Inductive Two-layer Modeling with Parametric Bregman Transfer

Vignesh Ganapathiraman [1] **Zhan Shi** [1] **Xinhua Zhang** [1] **Yaoliang Yu** [2]

Abstract

Latent prediction models, exemplified by multi-layer networks, employ hidden variables that automate abstract feature discovery. They typically pose nonconvex optimization problems and effective semi-definite programming (SDP) relaxations have been developed to enable global solutions (Aslan et al., 2014). However, these models rely on nonparametric training of layer-wise kernel representations, and are therefore restricted to *transductive* learning which slows down test prediction. In this paper, we develop a new *inductive* learning framework for *parametric* transfer functions using *matching losses*. The result for ReLU utilizes completely positive matrices, and the inductive learner not only delivers superior accuracy but also offers an order of magnitude speedup over SDP with *constant* approximation guarantees.

1. Introduction

The past decade has witnessed advances of deep learning in a broad range of application areas such as game playing (Silver et al., 2016), natural language processing (Sutskever et al., 2014), image processing and computer vision (He et al., 2016). Its effectiveness is often attributed to the automated learning of latent representations, in that salient and discriminative features are highly beneficial for the overall learning task. With abstract and semantic features synthesized, the predictive relations between observations can be captured with more ease despite the possible complications in the correlation. In unsupervised learning, latent models have been widely used for clustering (Banerjee et al., 2005), dimensionality reduction (Lawrence, 2005), and transformation-invariant visual data analysis (Ranzato et al., 2012).

The focus of this paper is conditional modeling for supervised learning, where latent variables are learned in the context of output information, so that accurate reconstruction of outputs can be facilitated through predictive intervening features. Such features can characterize latent clusters (Tishby et al., 1999), sparse coding (Elad & Aharon, 2006), invariant representation (Rifai et al., 2011), amongst others.

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 nonconvex 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, Choromanska et al. (2014) and Dauphin et al. (2014) showed that the lowest critical values of the random loss function are close to the global minimum, and Kawaguchi (2016) 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 (Tian, 2017), quadratic activations (Soltanolkotabi et al., 2017), and one-hidden-layer non-overlapping convolution nets (Brutzkus & Globerson, 2017). The global minima in over-parameterized settings were characterized on deep and wide nets and convolutional nets (Nguyen & Hein, 2017a;b). 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 (Bengio et al., 2005), spectral methods (Anandkumar et al., 2014; Zhong et al., 2017), kernel methods (Zhang et al., 2016; 2017), polynomial networks and sum-product networks (Livni et al., 2014; Gens & Domingos, 2012), and semidefinite relaxations (Fogel et al., 2015). 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.

[1] Department of Computer Science, Univ of Illinois at Chicago, USA [2] School of Computer Science, Univ of Waterloo, Canada. Correspondence to: Xinhua Zhang <zhangx@uic.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

A framework based on reformulation that accommodates more general latent variable structures was proposed by Aslan et al. (2013; 2014), where each pair of adjacent layers are conjoined through a prediction loss that favors nonlinear connections. A similar approach was designed by Carreira-Perpinnan & Wang (2014), which introduced "auxiliary coordinates" to allow deviation from layer-wise outputs with a penalty. In order to achieve a convex model, Aslan et al. (2013; 2014) further represent each layer's output as a kernel matrix, and the loss over adjacent kernels is relaxed in a jointly convex fashion, retaining nonlinear transformations that allow a rich set of salient latent features to be captured.

However, these models assume that all latent layers behave as a multi-label classifier, and the latent kernels are learned *nonparametrically*, *i.e.* there is no explicit *parametric* transfer function and nonlinear relations are introduced only through the loss functions between layers. This is more restrictive than state-of-the-art deep learners where the activation functions are parametric and continuously valued, with popular choices such as ReLU. As a result the model is restricted to a *transductive* setting, in that training examples are required to establish the data-dependent context of nonparametric kernel learning. This restriction significantly slows down predictions at test time, which is more important than the training cost.

Such a challenge in efficiency is exacerbated as the kernel-based learning leads to an expensive semi-definite programming (SDP), whose computational cost limited their experiments to only 200 examples.

The goal of this paper, therefore, is to develop an *inductive* and *efficient* learning strategy for two-layer conditional models with global optimality guarantees. This allows predictions to be made as efficiently as a feedforward neural network (FFNN) does, obviating retraining at test time. It is achieved by directly constructing a convex relaxation based on a parametric transfer function (e.g. ReLU) specified *a priori*. In particular, we first make a new observation that no inter-layer loss satisfying nonlinear recovery and grounding can be jointly convex (§2). However by using the matching loss, the non-convexity can be encapsulated entirely by a bilinear term, facilitating a convex relaxation for ReLU based on completely positive (CP) cones (§3). The result provides a direct initialization of FFNN for finer tuning, which yields, inductively, more accurate predictions than baseline training methods (§5).

Different from the SDP used by Aslan et al. (2013; 2014), our CP-based model allowed us to develop a new efficient algorithm using low-rank approximation, scaling up the size of solvable problems by an order of magnitude (§4). A new *constant* approximation guarantee is also proved.

