Sparse Multi-Prototype Classification

Vikas K. Garg
CSAIL, MIT
vgarg@csail.mit.edu

Lin Xiao
Microsoft Research
lin.xiao@microsoft.com

Ofer Dekel
Microsoft Research
oferd@microsoft.com

Abstract

We introduce a new class of sparse multi-prototype classifiers, designed to combine the computational advantages of sparse predictors with the non-linear power of prototype-based classification techniques. This combination makes sparse multi-prototype models especially well-suited for resource constrained computational platforms, such as the IoT devices. We cast our supervised learning problem as a convex-concave saddle point problem and design a provably-fast algorithm to solve it. We complement our theoretical analysis with an empirical study that demonstrates the merits of our methodology.

1 INTRODUCTION

As modern machine-learned models become more accurate, they also tend to grow bigger and become more expensive to compute. Deep neural networks, massive decision tree ensembles, and other contemporary machine learning predictors can have hundreds of millions of parameters, resulting in a large memory footprint and a high computational cost. These models become especially prohibitive when the goal is to deploy them on resource impoverished platforms, such as wearable computers or IoT devices (Kumar *et al.* , 2017). Similarly, their high cost makes it difficult to build systems that need to keep track of many different models, such as those that maintain a separate model per user. Unsurprisingly, these issues have fostered a renewed interest in learning models that strike a better balance between accuracy and cost.

Efforts to develop machine learning techniques that produce more compact models can be broadly bifurcated into two schools of thought. The first approach is to train a large and accurate model topology, and subsequently compress it using an approximation method. Some of these approximation techniques include pruning (Han *et al.* , 2016; Nan *et al.* , 2016; Luo *et al.* , 2017), low-rank matrix approximation (Sainath *et al.* , 2013; Nakkiran *et al.* , 2015), hashing (Chen *et al.* , 2015), and parameter quantization or binarization (Hubara *et al.* , 2016; Han *et al.* , 2016). Another popular technique is to use a large model to generate training data for a smaller model (Bucila *et al.* , 2006).

The second approach incorporates compression more intimately into the training objective. For example, the well-known Lasso (Tibshirani, 1996) and Elastic-Net (Zou & Hastie, 2005) algorithms use a sparsity-inducing regularization term to control the sparsity of a linear model. The resulting sparse predictor relies only on a small subset of the available features, and is therefore economical to store and make predictions. The weakness of these approaches is that linear predictors are typically not expressive enough to achieve state-of-the-art accuracy. Another common idea is to define models that are specified by a small number of prototypes, for example, by learning a Support Vector Machine (SVM) with a small support set (Dekel & Singer, 2007; Dekel *et al.* , 2008), or by finding a compressed set of reference points for a Nearest Neighbor model (Kusner *et al.* , 2014; Zhong *et al.* , 2017; Gupta *et al.* , 2017). A main drawback of these techniques is that they typically require solving highly non-convex optimization problems, which makes it difficult to guarantee their convergence and optimality. Another shortcoming of many of these approaches is that the prototypes that they learn are typically dense.

In this paper, we subscribe to the second approach mentioned above, and address the cost-accuracy

tradeoff by designing the training objective appropriately. Specifically, we introduce a class of models that we call *Sparse Multi-Prototype* (SMP) classifiers. SMP classifiers attempt to combine the sparsity benefits of linear models with the non-linear power of multi-prototype methods. Namely, each class is associated with a small set of prototypes, and each of those prototypes is sparse. But for their sparsity, SMP classifiers are reminiscent of multiclass SVMs (Weston & Watkins, 1999; Crammer & Singer, 2001), and their multi-prototype extensions (Aiolli & Sperduti, 2005).

We formulate the training procedure for SMP classifiers as a convex optimization problem. Specifically, we cast the SMP training problem as a convex-concave saddle point optimization problem and show that this formulation admits fast convergence via a primal-dual proximal point algorithm due to Chambolle and Pock (Chambolle & Pock, 2011, 2016; He *et al.* , 2017; Zhang & Xiao, 2015; Yu *et al.* , 2017). On one hand, our formulation induces sparsity by incorporating a regularization term, similar to the ℓ_1 term used in Lasso and Elastic-Net. On the other hand, it controls the number of prototypes using another regularization term, similar to the one used to derive convex formulations of clustering (Hocking *et al.* , 2011) and regression (Feng *et al.* , 2012). Our optimization formulation and algorithm are different from the ones used in these papers.

The rest of the paper is organized as follows. We set up the problem in Section 2. We then show, in Section 3, how our problem can be posed as a saddle point problem that admits fast and provable convergence via the Chambolle-Pock procedure. We present the results of our experiments in Section 4.

