorporating non-trivial constraints on such models for practical applications is still hard. For instance, consider the problem of solving a least squares objective for low-rank matrix recovery. $\min_U \|y_i - a_i U U^T a_i\|$, for fixed measurements a_i and $U \in \mathbb{R}^{n \times r}$ is low rank [126], a non-convex optimization problem. Convex reformulations of this problems have been studied, but known have high computational complexity. However, under gaussian assumptions on the random vectors a_i, [126] showed that one can find an initialization on a region of local strong convexity for training using SGD, which will guarantee linear convergence. In Figure 8, the plot on the left is a convex function and on the right is a function that is convex in a restricted neighborhood. Extensions of this work to (two-layer) neural network models under the recoverability setting was studied by [125]. However similar to [126], their technique requires additional distributional assumptions for the analysis to go through. Based on our analysis so far, we believe our convex reformulations can be incorporated into these frameworks [16, 125], with little assumptions, providing improved analysis.

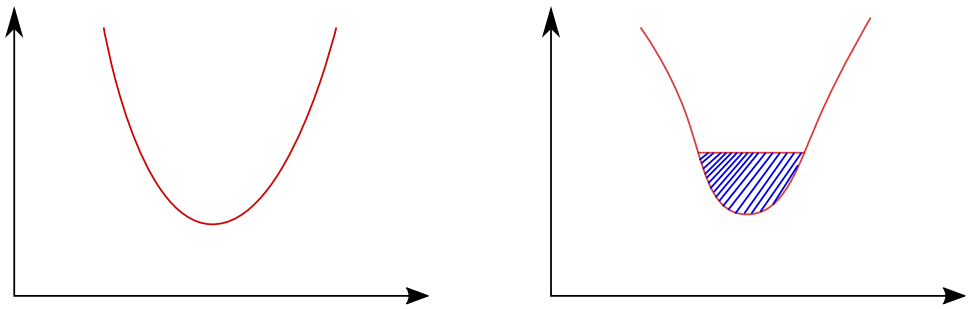


Figure 8: **Left.** A convex function. **Right.** A non-convex function with a region of local strong convexity (region shaded with stripes, best viewed in color)

.

Principled fine-grained domain transfer.

Unsupervised domain adaptation is an important machine learning problem, related to the continual learning setup discussed in Chapter 5. An ubiquitous question in domain adaptation literature is *what information is transferable across domains?* Forcefully aligning feature distributions among disparate domains might lead to a phenomenon called *negative transfer*, which can seriously deteriorate the quality of learning . Harnessing and transferring fine-grained structures across domains can minimize negative transfer [127]. As an example, one immediate way to achieve fine-grained control in transfer could be by masking parts of input from both S and T that are detrimental to transfer, using simple sufficient statistics. However it is not immediately clear what level of granularity in transfer is necessary to achieve optimal transfer. One direction I wish to pursue involves studying different structured representations that are amenable for fine-grained principled domain transfer.

Guaranteed representations for deep domain transfer.

With a flurry of recent algorithms on obtaining transferable representations of data using deep domain adaptation techniques, there is now interest in analyzing these algorithms theoretically. As an immediate future work, I plan to develop new techniques for analyzing the training of deep domain adaptation models. With our recent work on modeling invariance [24] using *kernel warping*, we believe domain invariance can be efficiently modeled and analysed using the recently released theoretical framework called the Neural Tangent Kernels (NTK) [128].

CITED LITERATURE

1. Hu, W., Lin, Z., Liu, B., Tao, C., Tao, Z., Ma, J., Zhao, D., and Yan, R.: Overcoming catastrophic forgetting for continual learning via model adaptation. In International Conference on Learning Representations, 2018.

2. He, K., Zhang, X., Ren, S., and Sun, J.: Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016.

3. Sutskever, I., Vinyals, O., and Le, Q. V.: Sequence to sequence learning with neural networks. 2014.

4. Graves, A., Mohamed, A.-r., and Hinton, G.: Speech recognition with deep recurrent neural networks. In 2013 IEEE international conference on acoustics, speech and signal processing, pages 6645–6649. IEEE, 2013.

5. Ma, Y., Ganapathiraman, V., and Zhang, X.: Learning invariant representations with kernel warping. In The 22nd International Conference on Artificial Intelligence and Statistics, pages 1003–1012, 2019.

6. Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y.: The loss surfaces of multilayer networks. In International Conference on Artificial Intelligence and Statistics (AISTATS), 2014.

7. Dauphin, Y., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y.: Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. 2014.

8. Kawaguchi, K.: Deep learning without poor local minima. 2016.

9. Tian, Y.: An analytical formula of population gradient for two-layered ReLU network and its applications in convergence and critical point analysis. In International Conference on Machine Learning (ICML), 2017.

10. Soltanolkotabi, M., Javanmard, A., and Lee, J. D.: Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. In International Conference on Machine Learning (ICML), 2017.

11. Brutzkus, A. and Globerson, A.: Globally optimal gradient descent for a ConvNet with gaussian inputs. In International Conference on Machine Learning (ICML), 2017.

12. Nguyen, Q. and Hein, M.: The loss surface of deep and wide neural networks. In International Conference on Machine Learning (ICML), 2017.

13. Nguyen, Q. and Hein, M.: The loss surface and expressivity of deep convolutional neural networks. arXiv:1710.10928, 2017.

14. Bengio, Y., Roux, N. L., Vincent, P., Delalleau, O., and Marcotte, P.: Convex neural networks. 2005.

15. Anandkumar, A., Ge, R., Hsu, D., Kakade, S. M., and Telgarsky, M.: Tensor decompositions for learning latent variable models. Journal of Machine Learning Research, 15:2773–2832, 2014.

16. Zhong, K., Song, Z., Jain, P., Bartlett, P., and Dhillon, I.: Recovery guarantees for one-hidden-layer neural networks. In International Conference on Machine Learning (ICML), 2017.

17. Zhang, Y., Lee, J. D., and Jordan, M. I.: ℓ_1-regularized neural networks are improperly learnable in polynomial time. In International Conference on Machine Learning (ICML), 2016.

18. Zhang, Y., Liang, P., and Wainwright, M.: Convexified convolutional neural networks. In International Conference on Machine Learning (ICML), 2017.

19. Livni, R., Shalev-Shwartz, S., and Shamir, O.: An algorithm for training polynomial networks. arXiv:1304.7045v2, 2014.

20. Gens, R. and Domingos, P.: Discriminative learning of sum-product networks. 2012.

21. Fogel, F., Jenatton, R., Bach, F., and d'Aspremont, A.: Convex relaxations for permutation problems. SIAM Journal on Matrix Analysis and Applications, 36(4):1465–1488, 2015.

22. Ganapathiraman, V., Shi, Z., Zhang, X., and Yu, Y.: Inductive two-layer modeling with parametric bregman transfer. In International Conference on Machine Learning (ICML), 2018.

23. Ganapathiraman, V., Zhang, X., Yu, Y., and Wen, J.: Convex two-layer modeling with latent structure. 2016.

24. Ma, Y., Ganapathiraman, V., Yu, Y., and Zhang, X.: Convex Representation Learning for Generalized Invariance in Semi-Inner-Product Space. arXiv:2004.12209 [cs, stat], July 2020. arXiv: 2004.12209.

25. Auer, P., Herbster, M., and Warmuth, M. K.: Exponentially many local minima for single neurons. Technical Report UCSC-CRL-96-1, Univ. of Calif.Computer Research Lab, Santa Cruz, CA, 1996. In preparation.

26. Berman, A. and Shaked-Monderer, N.: Completely Positive Matrices. World Scientific, 2003.

27. Dickinson, P. J. C. and Gijben, L.: On the computational complexity of membership problems for the completely positive cone and its dual. Computational Optimization and Applications, 57(2):403–415, Mar 2014.

28. Jaggi, M.: Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In International Conference on Machine Learning (ICML), 2013.

29. Harchaoui, Z., Juditsky, A., and Nemirovski, A.: Conditional gradient algorithms for norm-regularized smooth convex optimization. Mathematical Programming, 152:75–112, 2015.

30. Freund, R. M. and Grigas, P.: New analysis and results for the conditional gradient method. preprint, 2013.

31. Lacoste-Julien, S. and Jaggi, M.: On the global linear convergence of Frank-Wolfe optimization variants. 2015.

32. Zhang, X., Yu, Y., and Schuurmans, D.: Accelerated training for matrix-norm regularization: A boosting approach. 2012.

33. Murty, K. G. and Kabadi, S. N.: Some NP-complete problems in quadratic and nonlinear programming. Mathematical Programming, 39(2):117–129, 1987.

34. Nemirovski, A., Roos, C., and Terlaky, T.: On maximization of quadratic form over intersection of ellipsoids with common center. Math. Program. Ser. A, 86:463–473, 1999.

35. Cheng, H., Yu, Y., Zhang, X., Xing, E., and Schuurmans, D.: Scalable and sound low-rank tensor learning. In International Conference on Artificial Intelligence and Statistics (AISTATS), 2016.

36. Woodbury, M. A.: Inverting modified matrices. Technical Report Memorandum Rept. 42, Statistical Research Group, Princeton University, Princeton, NJ, 1950.

37. Aslan, O., Zhang, X., and Schuurmans, D.: Convex deep learning via normalized kernels. 2014.

38. Lichman, M.: UCI machine learning repository, 2013.

39. Aslan, O., Cheng, H., Zhang, X., and Schuurmans, D.: Convex two-layer modeling. 2013.

40. Krizhevsky, A. and Hinton, G.: Learning multiple layers of features from tiny images. 2009.

41. Chapelle, O.: http://olivier.chapelle.cc/ssl-book/benchmarks.html.

42. Chang, M.-W., Goldwasser, D., Roth, D., and Srikumar, V.: Discriminative learning over constrained latent representations. In Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), 2010.

43. Chang, M.-W., Srikumar, V., Goldwasser, D., and Roth, D.: Structured output learning with indirect supervision. In International Conference on Machine Learning (ICML), 2010.

44. Ammar, W., Dyer, C., and Smith, N. A.: Conditional random field autoencoders for unsupervised structured prediction. 2014.

45. Hinton, G. E.: Training products of experts by minimizing contrastive divergence. Neural Computation, 14(8):1771–1800, 2002.

46. Gane, A., Hazan, T., and Jaakkola, T.: Learning with maximum a-posteriori perturbation models. In Artificial Intelligence and Statistics, pages 247–256, 2014.

47. Hazan, T. and Jaakkola, T.: On the partition function and random maximum a-posteriori perturbations. arXiv preprint arXiv:1206.6410, 2012.

48. Banerjee, A., Merugu, S., Dhillon, I. S., and Ghosh, J.: Clustering with Bregman divergences. Journal of Machine Learning Research, 6:1705–1749, 2005.

49. Hazan, T. and Jaakkola, T.: On the partition function and random maximum a-posteriori perturbations. In International Conference on Machine Learning (ICML), 2012.

50. Gotovos, A., Hassani, H., and Krause, A.: Sampling from probabilistic submodular models. 2015.

51. Haffari, G. and Sarkar, A.: Analysis of semi-supervised learning with Yarowsky algorithm. In Conference on Uncertainty in Artificial Intelligence (UAI), 2007.

52. Daumé III, H.: Unsupervised search-based structured prediction. In International Conference on Machine Learning (ICML), pages 209–216, 2009.

53. Xu, L., White, M., and Schuurmans, D.: Optimal reverse prediction: a unified perspective on supervised, unsupervised and semi-supervised learning. In International Conference on Machine Learning (ICML), 2009.

54. Nesterov, Y.: Smooth minimization of non-smooth functions. Mathematical Programming, 103(1):127–152, 2005.

55. Meshi, O., Mahdavi, M., and Schwing, A. G.: Smooth and strong: Map inference with linear convergence. 2015.

56. Druck, G., Pal, C., Zhu, X., and McCallum, A.: Semi-supervised classification with hybrid generative/discriminative methods. In the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2007.

57. Chandrasekaran, V., Recht, B., Parrilo, P. A., and S.Willsky, A.: The convex geometry of linear inverse problems. Foundations of Computational Mathematics, 12(6):805–849, 2012.

58. Pataki, G.: On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. Mathematics of Operations Research, 23(2):339–358, 1998.

59. Goldwasser, D. and Roth, D.: Transliteration as constrained optimization. In Conference on Empirical Methods in Natural Language Processing (EMNLP), 2008.

60. https://cogcomp.cs.illinois.edu/page/resource_view/2.

61. Simard, P., LeCun, Y., Denker, J. S., and Victorri, B.: Transformation invariance in pattern recognition-tangent distance and tangent propagation. In Neural Networks: Tricks of the Trade, pages 239–274, 1996.

62. Ferraro, M. and Caelli, T. M.: Lie transformation groups, integral transforms, and invariant pattern recognition. Spatial Vision, 8:33–44, 1994.

63. Cohen, T. S. and Welling, M.: Group Equivariant Convolutional Networks. In International Conference on Machine Learning (ICML), 2016.

64. Graham, D. and Ravanbakhsh, S.: Equivariant entity-relationship networks. arXiv:1903.09033, 2019.

65. Bengio, Y., Courville, A., and Vincent, P.: Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8):1798–1828, 2013.

66. Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A.: β-vae: Learning basic visual concepts with a constrained variational framework. In International Conference on Learning Representations (ICLR), 2017.

67. Eliassi-Rad, T. and Faloutsos, C.: Discovering roles and anomalies in graphs:theory and applications. Tutorial at SIAM International Conference on Data Mining (ICDM), 2012.

68. Creager, E., Madras, D., Jacobsen, J.-H., Weis, M., Swersky, K., Pitassi, T., and Zemel, R.: Flexibly fair representation learning by disentanglement. In International Conference on Machine Learning (ICML), 2019.

69. Wang, W., Arora, R., Livescu, K., and Bilmes, J.: On deep multi-view representation learning. In International Conference on Machine Learning (ICML), 2015.

70. Kumar, A., Sattigeri, P., Wadhawan, K., Karlinsky, L., Feris, R., Freeman, W. T., and Wornell, G.: Co-regularized alignment for unsupervised domain adaptation. 2018.

71. Ravanbakhsh, F. M. S., Ding, N., and Schuurmans, D.: Embedding inference for structured multilabel prediction. 2015.

72. Deng, J., Ding, N., Jia, Y., Frome, A., Murphy, K., Bengio, S., Li, Y., Neven, H., and Adam, H.: Large-scale object classification using label relation graphs. In European Conference on Computer Vision (ECCV), 2012.

73. Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In International Conference on Learning Representations (ICLR), 2018.

74. Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R., and Smola, A.: Deep sets. 2017.

75. Ravanbakhsh, S., Schneider, J., and Poczos, B.: Equivariance Through Parameter-Sharing. In International Conference on Machine Learning (ICML), 2017.

76. Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y.: Contractive auto-encoders: Explicit invariance during feature extraction. In International Conference on Machine Learning (ICML), 2011.

77. Smola, A.: Sets and symmetries. NeurIPS Workshop on Sets & Partitions, 2019.

78. Teo, C. H., Globerson, A., Roweis, S., and Smola, A.: Convex learning with invariances. 2007.

79. Haasdonk, B. and Keysers, D.: Tangent distance kernels for support vector machines. In Pattern Recognition, 2002. Proceedings. 16th International Conference on, volume 2, pages 864–868. IEEE, 2002.

80. Raj, A., Kumar, A., Mroueh, Y., Thomas Fletcher, P., and Schoelkopf, B.: Local group invariant representations via orbit embeddings. In International Conference on Artificial Intelligence and Statistics (AISTATS), 2017.

81. Mroueh, Y., Voinea, S., and Poggio, T.: Learning with group invariant features: A kernel perspective. 2015.

82. Haasdonk, B. and Burkhardt, H.: Invariant kernel functions for pattern analysis and machine learning. Machine Learning, 68(1):35–61, 2007.

83. Bhattacharyya, C., Pannagadatta, K. S., and Smola, A. J.: A second order cone programming formulation for classifying missing data. pages 153–160, 2005.

84. Rahimian, H. and Mehrotra, S.: Distributionally robust optimization: A review. arXiv:1908.05659, 2019.

85. Ma, Y., Ganapathiraman, V., and Zhang, X.: Learning invariant representations with kernel warping. In International Conference on Artificial Intelligence and Statistics (AISTATS), 2019.

86. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A.: Towards deep learning models resistant to adversarial attacks. In International Conference on Learning Representations (ICLR), 2018.

87. Lumer, G.: Semi-inner-product spaces. Transactions of the American Mathematical Society, 100:29–43, 1961.

88. Zhang, H., Xu, Y., and Zhang, J.: Reproducing kernel Banach spaces for machine learning. Journal of Machine Learning Research (JMLR), 10:2741–2775, 2009.

89. Salzo, S., Rosasco, L., and Suykens, J.: Solving ℓ^p-norm regularization with tensor kernels. In International Conference on Artificial Intelligence and Statistics (AISTATS), eds. A. Storkey and F. Perez-Cruz, volume 84, 2018.