2. Matching Loss for Transfer Functions

Two-layer neural networks are composed of two nonlinear conditional models. The latent layer is characterized by a nonlinear transfer function $\mathbf{f} : \mathbb{R}^h \to \mathbb{R}^h$, 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. We use regular lowercase letters for scalar, bold lowercase letters for vector, and capital letters for matrix. 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 $\mathbf{y} \in \mathbb{R}^m$ 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 $m = 1$, where $y \in \{-1, 1\}$ and $[z]_+ := \max\{0, z\}$. For multi-class 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: $\mathbb{R} \to \mathbb{R}$. ReLU uses $f_r(z) = [z]_+$, and 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 $f_s(z) = (1 + e^{-z})^{-1}$. These transfers are illustrated in Figure 1. Non-elementwise transfers are also available, *e.g.* the soft-max function with $\mathbf{f}(\mathbf{z}) = (e^{z_1}, \ldots, e^{z_h})' / \sum_{k=1}^h e^{z_k}$.

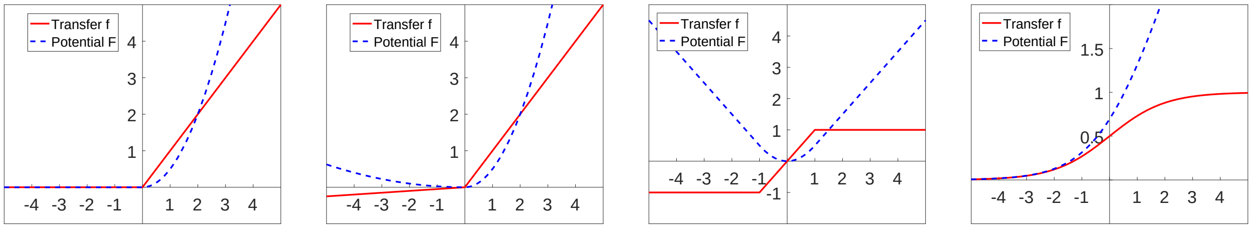
A major source of non-convexity in neural network is the nonlinear transfer function. To cope with it, a natural approach is to replace the exact connection of $\boldsymbol{\phi} = \mathbf{f}(\mathbf{z})$ by a loss function that penalizes the deviation between $\boldsymbol{\phi}$ and $\mathbf{f}(\mathbf{z})$. Formally, it attempts to construct 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 (see the proof in Appendix A):

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

This result motivates us to resort to weaker versions of loss. Interestingly, the matching loss (Auer et al., 1996) meets



(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

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 : $\mathbf{f} = \nabla F$, with $F : \mathbb{R}^h \to \mathbb{R}$. If $\mathbf{f}$ 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 the ReLU $f_r(z)$ is not strictly increasing in the negative hallf line, it can be approximated arbitrarily closely via $\max\{\epsilon z, z\}$ for infinitesimally small $\epsilon > 0$. Similar alterations 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. Under this assumption, we adopt the following loss function based on Bregman divergence:

$$L(\boldsymbol{\phi}, \mathbf{z}) = D_{F^*}(\boldsymbol{\phi}, \mathbf{f}(\mathbf{z})) = F^*(\boldsymbol{\phi}) + F(\mathbf{z}) - \boldsymbol{\phi}'\mathbf{z}, \quad (1)$$

where 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 nonconvex part is the bilinear term $\boldsymbol{\phi}'\mathbf{z}$, while both F^* and F are convex. Such a decoupling of nonconvex terms from the transfer functions is the key enabler for our convex reformulation.

3. Convex Two-layer Modeling

Suppose we have 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

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

where $\ell_j(U'\boldsymbol{\phi}_j + \mathbf{b}) := \ell(U'\boldsymbol{\phi}_j + \mathbf{b}, \mathbf{y}_j)$. We introduced regularizations via Frobenius norms. 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 $\operatorname{dom} \ell_j$ is the entire space. To keep our notation neat we write vector-input 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 compactly as

$$\min_{\Phi,W,U,\mathbf{b}} F^*(\Phi) - \operatorname{tr}(\Phi' W X) + F(WX) + \ell(U'\Phi + \mathbf{b}\mathbf{1}')$$
$$+ \tfrac{1}{2}\|W\|^2 + \tfrac{1}{2}\|U\|^2. \quad (3)$$