2 PROBLEM FORMULATION

Let $\mathcal{Y}$ be a finite set of labels. Suppose that we are given a set of training examples $\{(x_i, y_i)\}_{i=1}^m$, where each $x_i \in \mathbb{R}^n$ and $y_i \in \mathcal{Y}$. Without loss of generality, we assume an ordering of the training examples: examples from first class precede those in second, examples from second precede those in third, and so on.

Our goal is to learn a classifier $c : \mathbb{R}^n \mapsto \mathcal{Y}$. Assume (without loss of generality) that c is defined by a set of scoring functions $\{\phi_y\}_{y\in\mathcal{Y}}$, where the value $\phi_y(x)$ is interpreted as the score of predicting the label y for the instance x. Using these scoring functions, our classifier takes the form

$$c(x) \;=\; \arg\max_{y\in\mathcal{Y}} \; \phi_y(x).$$

The classifier $c(x)$ correctly classifies the example (x, y) if and only if

$$\phi_y(x) - \max_{y'\neq y} \phi_{y'}(x) \;>\; 0. \tag{1}$$

We use this property to define the empirical loss

$$\sum_{i=1}^{m} \ell\left(\phi_{y_i}(x_i) - \max_{y\neq y_i} \phi_y(x)\right),$$

where ℓ is a convex monotonically non-increasing loss function that upper bounds the error indicator function (for instance, ℓ could be hinge-loss or log-loss). Clearly, this loss is an upper-bound on the number of multiclass classification mistakes.

If we use a linear score function for each class, i.e.,

$$\phi_y(x) = w_y \cdot x, \quad y \in \mathcal{Y},$$

where each $w_y \in \mathbb{R}^n$ is called a *class prototype*, and $\cdot$ denotes the inner product of two vectors, then we obtain the multi-class support vector machine (Weston & Watkins, 1999; Crammer & Singer, 2001).

In this paper, we allow multiple prototypes for each class (Aiolli & Sperduti, 2005). Suppose we have in total N prototypes $w_1, \ldots, w_N \in \mathbb{R}^n$, and let J_y be the set of the prototype indices associated with class y for each $y \in \mathcal{Y}$. We let the score function for class label y be

$$\phi_y(x) \;=\; \max_{j\in J_y} \; w_j \cdot x.$$

Since ℓ is monotonically non-increasing, we have

$$\begin{aligned}
& \ell\left(\phi_{y_i}(x_i) - \max_{y\neq y_i} \phi_y(x_i)\right) \\
= \;& \ell\left(\max_{j\in J_{y_i}} w_j \cdot x_i \;- \max_{j\notin J_{y_i}} w_j \cdot x_i\right) \\
\leq \;& \ell\left(w_{j(i)} \cdot x_i \;- \max_{j\notin J_{y_i}} w_j \cdot x_i\right) \\
= \;& \max_{j\notin J_{y_i}} \ell\big((w_{j(i)} - w_j) \cdot x_i\big),
\end{aligned}$$

where $j(i) \in J_{y_i}$ is any fixed assignment of prototype to the example (x_i, y_i). The last expression above is a convex function in all the prototypes $w_1, \ldots, w_N$, and so is the average loss function

$$\frac{1}{m}\sum_{i=1}^{m} \max_{j\notin J_{y_i}} \ell\big((w_{j(i)} - w_j) \cdot x_i\big). \tag{2}$$

Note that $j(i)$ needs to be fixed before we optimize over the prototypes, but is not required to maximize $w_j \cdot x_i$ over $j \in J_{y_i}$. This relaxation helps us to obtain a convex upper bound on the loss in the general case.

Setting $N = |\mathcal{Y}|$ and $|J_y| = 1$ recovers the loss for the multi-class SVM, and we have $j(i) = \arg\max_{j \in J_{y_i}} w_j \cdot x_i$. In the other extreme case, we can let $N = m$ and associate each training example (x_i, y_i) with a prototype w_i. In this case, we can have $\phi_{y_i}(x_i) = w_i \cdot x_i = \arg\max_{j \in J_{y_i}} w_j \cdot x_i$. However, this approach requires excessive amount of storage and computation, and also may cause significant overfitting.

In practice, we can cluster the training examples in each class into p groups, where p is much smaller than the number of examples in the class. Then we can have p prototypes for each class $y \in \mathcal{Y}$, and associate the examples in each cluster within the class with a common prototype: $j(i) = j(i')$ if $y_i = y_{i'}$, and x_i and $x_{i'}$ belong to the same cluster.

2.1 SMOOTHING THE LOSS

In order to leverage the fast algorithms designed for smooth convex optimization, we focus on smooth loss functions. In particular, we use the log-loss

$$\ell(\alpha) = \log(1 + \exp(-\alpha)).$$

Although this is a smooth function, the average loss function defined in (2) is non-smooth, due to the max operators in the sum. We can make the loss function smooth using the usual trick of softmax. Specifically, we can replace the function $u(z) = \max_j \ell(z_j)$ with

$$\tilde{u}(z) = \log\left(1 + \textstyle\sum_j \exp(-z_j)\right). \tag{3}$$

As a result, the smoothed loss function is

$$f(W) = \frac{1}{m}\sum_{i=1}^{m} \log\left(1 + \sum_{j \notin J_{y_i}} \exp\left((w_j - w_{j(i)}) \cdot x_i\right)\right), \tag{4}$$

where $W \in \mathbb{R}^{N \times n}$ is a matrix formed by stacking the vectors $w_1^T, \ldots, w_N^T$ as its rows.