90. Der, R. and Lee, D.: Large-margin classification in Banach spaces. In International Conference on Artificial Intelligence and Statistics (AISTATS), 2007.

91. Bennett, K. P. and Bredensteiner, E. J.: Duality and geometry in SVM classifiers. In International Conference on Machine Learning (ICML), ed. P. Langley, pages 57–64, San Francisco, California, 2000. Morgan Kaufmann Publishers.

92. Hein, M., Bousquet, O., and Schölkopf, B.: Maximal margin classification for metric spaces. J. Comput. System Sci., 71:333–359, 2005.

93. von Luxburg, U. and Bousquet, O.: Distance-based classification with lipschitz functions. Journal of Machine Learning Research, 5:669–695, 2004.

94. Zhou, D., Xiao, B., Zhou, H., and Dai, R.: Global geometry of svm classifiers. Technical Report 30-5-02, Institute of Automation, Chinese Academy of Sciences, 2002.

95. Smola, A. J. and Schölkopf, B.: On a kernel-based method for pattern recognition, regression, approximation and operator inversion. Algorithmica, 22:211–231, 1998.

96. Zhang, X., Lee, W. S., and Teh, Y. W.: Learning with invariance via linear functionals on reproducing kernel hilbert space. 2013.

97. Borwein, J. M. and Vanderwerff, J. D.: Convex Functions: Constructions, Characterizations and Counterexamples. Cambridge University Press, 2010.

98. Giles, J. R.: Classes of semi-inner-product spaces. Transactions of the American Mathematical Society, 129(3):436–446, 1967.

99. Faulkner, G. D.: Representation of linear functionals in a Banach space. Rocky Mountain Journal of Mathematics, 7(4):789–792, 1977.

100. Combettes, P. L., Salzo, S., and Villa, S.: Regularized learning scheme in feature Banach spaces. Analysis and Applications, 16(1):1–54, 2018.

101. Rahimi, A. and Recht, B.: Random features for large-scale kernel machines. Cambridge, MA, MIT Press, 2008.

102. Shi, Q., Petterson, J., Dror, G., Langford, J., Smola, A., and Vishwanathan, S.: Hash kernels for structured data. Journal of Machine Learning Research (JMLR), 10:2615–2637, 2009.

103. Williams, C. K. I. and Seeger, M.: Using the Nyström method to speed up kernel machines. 2000.

104. link: Multilabel dataset. `https://sites.google.com/site/hrsvmproject/datasets-hier`.

105. Klimt, B. and Yang, Y.: The enron corpus: A new dataset for email classification research. In European Conference on Machine Learning (ECML), pages 217–226. Springer, 2004.

106. Rousu, J., Saunders, C., Szedmak, S., and Shawe-Taylor, J.: Kernel-based learning of hierarchical multilabel classification models. Journal of Machine Learning Research (JMLR), 7(Jul):1601–1626, 2006.

107. Lewis, D. D., Yang, Y., Rose, T. G., and Li, F.: Rcv1: A new benchmark collection for text categorization research. Journal of Machine Learning Research (JMLR), 5(Apr):361–397, 2004.

108. Vateekul, P., Kubat, M., and Sarinnapakorn, K.: Top-down optimized svms for hierarchical multi-label classification: A case study in gene function prediction. Intelligent Data Analysis, 2012.

109. Pal, D. K., Kannan, A. A., Arakalgud, G., and Savvides, M.: Max-margin invariant features from transformed unlabeled data. 2017.

110. Fukumizu, K., Lanckriet, G. R., and Sriperumbudur, B. K.: Learning in Hilbert vs. Banach spaces: A measure embedding viewpoint. 2011.

111. Zügner, D., Akbarnejad, A., and Günnemann, S.: Adversarial attacks on neural networks for graph data. In ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), pages 2847–2856. ACM, 2018.

112. Mirzazadeh, F., White, M., György, A., and Schuurmans, D.: Scalable metric learning for co-embedding. In European Conference on Machine Learning (ECML), 2015.

113. Chen, Z. and Liu, B.: Lifelong machine learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 12(3):1–207, 2018.

114. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences, 114(13):3521–3526, 2017.

115. Nguyen, C. V., Achille, A., Lam, M., Hassner, T., Mahadevan, V., and Soatto, S.: Toward understanding catastrophic forgetting in continual learning. arXiv preprint arXiv:1908.01091, 2019.

116. Yoon, J., Yang, E., Lee, J., and Hwang, S. J.: Lifelong learning with dynamically expandable networks. arXiv preprint arXiv:1708.01547, 2017.

117. Li, Z. and Hoiem, D.: Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence, 40(12):2935–2947, 2017.

118. Lopez-Paz, D. and Ranzato, M.: Gradient episodic memory for continual learning. In Advances in neural information processing systems, pages 6467–6476, 2017.

119. Shin, H., Lee, J. K., Kim, J., and Kim, J.: Continual learning with deep generative replay. In Advances in Neural Information Processing Systems, pages 2990–2999, 2017.

120. Lee, S.-W., Kim, J.-H., Jun, J., Ha, J.-W., and Zhang, B.-T.: Overcoming catastrophic forgetting by incremental moment matching. In Advances in neural information processing systems, pages 4652–4662, 2017.

121. Rosenberg, D. S. and Bartlett, P. L.: The Rademacher complexity of co-regularized kernel classes. In International Conference on Artificial Intelligence and Statistics (AISTATS), 2007.

122. Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Wortman Vaughan, J.: A theory of learning from different domains. Machine Learning Journal, 72(1-2):151–175, 2010.

123. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al.: Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

124. Shin, H., Lee, J. K., Kim, J., and Kim, J.: Continual learning with deep generative replay. In Advances in Neural Information Processing Systems, pages 2990–2999, 2017.

125. Zhong, K., Song, Z., and Dhillon, I. S.: Learning non-overlapping convolutional neural networks with multiple kernels. arXiv preprint arXiv:1711.03440, 2017.

126. White, C. D., Sanghavi, S., and Ward, R.: The local convexity of solving systems of quadratic equations. arXiv preprint arXiv:1506.07868, 2015.

127. Wen, J., Liu, R., Zheng, N., Zheng, Q., Gong, Z., and Yuan, J.: Exploiting local feature patterns for unsupervised domain adaptation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 5401–5408, 2019.

128. Jacot, A., Gabriel, F., and Hongler, C.: Neural tangent kernel: Convergence and generalization in neural networks. In Advances in neural information processing systems, pages 8571–8580, 2018.

129. Mohri, M., Rostamizadeh, A., and Talwalkar, A.: Foundations of Machine Learning. MIT press, 2018.

130. Boyd, S. and Vandenberghe, L.: Convex optimization. Cambridge university press, 2004.

131. Luo, Z.-q., Ma, W.-k., So, A., Ye, Y., and Zhang, S.: Semidefinite Relaxation of Quadratic Optimization Problems. IEEE Signal Processing Magazine, 27(3):20–34, May 2010.

132. Aslan, O.: Convex Latent Modeling. https://era.library.ualberta.ca/items/1066d3e3-e32d-4cb9-a867-c666e5efaa9a, June 2017.

133. Steinberg, D.: Computation of matrix norms with applications to Robust Optimization. Doctoral dissertation, Faculty of Industrial Engineering and Management, Technion, 2005.

134. UCI: University of California Irvine: Machine Learning Repository, 1990.

135. Hörmander, L.: Sur la fonction d'appui des ensembles convexes dans un espace loealement convexe. Arkiv För Matematik, 3(12):181–186, 1954.

VITA

EDUCATION

University of Illinois at Chicago — Aug 2015 - Present
Ph.D. in Computer Science — *Chicago, Illinois, United States*

Chennai Mathematical Institute — Aug 2010 - May 2012
M.Sc. Computer Science — *Chennai, India*

Anna University — Aug 2003 - May 2007
B. Engg., Computer Science & Engineering — *Chennai, India*

WORK EXPERIENCE

Amazon Rekognition — May 2017 - August 2017
Applied Scientist Intern — *Seattle, United States*

HERE — May 2016 - August 2016
NLP Research Intern — *Chicago, United States*

Perspica Networks Pvt Limited — Jun 2014 - June 2015
Senior Data Scientist — *Chennai, India*

GE Global Research Center — Aug 2012 - Jun 2014
Scientist - Data Mining — *Bengaluru, India*

Mu Sigma Business Solutions — May 2011 - Jul 2011
Graduate Summer Intern — *Bengaluru, India*

PUBLICATIONS

1. Vignesh Ganapathiraman, Xinhua Zhang, Yaoliang Yu, and Junfeng Wen. Convex two-layer modeling with latent structure. In D. D. Lee, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances In Neural Information Processing Systems 29*, pages 1280–1288. Curran Associates, Inc., 2016

2. Yingyi Ma, Vignesh Ganapathiraman, and Xinhua Zhang. Learning invariant representations with kernel warping. volume 89 of *Proceedings of Machine Learning Research*, pages 1003–1012. PMLR, 16–18 Apr 2019

3. Vignesh Ganapathiraman, Zhan Shi, Xinhua Zhang, and Yaoliang Yu. Inductive two-layer modeling with parametric Bregman transfer. volume 80 of *Proceedings of Machine Learning Research*, pages 1636–1645, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR

4. Yingyi Ma, Vignesh Ganapathiraman, Yaoliang Yu, and Xinhua Zhang. Convex representation learning for generalized invariance in semi-inner-product space, 2020

APPENDIX

BACKGROUND MATERIAL

A.1 Convex optimization

In this section, we introduce some of the core concepts of *Convex Optimization* and results that are used in the subsequent material. We borrow heavily from popular textbooks on machine learning and convex optimization such as [129] and [130] in presenting some of the definitions in this section

A.1.1 Basic definitions

Definition 7. *Convex Set. A set $X \in \mathbb{R}^n$ is said to be convex if for any two points $x_1, x_2 \in X$, all points in the line segment l joining x_1 and x_2 are also in X.*

$$\{\alpha x_1 + (1-\alpha)x_2 : 0 \leq \alpha \leq 1\} \subseteq X$$

A.1.1.0.1 Operations that preserve convexity.

In developing algorithms for convex optimization, it is important to test if a given set X is convex. To this end, we present some of the commonly used theorems(without proofs) for proving convexity of sets.

Theorem 12. *Given two convex sets X_1 and X_2, their sum $S = X_1 + X_2 = \{x_1 + x_2 : x_1 \in X_1, x_2 \in X_2\}$ is convex.*

APPENDIX (Continued)

Theorem 13. *Intersection of a set of convex sets $S = \cap\{X_i\}, i \in \mathbb{N}$ is a convex set.*

Theorem 14. *The projection of a convex set onto some of its coordinates is convex. If $X \in \mathbb{R}^m \times \mathbb{R}^n$, then*

$$X_{proj} = \{x_1 \in \mathbb{R}^m : (x_1, x_2) \in X, x_2 \in \mathbb{R}^n\}$$

is convex.

Definition 8. *Convex hull. The convex hull of a set X, denoted by conv X is the set of all convex combinations of the elements of X. It is defined as,*

$$conv\ X = \left\{ \sum_{i=1}^{k} \theta_i x_i : x_i \in X, k \geq 1, \forall i \in [k], \sum_{i}^{k} \theta_i = 1 \right\}.$$

Intuitively, conv X is the smallest convex set that contains X. Please note that while conv X is convex, X is not required to be convex.

Definition 9. *Convex function. Let X be a convex set and let $f : X \to \mathbb{R}$. f is said to be convex if for all $x, y \in X$ and $\theta \in [0, 1]$,*

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta) f(y).$$

*Geometrically, this means that the line between $(x, f(x))$ and $(y, f(y))$, upper bounds the graph of $f(x)$. f is said to be **strictly convex**, if the above inequality is strict.*

APPENDIX (Continued)

A.1.1.0.2 Other characterizations of convex functions.

Convex functions can be characterized by the nature of their first order and second order derivatives. They are popularly referred to as the **First order conditions** and **Second order conditions**.

Theorem 15. *First order conditions. Let f be a differentiable function and* $\operatorname{dom} f$ *is convex. f is convex if and only if for all $x, y \in \operatorname{dom} f$,*

$$f(y) \geq f(x) + \nabla f(x)^t (y - x).$$

The term on the right hand side of the inequality, is the *first-order Taylor expansion* of f. Intuitively, the above definition states that, the first-order Taylor expansion always lower bounds the function globally.

Theorem 16. *Second order conditions. If f is a twice-differentiable function(that is the Hessian of f exists everywhere) and* $\operatorname{dom} f$ *is convex, then f is convex if and only if the Hessian is positive semi-definite. That is for all $x \in \operatorname{dom} f$*

$$\nabla^2 f(x) \succcurlyeq 0.$$

Examples of convex functions

1. **Linear functions.** Linear functions are both convex and concave, by definition.
2. **Quadratic functions.** The function $f : x \to x^2$ over a convex set X is convex.

APPENDIX (Continued)

3. **Norms.** Norms $\|.\|$ defined over a convex set X are convex. For $\alpha \in [0,1], x_1, x_2 \in X$, we can write

$$\|\alpha x_1 + (1-\alpha)x_2\| \leq \|\alpha x_1\| + \|(1-\alpha)x_2\| = \alpha\|x_1\| + (1-\alpha)\|x_2\|.$$

where the first inequality is due to triangle law and the equality is due to homogeneity of norms.

A.1.2 Optimization problems

In this section, we will formally introduce the idea of a *Convex optimization problem*, by first introducing an optimization problem and its constrained variants. The most general form of optimization problem is the *unconstrained optimization problem*, where we would like to minimize(or maximize) a function f over $\operatorname{dom} f$. This is formally written as,

$$\min_{x \in \operatorname{dom} f} f(x)$$

x is called the optimization variable. Note that there are no restrictions on the range of values that x can take. The **optimum value** $x^\star$ is the $x \in \operatorname{dom} f$, where f attains its minimum value.

Typically problems in real life have additional restrictions or *constraints*. We now define the a constrained version of the above problem, which incorporates these additional constraints.

APPENDIX (Continued)

Definition 10. *Constrained optimization problem. Let* $X \in \mathbb{R}^n$ *and let* $f, g_i, h_i : X \to \mathbb{R}\ \forall i \in [n]$. *A* constrained optimization problem *is written as*

$$\min_{x \in X} f(\mathbf{x})) \tag{A.1}$$

$$\textit{subject to } g_i(\mathbf{x}) \leq 0 \tag{A.2}$$

$$h_i(\mathbf{x}) = 0 \tag{A.3}$$

This is the most general form of a constrained optimization problem, where we do not make any assumptions on the convexity of X or the functions f, g, h. This form is also called the *primal form*, related to the *dual form* which will be defined below.

The *optimal* solution to equation (Equation A.1) is denoted as $x^\star$ and is also known as the *primal optimal*. Constrained optimization problem are often solved or analyzed by another form, called the *dual form*. This is based on the so-called **Lagrangian** of the primal problem, which is defined as follows.

Definition 11. *The Lagrangian of a constrained optimization problem is a function defined over* $X \times \mathbb{R}_+$. *It has the following form*

$$\mathcal{L}(\mathbf{x}, \gamma, \nu) = f(\mathbf{x}) + \sum_{i=1}^{n} \gamma_i g_i(\mathbf{x}) + \sum_{i=1}^{n} \nu_i h_i(x)$$

APPENDIX (Continued)

The variables γ_i and ν_i are called the *Langrange dual variables*. Now, we are ready to define the *Lagrange dual function*.

$$\mathcal{F}(\gamma, \nu) = \min_{\mathbf{x} \in X} \mathcal{L}(\mathbf{x}, \gamma, \nu) = \min_{\mathbf{x} \in X} (f(\mathbf{x}) + \sum_{i=1}^{n} \gamma_i g_i(\mathbf{x}) + \sum_{i=1}^{n} \nu_i h_i(x))$$

where, $\gamma_i \geq 0, \nu_i \geq 0$.

We have rewritten the objective (Equation A.1) in terms of the dual variables γ, ν. Note that $\mathcal{F}$ is concave in both the parameters; in fact a linear function in the dual variables. This formulation gives rise to the so-called *dual-formulation* of the original primal form (Equation A.1) of general unconstrained optimization problems

Definition 12. *Dual optimization problem. The following problem is called the dual problem corresponding to the primal constrained optimization problem*

$$\max_{\gamma, \nu} \mathcal{F}(\gamma, \nu)$$

$$\textit{subject to } \gamma \geq 0,$$

$$\nu \geq 0.$$

The optimal solution to the dual problem is often denoted as $d^\star$.

A.2 Convex Conjugate

Convex conjugate or *Fenchel Conjugate* is an important tool in convex optimization and analysis. It builds up towards the concept of duality in convex analysis. It is the counterpart of

APPENDIX (Continued)

Fourier transform in the Harmonic Analysis literature []. In this section we present some of the key definitions and properties of Fenchel Conjugate, which we use extensively in the relaxation steps in convexifying non-convex objectives.