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, $F(WX) = \max_\Lambda\{\operatorname{tr}(\Lambda' W X) - F^*(\Lambda)\}$. So we can rewrite (2) into

$$\min_{W,U,\mathbf{b},\Phi} \max_{R,\Lambda} \; F^*(\Phi) - \operatorname{tr}(\Phi' W X) + \operatorname{tr}(\Lambda' W X)$$
$$- F^*(\Lambda) + \operatorname{tr}(R'(U'\Phi + \mathbf{b}\mathbf{1}')) - \ell^*(R) + \tfrac{\|W\|^2 + \|U\|^2}{2}$$
$$= \min_{\Phi} \max_{R,\Lambda} \min_{W,U,\mathbf{b}} \; F^*(\Phi) - \operatorname{tr}(\Phi' W X) + \operatorname{tr}(\Lambda' W X)$$
$$- F^*(\Lambda) + \operatorname{tr}(R'(U'\Phi + \mathbf{b}\mathbf{1}')) - \ell^*(R) + \tfrac{\|W\|^2 + \|U\|^2}{2}$$
$$= \min_{\Phi} \max_{R\mathbf{1}=\mathbf{0},\Lambda} \; F^*(\Phi) - \tfrac{1}{2}\|(\Phi - \Lambda)X'\|^2 - \tfrac{1}{2}\|\Phi R'\|^2$$
$$- F^*(\Lambda) - \ell^*(R). \quad (4)$$

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 squared Frobenius norm (see the proof in Appendix A):

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*

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

3.1. Convex relaxation

We now derive a convex relaxation for (4). To be concrete, consider the ReLU transfer with $F_r(Z) = \frac{1}{2}\|[Z]_+\|^2$. Its Fenchel dual is $F_r^*(\Phi) = \frac{1}{2}\|\Phi\|^2$ for $\Phi \geq \mathbf{0}$ (elementwise), and $+\infty$ otherwise. Therefore (4) can be specialized into

$$\min_{\Phi \geq \mathbf{0}} \max_{R\mathbf{1}=\mathbf{0}, \Lambda \geq \mathbf{0}} \tfrac{1}{2}\|\Phi\|^2 - \tfrac{1}{2}\|(\Phi - \Lambda)X'\|^2 - \tfrac{1}{2}\|\Phi R'\|^2 - \tfrac{1}{2}\|\Lambda\|^2 - \ell^*(R). \quad (5)$$

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 (5) as

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

Although this is still not convex, all occurrences of Φ are now in the form of $\Phi'\Phi$, leading 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

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

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 nonconvexity 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 (Berman & Shaked-Monderer, 2003). 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. Letting $\mathcal{T} := \mathcal{T}_\infty$ yields our final objective

$$\min_{T \in \mathcal{T}} \max_{R\mathbf{1}=\mathbf{0}, A \geq \mathbf{0}} \tfrac{1}{2}\operatorname{tr}(T) - \tfrac{1}{2}\operatorname{tr}(T(I - A)X'X(I - A')) - \tfrac{1}{2}\operatorname{tr}(TR'R) - \tfrac{1}{2}\operatorname{tr}(TAA') - \ell^*(R). \quad (6)$$

It turns out that the convex relaxation does not require prespecifying the number of hidden nodes; h can be figured out automatically through the rank of the optimal T. We will see 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.

Generality of the convexification scheme. We note in passing that the above technique is general, and can be extended beyond ReLU. For example, when using the hard tanh transfer, we have $F_h^*(\Phi) = \frac{1}{2}\|\Phi\|^2$ if the L_∞ norm $\|\Phi\|_\infty := \max_{ij}|\Phi_{ij}| \leq 1$, and ∞ otherwise. Then we get the same objective function as in (6), only with $\mathcal{T}_h$ changed into $\{\Phi'\Phi : \|\Phi\|_\infty \leq 1\}$ and the domain of A changed into $\{A : \sum_i |A_{ij}| \leq 1, \forall j\}$.

Even more general extensions to *non-elementwise* transfer functions can also be developed in our framework. The details on convexifying the *soft-max* transfer (and hard tanh) are deferred to Appendix B, and the space saved will be devoted to the more important issue of efficiently optimizing the model, hence overcoming the key bottleneck that has much confined the applicability of (Aslan et al., 2014).

4. Optimization

Although the problem (6) 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 (Berman & Shaked-Monderer, 2003), and even projection to $\mathcal{T}$ is NP-hard (Dickinson & Gijben, 2014).[1] Therefore we resort to conditional gradient (Frank-Wolfe) methods that are free of projection (CG, Jaggi, 2013; Harchaoui et al., 2015). 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 (Freund & Grigas, 2016), 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 (Lacoste-Julien & Jaggi, 2015). This is partly because at each step only the weights on the existing bases are optimized, while the bases themselves are not. To alleviate this problem, Zhang et al. (2012) 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 (6) with a multiplicative approximation bound of $\frac{1}{4}$.