2.2 ENFORCING GROUP SPARSITY

Instead of relying on a separate clustering stage to reduce the number of prototypes, we can use a more principled approach based on convex optimization. Suppose we start with a large number of prototypes, for example, by having a separate prototype for each training example. While minimizing the average loss function, we may add the regularization term

$$\sum_{y \in \mathcal{Y}} \sum_{\substack{j > i \\ i,j \in J_y}} \|w_i - w_j\|_\infty, \tag{5}$$

which encourages some of the prototypes in each class to merge, forming a smaller set of distinct prototypes.

We introduce some notations to simplify our presentation. Let $W_y \in \mathbb{R}^{|J_y| \times n}$ be the matrix formed by stacking the set of prototypes $\{w_j^T : j \in J_y\}$ as its rows. For each class $y \in \mathcal{Y}$, we form a $b_y \times |J_y|$ matrix B_y, where $b_y = \binom{|J_y|}{2}$. Specifically, each row of B_y corresponds to a pair (i, j) such that $i < j$ and $i, j \in J_y$, with value 1 at index i, -1 at index j, and 0 elsewhere. Then the penalty function in (5) can be written as

$$\sum_{y \in \mathcal{Y}} \|B_y W_y\|_{\infty,1},$$

where the matrix norm $\|\cdot\|_{\infty,1}$ is defined as

$$||U||_{\infty,1} = \sum_i ||U_{i,\cdot}||_\infty = \sum_i \max_{r \in [n]} |U_{i,r}|.$$

Therefore, the regularized loss can be written as

$$f(W) + \lambda \sum_{y \in \mathcal{Y}} \|B_y W_y\|_{\infty,1},$$

where $\lambda > 0$ is a regularization parameter and f is the smoothed average loss defined in (4).

2.3 IMPOSING PROTOTYPE SPARSITY

In addition to the group sparsity aimed at having fewer prototypes, we can also induce sparsity in each prototype by adding the following regularization:

$$h_\eta(W_y) \triangleq \quad ||W_y||_{1,1} + \frac{\eta}{2}||W_y||_F^2,$$

where $\|\cdot\|_F$ denotes the matrix Frobenius norm and

$$\|U\|_{1,1} = \sum_i ||U_{i,\cdot}||_1 = \sum_i \sum_{r \in [n]} |U_{i,r}|.$$

In other words, h_η is an elastic-net type of regularization, where η is a parameter to trade off between the ℓ_1 and ℓ_2 regularizations.

In summary, we would like to solve the following sparse multi-prototype (SMP) classification problem:

$$\min_W \left\{ f(W) + \lambda \sum_{y \in \mathcal{Y}} \|B_y W_y\|_{\infty,1} + \mu \sum_{y \in \mathcal{Y}} h_\eta(W_y) \right\}, \tag{6}$$

Algorithm 1 The Chambolle-Pock (CP) Algorithm

input: parameters τ, σ, and initial point (w^0, v^0)
Set $\bar{w}^0 = w^0$
for $t = 0, 1, 2, \ldots$ **do**
$v^{t+1} = \text{prox}_{\sigma g}\left(v^t + \sigma K \bar{w}^t\right)$
$w^{t+1} = \text{prox}_{\tau h}\left(w^t - \tau\left(\nabla f(w^t) + K^T v^{t+1}\right)\right)$
$\bar{w}^{t+1} = 2w^{t+1} - w^t$

where $\lambda, \mu > 0$ are regularization hyperparameters. This is a convex optimization problem. However, due to the complex structure of the regularization terms, it is not clear how to solve this minimization problem directly in an efficient manner (e.g., how to compute the proximal mapping of the group sparsity regularization). In the next section, we tackle this problem using a primal-dual first-order algorithm.

3 PRIMAL-DUAL ALGORITHM

Chambolle & Pock (2011, 2016) developed a class of primal-dual first-order algorithms for solving the following form of convex-concave saddle-point problems with bilinear coupling:

$$\min_{w \in \mathbb{R}^d} \max_{v \in \mathbb{R}^{d'}} f(w) + h(w) + \langle Kw, v\rangle - g^\star(v), \quad (7)$$

where f is convex and differentiable, and both h and $g^\star$ are convex but may be non-differentiable. In particular, $g^\star$ can be considered as the conjugate function of some convex function g. Here K is a bilinear coupling matrix of dimension $d' \times d$. In addition, it is assumed that the proximal mappings of h and $g^\star$,

$$\text{prox}_h(w) = \arg\min_{u \in \mathbb{R}^d} \left\{ f(u) + \frac{1}{2}||u - w||_2^2 \right\},$$
$$\text{prox}_{g^\star}(v) = \arg\min_{z \in \mathbb{R}^{d'}} \left\{ g^\star(z) + \frac{1}{2}||z - v||_2^2 \right\},$$

can be computed efficiently. Intuitively, the proximal map $\text{prox}_h(w)$ looks for a point u that has a low cost $f(u)$ *and* is not too far from w.