Definition 13. *Fenchel Conjugate. Fenchel conjugate of a function* $f : \mathbb{R}^n \to \mathbb{R}$

$$f^*(\mathbf{y}) - \sup_{\mathbf{x} \in X} y^T \mathbf{x} - f(\mathbf{x}). \tag{A.4}$$

The Fenchel Conjugate, f^* is a max over a set of affine functions (which is convex, by definition), hence is always convex, irrespective of the convexity of f.

Lemma 3. *Let* $f, g : \mathbb{R}^n \to \mathbb{R}$ *be two functions such that*

$$f(\mathbf{y}) \leq g(\mathbf{y}); \forall \mathbf{y} \in \mathbb{R}^n$$

Then,

$$f^*(\mathbf{y}) \geq g^*(\mathbf{y}); \ \forall \mathbf{y} \in \mathbb{R}^n$$

Proof. Let $\phi_f(\mathbf{y}) = \langle \mathbf{y}, \mathbf{x} \rangle - f(\mathbf{x})$. Now, since $f(\mathbf{x}) \leq g(\mathbf{x}); \ \forall \mathbf{x} \in \mathbb{R}^n$, we have,

$$\phi_f(\mathbf{x}) \geq \phi_g(\mathbf{x}) \implies \sup_{\mathbf{x} \in X} \phi_f(\mathbf{x}) \geq \sup_{\mathbf{x} \in X} \phi_g(\mathbf{x}).$$

□

APPENDIX (Continued)

Examples

1. **Affine function.** Let $f(\mathbf{y}) = \langle \mathbf{y}, \mathbf{a} \rangle - b$ for $\mathbf{y}, \mathbf{a} \in \mathbb{R}^n$. Then, $f^*(\mathbf{y})$ can be computed as

$$\begin{aligned} f^*(\mathbf{y}) &= \sup_{\mathbf{x} \in X} \langle \mathbf{y}, \mathbf{x} \rangle - f(\mathbf{x}) \\ &= \sup_{\mathbf{x} \in X} \{ \langle \mathbf{y}, \mathbf{x} \rangle - \{ \langle \mathbf{y}, \mathbf{a} \rangle - b \} \\ &= \sup_{\mathbf{x} \in X} \{ \langle (\mathbf{x} - \mathbf{a}), \mathbf{y} \rangle - b \} \\ &= \sup_{\mathbf{x} \in X} \{ \langle (\mathbf{x} - \mathbf{a}), \mathbf{y} \rangle \} - b \end{aligned}$$

This gives us,

$$f^*(\mathbf{y}) = \begin{cases} 0 & \text{if } \mathbf{x} = \mathbf{a} \\ \infty & \text{otherwise} \end{cases}$$

Lemma 4. *Fenchel-Young inequality. Let $f : \mathbb{R}^n \to \mathbb{R}$, then it follows from the definition of Fenchel conjugate that*

$$\langle \mathbf{y}, \mathbf{x} \rangle \leq f(\mathbf{x}) + f^*(\mathbf{y}); \quad \textit{for } \mathbf{x}, \mathbf{y} \in \mathbb{R}^n \tag{A.5}$$

Proof. From the definition of Fenchel Conjugate, we have

$$\begin{aligned} f^*(\mathbf{y}) &= \sup_{\mathbf{x} \in \mathbb{R}^n} \langle \mathbf{y}, \mathbf{x} \rangle - f(\mathbf{x}) \\ &\geq \langle \mathbf{y}, \mathbf{x} \rangle - f(\mathbf{x}) \\ \implies \langle \mathbf{y}, \mathbf{x} \rangle &\leq f(\mathbf{x}) + f^*(\mathbf{y}) \end{aligned}$$

□

Now, we will look at another lemma that establishes the relationship between the *Fenchel Double Conjugate* of a function f and f itself.

Lemma 5. *Let* $f : \mathbb{R}^n \to \mathbb{R}$*, then*

$$f^{**}(\mathbf{x}) \leq f(\mathbf{x}).$$

Proof. To prove this, let us first try to define $f^{**}(\mathbf{x})$ in terms of $f^*(x)$. Just by the definition of Fenchel Conjugate, we know that ,

$$f^{**}(\mathbf{x}) = \sup_{\mathbf{y}\in\mathbb{R}^n} \langle \mathbf{y}, \mathbf{x}\rangle - f^*(\mathbf{y});\ \mathbf{y} \in \mathbb{R}^n.$$

Now, we will prove this result using the *Fenchel-Young inequality* defined in equation (Equation A.5). For any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we have

$$\langle \mathbf{x}, \mathbf{y}\rangle \leq f(\mathbf{x}) + f^*(\mathbf{y})$$

$$\langle \mathbf{x}, \mathbf{y}\rangle - f^*(\mathbf{y}) \leq f(\mathbf{x})$$

$$\underbrace{\sup\{\langle \mathbf{x}, \mathbf{y}\rangle - f^*(\mathbf{y})\}}_{f^{**}(\mathbf{x})} \leq f(\mathbf{x}).$$

□

APPENDIX (Continued)

A.3 Semi-definite programming and relaxation

In this section, we will look at an important topic in optimization theory, especially convex optimization theory, *Semi-definite* programming (SDP) and *Semi-definite relaxation* (SDR). The reason SDPs and SDRs are so important is that, researchers have found ways to efficiently approximate really hard optimization problems via SDR. For instance, the well known *Quadratic Constrained Quadratic Program (QCQP)* is a hard optimization problem which has been studied extensively and commonly solved using an SDP by the SDR relaxation procedure. SDPs are often easy to formulate, implement and solved in polynomial time. A general QCQP is formulated as follows.

$$\min_{\mathbf{x}\in\mathbb{R}^n} \mathbf{x}^T C\mathbf{x} \tag{A.6}$$

$$\text{subject to } \mathbf{x}^T F_i\mathbf{x} \geq g_i \ \forall i \in [m] \tag{A.7}$$

$$\mathbf{x}^T H_i\mathbf{x} = l_i \ \forall i \in [q], \tag{A.8}$$

where C, F, H are general symmetric matrices and possibly *indefinite* and $i \in [m]$ means $i = 1 \ldots m$. Indefinite matrices are matrices that are neither positive semi-definite nor negative semi-definite. Equation (Equation A.6) is the general QCQP form, which is highly non-convex, since the constraint set (equations (Equation A.7), (Equation A.8)) is non-convex.

APPENDIX (Continued)

Another popular problem in the signal processing community, whose special case is the famous *max-cut* problem in computer science, is the BQP problem, which is defined as

$$\min_{\mathbf{x}\in\mathbb{R}^n} \mathbf{x}^T C\mathbf{x}$$

$$\text{subject to } \mathbf{x}_i^2 = 1.$$

This can also proved to be NP-hard and can be shown to be approximated very well using semi-definite relaxations. We will next introduce the semi-definite programming formulation and then introduce semi-definite relaxations via a simple example as in [131].

Definition 14. *Semi-definite program. A* semi-definite program *is an optimization problem that minimizes a linear function of the optimization variable* $\mathbf{x}$*, subject to certain linear inequality constraints.*

$$\min_{x\in\mathbb{R}^n} c^T\mathbf{x}$$

$$\textit{subject to} \quad F(\mathbf{x}) \geq 0,$$

where $F(x) = F_0 + \sum_{i=0}^{n} \mathbf{x}_i F_i$.

APPENDIX (Continued)

The other popularly used form of SDPs, also known as the *pure form* is given below.

$$\min_{X \in \mathbb{S}^n} \ \mathrm{tr}(CX)$$

$$\text{subject to } \ \mathrm{tr}(A_i X) = \mathbf{b}_i; \ i \in [n]$$

$$X \succeq 0.$$

where $\mathbb{S}^n$ is the set of all real symmetric matrices of size $n \times n$ and $A_i \in \mathbb{S}^n$ and $\mathbf{b}_i \in \mathbb{R}$.

Semi-definite relaxations (SDRs).

We will first describe the key steps in relaxing a hard optimization problem into an SDP. Let us consider the following (simplified) version of the QCQP problem introduced earlier.

$$\min_{\mathbf{x} \in \mathbb{R}^n} \ \mathbf{x}^T C \mathbf{x} \tag{A.9}$$

$$\text{subject to } \ \mathbf{x}^T A_i \mathbf{x} \geq \mathbf{b}_i, \tag{A.10}$$

where $A_i \in \mathbb{S}^n$, the set of all real-symmetric matrices and $\mathbf{b}_i \in \mathbb{R}$.

1. First step towards solving the above hard problem using SDP is to observe that

$$\mathbf{x}^T C \mathbf{x} = \mathrm{tr}(\mathbf{x}^T C \mathbf{x}) = \mathrm{tr}(C \mathbf{x} \mathbf{x}^T).$$

Similarly we can write,

$$\mathbf{x}^T A_i \mathbf{x} = \mathrm{tr}(A_i \mathbf{x} \mathbf{x}^T).$$

APPENDIX (Continued)

2. Now, we denote by the matrix $X = \mathbf{x}\mathbf{x}^T$. Substituting this, objective (Equation A.9) becomes,

$$\min_{X \in \mathbb{S}^n} \ \text{tr}(CX)$$
$$\text{subject to } \ \text{tr}(A_i X) \geq \mathbf{b}_i$$
$$X = \mathbf{x}\mathbf{x}^T.$$

3. Now, it can be observed that the constraint $X = \mathbf{x}\mathbf{x}^T$ can be rewritten as $X \succeq 0$ and $\text{rank}(X) = 1$. Noting this, the objective becomes,

$$\min_{X \in \mathbb{S}^n} \ \text{tr}(CX) \tag{A.11}$$
$$\text{subject to } \ \text{tr}(A_i X) \geq \mathbf{b}_i \tag{A.12}$$
$$X \succeq 0; \ \text{rank}(X) = 1. \tag{A.13}$$

4. Now, equation (Equation A.11) is convex in X; in fact linear in X and all the constraints, except the rank constraint are also linear in X. However, this rewrite doesn't decrease the complexity of the original problem. Indeed, equation (Equation A.11) is as hard as equation (Equation A.9). However, as mentioned earlier, the only "hard" constraint is the

rank constraint. Thus, it is common to drop the rank constraint to obtain the following *relaxed* objective, which is an *SDP*.

$$\min_{X \in \mathbb{R}^n} \ \mathrm{tr}(CX)$$
$$\text{subject to } \ \mathrm{tr}(A_i X) \geq \mathbf{b}_i$$
$$X \succeq \mathbf{0}.$$

The above procedure, is a general recipe for performing semi-definite relaxations of hard-to-solve combinatorial optimization problems. A similar relaxation procedure can be derived for the max-cut problem. For details the reader is referred to [132].

A.4 Projected Gradient Descent (PGD) and Conditional Gradient algorithm (Frank Wolfe

In the previous few subsections, we looked at constrained optimization problems, in general, and SDP - a special constrained optimization problem. In this section, we will look at two commonly used algorithms to solve constrained optimization problems.

Projected Gradient Descent (PGD) algorithm.

Let us consider the following general version of a constrained optimization problem

$$\min_{\mathbf{x} \in \mathbb{R}^n} \ f(\mathbf{x}) \tag{A.14}$$
$$\text{subject to } \mathbf{x} \in C \tag{A.15}$$

where C represents some constraint set, which are typically expressed by some function of $\mathbf{x}$.

APPENDIX (Continued)

One of the common ways to solve this optimization problem is the *projected gradient descent*, where in each iteration $t+1$ the algorithm performs two operations.

First, it proposes a new point x^{t+1} in the direction opposite to the negative of the gradient of f.

$$\mathbf{x}^{t+1} \leftarrow \mathbf{x}^t - \eta \nabla f(\mathbf{x}^t)$$

where η is the step size - a hyper parameter to be tuned.

The above update rule is the vanilla gradient descent step. The gradient of a function gives the direction of steepest ascent. Therefore, to minimize the function, we pursue the direction opposite to the gradient. This is done conservatively by multiplying the negative gradient with a step size parameter $\eta \in [0, 1]$.

Note that the above *proposed* point, need not be in the feasible set C - since the update rule doesn't enforce this. So, in order to make sure that the proposed point is in the feasible region, we perform a "projection" step, which typically another optimization problem(depending on how the constraint set is), written as

$$P_c(\mathbf{x}) = \min_{\mathbf{z} \in C} \|\mathbf{x} - \mathbf{z}\|$$

The projected point $P_c(\mathbf{x})$ is the optimal "feasible" point for the next iteration of the algorithm.

Conditional Gradient Algorithm (Frank-Wolfe algorithm).

Let us consider the same optimization problem (Equation A.14). Additionally, let us assume that f has L-Lipschitz gradient, which is a measure of the smoothness of f. The key step in

APPENDIX (Continued)

the Projected Gradient Descent algorithm introduced earlier is the *projection* operation $P_c(\mathbf{x})$, which keeps the domain of the optimization within the feasible region. There are however two drawbacks of this method.

1. If the proposed optimal point $\mathbf{x}$ (before projection), lies outside the feasible region, then the projection operation $P_c(\mathbf{x})$ always projects the point on the boundary of the feasible region. This rules out points on the int C.
2. The projection operation is often computationally expensive and sometimes hard to compute. Typically the constraint set of an optimization problem is the intersection of multiple regions defined by functions of $\mathbf{x}$.

To this end, researchers have developed many "projection-free" approaches. The *Frank-Wolfe* or *Conditional Gradient* algorithm is one of the important projection-free approaches to solve constrained optimization problems. An illustration of Frank-Wolfe is given in figure Figure 9.

The Conditional Gradient algorithm is surprisingly simple and has two major steps in building iterative solutions towards the optimal $\mathbf{x}$ for the constrained problem (Equation A.14).

1. At each step, find a point $\mathbf{s}^t \in \mathbb{R}^n$ that maximizes the negative gradient or (equivalently) minimizes the gradient.

$$\mathbf{s}^t \in \arg\max_{\mathbf{s}\in\mathbb{R}^n} \langle -\nabla f(\mathbf{x}^t), \mathbf{s} \rangle,$$

where $\mathbf{x}^t$ is the current estimate of the optimal solution.

APPENDIX (Continued)

2. Now, obtain the new estimate of the solution $\mathbf{x}^s t + 1$ which is a convex combination of the current iterate $\mathbf{x}^t$ and $\mathbf{s}^t$.

$$\mathbf{x}^{t+1} = \eta_t \mathbf{x}^t + (1 - \eta_t)\mathbf{s}^t.$$

Choosing the step size η. There are multiple variants of Frank-Wolfe, depending on how we choose η_t. One possibility is to view the problem of choosing η_t as another linear minimization problem. It's often referred to as the *line search* step in Frank-Wolfe.

$$\eta_t = \arg\min_{\eta \in [0,1]} f(\eta_t \mathbf{x}^t + (1 - \eta_t)\mathbf{s}^t).$$

Other heuristic choices of *eta* also exists.

$$\eta_t = \frac{2}{t+1}$$

APPENDIX (Continued)

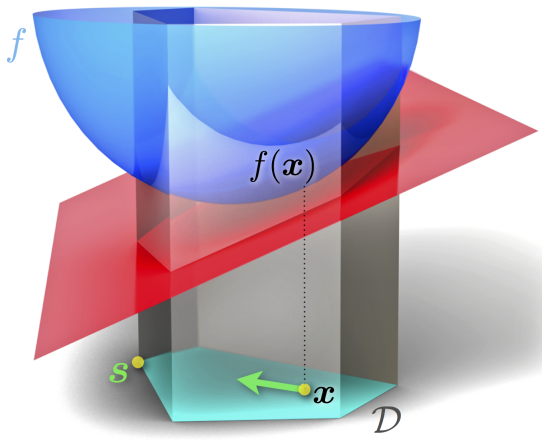


Figure 9: An illustration of Frank-Wolfe.
[1]

APPENDIX

ADDITIONAL PROOFS FOR CHAPTER 2

Proof for Theorem 1

Restatement of Theorem 1: There exists a loss L that satisfies all the three conditions if, and only if, $\mathbf{f}$ is affine.