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

Since GCG operates on gauge regularized objectives, our first step is to take a nontrivial path of rewriting (6). 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 (6) 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 $\operatorname{tr}(T)$ since $T \succeq \mathbf{0}$:

$$\begin{aligned}\mathcal{S} &:= \mathcal{T} \cap \{T : \operatorname{tr}(T) \leq 1\} &(7)\\ &= \operatorname{conv}\mathcal{T}_1 \cap \{T : \operatorname{tr}(T) \leq 1\} &(8)\\ &= \operatorname{conv}\left\{\mathbf{x}\mathbf{x}' : \mathbf{x} \in \mathbb{R}^t_+, \|\mathbf{x}\| \leq 1\right\}. &(9)\end{aligned}$$

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*

$$\gamma_{\mathcal{S}}(T) = \begin{cases} \operatorname{tr}(T) & T \in \mathcal{T} \\ +\infty & \textit{otherwise} \end{cases}. \quad (10)$$

The proof is relegated to Appendix A. In fact, it is easy to show that for any convex cone C, the gauge function of its intersection with a half-space $\operatorname{tr}(A'T) \leq 1$ is exactly $\operatorname{tr}(A'T)$ over C. The significance of Lemma 1 is that it provides the cornerstone for solving the problem (6) by GCG. Indeed, (6) can be equivalently rewritten as

$$\min_T \; J(T) := \tfrac{1}{2}\gamma_{\mathcal{S}}(T) + g(T) \quad \text{where} \quad (11)$$

$$\begin{aligned} g(T) := \max_{R\mathbf{1}=\mathbf{0}, A \geq \mathbf{0}} & -\tfrac{1}{2}\operatorname{tr}(T(I-A)X'X(I-A')) \quad (12)\\ & - \tfrac{1}{2}\operatorname{tr}(TR'R) - \tfrac{1}{2}\operatorname{tr}(TAA') - \ell^*(R). \end{aligned}$$

This objective finally falls into the framework of GCG sketched in Algorithm 1 (Zhang et al., 2012; Harchaoui et al., 2015). 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 our problem.

4.1. 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 (9):

Algorithm 1: General GCG algorithm

```
1 Randomly sample $\Phi_1 \in [0,1]^t$, and set $T_1 = \Phi_1'\Phi_1$.
2 while $k = 1, 2, \ldots$ do
3     Find $\nabla g(T_k)$ with $T_k = \Phi_k'\Phi_k$ by solving the inner
        maximization problem in $g(T_k)$ of (12).
4     Polar operator: find a new basis via
        $T^{\text{new}} = \arg\max_{T \in \mathcal{S}} \langle T, -\nabla g(T_k)\rangle$.
5     Compute the optimal combination weight
        $(\alpha, \beta) := \arg\min_{\alpha \geq 0, \beta \geq 0} J(\alpha T_k + \beta T^{\text{new}})$.
6     Locally optimize $T$: $\Phi_{k+1} = \arg\min_{\Phi \geq \mathbf{0}} J(\Phi'\Phi)$
        with $\Phi$ initialized by the value corresponding to
        $\Phi'\Phi = \alpha T_k + \beta T^{\text{new}}$ (see Section 4.1).
7 Return $T_{k+1}$
```

$$\max_{T \in \mathcal{S}} \operatorname{tr}(G'T) \quad \Longleftrightarrow \quad \max_{\mathbf{x} \in \mathbb{R}^t_+,\ \|\mathbf{x}\| \leq 1} \operatorname{tr}(\mathbf{x}'G\mathbf{x}). \quad (13)$$

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 (Murty & Kabadi, 1987). Usually problems like (13) are approached by semi-definite relaxations (SDP), and Nemirovski et al. (1999) 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 (13) with a computational procedure that is much more efficient than SDP. Furthermore, our problem does satisfy $G \succeq \mathbf{0}$.

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

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

Theorem 3. *Assume $G \succeq \mathbf{0}$. Then a $\frac{1}{4}$-approximate solution to* (13) *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 (13) 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:

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

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 PSD, it turns out very hard to extend this technique while retaining a constant bound. However the SDP-based technique in (Nemirovski et al., 1999) 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 (7) 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 fulfilled by our particular problem (11). Notice the gradient of g is simply

$$-\tfrac{1}{2}(I-A)X'X(I-A') - \tfrac{1}{2}R'R - \tfrac{1}{2}AA', \qquad (14)$$

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

Optimality of GCG and rates of convergence We finally translate the bound on the polar operator to that of the original objective (11). As shown by Theorem 1 of (Cheng et al., 2016), any α-approximate polar operator allows GCG to converge to an α-approximate solution to the original problem, and the convergence rate is $O(1/\epsilon)$. Hence we are guaranteed to find a $\frac{1}{4}$-approximate solution to (11). The overall method is summarized in Algorithm 2.

4.2. Accelerating local optimization by converting min-max into min-min

The computational bottleneck of applying GCG to our problem (11) 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 (12) employs a *nested maximization*, we are now faced with a *min-max* problem. Different from *min-min* optimizations $\min_x \min_y f(x,y)$ which can be solved very efficiently by alternating between optimizing x and y, a min-max problem like $\min_x \max_y f(x,y)$ 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.