Algorithm 1 shows the CP algorithm (Chambolle & Pock, 2016) for solving the convex-concave saddle-point problem (7). Suppose ∇f is Lipschitz continuous with Lipschitz constant L_f, i.e.,

$$\|\nabla f(u) - \nabla f(w)\|_2 \le L_f \|u - w\|_2, \quad \forall u, w \in \mathbb{R}^d,$$

and the spectral norm of K is bounded by L, i.e., $\|K\| \le L$. Chambolle & Pock (2016) showed that this algorithm enjoys an $O(1/t)$ convergence rate (the reduction of optimization error after t iterations) provided that the step size parameters σ and τ satisfy the condition

$$\left(\frac{1}{\tau} - L_f\right)\frac{1}{\sigma} \ge L^2. \quad (8)$$

In the rest of this section, we show how to transform the SMP problem (6) into the form of (7), and how to compute the relevant proximal mappings as well as choose the step sizes.

3.1 SADDLE-POINT FORMULATION

Let $g(U) = \|U\|_{\infty,1}$, and let $\langle U, V\rangle$ denote the inner product between the two matrices, i.e., $\langle U, V\rangle = \text{Tr}(U^T V)$. The conjugate function of g is defined as

$$g^\star(V) = \max_U \big\{\langle U, V\rangle - g(U)\big\} = \begin{cases} 0 & \text{if } \|V\|_{1,\infty} \le 1 \\ +\infty & \text{otherwise,} \end{cases}$$

where $\|V\|_{1,\infty} = \max_i \|V_{i,\cdot}\|_1 = \max_i \sum_j |V_{i,j}|$ is the dual norm of $\|\cdot\|_{\infty,1}$. We replace the group sparsity regularizations $g_y(B_y W_y) = \|B_y W_y\|_{\infty,1}$ in (6) by

$$\max_{V_y} \left\{\langle B_y W_y, V_y\rangle - g_y^\star(V_y)\right\},$$

which yields the convex-concave saddle-point problem

$$\min_W \max_V \Bigg\{ f(W) + \mu \sum_{y \in \mathcal{Y}} h_\eta(W_y) + \lambda \sum_{y \in \mathcal{Y}} \Big(\langle B_y W_y, V_y\rangle - g_y^\star(V_y) \Big) \Bigg\}. \quad (9)$$

Here the subscript y in g_y and $g_y^\star$ indicates that their arguments may have different dimensions; more specifically, $V_y \in \mathbb{R}^{b_y \times n}$ with $b_y = \binom{|J_y|}{2}$. With some delicate vectorization of the matrix variables, we can put the formulation from (9) in the exact form of (7).

Without loss of generality, let the multi-class labels be $\{1, 2, \ldots, |\mathcal{Y}|\}$. Denote by $\text{vec}(A)$ the column vector formed by stacking the columns of matrix A on top of one another. By a slight abuse of notation, we define

$$\text{vec}(W) \triangleq [\text{vec}(W_1^\top)^\top \, \text{vec}(W_2^\top)^\top \ldots \text{vec}(W_{|\mathcal{Y}|}^\top)]^\top.$$

Note that $\text{vec}(W) \in \mathbb{R}^{Nn}$, where N is the total number of prototypes. Likewise, we form $\text{vec}(V) \in \mathbb{R}^{bn}$, where $b \triangleq \sum_{y \in \mathcal{Y}} b_y$, by concatenating the vectorizations of $\{V_y\}$. Let I_d and 0_d be, respectively, the identity matrix and the zero matrix of order d. Let $\mathbb{1}_d$ be a d-dimensional vector with all coordinates set to 1. We use $A_1 \otimes A_2$ to denote the Kronecker

product of any two vectors or matrices A_1 and A_2. Finally, we represent the k^{th} standard basis in $\mathbb{R}^{|\mathcal{Y}|}$ by e_k, i.e., e_k has coordinate k set to 1 and all the others set to 0.