Proof. The "if" part is trivial as we just need to set $L(\boldsymbol{\phi}, \mathbf{z}) = ||\boldsymbol{\phi} - \mathbf{f}(\mathbf{z})||^2$. To see the "only if" part, consider the sublevel set of L at 0: $S = \{(\boldsymbol{\phi}, \mathbf{z}) : L(\boldsymbol{\phi}, \mathbf{z}) \leq 0\}$. By grounding and unique recovery, $S = \{(\mathbf{f}(\mathbf{z}), \mathbf{z}) : \mathbf{z}\}$. And by the joint convexity of L, S is convex. So for any $\mathbf{z}_1, \mathbf{z}_2$, $(\frac{1}{2}(\mathbf{f}(\mathbf{z}_1) + \mathbf{f}(\mathbf{z}_2)), \frac{1}{2}(\mathbf{z}_1 + \mathbf{z}_2))$ is in S. But $(\mathbf{f}(\frac{1}{2}(\mathbf{z}_1 + \mathbf{z}_2)), \frac{1}{2}(\mathbf{z}_1 + \mathbf{z}_2))$ is the only element in S with the second component being $\frac{1}{2}(\mathbf{z}_1 + \mathbf{z}_2)$. So $\frac{1}{2}(\mathbf{f}(\mathbf{z}_1) + \mathbf{f}(\mathbf{z}_2)) = \mathbf{f}(\frac{1}{2}(\mathbf{z}_1 + \mathbf{z}_2))$. So $\mathbf{f}$ is affine. □

Proof for Theorem 2

Restatement of Theorem 2: For any $W, U, \mathbf{b}$, denote $\mathcal{L}(\Phi, R) = F^*(\Phi) - \mathrm{tr}(\Phi' W X) + \mathrm{tr}(R'(U'\Phi + \mathbf{b}\mathbf{1}')) - \ell^*(R)$. Then

$$\text{(LHS)} \quad \min_{\Phi} \max_{R} \mathcal{L}(\Phi, R) = \max_{R} \min_{\Phi} \mathcal{L}(\Phi, R) \quad \text{(RHS)}.$$

APPENDIX (Continued)

Proof. Trivially LHS $\geq$ RHS (weak duality). Define $h(Z) := \min_{\Phi} F^*(\Phi) - \mathrm{tr}(\Phi' W X) + \ell(U'\Phi + \mathbf{b1}' + Z)$. Since F^* is strongly convex, $h(Z) > -\infty$ for all Z. Since $\ell(U'\Phi + \mathbf{b1}' + Z)$ is jointly convex in Φ and Z, $h(Z)$ is convex in Z (after minimizing out Φ). Suppose $R \in \partial h(\mathbf{0})$, then

$$h(\mathbf{0}) \leq h(Z) - \mathrm{tr}(R'Z) \leq F^*(\Phi) - \mathrm{tr}(\Phi' W X) + \ell(U'\Phi + \mathbf{b1}' + Z) - \mathrm{tr}(R'Z) \tag{B.1}$$

$$= \{F^*(\Phi) - \mathrm{tr}(\Phi' W X) + \mathrm{tr}(\Phi' U R)\} \tag{B.2}$$

$$+ \{\ell(U'\Phi + \mathbf{b1}' + Z) - \mathrm{tr}(R'(Z + U'\Phi + \mathbf{b1}'))\} + \mathrm{tr}(R'\mathbf{b1}'). \tag{B.3}$$

Take infimum over Z for the second group of terms, and then take infimum over Φ for the first group of terms. We finally arrive at

$$h(\mathbf{0}) \leq -F(WX - UR) - \ell^*(R) + \mathrm{tr}(R'\mathbf{b1}') \leq \max_R -F(WX - UR) - \ell^*(R) + \mathrm{tr}(R'\mathbf{b1}').$$

That is LHS $\leq$ RHS. So in summary LHS = RHS. □

Proof for Lemma 1

Restatement of Lemma 1: $\mathcal{S}$ is convex, bounded, and closed. In addition,

$$\gamma_{\mathcal{S}}(T) = \begin{cases} \mathrm{tr}(T) & T \in \mathcal{T} \\ +\infty & \text{otherwise} \end{cases}. \tag{B.4}$$

Proof. Since $\mathcal{T}$ is a convex cone, the right-hand side is a sublinear function. To show two sublinear functions f and g are equal, it suffices to show that their "unit balls" are equal, *i.e.*

APPENDIX (Continued)

$\{\mathbf{x} : f(\mathbf{x}) \leq 1\} = \{\mathbf{x} : g(\mathbf{x}) \leq 1\}$. The unit ball of the left-hand side, by definition, is $\mathcal{S}$. The unit ball of the right-hand side is: $\{T : T \in \mathcal{T}, \mathrm{tr}(T) \leq 1\}$. But this is exactly the definition of $\mathcal{S}$ in (Equation 2.11). □

B.1 Extensions to hard tanh and non-elementwise transfers

B.1.1 Elementwise transfer.

When using the **hard tanh** transfer, we have $F_h^*(\Phi) = \frac{1}{2}\|\Phi\|^2$ if the L_∞ norm $\|\Phi\|_\infty := \max_{ij}|\Phi_{ij}| \leq 1$, and ∞ otherwise. As a result, we get the same objective function as in (Equation 2.10), only with $\mathcal{T}_h$ changed into $\{\Phi'\Phi : \|\Phi\|_\infty \leq 1\}$ and the domain of A changed into $\{A : \sum_i |A_{ij}| \leq 1,\ \forall\ j\}$. Given the negative gradient $G \succeq \mathbf{0}$ of the objective, the polar operator boils down to solving

$$\max_{\Phi\in\mathbb{R}^{h\times t}:\|\Phi\|_\infty\leq 1} \mathrm{tr}(G'\Phi'\Phi) \quad = \quad h \max_{\boldsymbol{\phi}\in[0,1]^t} \boldsymbol{\phi}' G \boldsymbol{\phi} \quad = \quad h \max_{\boldsymbol{\phi}\in[0,1]^t} \|A\boldsymbol{\phi}\|^2, \quad \text{where} \quad A'A = G. \tag{B.5}$$

This problem is NP-hard, but an approximate solution with constant multiplicative guarantee can be found in $O(t^2)$ time [133]. Note for computation we do not even need an expression of the convex hull of $\mathcal{T}_h$.

APPENDIX (Continued)

B.1.2 Non-elementwise transfer.

The Bregman divergence can be further leveraged to convexify transfer functions that are not applied elementwise. For example, consider the soft-max function that is commonly used in machine learning and deep learning:

$$\mathbf{f}(\mathbf{x}) = \left(\sum_{k=1}^{h} e^{x_k}\right)^{-1} (e^{x_1}, \ldots, e^{x_h})'.$$

Clearly the range of $\mathbf{f}$ is $S^h = \{\mathbf{z} \in \mathbb{R}^h : \mathbf{z} > \mathbf{0}, \mathbf{1}'\mathbf{z} = 1\}$. The potential function $F(\mathbf{x})$ is simply

$$F(\mathbf{x}) = \log \sum_{k=1}^{h} e^{x_k}, \tag{B.6}$$

and its Fenchel dual is

$$F^*(\boldsymbol{\phi}) = \begin{cases} \sum_{k=1}^{h} \phi_k \log \phi_k & \text{if } \boldsymbol{\phi} \in S^h \\ \infty & \text{otherwise} \end{cases}. \tag{B.7}$$

Therefore the objective in (Equation 2.7) can be instantiated into

$$\min_{\boldsymbol{\phi}_j \in \mathcal{S}^h} \max_{R\mathbf{1}=\mathbf{0}, \boldsymbol{\lambda}_j \in \mathcal{S}^h} \sum_{j=1}^{t} F^*(\boldsymbol{\phi}_j) - \frac{1}{2}\left\|(\Phi - \Lambda)X'\right\|^2 - \frac{1}{2}\left\|\Phi R'\right\|^2 - F^*(\Lambda) - \ell^*(R). \tag{B.8}$$

where $\Phi = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_t) \in \mathbb{R}^{h \times t}$ and $\Lambda = (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t) \in \mathbb{R}^{h \times t}$. Here $\mathcal{S}^h$ is the closure of S^h: $\{\mathbf{z} \in \mathbb{R}^h_+ : \mathbf{1}'\mathbf{z} = 1\}$, *i.e.* the h dimensional probability simplex.

APPENDIX (Continued)

When $h = 2$, $F^*(\boldsymbol{\phi})$ is the negative entropy function, and it can be approximated by $\frac{a}{2}[(\phi_1 - 0.5)^2 + (\phi_2 - 0.5)^2] + c$, where a and c are chosen such that $c = F^*(\frac{1}{2}\mathbf{1}) = \log\frac{1}{2}$ and $\frac{a}{2}(0.5^2 + 0.5^2) + c = F^*((0,1)') = 0$. For general h, we can similarly approximate $F^*(\boldsymbol{\phi})$ by $\frac{a}{2}\left\|\boldsymbol{\phi} - \frac{1}{h}\mathbf{1}\right\|^2 + c$, with $c = F^*(\frac{1}{h}\mathbf{1}) = \log\frac{1}{h}$ and $\frac{a}{2}[(1-\frac{1}{h})^2 + \frac{h-1}{h^2}] + c = F^*((1,0,\ldots,0)') = 0$. Since $\mathbf{1}'\boldsymbol{\phi} = 1$, this approximation is in turn equal to $a\left\|\boldsymbol{\phi}\right\|^2 + d$ where $d = c - a/(2h)$. As a result, (Equation B.8) can be approximated by (setting $a = 1$ to ignore scaling)

$$\min_{\boldsymbol{\phi}_j \in \mathcal{S}^h} \max_{R\mathbf{1}=\mathbf{0}, \boldsymbol{\lambda}_j \in \mathcal{S}^h} \quad \frac{1}{2}\left\|\Phi\right\|^2 - \frac{1}{2}\left\|(\Phi - \Lambda)X'\right\|^2 - \frac{1}{2}\left\|\Phi R'\right\|^2 - \frac{1}{2}\left\|\Lambda\right\|^2 - \ell^*(R). \tag{B.9}$$

Once more we can apply change of variable by $\Lambda = \Phi A$. Since $\Phi \geq \mathbf{0}$, $\Lambda \geq \mathbf{0}$, $\Phi'\mathbf{1} = \mathbf{1}$, and $\Lambda'\mathbf{1} = \mathbf{1}$, we easily derive the domain of A as $A'\mathbf{1} = \mathbf{1}$ and $A \geq \mathbf{0}$. So using $T = \Phi'\Phi$, we finally arrive at the convexified objective:

$$\min_{T \in \mathcal{T}_h} \max_{R\mathbf{1}=\mathbf{0}, A \geq \mathbf{0}, A'\mathbf{1}=\mathbf{1}} \frac{1}{2}\operatorname{tr}(T) - \frac{1}{2}\operatorname{tr}(T(I-A)X'X(I-A')) - \frac{1}{2}\operatorname{tr}(TR'R) - \frac{1}{2}\operatorname{tr}(TAA') - \ell^*(R), \tag{B.10}$$

where $\mathcal{T}_h$ is the convex hull of $\{\Phi'\Phi : \Phi \in \mathbb{R}_+^{h\times t}, \Phi'\mathbf{1} = \mathbf{1}\}$. So given the negative gradient $G \succeq \mathbf{0}$ of the objective, the polar operator aims to compute

$$\max_{\Phi \in \mathbb{R}_+^{h\times t}: \Phi'\mathbf{1}=\mathbf{1}} \operatorname{tr}(G'\Phi'\Phi) \quad = \quad \max_{\boldsymbol{\phi}_1,\ldots,\boldsymbol{\phi}_h \in \mathbb{R}_+^t} \sum_{k=1}^h \left\|A\boldsymbol{\phi}_k\right\|^2 \quad s.t. \quad \sum_{k=1}^h \boldsymbol{\phi}_k = \mathbf{1}, \quad \text{where} \quad A'A = G. \tag{B.11}$$

This problem is NP-hard [133], but an approximate solution with provable guarantee is still possible. For example, in the case that $h = 2$, we have $\boldsymbol{\phi}_2 = \mathbf{1} - \boldsymbol{\phi}_1$, and the problem becomes

$$\max_{\boldsymbol{\phi}_1 \in [0,1]^t} \|A\boldsymbol{\phi}_1\|^2 + \|A(\mathbf{1} - \boldsymbol{\phi}_1)\|^2 = \max_{\boldsymbol{\phi}_1 \in [0,1]^t} \left\|A\left(\boldsymbol{\phi}_1 - \tfrac{1}{2}\mathbf{1}\right)\right\|^2 + \text{constant} \tag{B.12}$$

$$= \max_{\boldsymbol{\phi} \in [-\frac{1}{2}, \frac{1}{2}]^t} \|A\boldsymbol{\phi}\|^2 + \text{constant}. \tag{B.13}$$

This again admits an approximate solution with constant multiplicative guarantee that can be computed in $O(t^2)$ time [133].

Note the $\mathcal{T}_h$ in this case, as well as that in the hard tanh case above, is closely related to the completely positive matrix cone, because $\Phi \in \mathbb{R}_+^{h \times t}$.

B.2 Dataset description

The experiments made use of 4 "real" world datasets - G241N (241×1500) from [41], Letter (vowel letters A-E vs non vowel letters B-F) (16×20000) from [134], CIFAR-SM (bicycle and motorcycle vs lawn- mower and tank) (256×1526) from [39] and [40] and CIFAR-10 (ship vs truck) (256×12000) from [40], where red channel features are preprocessed by averaging pixels in both the CIFAR datasets.

B.3 Proof of Lemma 2

Let X be an extreme point of $\mathcal{R}$ and $\text{rank}(X) = r$. We prove by contradiction, and suppose (Equation 3.21) does not hold. Then $\frac{1}{2}r(r+1) > m$. Write the eigen-decomposition of X as $X = Q\Lambda Q'$, where $Q'Q = I$ and $\Lambda \in \mathbb{R}^{r \times r}$ is positive *definite* and diagonal. By (Equation 3.20), $\text{tr}(A_i X) = \text{tr}(A_i Q \Lambda Q') = \text{tr}(\Lambda Q' A_i Q) \lessgtr b_i$.

APPENDIX (Continued)

Let S^r be the set of symmetric r-by-r matrices. It is a subspace with dimension $\frac{1}{2}r(r+1)$. Since $m < \frac{1}{2}r(r+1)$ and $Q'A_iQ \in \mathrm{S}^r$, there must exist a nonzero $\Delta \in \mathrm{S}^r$ such that $\mathrm{tr}(\Delta Q'A_iQ) = 0$ for all i. In addition, as Λ is positive *definite*, there exists $\epsilon > 0$ such that $\Lambda \pm \epsilon\Delta$ are both positive semi-definite. Now consider two matrices $X_+ = Q(\Lambda + \epsilon\Delta)Q'$ and $X_- = Q(\Lambda - \epsilon\Delta)Q'$. Clearly X_+, X, X_- are distinct because $Q\Delta Q' = \mathbf{0}$ if, and only if, $\Delta = \mathbf{0}$. Furthermore X_+ and X_- are both in $\mathcal{R}$ because i) $X_+ \succeq \mathbf{0}$ and $X_- \succeq \mathbf{0}$, and ii) $\mathrm{tr}(A_iX_+) = \mathrm{tr}(A_iX_-) = \mathrm{tr}(A_iX)$. Now the fact that $X = (X_+ + X_-)/2$ contradicts with the assumption that X is an extreme point of $\mathcal{R}$.

B.4 Chain model with sets of output bases

Finally to reconstruct the image for each letter, we assume that they are the convex combination of p bases (or principal components) that are specific to each letter. Suppose for letter j, the bases are the columns of $R_j \in \mathbb{R}^{m\times p}$. Denote the combination weights for the i-th letter in a word as $Q_i \in \mathbb{R}_+^{p\times h}$, where the j-th column corresponds to the case where the letter is j. We postulate that Q_i is related to W_i in the sense that its j-th column is nonzero only if W_i represents letter j: $Q_i'\mathbf{1} = W_i\mathbf{1}$. As a result, the expected reconstruction is $\sum_{j=1}^h R_jQ_i(:,j)$

APPENDIX (Continued)

where $Q_i(:,j)$ is the j-th column of Q_i. Enforcing this constraint with a Lagrange multiplier $\boldsymbol{\alpha}_i$, we finally obtain our objective

$$\min_{\|U\|\le\lambda_1}\min_{\|R\|\le\lambda_2}\mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{z})\sim\tilde{p}}\Big[\max_{\boldsymbol{\alpha}_i\ge\mathbf{0},\Pi}\max_{(\Theta,\gamma)\in\mathcal{C}}\max_{\mathbf{v}}\min_{W}\min_{Q_i\in[0,1]^{p\times h}} \tag{B.14}$$

$$\sum_i\left((\mathbf{v}_i-\mathbf{z}_i)'\sum_j R_jQ_i(:,j)-G^*(\mathbf{v}_i)+\frac{\sigma'}{2}\|Q_i\|^2\right)+\operatorname{tr}(\Pi'W)-\sigma_S(\Pi) \tag{B.15}$$

$$+\sum_i\operatorname{tr}((U_v\mathbf{x}_i\mathbf{1}'+U_e+\sigma W_i)'(\gamma W_i-\Theta_i))+\sum_i\boldsymbol{\alpha}_i'(Q_i'\mathbf{1}-W_i\mathbf{1})\Big]. \tag{B.16}$$

Here we added an extra small L_2 penalty on Q_i and its weight σ' is a small positive number. Then we proceed by

$$\min_{\|U\|\le\lambda_1}\min_{\|R\|\le\lambda_2}\mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{z})\sim\tilde{p}}\Big[\max_{\Pi}\max_{(\Theta,\gamma)\in\mathcal{C}}\max_{\mathbf{v}}-\sigma_S(\Pi)-\sum_i\left(G^*(\mathbf{v}_i)+\operatorname{tr}(\Theta_i'(U_v\mathbf{x}_i\mathbf{1}'+U_e)\right) \tag{B.17}$$

$$-\frac{1}{4\sigma\gamma}\sum_i\left\|\gamma U_v\mathbf{x}_i\mathbf{1}'+\gamma U_e+\Pi_i-\sigma\Theta_i-\boldsymbol{\alpha}_i\mathbf{1}'\right\|^2 \tag{B.18}$$

$$+\sum_i\min_{Q_i\in[0,1]^{p\times h}}\left\{\frac{\sigma'}{2}\|Q_i\|^2+(\mathbf{v}_i-\mathbf{z}_i)'\sum_j R_jQ_i(:,j)+\boldsymbol{\alpha}_i'Q_i'\mathbf{1}\right\}\Big]. \tag{B.19}$$

The last term is minimizing a quadratic form of Q_i over $[0,1]$ constraints. This is exactly the same as we discussed in Section B.5. So following the same derivations there we get an SDP relaxation again.