Algorithm 2: Solve (6) for T by the GCG algorithm

```
1  Randomly sample $\Phi_1 \in [0,1]^t$, and set $T_1 = \Phi_1'\Phi_1$.
2  while $k = 1, 2, \ldots$ do
3      if $k = 1$ then
4          $(U_k, \mathbf{b}_k)$ = optimal $U$ and $\mathbf{b}$ in (15) for $\Phi_1$.
5          $M_k$ = optimal $M$ in (15) for $\Phi_1$.
6      Recover the optimal $R$: $R_k = \nabla\ell(U_k'\Phi_k + \mathbf{b}_k\mathbf{1}')$.
7      Recover the optimal $A$ by (17).
8      Compute the gradient $G_k$ of $g_\mu$ at $T_k = \Phi_k'\Phi_k$ via
         (14), with $R$ and $A$ served by $R_k$ and $A_k$, resp
9      Compute a new basis $\mathbf{x}_k$ by approximately solving
         $\arg\max_{\mathbf{x}\in\mathbb{R}_+^t, \|\mathbf{x}\|\leq 1} \mathbf{x}'(-G_k)\mathbf{x}$ (c.f. Theorem 3).
10     Line search:
         $(\alpha, \beta) := \arg\min_{\alpha\geq 0, \beta\geq 0} J(\alpha T_k + \beta\mathbf{x}_k\mathbf{x}_k')$.
11     Set $\Phi_{\text{tmp}} = (\sqrt{\alpha}\Phi_k', \sqrt{\beta}\mathbf{x}_k)'$.
12     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.
13 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 the change is small
    $(U, \mathbf{b}) = \arg\min_{U,\mathbf{b}}\{\ell(U'\Phi + \mathbf{b}\mathbf{1}') + \frac{1}{2}\|U\|^2\}$.
    $M = \arg\min_{M\geq\mathbf{0}} h(M, \Phi)$.
    $\Phi = \arg\min_{\Phi\geq\mathbf{0}}\{\ell(U'\Phi + \mathbf{b}\mathbf{1}') + h(M, \Phi)\}$.
Return $(\Phi, U, \mathbf{b}, M)$.
```

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:

$$\begin{aligned}
g(\Phi'\Phi) &= \max_{R\mathbf{1}=\mathbf{0}}\left\{-\tfrac{1}{2}\|\Phi R'\|^2 - \ell^*(R)\right\} \\
&\quad + \max_{A\geq\mathbf{0}}\left\{-\tfrac{1}{2}\|\Phi(I-A)X'\|^2 - \tfrac{1}{2}\|\Phi A\|^2\right\} \\
&= \max_R \min_{\mathbf{b}}\left\{\mathbf{b}'R\mathbf{1} - \ell^*(R) - \max_U - \operatorname{tr}(U'\Phi R') - \tfrac{\|U\|^2}{2}\right\} \\
&\quad + \max_A \min_{M\geq\mathbf{0}}\left\{-\tfrac{\|\Phi(I-A)X'\|^2}{2} - \tfrac{\|\Phi A\|^2}{2} + \operatorname{tr}(M'A)\right\} \\
&= \min_{U,\mathbf{b}}\left\{\ell(U'\Phi + \mathbf{b}\mathbf{1}') + \tfrac{1}{2}\|U\|^2\right\} + \min_{M\geq\mathbf{0}} h(M,\Phi), \qquad (15)
\end{aligned}$$

$$\text{where} \quad h(M,\Phi) := \max_A\Big\{-\tfrac{1}{2}\|\Phi(I-A)X'\|^2 - \tfrac{1}{2}\|\Phi A\|^2 + \operatorname{tr}(M'A)\Big\}. \qquad (16)$$

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 (Woodbury, 1950).

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 that 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 (16), akin to the standard smoothing technique (Nesterov, 2005):

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

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 (12), which we will denote as g_μ. Then the optimal A can be found by setting the gradient to zero:

$$A = (\Phi'\Phi + \mu I)^{-1}(M + \Phi'\Phi X'X)(X'X + I)^{-1}. \quad (17)$$

To efficiently compute A, we apply the Woodbury formula:

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

Computational complexity 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 line can be computed in $O(ht^2)$ time as we can precompute $(X'X + I)^{-1}$. 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.

Overall, the complexity is $\frac{1}{\epsilon} \cdot nt^2$ multiplied with: i) #round of alternating in Algorithm 3, and ii) #iteration of LBFGS in steps 4-6. In practice, with warm start these two numbers are about 10 before the relative change becomes small.

5. 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 Aslan et al. (2014); LOCAL: a model obtained by alternative minimization of the two-layer objective (3); and CVX-TR: our model learned transductively (see below). SVM was not included since it was already shown inferior to Ker-CVX by Aslan et al. (2014).

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 (3) with fixed $(\Phi, U, \mathbf{b})$, and it is a simple convex problem. With this initialization, we finely tuned all parameters by backpropagation.

Transductive learning. As Ker-CVX is transductive, we also considered the following transductive variant of CVX-IN. The objective (11) 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 $\ell(U'\Phi + \mathbf{b1}')$ 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 (16). 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.

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 (Lichman, 2013), CIFAR-SM, a binary classification dataset from (Aslan et al., 2013) based on CIFAR-100 (Krizhevsky & Hinton, 2009), and G241N (Chapelle).

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 1 and 2 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	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 1. 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 2. Mean test error for 200 training and 200 test examples

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.

Comparison on larger datasets. Thanks to the fast local optimization enabled by the new min-min alternating (§4.2), our model enjoys significant speedup compared with Aslan et al. (2013; 2014). To demonstrate this, we applied CVX-IN to Letter, XOR, and CIFAR-10 (Krizhevsky & Hinton, 2009) 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 preprocessing are available in Appendix C.