With the above notations, and letting $\tilde{w} = \text{vec}(W)$ and $\tilde{v} = \text{vec}(V)$, we can show that the saddle-point problem in (9) can be expressed as

$$\min_{\tilde{w}\in\mathbb{R}^{Nn}} \min_{\tilde{v}\in\mathbb{R}^{bn}} \tilde{f}(\tilde{w}) + \langle \tilde{B}\tilde{w}, \tilde{v}\rangle - \tilde{g}^\star(\tilde{v}),$$

with appropriate definitions of $\tilde{f}$, $\tilde{B}$ and $\tilde{g}^\star$. First, we have (with some tedious algebra)

$$\lambda \sum_{y\in\mathcal{Y}} \langle B_y W_y, V_y\rangle = \langle \tilde{B}\tilde{w}, \tilde{v}\rangle,$$

where

$$\tilde{B} \triangleq \left(\sum_{k=1}^{|\mathcal{Y}|} e_k e_k^\top \otimes B_k\right) \otimes \lambda I_n. \tag{10}$$

We define $\text{abs}(z) \triangleq [|z_1|, |z_2|, \ldots, |z_k|]$ for any vector $z \in \mathbb{R}^k$. We note that $\sum_{y\in\mathcal{Y}} \lambda g_y^\star(V_y)$ is finite (when it is 0) only if $g_y^\star(V_y) = 0$, i.e. only if $||V_y||_{1,\infty} \le 1$, for all $y \in \mathcal{Y}$. Moreover, for λ finite and positive, we have $\lambda g_y^\star(V_y) = g_y^\star(V_y)$. This lets us define

$$\tilde{g}^\star(\tilde{v}) \triangleq g^\star(C_g\ \text{abs}(\tilde{v})), \qquad \text{where}$$

$$C_g = \sum_{k=1}^{|\mathcal{Y}|} e_k e_k^\top \otimes \mathbb{1}_n^\top$$

is a block diagonal matrix with $|\mathcal{Y}|$ blocks each equal to $\mathbb{1}_n^\top$, and $g^\star(z) = 0$ if $z_k \in [-1, 1]$ for all $k \in \{1, \ldots, |\mathcal{Y}|\}$ and ∞ otherwise.

Finally, we can write the smoothed loss function $f(W)$ defined in (4) as

$$f(W) = \tilde{f}(\tilde{w}) = \frac{1}{m}\sum_{i=1}^{m} \tilde{u}(A_i\tilde{w}), \tag{11}$$

where $\tilde{u}$ is the soft-max function defined in (3), and

$$A_i = C_i(I_N \otimes x_i^\top).$$

Here C_i is a matrix with $(N - |I_{y_i}|)$ rows and N columns. Each row of C_i corresponds to some $j \notin J_{y_i}$, with its $j(i)$th coordinate being 1, jth coordinate being -1, and all the other coordinates being 0.

3.2 BOUNDS ON THE LIPSCHITZ CONSTANTS

In order to choose the step sizes σ and τ in the CP algorithm appropriately, we need to estimate the two parameters L_f and L that appeared in (8). First, we give an upper bound on L_f.

Proposition 1. *The Lipschitz constant L_f of $\nabla\tilde{f}(\tilde{w})$ defined in (11) is bounded as*

$$L_f \le NR^2,$$

where N is the total number of prototypes and R is an upper bound on $\|x_i\|$ for all $i = 1, \ldots, m$.

Proof. We derive the desired result by bounding the spectral norm of the Hessian matrix of $\tilde{f}$ defined in (11). It is sufficient to consider the Hessian of each $\tilde{u}(A_i\tilde{w})$, which can be written as

$$H_i = A_i^\top\, \nabla^2\tilde{u}(A_i\tilde{w})\, A_i.$$

It is not hard to check that $\nabla^2\tilde{u}(A_i\tilde{w}) \preceq I$, i.e., the matrix $I - \nabla^2\tilde{u}(A_i\tilde{w})$ is positive semidefinite for any $\tilde{w}$. Therefore we have

$$H_i \preceq A_i^\top A_i = (I_N \otimes x_i^\top)^\top C_i^\top C_i (I_N \otimes x_i^\top).$$

In terms of their spectral norm, we have

$$\|H_i\| \le \|C_i^\top C_i\|\ \|I_N \otimes x_i^\top\|^2. \tag{12}$$

If $\|x_i\|_2 \le R$, then we have $\|I_N \otimes x_i^\top\| \le R$.

It remains to bound $\|C_i^\top C_i\|$, which is the same as $\|C_i C_i^\top\|$. By construction of C_i at the end of Section 3.1, we have

$$C_i C_i^\top = I_{d_i} + \mathbb{1}_{d_i}\mathbb{1}_{d_i}^\top$$

where $d_i = N - |J_{y_i}|$. Therefore we have

$$\|C_i^\top C_i\| = \|C_i C_i^\top\| = N - |J_{y_i}| + 1 \le N.$$

Combining with (12), we conclude that $\|H_i\| \le NR^2$. Finally, since $\|\nabla^2\tilde{f}(\tilde{w})\| \le (1/m)\sum_{i=1}^m \|H_i\|$, we obtain the desired result. □

Next we derive the precise value of $L = \|\tilde{B}\|$.