APPENDIX (Continued)

An even more careful look reveals that the terms related to U are only in (Equation B.17) and (Equation B.18), while the terms related to R are only in (Equation B.19). The absence of cross terms allows us to carry out SDP for U and R separately by considering matrices

$$\begin{pmatrix} I \\ U' \end{pmatrix} \begin{pmatrix} I, U \end{pmatrix} = \begin{pmatrix} I & U \\ U' & U'U \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} I \\ R' \end{pmatrix} \begin{pmatrix} I, R \end{pmatrix} = \begin{pmatrix} I & R \\ R' & R'R \end{pmatrix}. \tag{B.20}$$

Note the technique of decoupling the hidden variables can also be applied to the more general framework in (Equation 3.12). The trade-off is delicate between the size of SDP size and the complexity of solving inner maximization given M. We conjecture that the SDP relaxation over U and R separately may lead to tighter approximation and higher sample efficiency. We leave the investigation for future work.

B.5 Simplification via Partial Lagrangian Formulation

In many applications, the dimensionality of $\mathbf{w}$ is much higher than the number of non-box constraints. For example, in the homogeneous linear chain model, there are $O(C^2)$ variables (hence that number of $[0, 1]$ box constraints), while the number of non-box constraints is $O(C)$. So a partial Lagrangian approach turns out more effective by retaining the box constraints in the optimization of $\mathbf{w}$ in (Equation 3.12), while the non-box constraints are enforced by

APPENDIX (Continued)

Lagrange multipliers. This leads to the following objective that replaces the expression inside the expectation operator of (Equation 3.12):

$$\max_{\boldsymbol{\beta}\geq\mathbf{0}} \max_{(\boldsymbol{\theta},\gamma)\in\mathcal{N}} \max_{\mathbf{v}} \min_{\mathbf{w}\in[0,1]^h} -\mathbf{z}'R'\mathbf{w} + \mathbf{v}'R'\mathbf{w} - G^*(\mathbf{v}) + \boldsymbol{\beta}'(A\mathbf{w}-\mathbf{c}) + (U\mathbf{x}+\sigma\mathbf{w})'(\gamma\mathbf{w}-\boldsymbol{\theta})$$

$$= \max_{\boldsymbol{\beta}\geq\mathbf{0}} \max_{(\boldsymbol{\theta},\gamma)\in\mathcal{N}} \max_{\mathbf{v}} -G^*(\mathbf{v}) - \boldsymbol{\beta}'\mathbf{c} - \boldsymbol{\theta}'U\mathbf{x} - 2\gamma\sigma g\left(\frac{R(\mathbf{v}-\mathbf{z}) + \gamma U\mathbf{x} + A'\boldsymbol{\beta} - \sigma\boldsymbol{\theta}}{-2\gamma\sigma}\right), \quad \text{(B.21)}$$

$$\text{where} \quad g(\mathbf{s}) = \frac{1}{2}\|\mathbf{s}\|^2 - \frac{1}{2}\left\|\left[\left|\mathbf{s}-\tfrac{1}{2}\mathbf{1}\right| - \tfrac{1}{2}\mathbf{1}\right]_+\right\|^2. \quad \text{(B.22)}$$

Here $[x]_+ = \max\{0, x\}$, and is taken elementwise along with absolute value. The expression of g is derived from the fact that $\min_{w\in[0,1]} \frac{1}{2}w^2 - sw = \frac{1}{2}(d^2 - 2sd)$, where d is the median of $\{s, 0, 1\}$.

Compared with (Equation 3.12) and (Equation 3.14), the $\boldsymbol{\beta}$ used here has much lower dimension than $\boldsymbol{\pi}$. In addition, $g(\mathbf{s})$ is convex and therefore given (U, R), the optimal $(\boldsymbol{\beta}, \boldsymbol{\theta}, \gamma, \mathbf{v})$ can be solved efficiently. More interestingly, thanks to the expression of $g(\mathbf{s})$ in (Equation B.22), which is a quadratic minus a convex function in $\mathbf{s}$, our SDP relaxation can be easily extended to this partial Lagrangian framework. In fact, just replace $\|\mathbf{s}\|^2$ by using an affine function of M, and then we obtain a convex objective in (U, R).

Although the derivation of (Equation B.21) is based on the generic form in (Equation 3.12), it is straightforward to apply the same technique to specialized formulations in §3.6.1 and 3.6.2.

APPENDIX (Continued)

B.6 Projection to $\mathcal{M}_1$

The projection to $\mathcal{M}_1$ means solving for a given $\hat{M}$:

$$\min_{M} \tfrac{1}{2}||M - \hat{M}||^2, \quad s.t. \quad M \succeq \mathbf{0},\ M_1 = I,\ \mathrm{tr}(M_{u,u}) \leq 1,\ \mathrm{tr}(M_{r,r}) \leq 1. \tag{B.23}$$

To solve it efficiently, we resort to a partial Lagrangian approach with the last three constraints enforced by Lagrange multipliers $\Lambda \in \mathbb{R}^{h\times h}$, $\alpha \geq 0$, and $\beta \geq 0$, keeping the $M \succeq \mathbf{0}$ in closed form:

$$\max_{\alpha\geq 0,\beta\geq 0,\Lambda\in\mathbb{R}^{h\times h}} \min_{M\succeq\mathbf{0}} \ \tfrac{1}{2}||M - \hat{M}||^2 - \mathrm{tr}(\Lambda'(M_1 - I)) + \alpha(\mathrm{tr}(M_{u,u}) - 1) + \beta(\mathrm{tr}(M_{r,r}) - 1).$$

Given (Λ, α, β), the optimal M has a closed form solution via eigenvalue thresholding.

APPENDIX

ADDITIONAL EXPERIMENTS AND PROOFS FOR CHAPTER 4

C.1 Proofs

Proposition 1. $R(f)$ satisfies Assumption 2 if, and only if, $R(f) = \sup_{g \in S} \langle f, g \rangle_{\mathcal{H}}$, where $S \subseteq \mathcal{H}$ is bounded in the RKHS norm and is symmetric ($g \in S \Leftrightarrow -g \in S$).

Recall

Assumption 1. We assume that $R : \mathcal{F} \to \mathbb{R}$ is a semi-norm. Equivalently, $R : \mathcal{F} \to \mathbb{R}$ is convex and $R(\alpha f) = |\alpha| R(f)$ for all $f \in \mathcal{F}$ and $\alpha \in \mathbb{R}$ (absolute homogeneity). Furthermore, we assume R is closed (i.e., lower semicontinuous) *w.r.t.* the topology in $\mathcal{H}$.

Proposition 1 (in a much more general form), to our best knowledge, is due to [135]. We give a "modern" proof below for the sake of completeness.

Proof for Proposition 1.

The "if" part: convexity and absolute homogeneity are trivial. To show the lower semicontinuity, we just need to show the epigraph is closed. Let (f_n, t_n) be a convergent sequence in the epigraph of R, and the limit is (f, t). Then $\langle f_n, g \rangle_{\mathcal{H}} \leq t_n$ for all n and $g \in S$. Tending n to infinty, we get $\langle f, g \rangle_{\mathcal{H}} \leq t$. Take supremum over g on the left-hand side, and we obtain $R(f) \leq t$, i.e., (f, t) is in the epigraph of R.

APPENDIX (Continued)

The "only if" part: A sublinear function R vanishing at the origin is a support function if, and only if, it is closed. Indeed, if R is closed, then its conjugate function

$$\lambda R^*(f^*) = \lambda \left(\sup_f \langle f, f^* \rangle_{\mathcal{H}} - R(f) \right) = \sup_f \langle \lambda f, f^* \rangle_{\mathcal{H}} - R(\lambda f) = R^*(f^*), \tag{C.1}$$

is scaling invariant for any positive λ, i.e., R^* is an indicator function. Conjugating again we have $R = (R^*)^*$ is a support function. So, R is the support function of

$$S = \mathrm{dom}(R^*) = \{g : \langle f, g \rangle_{\mathcal{H}} \leq R(f) \text{ for all } f \in \mathcal{H}\},$$

which is obviously closed. S is also symmetric, because the symmetry of R implies the same for its conjugate function R^*, hence its domain S.

To see S is bounded, assume to the contrary we have $\lambda_n g_n \in S$ with $\|g_n\|_{\mathcal{H}} = 1$ and $\lambda_n \to \infty$. Since R is finite-valued and closed, it is continuous, see [97]. Thus, for any $\delta > 0$ there exists some $\epsilon > 0$ such that $\|f\|_{\mathcal{H}} \leq \epsilon \implies R(f) \leq \delta$. Choose $f = \epsilon g_n$ in the definition of S above we have:

$$\epsilon \lambda_n = \langle \epsilon g_n, \lambda_n g_n \rangle_{\mathcal{H}} \leq R(\epsilon g_n) \leq \delta, \tag{C.2}$$

which is impossible as $\lambda_n \to \infty$. □

APPENDIX (Continued)

Proof of Theorem 8.

a): since $\sum_i \alpha_i G^*_{x_i} = \sum_j \beta_j G^*_{z_j}$, it holds that

$$\left\langle h; \sum_i \alpha_i G^*_{x_i} \right\rangle = \left\langle h; \sum_j \beta_j G^*_{z_j} \right\rangle, \ \forall\ h \in \mathcal{F} \tag{C.3}$$

which implies that

$$\sum_i \alpha_i h(x_i) = \sum_j \beta_i h(z_j), \quad \forall\, h \in \mathcal{F}. \tag{C.4}$$

Therefore

$$\sum_i \alpha_i k(x_i, \cdot) = \sum_j \beta_j k(z_j, \cdot). \tag{C.5}$$

Then apply the linear map T on both sides, and we immediately get $\sum_i \alpha_i \tilde{k}_{x_i} = \sum_j \beta_j \tilde{k}_{z_j}$.

b): suppose otherwise that the completion of $\text{span}\{G^*_x : x \in \mathcal{X}\}$ is not $\mathcal{B}^*$. Then by the Hahn-Banach theorem, there exists a nonzero function $f \in \mathcal{B}$ such that $\langle f; G^*_x \rangle = 0$ for all $x \in \mathcal{X}$. By (Equation 4.8), this means $f(x) = 0$ for all x. Since $\mathcal{B}$ is a Banach space of functions on $\mathcal{X}$, $f = 0$ in $\mathcal{B}$. Contradiction.

The linearity of ι^* follows directly from a) and b). □

APPENDIX (Continued)

To prove Theorem 9, we first introduce five lemmas. To start with, we set up the concept of *polar operator* that will be used extensively in the proof:

$$\mathrm{PO}_{\tilde{\mathsf{B}}}(u) := \arg\max_{v \in \tilde{\mathsf{B}}} \langle v, u \rangle , \quad \forall\, u \in \mathbb{R}^d. \tag{C.6}$$

Here the optimization is convex, and the argmax is uniquely attained because $\tilde{\mathsf{B}}$ is strictly convex. So $\|\cdot\|_{\tilde{\mathcal{B}}^*}$ is differentiable at all u, and the gradient is

$$\nabla \|u\|_{\tilde{\mathcal{B}}^*} = \mathrm{PO}_{\tilde{\mathsf{B}}}(u). \tag{C.7}$$

Lemma 6. *Under Assumptions 3 and 4,*

$$\|g\|_{\mathcal{B}} = \|g^*\|_{\mathcal{B}^*} = \|\iota^*(g^*)\|_{\tilde{\mathcal{B}}^*} = \|\iota(g)\|_{\tilde{\mathcal{B}}} , \quad \forall\ g \in \mathcal{B}. \tag{C.8}$$

Proof. The first equality is trivial, and the third equality is by the definition of $\iota(g)$ in (Equation 4.22). To prove the second equality, let us start by considering $g^* = \sum_i \alpha_i G^*_{x_i}$. Then

$$\|\iota^*(g^*)\|_{\tilde{\mathcal{B}}^*} = \max_{v \in \tilde{\mathsf{B}}} \langle v, \iota^*(g^*) \rangle = \max_{v \in \tilde{\mathsf{B}}} \sum_i \alpha_i \left\langle v, \tilde{k}_{x_i} \right\rangle \tag{C.9}$$

$$\|g^*\|_{\mathcal{B}^*} = \max_{f \in \mathsf{B}} \langle f; g^* \rangle = \max_{f \in \mathsf{B}} \sum_i \alpha_i f(x_i) = \max_{f \in \mathsf{B}} \sum_i \alpha_i \langle f, k(x_i, \cdot) \rangle_{\mathcal{H}} = \max_{f \in \mathsf{B}} \sum_i \alpha_i \left\langle \tilde{f}, \tilde{k}_{x_i} \right\rangle , \tag{C.10}$$

where the last equality is by Assumption 4. So it suffices to show that $\tilde{\mathsf{B}} = \{\tilde{f} : f \in \mathsf{B}\}$.

APPENDIX (Continued)

"$\supseteq$" is trivial because for all $f \in \mathsf{B}$, by Assumption 4,

$$\left\|\tilde{f}\right\|^2 + \max_{z \in S} \left\langle \tilde{z}, \tilde{f} \right\rangle^2 = \|f\|_{\mathcal{H}}^2 + \max_{z \in S} \langle z, f \rangle_{\mathcal{H}}^2 \leq 1. \tag{C.11}$$

"$\subseteq$": for any $v \in \tilde{\mathsf{B}}$, Assumption 3 asserts that there exists $h_v \in \mathcal{H}$ such that $\tilde{h}_v = v$. Then by Assumption 4,

$$\|h_v\|_{\mathcal{H}}^2 + \max_{z \in S} \langle z, h_v \rangle_{\mathcal{H}}^2 = \|v\|^2 + \max_{z \in S} \langle \tilde{z}, v \rangle^2 \leq 1. \tag{C.12}$$

Since both $\|\cdot\|_{\mathcal{B}^*}$ and $\|\cdot\|_{\tilde{\mathcal{B}}^*}$ are continuous, applying the denseness result in part b) of Theorem 8 completes the proof of the second equality in (Equation C.8). □

Lemma 7. *Under Assumptions 3 and 4,*

$$\langle \iota(f), \iota^*(g^*) \rangle = \langle f; g^* \rangle, \quad \forall\ f \in \mathcal{B}, g^* \in \mathcal{B}^*. \tag{C.13}$$

Proof.

$$\langle f; g^* \rangle \overset{\text{by (Equation 4.7)}}{=} [g^*, f^*]_{\mathcal{B}^*} \tag{C.14}$$

$$= \lim_{t \to 0} \frac{1}{2t} \left(\|f^* + t g^*\|_{\mathcal{B}^*}^2 - \|f^*\|_{\mathcal{B}^*}^2 \right) \text{ (by [98])} \tag{C.15}$$

$$= \lim_{t \to 0} \frac{1}{2t} \left[\|\iota^*(f^*) + t\iota^*(g^*)\|_{\tilde{\mathcal{B}}^*}^2 - \|\iota^*(f^*)\|_{\tilde{\mathcal{B}}^*}^2 \right], \tag{C.16}$$

APPENDIX (Continued)

where the last equality is by Lemma 6 and Theorem 8. Now it follows from the polar operator as discussed above that

$$\langle f; g^* \rangle = \left\langle \|\iota^*(f^*)\|_{\tilde{\mathcal{B}}^*} \cdot \mathrm{PO}_{\tilde{\mathsf{B}}}(\iota^*(f^*)), \iota^*(g^*) \right\rangle = \langle \iota(f), \iota^*(g^*) \rangle . \qquad \square$$

Lemma 8. *Under Assumptions 3 and 4,*

$$\tilde{\mathsf{B}} = \iota(\mathsf{B}) := \{\iota(f) : \|f\|_{\mathcal{B}} \leq 1\}. \tag{C.17}$$

Proof. "LHS $\supseteq$ RHS": by Lemma 6, it is obvious that $\|f\|_{\mathcal{B}} \leq 1$ implies $\|\iota(f)\|_{\tilde{\mathcal{B}}} \leq 1$.