As Table 3 and 4 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 5, 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 §4.2.

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 2 demonstrates the original features in the input data X_{train} and the learned intermediate representation Φ_{train}, 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.

	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

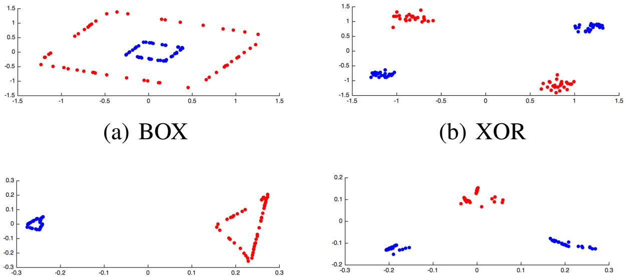
Table 3. Mean test error for 1000 training and 1000 test examples

	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 4. 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 5. Training times (in minutes) for CVX-IN on 100, 200, 1000, and 2000 training examples



(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.

6. Conclusions and Future Work

We developed a convex relaxation for parametric transfer functions such as ReLU based on matching loss. An efficient optimization method was designed with a constant approximation bound. For future work we will explore other transfer functions and their influence. To the best of our knowledge, no nontrivial recovery properties are known about *nonlinear* CP or SDP relaxation. Although our empirical results demonstrate compelling promise, it will be interesting to rigorously establish its theoretical guarantees.

References

Anandkumar, Animashree, Ge, Rong, Hsu, Daniel, Kakade, Sham M., and Telgarsky, Matus. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15:2773–2832, 2014.

Aslan, Ozlem, Cheng, Hao, Zhang, Xinhua, and Schuurmans, Dale. Convex two-layer modeling. In *Neural Information Processing Systems*, 2013.

Aslan, Ozlem, Zhang, Xinhua, and Schuurmans, Dale. Convex deep learning via normalized kernels. In *Neural Information Processing Systems*, 2014.

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.

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

Bengio, Yoshua, Roux, Nicolas Le, Vincent, Pascal, Delalleau, Olivier, and Marcotte, Patrice. Convex neural networks. In *Neural Information Processing Systems*, 2005.

Berman, Abraham and Shaked-Monderer, Naomi. *Completely Positive Matrices*. World Scientific, 2003.

Brutzkus, Alon and Globerson, Amir. Globally optimal gradient descent for a ConvNet with gaussian inputs. In *Proc. Intl. Conf. Machine Learning*, 2017.

Carreira-Perpinnan, M.A. and Wang, Weiran. Distributed optimization of deeply nested systems. In *Proc. Intl. Conference on Artificial Intelligence and Statistics*, 2014.

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

Cheng, Hao, Yu, Yaoliang, Zhang, Xinhua, Xing, Eric, and Schuurmans, Dale. Scalable and sound low-rank tensor learning. In *Proc. Intl. Conference on Artificial Intelligence and Statistics*, 2016.

Choromanska, Anna, Henaff, Mikael, Mathieu, Michael, Arous, Gerard Ben, and LeCun, Yann. The loss surfaces of multilayer networks. In *Proc. Intl. Conference on Artificial Intelligence and Statistics*, 2014.

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. In *Neural Information Processing Systems*, 2014.

Dickinson, Peter J. C. and Gijben, Luuk. 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. ISSN 1573-2894. doi: 10.1007/s10589-013-9594-z. URL `https://doi.org/10.1007/s10589-013-9594-z`.

Elad, M. and Aharon, M. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, 2006.

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.

Freund, Robert and Grigas, Paul. New analysis and results for the Frank-Wolfe method. *Mathematical Programming*, 155(1):199–230, 2016.

Gens, Robert and Domingos, Pedro. Discriminative learning of sum-product networks. In *Neural Information Processing Systems*, 2012.

Harchaoui, Zaid, Juditsky, Anatoli, and Nemirovski, Arkadi. Conditional gradient algorithms for norm-regularized smooth convex optimization. *Mathematical Programming*, 152:75–112, 2015.

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)*, pp. 770–778, 2016.

Jaggi, Martin. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proc. Intl. Conf. Machine Learning*, 2013.

Kawaguchi, Kenji. Deep learning without poor local minima. In *Neural Information Processing Systems*, 2016.

Krizhevsky, Alex and Hinton, Geoffrey. Learning multiple layers of features from tiny images. 2009.

Lacoste-Julien, S. and Jaggi, M. On the global linear convergence of Frank-Wolfe optimization variants. In *Neural Information Processing Systems*, 2015.

Lawrence, Neil. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *J. Mach. Learn. Res.*, 6:1783–1816, 2005.

Lichman, M. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

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

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

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.

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

Nguyen, Quynh and Hein, Matthias. The loss surface of deep and wide neural networks. In *Proc. Intl. Conf. Machine Learning*, 2017a.