Proposition 2. *The singular values of the matrix $\tilde{B}$ defined in (10) belong to the set*

$$\left\{\lambda\sqrt{|J_1|}, \ldots, \lambda\sqrt{|J_{|\mathcal{Y}|}|}, 0\right\}.$$

Therefore, the spectral norm of $\tilde{B}$ is

$$L = \max_{k=1,\ldots,|\mathcal{Y}|} \lambda\sqrt{|J_k|}.$$

Proof. We first claim that B_k has two distinct singular values, viz., $\sqrt{|J_k|}$ (with multiplicity $|J_k|$ - 1) and 0 (with multiplicity 1). We invoke a characterization of the singular values in terms of the eigen decomposition of positive semidefinite matrix $B_k^\top B_k \in \mathbb{R}^{|J_k|\times|J_k|}$ to argue for the full spectrum of B_k. Specifically, if η^2 is an eigenvalue of the matrix $B_k^\top B_k$, then η is a singular value of B_k.

Now note that $B_k^\top B_k$ has all the off-diagonal coordinates equal to -1 and all the diagonal coordinates equal to $|J_k|-1$. This is precisely the Laplacian of the complete graph with $|J_k|$ nodes, which is known to have the eigenvalue $|J_k|$ with multiplicity $|J_k|-1$ and 0 with multiplicity one. This completes the analysis for the spectrum of B_k.

Next, we note the term within the parentheses in the definition of $\tilde{B}$ is a block diagonal matrix with blocks $B_1,\ldots,B_{|\mathcal{Y}|}$, and therefore, the set of the singular values of this set is simply the union of the singular values of each block, i.e., the set $\{\sqrt{|J_1|},\ldots,\sqrt{|J_{|\mathcal{Y}|}|},0\}$. Finally, the distinct singular values of $\tilde{B}$ are $\{\lambda\sqrt{|J_1|},\ldots,\lambda\sqrt{|J_{|\mathcal{Y}|}|},0\}$, since λ is the unique singular value (multiplicity n) of the matrix λI_n. This follows since every singular value μ_{12} of $A_1\otimes A_2$ can be expressed as the product of singular values μ_1 of A_1 and μ_2 of A_2. □

We can rewrite the condition in (8) as

$$\tau \le \frac{1}{L_f}, \qquad \frac{\sigma\tau}{1-\tau L_f}L^2 \le 1.$$

Given the bounds for L_f and L derived in Proposition 1 and Proposition 2, we can choose the step sizes τ and σ to satisfy the conditions above, which lead to $O(1/t)$ convergence of the CP algorithm. For example, if we start with p prototypes for each class, then $|J_k| = p$ for $k \in \{1,\ldots,|\mathcal{Y}|\}$ and $N = p|\mathcal{Y}|$. Therefore we have $L_f \le p|\mathcal{Y}|R^2$ and $L^2 = \lambda^2 p$, and can choose

$$\tau = \frac{1}{2pR^2|\mathcal{Y}|}, \quad \sigma \le \frac{1}{2\lambda^2 p}\frac{1}{\tau}.$$

We note that the bound on L_f can be very loose. Some trial-and-error for tuning the step sizes is expected in practice.

3.3 COMPUTING PROXIMAL MAPS

We denote the sign of a real number p by $\text{sign}(p) \in \{0,\pm 1\}$, and the positive part $\max(0,p)$ by $(p)_+$. For clarity of presentation, in this section, we use $A(i,j)$ to denote the entry of the matrix A at the ith row and jth column. Our next result shows that the Chambolle-Pock (CP) update for each W_y, $y \in \mathcal{Y}$, can be computed via a closed form expression.

Proposition 3. *The CP update rule for W_y is*

$$W_y^{t+1}(i,j) = \frac{\text{sign}(U_y^t(i,j))}{\mu\tau\eta+1}\ \left(|U_y^t(i,j)| - \mu\tau\right)_+,$$

where $U_y^t \triangleq W_y^t - \tau\left(\nabla_y f(W^t) + \lambda B_y^\top V_y^{t+1}\right)$.