"LHS $\subseteq$ RHS": we are to show that for all $v \in \tilde{\mathsf{B}}$, there must exist a $f_v \in \mathsf{B}$ such that $v = \iota(f)$. If $v = 0$, then trivially set $f_v = 0$. In general, due to the polar operator definition (Equation C.6), there must exist $u \in \mathbb{R}^d$ such that

$$v / \|v\|_{\tilde{\mathcal{B}}} = \mathrm{PO}_{\tilde{\mathsf{B}}}(u). \tag{C.18}$$

We next reverse engineer a $q^* \in \mathcal{B}^*$ so that $\iota^*(g^*) = u$. By Assumption 3, there exists $h_u \in \mathcal{H}$ such that $\tilde{h}_u = u$. Suppose $h_u = \sum_i \alpha_i k_{x_i}$. Then define $q^* = \sum_i \alpha_i G^*_{x_i}$, and we recover u by

$$\iota^*(q^*) = \sum_i \alpha_i \tilde{k}_i = \tilde{h}_u = u. \tag{C.19}$$

APPENDIX (Continued)

Apply Lemma 6 and we obtain

$$\|q\|_{\mathcal{B}} = \|\iota^*(q^*)\|_{\tilde{\mathcal{B}}^*} = \|u\|_{\tilde{\mathcal{B}}^*}. \tag{C.20}$$

Now construct

$$f_v = \frac{\|v\|_{\tilde{\mathcal{B}}}}{\|q\|_{\mathcal{B}}}\, q. \tag{C.21}$$

We now verify that $v = \iota(f_v)$. By linearity of ι^*,

$$\iota^*(f_v^*) = \frac{\|v\|_{\tilde{\mathcal{B}}}}{\|q\|_{\mathcal{B}}}\, \iota^*(q^*) = \frac{\|v\|_{\tilde{\mathcal{B}}}}{\|q\|_{\mathcal{B}}}\, u. \tag{C.22}$$

So $\mathrm{PO}_{\tilde{\mathsf{B}}}(\iota^*(f_v^*)) = v/\|v\|_{\tilde{\mathcal{B}}}$ and plugging into (Equation 4.22),

$$\begin{aligned}\iota(f_v) &= \|\iota^*(f_v^*)\|_{\tilde{\mathcal{B}}^*}\, \mathrm{PO}_{\tilde{\mathsf{B}}}(\iota^*(f_v^*)) = \frac{\|v\|_{\tilde{\mathcal{B}}}}{\|q\|_{\mathcal{B}}}\, \|u\|_{\tilde{\mathcal{B}}^*}\, \frac{1}{\|v\|_{\tilde{\mathcal{B}}}} v \qquad \text{(C.23)}\\ &= v. \quad \text{(by (Equation C.20))}\end{aligned}$$

□

Lemma 9. *Under Assumptions 3 and 4,*

$$\tilde{\mathsf{B}}^* = \iota^*(\mathsf{B}^*) := \{\iota^*(g^*) : \|g^*\|_{\mathcal{B}^*} \le 1\}. \tag{C.24}$$

APPENDIX (Continued)

Proof. "LHS $\supseteq$ RHS": By definition of dual norm, any $g^* \in \mathsf{B}^*$ must satisfy

$$\langle f; g^* \rangle \leq 1, \quad \forall\ f \in \mathsf{B}. \tag{C.25}$$

Again, by the definition of dual norm, we obtain

$$\begin{aligned} \|\iota^*(g^*)\|_{\tilde{\mathcal{B}}^*} &= \sup_{v \in \tilde{\mathsf{B}}} \langle v, \iota^*(g^*) \rangle && \text{(C.26)} \\ &= \sup_{f \in \mathsf{B}} \langle \iota(f), \iota^*(g^*) \rangle \quad \text{(Lemma 8)} && \text{(C.27)} \\ &= \sup_{f \in \mathsf{B}} \langle f; g^* \rangle \quad \text{(by Lemma 7)} && \text{(C.28)} \\ &\leq 1. && \text{(C.29)} \end{aligned}$$

"LHS $\subseteq$ RHS": Any $u \in \mathbb{R}^d$ with $\|u\|_{\tilde{\mathcal{B}}^*} = 1$ must satisfy

$$\max_{v \in \tilde{\mathsf{B}}} \langle u, v \rangle = 1. \tag{C.30}$$

Denote $v = \arg\max_{v \in \tilde{\mathsf{B}}} \langle u, v \rangle$ which must be uniquely attained. So $\|v\|_{\tilde{\mathcal{B}}} = 1$. Then Lemma 8 implies that there exists a $f \in \mathsf{B}$ such that $\iota(f) = v$. By duality,

$$\max_{u \in \tilde{\mathsf{B}}^*} \langle v, u \rangle = 1, \tag{C.31}$$

and u is the unique maximizer. Now note

$$\langle v, \iota^*(f^*)\rangle = \langle \iota(f), \iota^*(f^*)\rangle = \langle f; f^*\rangle = 1, \tag{C.32}$$

where the last equality is derived from Lemma 6 with

$$\|f\|_{\mathcal{B}} = \|\iota(f)\|_{\tilde{\mathcal{B}}} = \|v\|_{\tilde{\mathcal{B}}} = 1. \tag{C.33}$$

Note from Lemma 6 that $\|\iota^*(f^*)\|_{\tilde{\mathcal{B}}^*} = \|f\|_{\mathcal{B}} = 1$. So $\iota^*(f^*)$ is a maximizer in (Equation C.31), and as a result, $u = \iota^*(f^*)$.

If $\|u\|_{\tilde{\mathcal{B}}^*} < 1$, then just construct f as above for $u/\|u\|_{\tilde{\mathcal{B}}^*}$, and then multiply it by $\|u\|_{\tilde{\mathcal{B}}^*}$. The result will meet our need thanks to the linearity of ι^* from Theorem 8. □

Lemma 10. *Under Assumptions 3 and 4,*

$$\max_{v\in\tilde{\mathsf{B}}} \langle v, \iota^*(g^*)\rangle = \max_{f\in\mathsf{B}} \langle f; g^*\rangle, \quad \forall g^* \in \mathcal{B}^*. \tag{C.34}$$

Moreover, by Theorem 7, the argmax of the RHS is uniquely attained at $f = g/\|g\|_{\mathcal{B}}$, and the argmax of the LHS is uniquely attained at $v = \iota(g)/\|\iota(g)\|_{\tilde{\mathcal{B}}}$.

APPENDIX (Continued)

Proof. LHS $\geq$ RHS: Let f^{opt} be an optimal solution to the RHS. Then by Lemma 8, $\iota(f^{opt}) \in \tilde{\mathsf{B}}$, and so

$$\begin{aligned}
\text{RHS} &= \left\langle f^{opt}; g^* \right\rangle && \text{(C.35)}\\
&= \left\langle \iota(f^{opt}), \iota^*(g^*) \right\rangle \quad \text{(by Lemma 7)} && \text{(C.36)}\\
&\leq \max_{v \in \tilde{\mathsf{B}}} \langle v, \iota^*(g^*) \rangle && \text{(C.37)}\\
&= \text{LHS}. && \text{(C.38)}
\end{aligned}$$

LHS $\leq$ RHS: let v^{opt} be an optimal solution to the LHS. Then by Lemma 8, there is $f_{v^{opt}} \in \mathsf{B}$ such that $\iota(f_{v^{opt}}) = v^{opt}$. So

$$\begin{aligned}
\text{LHS} &= \left\langle v^{opt}, \iota^*(g^*) \right\rangle && \text{(C.39)}\\
&= \langle \iota(f_{v^{opt}}), \iota^*(g^*) \rangle && \text{(C.40)}\\
&= \langle f_{v^{opt}}; g^* \rangle \quad \text{(by Lemma 7)} && \text{(C.41)}\\
&\leq \max_{f \in \mathsf{B}} \langle f; g^* \rangle \quad \text{(since } f_{v^{opt}} \in \mathsf{B}\text{)} && \text{(C.42)}\\
&= \text{RHS}. && \square
\end{aligned}$$

Proof of Theorem 9. Let $f \in \mathcal{B}$ and $\alpha \in \mathbb{R}$. Then $(\alpha f)^* = \alpha f^*$, and by (Equation 4.22) and Theorem 8,

$$\iota(\alpha f) = \|\iota^*(\alpha f^*)\|_{\tilde{\mathcal{B}}^*} \cdot \text{PO}_{\tilde{\mathsf{B}}}(\iota^*(\alpha f^*)) = |\alpha| \, \|\iota^*(f^*)\|_{\tilde{\mathcal{B}}^*} \cdot \text{PO}_{\tilde{\mathsf{B}}}(\alpha \iota^*(f^*)). \tag{C.43}$$

APPENDIX (Continued)

By the symmetry of $\tilde{\mathsf{B}}$,

$$\iota(\alpha f) = |\alpha| \, \|\iota^*(f^*)\|_{\tilde{\mathcal{B}}^*} \cdot \operatorname{sign}(\alpha) \operatorname{PO}_{\tilde{\mathsf{B}}}(\iota^*(f^*)) = \alpha \, \iota(f). \tag{C.44}$$

Finally we show $\iota(f_1 + f_2) = \iota(f_1) + \iota(f_2)$ for all $f_1, f_2 \in \mathcal{B}$. Observe

$$\langle \iota(f_1) + \iota(f_2), \iota^*((f_1 + f_2)^*) \rangle \tag{C.45}$$

$$= \langle \iota(f_1), \iota^*((f_1 + f_2)^*) \rangle + \langle \iota(f_2), \iota^*((f_1 + f_2)^*) \rangle \tag{C.46}$$

$$= \langle f_1; (f_1 + f_2)^* \rangle + \langle f_2; (f_1 + f_2)^* \rangle \tag{C.47}$$

$$= \langle f_1 + f_2; (f_1 + f_2)^* \rangle \, . \tag{C.48}$$

Therefore

$$\langle v, \iota^*((f_1 + f_2)^*) \rangle = \left\langle \frac{f_1 + f_2}{\|f_1 + f_2\|_{\mathcal{B}}}; (f_1 + f_2)^* \right\rangle, \tag{C.49}$$

$$\text{where} \quad v = \frac{\iota(f_1) + \iota(f_2)}{\|f_1 + f_2\|_{\mathcal{B}}}. \tag{C.50}$$

We now show $\|v\|_{\tilde{\mathcal{B}}} = 1$, which is equivalent to

$$\|\iota(f_1) + \iota(f_2)\|_{\tilde{\mathcal{B}}} = \|f_1 + f_2\|_{\mathcal{B}} \, . \tag{C.51}$$

Indeed, this can be easily seen from

$$\begin{aligned}
\text{LHS} &= \sup_{u \in \tilde{\mathsf{B}}^*} \langle \iota(f_1) + \iota(f_2), u \rangle && \text{(C.52)} \\
&= \sup_{g^* \in \mathsf{B}^*} \langle \iota(f_1) + \iota(f_2), \iota^*(g^*) \rangle \quad \text{(Lemma 9)} && \text{(C.53)} \\
&= \sup_{g^* \in \mathsf{B}^*} \langle f_1 + f_2; g^* \rangle \quad \text{(by Lemma 7)} && \text{(C.54)} \\
&= \text{RHS}. && \text{(C.55)}
\end{aligned}$$

By Lemma 10,

$$\max_{v \in \tilde{\mathsf{B}}} \langle v, \iota^*((f_1 + f_2)^*) \rangle = \max_{f \in \mathsf{B}} \langle f; (f_1 + f_2)^* \rangle . \tag{C.56}$$

Since the right-hand side is optimized at $f = (f_1 + f_2)/\|f_1 + f_2\|_{\mathcal{B}}$, we can see from (Equation C.49) and $\|v\|_{\tilde{\mathcal{B}}} = 1$ that $v = \mathrm{PO}_{\tilde{\mathsf{B}}}(\iota^*((f_1 + f_2)^*))$. Finally by definition (Equation 4.22), we conclude

$$\begin{aligned}
\iota(f_1 + f_2) &= \|\iota^*((f_1 + f_2)^*)\|_{\tilde{\mathcal{B}}^*} \cdot \mathrm{PO}_{\tilde{\mathsf{B}}}(\iota^*((f_1 + f_2)^*)) && \text{(C.57)} \\
&= \|f_1 + f_2\|_{\mathcal{B}}\, v \quad \text{(by Lemma 6)} && \text{(C.58)} \\
&= \iota(f_1) + \iota(f_2). && \square
\end{aligned}$$

Proof of Theorem 11. We assume that the kernel k is smooth and the function

$$z_{ij}(\lambda) = \tfrac{\partial}{\partial \lambda} k((\tilde{x}_\lambda, \tilde{y}_\lambda), (\cdot, \cdot)).$$

is in L_p so that R_{ij} is well-defined and finite-valued.

Clearly, using the representer theorem we can rewrite

$$R_{ij}(f) = \| \langle f, z_{ij}(\lambda) \rangle_{\mathcal{H}} \|_p. \tag{C.59}$$

Thus, R_{ij} is the composition of the linear map $f \mapsto g(\lambda; f) := \langle f, z_{ij}(\lambda) \rangle_{\mathcal{H}}$ and the L_p norm $g \mapsto \|g(\lambda)\|_p$. It follows from the chain rule that R_{ij} is convex, absolutely homogeneous, and Gâteaux differentiable (recall that the L_p norm is Gâteaux differentiable for $p \in (1, \infty)$). □

C.2 Analysis under Inexact Euclidean Embedding

We first rigorously quantify the inexactness in the Euclidean embedding T: $\mathcal{H} \to \mathbb{R}^d$, where $Tf = \tilde{f}$. To this end, let us consider a subspace based embedding, such as Nyström approximation. Here let T satisfy that there exists a countable set of orthonormal bases $\{e_i\}_{i=1}^{\infty}$ of $\mathcal{H}$, such that

1. $Te_k = 0$ for all $k > d$,
2. $\langle Tf, Tg \rangle = \langle f, g \rangle_{\mathcal{H}}, \quad \forall f, g \in V := \text{span}\{e_1, \ldots, e_d\}$.

Clearly the Nyström approximation in (Equation 4.20) satisfies these conditions, where $d = n$, and $\{e_1, \ldots, e_d\}$ is any orthornormal basis of $\{k_{z_1}, \ldots, k_{z_d}\}$ (assuming d is no more than the dimensionality of $\mathcal{H}$).

APPENDIX (Continued)

As an immediate consequence, $\{Te_1, \ldots, Te_d\}$ forms an orthonormal basis of $\mathbb{R}^d$: $\langle Te_i, Te_j \rangle = \langle e_i, e_j \rangle_{\mathcal{H}} = \delta_{ij}$ for all $i, j \in [d]$. Besides, T is contractive because for all $f \in \mathcal{F}$,

$$\|Tf\|^2 = \left\| \sum_{i=1}^{d} \langle f, e_i \rangle_{\mathcal{H}} Te_i \right\|^2 = \sum_{i=1}^{d} \langle f, e_i \rangle_{\mathcal{H}}^2 \leq \|f\|_{\mathcal{H}}^2 . \tag{C.60}$$

By Definition 6, obviously k_{z_i} is 0-approximable under the Nyström approximation. If both f and g are ϵ-approximable, then $f + g$ must be (2ϵ)-approximable.