Nguyen, Quynh and Hein, Matthias. The loss surface and expressivity of deep convolutional neural networks. arXiv:1710.10928, 2017b.

Ranzato, Marc'aurelio, Monga, Rajat, Devin, Matthieu, Chen, Kai, Corrado, Greg, Dean, Jeff, Le, Quoc V., and Ng, Andrew Y. Building high-level features using large scale unsupervised learning. In *Proc. Intl. Conf. Machine Learning*, 2012.

Rifai, Salah, Vincent, Pascal, Muller, Xavier, Glorot, Xavier, and Bengio, Yoshua. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proc. Intl. Conf. Machine Learning*, 2011.

Silver, David, Huang, Aja, Maddison, Chris J., Guez, Arthur, Sifre, Laurent, van den Driessche, George, Schrittwieser, Julian, Antonoglou, Ioannis, Panneershelvam, Veda, Lanctot, Marc, Dieleman, Sander, Grewe, Dominik, Nham, John, Kalchbrenner, Nal, Sutskever, Ilya, Lillicrap, Timothy, Leach, Madeleine, Kavukcuoglu, Koray, Graepel, Thore, and Hassabis, Demis. Mastering the game of go with deep neural networks and tree search. *Science*, 529:484–489, 2016.

Soltanolkotabi, Mahdi, Javanmard, Adel, and Lee, Jason D. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. In *Proc. Intl. Conf. Machine Learning*, 2017.

Steinberg, Daureen. *Computation of matrix norms with applications to Robust Optimization*. PhD thesis, Faculty of Industrial Engineering and Management, Technion, 2005.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. In *Neural Information Processing Systems*, 2014.

Tian, Yuandong. An analytical formula of population gradient for two-layered ReLU network and its applications in convergence and critical point analysis. In *Proc. Intl. Conf. Machine Learning*, 2017.

Tishby, N., Pereira, F., and Bialek, W. The information bottleneck method. In *37-th Annual Allerton Conference on Communication, Control and Computing*, 1999.

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

Woodbury, Max A. Inverting modified matrices. Technical Report MR38136, Memorandum Rept. 42, Statistical Research Group, Princeton University, Princeton, NJ, 1950.

Zhang, X., Yu, Y., and Schuurmans, D. Accelerated training for matrix-norm regularization: A boosting approach. In *Neural Information Processing Systems*, 2012.

Zhang, Y., Lee, JD., and Jordan, MI. L1-regularized neural networks are improperly learnable in polynomial time. In *Proc. Intl. Conf. Machine Learning*, 2016.

Zhang, Yuchen, Liang, Percy, and Wainwright, Martin. Convexified convolutional neural networks. In *Proc. Intl. Conf. Machine Learning*, 2017.

Zhong, Kai, Song, Zhao, Jain, Prateek, Bartlett, Peter, and Dhillon, Inderjit. Recovery guarantees for one-hidden-layer neural networks. In *Proc. Intl. Conf. Machine Learning*, 2017.

A. Proofs

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

$$(\text{LHS}) \quad \min_{\Phi} \max_{R} \mathcal{L}(\Phi, R) = \max_{R} \min_{\Phi} \mathcal{L}(\Phi, R) \quad (\text{RHS}).$$

Proof. Trivially LHS $\geq$ RHS (weak duality). Define $h(Z) := \min_\Phi F^*(\Phi) - \mathrm{tr}(\Phi' W X) + \ell(U'\Phi + \mathbf{b}\mathbf{1}' + Z)$. Since F^* is strongly convex, $h(Z) > -\infty$ for all Z. Since $\ell(U'\Phi + \mathbf{b}\mathbf{1}' + Z)$ 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

$$\begin{aligned} h(\mathbf{0}) \leq h(Z) - \mathrm{tr}(R'Z) &\leq F^*(\Phi) - \mathrm{tr}(\Phi' W X) + \ell(U'\Phi + \mathbf{b}\mathbf{1}' + Z) - \mathrm{tr}(R'Z) &(18)\\ &= \{F^*(\Phi) - \mathrm{tr}(\Phi' W X) + \mathrm{tr}(\Phi' U R)\} &(19)\\ &\quad + \{\ell(U'\Phi + \mathbf{b}\mathbf{1}' + Z) - \mathrm{tr}(R'(Z + U'\Phi + \mathbf{b}\mathbf{1}'))\} + \mathrm{tr}(R'\mathbf{b}\mathbf{1}'). &(20) \end{aligned}$$

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

$$h(\mathbf{0}) \leq -F(WX - UR) - \ell^*(R) + \mathrm{tr}(R'\mathbf{b}\mathbf{1}') \leq \max_R -F(WX - UR) - \ell^*(R) + \mathrm{tr}(R'\mathbf{b}\mathbf{1}').$$

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,

$$\gamma_{\mathcal{S}}(T) = \begin{cases} \mathrm{tr}(T) & T \in \mathcal{T} \\ +\infty & \text{otherwise} \end{cases}. \tag{21}$$

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.* $\{\mathbf{x} : f(\mathbf{x}) \leq 1\} = \{\mathbf{x} : g(\mathbf{x}) \leq 1\}$. 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 (7). □