Proof. In the CP algorithm, the matrix W_y is updated as

$$W_y^{t+1} = \text{prox}_{\mu\tau h_\eta}\left(W_y^t - \tau(\nabla_y f(W^t) + \lambda B_y^\top V_y^{t+1})\right),$$

where $\nabla_y f(W^t)$ denotes the partial gradient of f with respect to W_y at the previous iterate W^t. We can equivalently write the above proximal mapping as

$$W_y^{t+1} = \underset{Z}{\text{argmin}}\left\{h_\eta(Z) + \frac{1}{2\mu\tau}||Z-U_y^t||_F^2\right\}. \quad (13)$$

Since W_y^{t+1} is optimal for the above minimization problem, the (sub-)gradient of the corresponding objective function must vanish. Recall that

$$h_\eta(W_y) = \|W_y\|_{1,1} + (\eta/2)\|W_y\|_F^2.$$

The optimality condition for (13) means that there exists $Z(i,j) \in \partial|W_y^{t+1}(i,j)|$ such that

$$Z(i,j)+\eta W_y^{t+1}(i,j)+\frac{1}{\mu\tau}(W_y^{t+1}(i,j)-U_y^t(i,j)) = 0.$$

When $W_y^{t+1}(i,j) = 0$, we have $\partial|W_y^{t+1}(i,j)| \in [-1,1]$, which implies $|U^t(i,j)| \le \mu\tau$. Otherwise, we have $\partial Z^*(i,j) = \text{sign}(Z^*(i,j))$, whence

$$|U^t(i,j)| > \mu\tau \text{ and } \text{sign}(Z^*(i,j)) = \text{sign}(U^t(i,j)).$$

So, we can perform soft thresholding to obtain Z^*, i.e. W_y^{t+1}, for all cases, and it turns out as claimed. □

We next derive an expression for updating V_y.

Proposition 4. *The CP update rule for V_y is*

$$V_y^{t+1} = \arg\min_{||Z||_{1,\infty}\le 1}\frac{1}{2}\left\|Z-(V_y^t+\lambda\sigma_t B_y\bar{W}_y^t)\right\|_F^2.$$

Proof. Since the empirical loss term and the sparsity term depend only on W, the update rule for V_y is

$$V_y^{t+1} = \arg\max_Z\left\{\lambda(\langle B_y\bar{W}_y^t, Z\rangle - g_y^\star(Z)) - \frac{1}{2\sigma}||Z-V_y^t||_F^2\right\}.$$

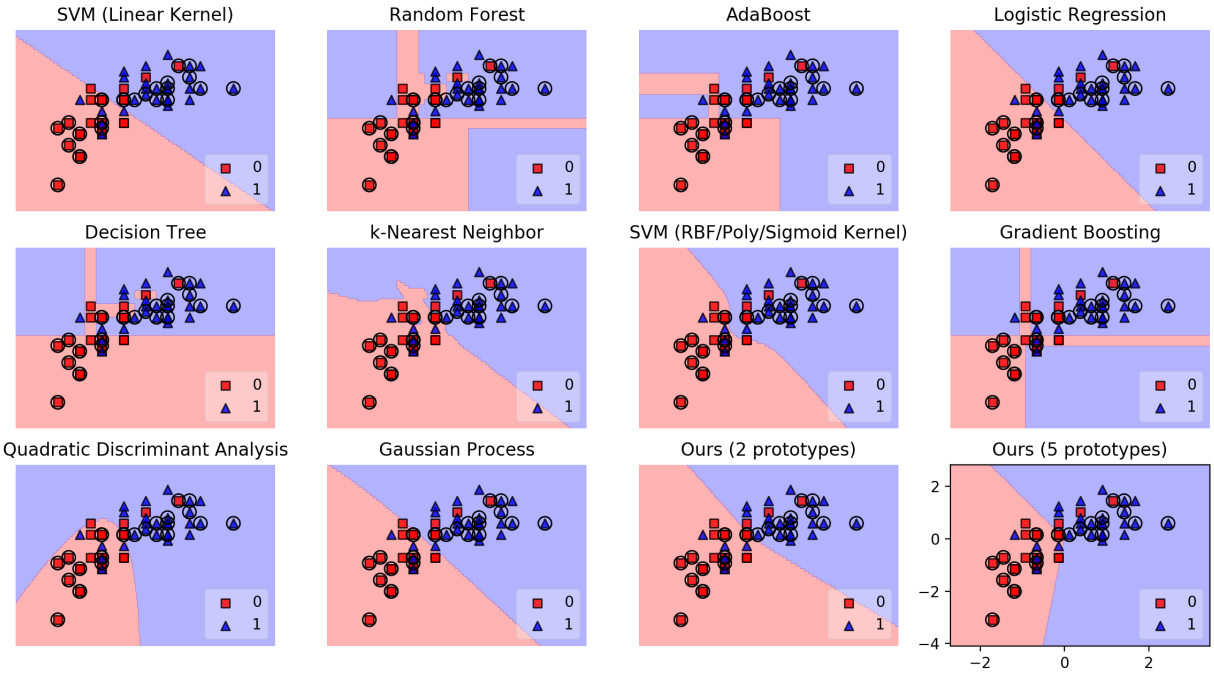

Figure 1: **Visual comparison of the decision boundary of classifiers on a run of the vineyard dataset.** The instances for two classes are shown in red and blue, with the test data, additionally, encircled.