Lemma 11. *Let $f \in \mathcal{H}$ be ϵ-approximable by T, then for all $u \in \mathcal{H}$,*

$$|\langle u, f \rangle_{\mathcal{H}} - \langle Tu, Tf \rangle| \leq \epsilon \|u\|_{\mathcal{H}} . \tag{C.61}$$

Proof. Let $f = \sum_{i=1}^{\infty} \alpha_i e_i$ and $u = \sum_{i=1}^{\infty} \beta_i e_i$. Then

$$\begin{aligned}
& |\langle u, f \rangle_{\mathcal{H}} - \langle Tu, Tf \rangle| && \text{(C.62)} \\
= & \left| \sum_{i=1}^{\infty} \alpha_i \beta_i - \left\langle \sum_{i=1}^{d} \alpha_i Te_i, \sum_{j=1}^{d} \beta_j Te_j \right\rangle \right| && \text{(C.63)} \\
= & \left| \sum_{i=d+1}^{\infty} \alpha_i \beta_i \right| && \text{(C.64)} \\
\leq & \left(\sum_{i=d+1}^{\infty} \alpha_i^2 \right)^{1/2} \left(\sum_{j=d+1}^{\infty} \beta_j^2 \right)^{1/2} && \text{(C.65)} \\
\leq & \epsilon \|u\|_{\mathcal{H}} . && \square
\end{aligned}$$

APPENDIX (Continued)

Proof of Theorem 10. We first prove (Equation 4.29). Note for any $u \in \mathcal{F}$,

$$\langle u; g^* \rangle = [u, g] = \lim_{t \to 0} \frac{1}{2} \left[\|tu + g\|_{\mathcal{B}}^2 - \|g\|_{\mathcal{B}}^2 \right] = \langle u, g + \nabla R^2(g) \rangle_{\mathcal{H}} . \tag{C.66}$$

The differentiability of R^2 is guaranteed by the Gâteaux differentiability. Letting $g^* = \sum_i \alpha_i G_{v_i}^*$, it follows that

$$\langle u; g^* \rangle = \sum_i \alpha_i u(v_i) = \left\langle u, \sum_i \alpha_i k_{v_i} \right\rangle_{\mathcal{H}} . \tag{C.67}$$

So $\sum_i \alpha_i k_{v_i} = g + \nabla R^2(g)$, and by the definition of ι^*

$$\iota^*(g^*) = \sum_i \alpha_i T k_{v_i} = T a_g \tag{C.68}$$

$$\text{where} \quad a_g := \sum_i \alpha_i k_{v_i} = g + \nabla R^2(g). \tag{C.69}$$

Similarly,

$$\iota^*(f^*) = T a_f, \quad \text{where} \quad a_f := f + \nabla R^2(f). \tag{C.70}$$

By assumption $\arg\max_{h \in S} \langle h, g \rangle_{\mathcal{H}}$ is ϵ-approximable, and hence a_g is $O(\epsilon)$-approximable. Similarly, a_f is also $O(\epsilon)$-approximable.

APPENDIX (Continued)

Now let us consider

$$v^\circ := \arg \max_{v \in \mathbb{R}^d : \|v\|^2 + \sup_{h \in S} \langle v, Th \rangle^2 \leq 1} \langle v, Ta_f \rangle \tag{C.71}$$

$$u^\circ := \arg \max_{u \in \mathcal{F} : \|u\|_{\mathcal{H}}^2 + \sup_{h \in S} \langle u, h \rangle_{\mathcal{H}}^2 \leq 1} \langle u, a_f \rangle_{\mathcal{H}} . \tag{C.72}$$

By definition, $\iota(f) = v^\circ$. Also note that $u^\circ = f$ because $\langle u, a_f \rangle_{\mathcal{H}} = \langle u; f^* \rangle$ for all $u \in \mathcal{F}$. We will then show that

$$\|\iota(f) - Tf\| = \|v^\circ - Tu^\circ\| = O(\sqrt{\epsilon}), \tag{C.73}$$

which allows us to derive that

$$\langle f; g^* \rangle = \langle f, a_g \rangle_{\mathcal{H}} \tag{C.74}$$

$$= \langle Tf, Ta_g \rangle + O(\epsilon) \quad \text{(by Lemma 11)} \tag{C.75}$$

$$= \langle Tu^\circ, Ta_g \rangle + O(\epsilon) \tag{C.76}$$

$$= \langle v^\circ, Ta_g \rangle + O(\sqrt{\epsilon}) \quad \text{(by (Equation C.73))} \tag{C.77}$$

$$= \langle \iota(f), \iota^*(g^*) \rangle + O(\sqrt{\epsilon}). \quad \text{(by (Equation C.68))} \tag{C.78}$$

Finally, we prove (Equation C.73). Denote

$$w^\circ := \arg \max_{w \in \mathcal{F} : \|w\|_{\mathcal{H}}^2 + \sup_{h \in S} \langle Tw, Th \rangle^2 \leq 1} \langle w, a_f \rangle_{\mathcal{H}} . \tag{C.79}$$

APPENDIX (Continued)

We will prove that $\|v^\circ - Tw^\circ\| = O(\epsilon^2)$ and $\|u^\circ - w^\circ\|_{\mathcal{H}} = O(\sqrt{\epsilon})$. They will imply (Equation C.73) because by the contractivity of T, $\|T(u^\circ - w^\circ)\| \le \|u^\circ - w^\circ\|_{\mathcal{H}}$.

Step 1: $\|v^\circ - Tw^\circ\| = O(\epsilon^2)$. Let $w = w_1 + w_2$ where $w_1 \in V$ and $w_2 \in V^\perp$. So $Tw = Tw_1$ and $\|Tw\| = \|w_1\|_{\mathcal{H}}$. Similarly decompose a_f as $a_1 + a_2$, where $a_1 = Ta_f \in V$ and $a_2 \in V^\perp$. Now the optimization over w becomes

$$\max_{w_1 \in V, w_2 \in V^\perp} \quad \langle w_1, a_1 \rangle_{\mathcal{H}} + \langle w_2, a_2 \rangle_{\mathcal{H}} \tag{C.80}$$

$$s.t. \quad \|w_1\|_{\mathcal{H}}^2 + \|w_2\|_{\mathcal{H}}^2 + \sup_{h \in S} \langle Tw_1, Th \rangle^2 \le 1. \tag{C.81}$$

Let $\|w_2\|^2 = 1 - \alpha$ where $\alpha \in [0, 1]$. Then the optimal value of $\langle w_2, a_2 \rangle_{\mathcal{H}}$ is $\sqrt{1-\alpha}\,\|a_2\|_{\mathcal{H}}$. Since $\langle w_1, a_1 \rangle_{\mathcal{H}} = \langle Tw_1, Ta_1 \rangle$, the optimization over w_1 can be written as

$$\min_{w_1 \in V} \langle Tw_1, Ta_1 \rangle \tag{C.82}$$

$$s.t. \ \|Tw_1\|^2 + \sup_{h \in S} \langle Tw_1, Th \rangle^2 \le \alpha. \tag{C.83}$$

Change variable by $v = Tw_1$. Then compare with the optimization of v in (Equation C.71), and we can see that $v^\circ = Tw_1^\circ / \sqrt{\alpha}$. Overall the optimal objective value of (Equation C.80) under $\|w_2\|^2 = 1 - \alpha$ is $\sqrt{1-\alpha}\,\|a_2\|_{\mathcal{H}} + \sqrt{\alpha} p$ where p is the optimal objective value of (Equation C.71). So the optimal α is $\frac{p^2}{p^2 + \|a_2\|_{\mathcal{H}}^2}$, and hence

$$\|v^\circ - Tw^\circ\| = \|v^\circ - Tw_1^\circ\| = \left\|v^\circ - \sqrt{\alpha} v^\circ\right\| = (1 - \sqrt{\alpha})\,\|v^\circ\| \le 1 - \sqrt{\alpha}. \tag{C.84}$$

APPENDIX (Continued)

Since a_f is $O(\epsilon)$-approximable, so $\|a_2\|_{\mathcal{H}} = O(\epsilon)$ and

$$1 - \sqrt{\alpha} = \frac{1-\alpha}{1+\sqrt{\alpha}} = O(\|a_2\|_{\mathcal{H}}^2) = O(\epsilon^2). \tag{C.85}$$

Step 2: $\|u^\circ - w^\circ\|_{\mathcal{H}} = O(\sqrt{\epsilon})$. Motivated by Theorem 17, we consider two equivalent problems:

$$\hat{u}^\circ = \arg \max_{u \in \mathcal{F}: \langle u, a_f \rangle_{\mathcal{H}} = 1} \left\{ \|u\|_{\mathcal{H}}^2 + \sup_{h \in S} \langle u, h \rangle_{\mathcal{H}}^2 \right\} \tag{C.86}$$

$$\hat{w}^\circ = \arg \max_{w \in \mathcal{F}: \langle w, a_f \rangle_{\mathcal{H}} = 1} \left\{ \|w\|_{\mathcal{H}}^2 + \sup_{h \in S} \langle Tw, Th \rangle^2 \right\}. \tag{C.87}$$

Again we can decompose u into $U := \text{span}\{a_f\}$ and its orthogonal space $U^\perp$. Since $\langle u, a_f \rangle_{\mathcal{H}} = 1$, the component of u in U must be $\bar{a}_f := a_f / \|a_f\|_{\mathcal{H}}^2$. So

$$\hat{u}^\circ = \bar{a}_f + \arg \max_{u^\perp \in U^\perp} \left\{ \left\| u^\perp \right\|_{\mathcal{H}}^2 + \sup_{h \in S} \left\langle u^\perp + \bar{a}_f, h \right\rangle_{\mathcal{H}}^2 \right\}. \tag{C.88}$$

Similarly,

$$w^\circ = \bar{a}_f + \arg \max_{w^\perp \in U^\perp} \left\{ \left\| w^\perp \right\|_{\mathcal{H}}^2 + \sup_{h \in S} \left\langle T(w^\perp + \bar{a}_f), Th \right\rangle_{\mathcal{H}}^2 \right\}. \tag{C.89}$$

APPENDIX (Continued)

We now compare the objective in the above two argmax forms. Since any $h \in S$ is ϵ-approximable, so for any $x \in \mathcal{F}$:

$$|\langle x, h\rangle_{\mathcal{H}} - \langle Tx, Th\rangle_{\mathcal{H}}| = O(\epsilon). \tag{C.90}$$

Therefore tying $u^\perp = w^\perp = x$, the objectives in the argmax of (Equation C.88) and (Equation C.89) differ by at most $O(\epsilon)$. Therefore their optimal objective values are different by at most $O(\epsilon)$. Since both objectives are (locally) strongly convex in $U^\perp$, the RKHS distance between the optimal $u^\perp$ and the optimal $w^\perp$ must be $O(\sqrt{\epsilon})$. As a result $\|\hat{u}^\circ - \hat{w}^\circ\|_{\mathcal{H}} = O(\sqrt{\epsilon})$.

Finally to see $\|u^\circ - w^\circ\|_{\mathcal{H}} = O(\epsilon)$, just note that by Theorem 17, u° and w° simply renormalize $\hat{u}^\circ$ and $\hat{w}^\circ$ to the unit sphere of $\|\cdot\|_{\mathcal{B}}$, respectively. So again $\|u^\circ - w^\circ\|_{\mathcal{H}} = O(\sqrt{\epsilon})$.

In the end, we prove (Equation 4.30). The proof of $\iota(\alpha f) = \alpha\iota(f)$ is exactly the same as that for Theorem 8. To prove (Equation 4.30), note that $f + g$ is (2ϵ)-approximable. Therefore applying (Equation C.73) on f, g, $f + g$, we get

$$\|\iota(f) - Tf\| = O(\sqrt{\epsilon}), \tag{C.91}$$

$$\|\iota(fg) - Tg\| = O(\sqrt{\epsilon}), \tag{C.92}$$

$$\|\iota(f + g) - T(f + g)\| = O(\sqrt{\epsilon}). \tag{C.93}$$

Combining these three relations, we conclude (Equation 4.30). □

APPENDIX (Continued)

C.3 Solving the Polar Operator

Theorem 17. *Suppose J is continuous and $J(\alpha x) = \alpha^2 J(x) \geq 0$ for all x and $\alpha \geq 0$. Then x is an optimal solution to*

$$P: \quad \max_x a^\top x, \quad s.t. \quad J(x) \leq 1, \tag{C.94}$$

if, and only if, $J(x) = 1$, $c := a^\top x > 0$, and $\hat{x} := x/c$ is an optimal solution to

$$Q: \quad \min_x J(x), \quad s.t. \quad a^\top x = 1. \tag{C.95}$$

Proof. We first show the "only if" part. Since $J(0) = 0$ and J is continuous, the optimal objective value of P must be positive. Therefore $c > 0$. Also note the optimal x for P must satisfy $J(x) = 1$ because otherwise one can scale up x to increase the objective value of P. To show $\hat{x}$ optimizes Q, suppose otherwise there exists y such that

$$a^\top y = 1, \quad J(y) < J(\hat{x}). \tag{C.96}$$

Then letting

$$z = J(y)^{-1/2} y, \tag{C.97}$$

we can verify that

$$J(z) = 1, \tag{C.98}$$

$$a^\top z = J(y)^{-1/2} a^\top y = J(y)^{-1/2} > J(\hat{x})^{-1/2} = cJ(x)^{-1/2} = c = a^\top x. \tag{C.99}$$

So z is a feasible solution for P, and is strictly better than x. Contradiction.

We next show the "if" part: for any x, if $J(x) = 1$, $c := a^\top x > 0$, and $\hat{x} := x/c$ is an optimal solution to Q, then x must optimize P. Suppose otherwise there exists y, such that $J(y) \leq 1$ and $a^\top y > a^\top x > 0$. Then consider $z := y/a^\top y$. It is obviously feasible for Q, and

$$J(z) = (a^\top y)^{-2} J(y) < (a^\top x)^{-2} J(y) \leq (a^\top x)^{-2} J(x) = J(\hat{x}). \tag{C.100}$$

This contradicts with the optimality of $\hat{x}$ for Q. □

C.3.0.0.1 Projection to hyperplane

To solve problem (Equation 4.27), we use LBFGS with each step projected to the feasible domain, a hyperplane. This requires solving, for given c and a,

$$\min_x \frac{1}{2} \|x - c\|^2, \quad s.t. \quad a^\top x = 1. \tag{C.101}$$

APPENDIX (Continued)

Write out its Lagrangian and apply strong duality thanks to convexity:

$$\min_x \max_\lambda \frac{1}{2}\|x-c\|^2 - \lambda(a^\top x - 1) \tag{C.102}$$

$$= \max_\lambda \min_x \frac{1}{2}\|x-c\|^2 - \lambda(a^\top x - 1) \tag{C.103}$$

$$= \max_\lambda \frac{1}{2}\lambda^2\|a\|^2 - \lambda^2\|a\|^2 - \lambda a^\top c + \lambda, \tag{C.104}$$

where $x = c + \lambda a$. The last step has optimal

$$\lambda = (1 - a^\top c)/\|a\|^2 . \tag{C.105}$$

C.4 Gradient in Dual Coefficients

We first consider the case where S is a finite set, and denote as z_i the RKHS Nyström approximation of its i-th element. When f^* has the form of (Equation 4.12), we can compute $\iota(f)$ by using the Euclidean counterpart of Theorem 7 as follows:

$$\arg\max_u \quad u^\top \sum\nolimits_j c_j k_j \tag{C.106}$$

$$s.t. \quad \|u\|^2 + (z_i^\top u)^2 \le 1, \quad \forall\, i, \tag{C.107}$$

where k_j the the Nyström approximation of $k(x_j, \cdot)$.

APPENDIX (Continued)

Writing out the Lagrangian with dual variables λ_i:

$$u^\top \sum_j c_j k_j + \sum_i \lambda_i \left(\|u\|^2 + (z_i^\top u)^2 - 1 \right), \tag{C.108}$$

we take derivative with respect to u:

$$X^\top c + 2\mathbf{1}^\top \lambda u + 2Z\Lambda Z^\top u = 0. \tag{C.109}$$

where $X = (k_1, k_2, \ldots)$, $Z = (z_1, z_2, \ldots)$, $\lambda = (\lambda_1, \lambda_2, \ldots)$, $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots)$ (diagonal matrix), and $\mathbf{1}$ is a vector of all ones. This will hold for $c + \Delta_c$, $\lambda + \Delta_\lambda$ and $u + \Delta_u$:

$$X^\top (c + \Delta_c) + 2\mathbf{1}^\top (\lambda + \Delta_\lambda)(u + \Delta u) + 2Z(\Lambda + \Delta_\Lambda)Z^\top (u + \Delta_u) = 0. \tag{C.110}$$

Subtract it by (Equation C.109), we obtain

$$X^\top \Delta_c + 2(\mathbf{1}^\top \Delta_\lambda)u + 2(\mathbf{1}^\top \lambda)\Delta_u + 2Z\Delta_\Lambda Z^\top u + 2Z\Lambda Z^\top \Delta_u = 0. \tag{C.111}$$

The complementary slackness writes

$$\lambda_i(\|u\|^2 + (z_i^\top u)^2 - 1) = 0. \tag{C.112}$$

APPENDIX (Continued)

This holds for $\lambda + \Delta_\lambda$ and $u + \Delta_u$:

$$(\lambda_i + \Delta_{\lambda_i})(\|u + \Delta_u\|^2 + (z_i^\top u + z_i^\top \Delta_u)^2 - 1) = 0. \tag{C.113}$$

Subtract it by (Equation C.112), we obtain

$$\Delta_{\lambda_i}(\|u\|^2 + (z_i^\top u)^2 - 1) + 2\lambda_i(u + (z_i^\top u)z_i)^\top \Delta_u = 0. \tag{C.114}$$

Putting together (Equation C.111) and (Equation C.114), we obtain

$$S \begin{pmatrix} \Delta_u \\ \\ \Delta_\lambda \end{pmatrix} = \begin{pmatrix} -X^\top \Delta_c \\ \\ 0 \end{pmatrix}, \tag{C.115}$$

where S is

$$\begin{pmatrix} 2(\mathbf{1}^\top \lambda)I + 2Z\Lambda Z^\top & 2u\mathbf{1}^\top + 2Z\,\mathrm{diag}(Z^\top u) \\ \\ 2\Lambda(\mathbf{1}u^\top + \mathrm{diag}(Z^\top u)Z^\top) & \mathrm{diag}(\|u\|^2 + (z_i^\top u)^2 - 1) \end{pmatrix}. \tag{C.116}$$

Therefore

$$\frac{\mathrm{d}u}{\mathrm{d}c} = \begin{pmatrix} I & 0 \end{pmatrix} S^{-1} \begin{pmatrix} -X^\top \\ \\ 0 \end{pmatrix}. \tag{C.117}$$

Finally we investigate the case when S is not finite. In such a case, the elements z in S that attain $\|u\|^2 + (z^\top u)^2 = 1$ for the optimal u are still finite in general. For all other z, the complementary slackness implies the corresponding λ element is 0. As a result, the corresponding diagonal entry in the bottom-right block of S is nozero, while the corresponding row in the bottom-left block of S is straight 0. So the corresponding entry in Δ_λ in (Equation C.115) plays no role, and can be pruned. In other words, all $z \in S$ such that $\|u\|^2 + (z^\top u)^2 < 1$ can be treated as nonexistent.