B. Extensions to hard tanh and non-elementwise transfers

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 (6), 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

$$\max_{\Phi \in \mathbb{R}^{h \times t} : \|\Phi\|_\infty \leq 1} \mathrm{tr}(G'\Phi'\Phi) \quad = \quad h \max_{\boldsymbol{\phi} \in [0,1]^t} \boldsymbol{\phi}' G \boldsymbol{\phi} \quad = \quad h \max_{\boldsymbol{\phi} \in [0,1]^t} \|A\boldsymbol{\phi}\|^2, \quad \text{where} \quad A'A = G. \tag{22}$$

This problem is NP-hard, but an approximate solution with constant multiplicative guarantee can be found in $O(t^2)$ time (Steinberg, 2005). Note for computation we do not even need an expression of the convex hull of $\mathcal{T}_h$.

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

$$F(\mathbf{x}) = \log \sum_{k=1}^{h} e^{x_k}, \tag{23}$$

and its Fenchel dual is

$$F^*(\boldsymbol{\phi}) = \begin{cases} \sum\limits_{k=1}^{h} \phi_k \log \phi_k & \text{if } \boldsymbol{\phi} \in S^h \\ \infty & \text{otherwise} \end{cases}. \tag{24}$$

Therefore the objective in (4) can be instantiated into

$$\min_{\boldsymbol{\phi}_j \in \mathcal{S}^h} \max_{R\mathbf{1}=\mathbf{0}, \boldsymbol{\lambda}_j \in \mathcal{S}^h} \sum_{j=1}^{t} F^*(\boldsymbol{\phi}_j) - \frac{1}{2}\left\|(\Phi - \Lambda)X'\right\|^2 - \frac{1}{2}\left\|\Phi R'\right\|^2 - F^*(\Lambda) - \ell^*(R). \tag{25}$$

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.

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, (25) can be approximated by (setting $a = 1$ to ignore scaling)

$$\min_{\boldsymbol{\phi}_j \in \mathcal{S}^h} \max_{R\mathbf{1}=\mathbf{0}, \boldsymbol{\lambda}_j \in \mathcal{S}^h} \frac{1}{2}\left\|\Phi\right\|^2 - \frac{1}{2}\left\|(\Phi - \Lambda)X'\right\|^2 - \frac{1}{2}\left\|\Phi R'\right\|^2 - \frac{1}{2}\left\|\Lambda\right\|^2 - \ell^*(R). \tag{26}$$

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:

$$\min_{T \in \mathcal{T}_h} \max_{R\mathbf{1}=\mathbf{0}, A \geq \mathbf{0}, A'\mathbf{1}=\mathbf{1}} \frac{1}{2}\operatorname{tr}(T) - \frac{1}{2}\operatorname{tr}(T(I - A)X'X(I - A')) - \frac{1}{2}\operatorname{tr}(TR'R) - \frac{1}{2}\operatorname{tr}(TAA') - \ell^*(R), \tag{27}$$

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

$$\max_{\Phi \in \mathbb{R}^{h \times t}_+ : \Phi'\mathbf{1}=\mathbf{1}} \operatorname{tr}(G'\Phi'\Phi) = \max_{\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_h \in \mathbb{R}^t_+} \sum_{k=1}^{h} \left\|A\boldsymbol{\phi}_k\right\|^2 \quad s.t. \quad \sum_{k=1}^{h} \boldsymbol{\phi}_k = \mathbf{1}, \quad \text{where} \quad A'A = G. \tag{28}$$

This problem is NP-hard (Steinberg, 2005), 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

$$\max_{\boldsymbol{\phi}_1 \in [0,1]^t} \left\|A\boldsymbol{\phi}_1\right\|^2 + \left\|A(\mathbf{1} - \boldsymbol{\phi}_1)\right\|^2 = \max_{\boldsymbol{\phi}_1 \in [0,1]^t} \left\|A\left(\boldsymbol{\phi}_1 - \tfrac{1}{2}\mathbf{1}\right)\right\|^2 + \text{constant} \tag{29}$$

$$= \max_{\boldsymbol{\phi} \in [-\frac{1}{2}, \frac{1}{2}]^t} \left\|A\boldsymbol{\phi}\right\|^2 + \text{constant}. \tag{30}$$

This again admits an approximate solution with constant multiplicative guarantee that can be computed in $O(t^2)$ time (Steinberg, 2005).

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 $\Phi \in \mathbb{R}^{h \times t}_+$.

C. Dataset description

The experiments made use of 4 “real” world datasets - G241N (241×1500) from (Chapelle), Letter (vowel letters A-E vs non vowel letters B-F) (16×20000) from (UCI, 1990), CIFAR-SM (bicycle and motorcycle vs lawn- mower and tank) (256×1526) from (Aslan et al., 2013) and (Krizhevsky & Hinton, 2009) and CIFAR-10 (ship vs truck) (256×12000) from (Krizhevsky & Hinton, 2009), where red channel features are preprocessed by averaging pixels in both the CIFAR datasets.