The empirical loss depends on $f(x_j)$, which can be computed by $\iota(f)^\top k_j$. Since $\iota(f) = (u^\top \sum_j c_j k_j)u$, (Equation C.117) allows us to backpropagate the gradient in $\iota(f)$ into the gradient in $\{c_j\}$.

C.5 Experiments

C.5.1 Mixup

We next investigated the performance of Embed on mixup.

Datasets.

We experimented with three image datasets: MNIST, USPS, and Fashion MNIST, each containing 10 classes. From each dataset, we drew n example for training and n examples for testing, with n varied in 500 and 1000. Based on the training data, p number of pairs were drawn from it.

Both Vanilla and Embed used Gaussian RKHS, along with Nyström approximation whose landmark points consisted of the entire training set. The vanilla mixup optimizes the objective (Equation 4.31) averaged over all sampled pairs. Following [73], The λ was generated from a

Beta distribution, whose parameter was tuned to optimize the performance. Again, Embed was trained with a linear classifier.

Algorithms.

We first ran mixup with stochastic optimization where pairs were drawn on the fly. Then we switched to batch training of mixup (denoted as Vanilla), with the number of sampled pair increased from $p = n$, $2n$, up to $5n$. It turned out when $p = 4n$, the performance already matches the best test accuracy of the online stochastic version, which generally witnesses much more pairs. Therefore we also varied p in $\{n, 2n, 4n\}$ when training Embed. each setting was evaluated 10 times with randomly sampled training and test data. The mean and standard deviation are reported in Table Table IX.

Results.

As Table Table IX shows, Embed achieves higher accuracy than Vanilla on almost all datasets and combinations of n and p. The margin tends to be higher when the training set size (n and p) is smaller. Besides, Vanilla achieves the highest accuracy at $p = 4n$.

C.5.2 Additional experiments for structured multilabel prediction

Here, we provide more detailed results for our method applied to structured multilabel prediction, as described in Section 4.6.

Accuracy on multiple runs.

We repeated the experiment, detailed in Section 4.7.3 and tabulated in Table Table X ten times for all the three algorithms. Figures Figure 12,Figure 13,Figure 14 show the accuracy plot of our method (Embed) compared with baselines (ML-SVM and HR-SVM) on Enron [105],

APPENDIX (Continued)

WIPO [106], Reuters [107] datasets with $100/100, 200/200, 500/500$ randomly drawn train/test examples over 10 runs.

Comparing constraint violations.

In this experiment, we demonstrate the effectiveness of the model's ability to embed structures explicitly. Recall that for the structured multilabel prediction task, we wanted to incorporate two types of constraints (i) *implication*, (ii) *exclusion*. To test if our model (Embed) indeed learns representations that respect these constraints, we counted the number of test examples that violated the implication and exclusion constraints from the predictions. We repeated the test for ML-SVM and HR-SVM.

We observed that HR-SVM and Embed successfully modeled implications on all the datasets. This is not surprising as HR-SVM takes the class hierarchy into account. The exclusion constraint, on the other hand, is a "derived" constraint and is not directly modeled by HR-SVM. Therefore, on datasets where Embed performed significantly better than HR-SVM, we might expect fewer exclusion violations by Embed compared to HR-SVM. To verify this intuition, we considered the Enron dataset with 200/200 train/test split where Embed performed better than HR-SVM. The constraint violations are shown as a line plot in Figure Figure 15, with the constraint index on the x-axis and number of examples violating the constraint on the y-axis.

Recall again that predictions in Embed for multilabel prediction are made using a linear classifier. Therefore the superior performance of Embed in this case, can be attributed to accurate representations learned by the model.

APPENDIX (Continued)

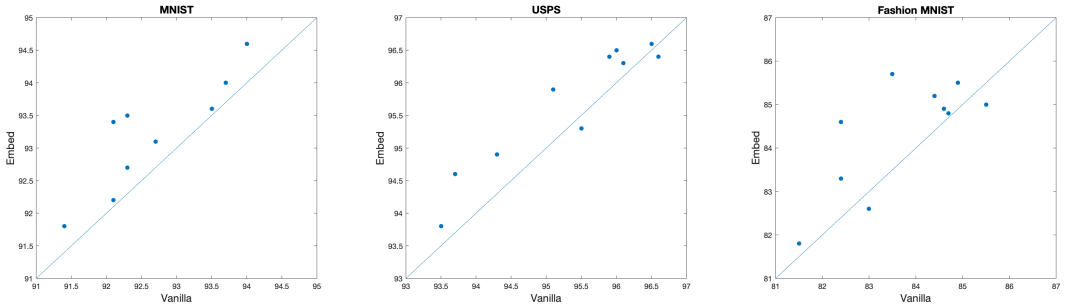


Figure 10: Scatter plot of test accuracy for mixup: $n = 1000$, $p = 4n$

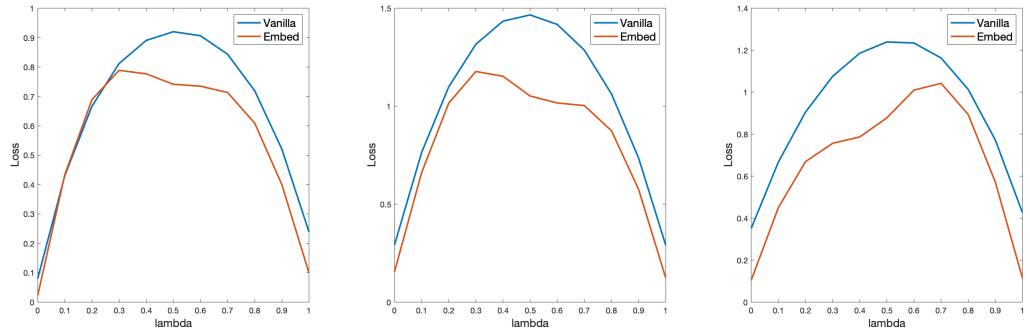
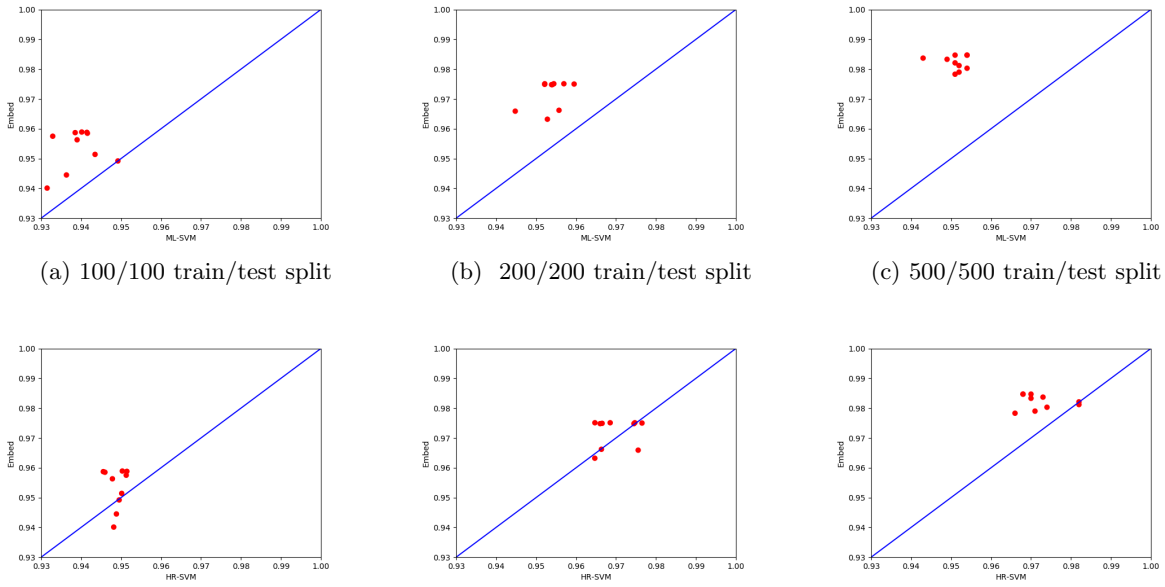


Figure 11: Plots of three different pairs of test examples, showing how loss values change as a function of λ

APPENDIX (Continued)

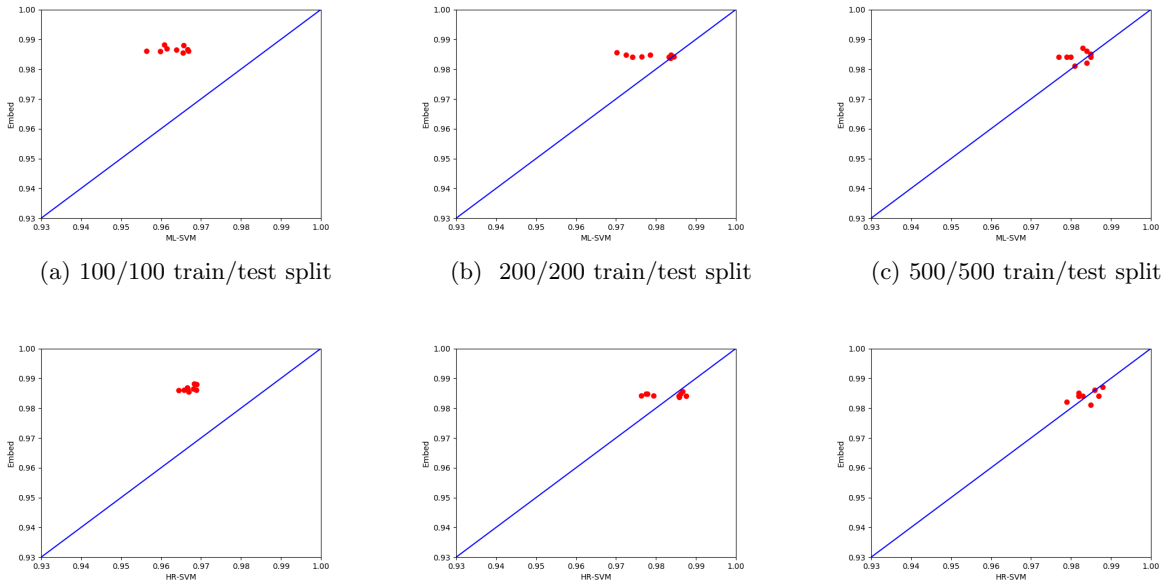


(a) 100/100 train/test split (b) 200/200 train/test split (c) 500/500 train/test split

(d) 100/100 train/test split (e) 200/200 train/test split (f) 500/500 train/test split

Figure 12: Test accuracy of ML-SVM vs Embed (top row) and HR-SVM vs Embed (bottom row) 10 runs on the Reuters dataset

APPENDIX (Continued)

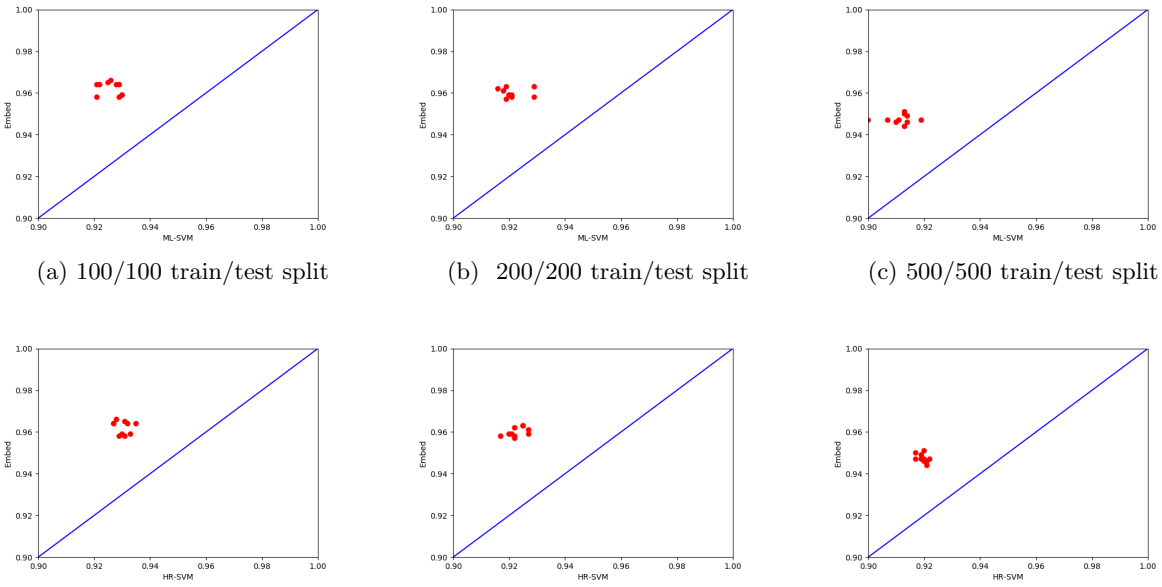


(a) 100/100 train/test split (b) 200/200 train/test split (c) 500/500 train/test split

(d) 100/100 train/test split (e) 200/200 train/test split (f) 500/500 train/test split

Figure 13: Test accuracy of ML-SVM vs Embed (top row) and HR-SVM vs Embed (bottom row) 10 runs on the WIPO dataset

APPENDIX (Continued)



(a) 100/100 train/test split (b) 200/200 train/test split (c) 500/500 train/test split

(d) 100/100 train/test split (e) 200/200 train/test split (f) 500/500 train/test split

Figure 14: Test accuracy of ML-SVM vs Embed (top row) and HR-SVM vs Embed (bottom row) 10 runs on the ENRON dataset

APPENDIX (Continued)

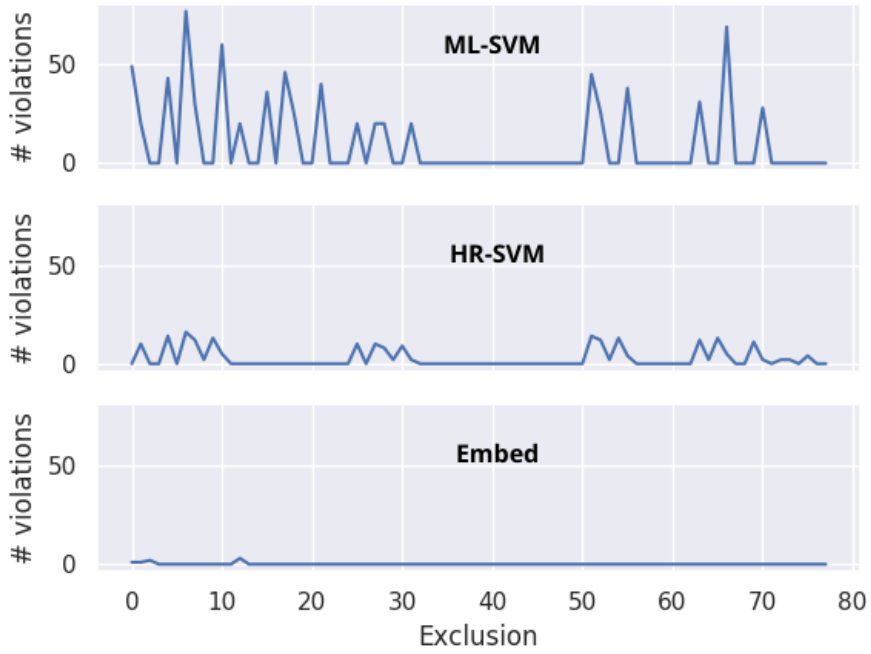


Figure 15: The number of violations for each exclusion constraint on the test set by (from top) ML-SVM, HR-SVM, and Embed on the Enron dataset with 200/200 train/test examples.