We note that

$$\begin{aligned} & \lambda(\langle B_y \bar{W}_y^t, Z\rangle - g_y^\star(Z)) \; - \; \frac{1}{2\sigma}||Z - V_y^t||_F^2 \\ = \; & -\lambda g_y^\star(Z) - \frac{1}{2\sigma}\left(||Z - V_y^t||_F^2 - 2\sigma\lambda\langle B_y \bar{W}_y{}^t, Z\rangle\right) \\ = \; & -\lambda g_y^\star(Z) - \frac{1}{2\sigma}||Z - (V_y^t + \sigma\lambda B_y \bar{W}_y{}^t)||_F^2. \end{aligned}$$

Since $g_y^\star(Z)$ is the indicator function of the unit norm ball $\|Z\|_{1,\infty}$, i.e., $g_y^\star(Z) = 0$ if $\|Z\|_{1,\infty} \leq 1$ and ∞ otherwise, we have V_y^{t+1}

$$\begin{aligned} = \; & \operatorname*{argmin}_Z \left\{ \lambda g_y^\star(Z) + \frac{1}{2\sigma}\left\|Z - (V_y^t + \sigma\lambda B_y \bar{W}_y^t)\right\|_F^2 \right\} \\ = \; & \arg\min_{||Z||_{1,\infty}\leq 1} \frac{1}{2}\left\|Z - (V_y^t + \lambda\sigma_t B_y \bar{W}_y^t)\right\|_F^2, \end{aligned}$$

which is what we set out to prove. □

Using the results of Propositions 3 and 4, we arrive at the customized CP algorithm in Algorithm 2 for solving the SMP problem. For the updates on V_y, we can compute V_y^{t+1} by projecting independently the rows of $(V_y^t + \sigma\lambda B_y \bar{W}_y^t)$ on the ℓ_1 unit ball. This can be done efficiently (Brucker, 1984; Pardalos & Kovoor, 1990; Duchi *et al.* , 2008).

4 EXPERIMENTS

We conducted several experiments[1] to substantiate the benefits of our framework. Our experiments are designed to convey two salient aspects of our approach. First, we attempt to position our method as an alternative to the standard classification algorithms. Second, we underscore the aptness of our approach as a viable means to obtaining highly sparse yet accurate representations. Before we dive into the details, we provide some visual intuition to differentiate our method from the other models.

We consider the following classifiers for the pictorial depiction: linear SVM (LSVM), SVM with a non-linear kernel (RSVM) selected from radial basis function, polynomial, and sigmoid via cross validation, Logistic Regression (LR), Decision Trees (DT), Random Forest (RF), k-Nearest Neighbor (kNN), Gaussian Process (GP), Gradient Boosting (GB), AdaBoost (AB), and Quadratic Discriminant Analysis (QDA). Our baselines are popular in the machine learning literature, have varying degrees of

[1] All our experiments used the average loss function from (2) in Algorithm 2 directly (i.e. without any smoothing), and we set $T = 200$.

Table 1: Comparison of test accuracy of the different classification algorithms on low dimensional OpenML datasets. The number of prototypes per class for the proposed algorithm, i.e. SMP, was set to 2.

	LSVM	RF	AB	LR	DT	kNN	RSVM	GB	QDA	GP	SMP
sleuth1714	.82±.03	.83±.08	.81±.14	.83±.04	.83±.06	.82±.04	.76±.03	.82±.06	.63±.13	.80±.03	**.87±.06**
vis_env	.66±.04	.65±.08	.66±.03	.65±.08	.62±.04	.57±.03	.69±.06	.64±.03	.62±.07	.65±.09	**.70±.03**
sleuth2016	.71±.04	.70±.03	.70±.05	.72±.03	.65±.07	.67±.06	.72±.04	.65±.03	.62±.07	.73±.03	**.74±.02**
sleuth1605	.66±.09	.70±.06	.66±.07	.70±.07	.63±.09	.66±.05	.65±.09	.65±.09	.62±.05	**.72±.07**	.70±.06
sleuth2002	.65±.04	.59±.04	.60±.04	.64±.04	.55±.04	.63±.07	.64±.04	.60±.05	.65±.05	.62±.04	**.68±.06**
rmftsa_cto	.75±.02	.70±.02	.74±.02	.75±.00	.69±.03	.71±.03	.74±.02	.72±.03	**.76±.02**	.75±.02	**.76±.02**
rabe266	.93±.04	.90±.04	.91±.04	.92±.04	.91±.03	.92±.03	.93±.04	.90±.04	.94±.03	**.95±.04**	.94±.04
rabe265	.58±.07	**.64±.05**	.60±.10	.63±.02	.54±.05	.55±.03	.62±.06	.56±.09	.61±.04	.60±.08	**.64±.06**
rabe148	.95±.04	.94±.02	.91±.08	.95±.04	.89±.07	.92±.05	.91±.06	.91±.08	.92±.09	.95±.02	**.96±.02**
prnn_synth	.82±.02	.84±.02	.84±.02	.83±.02	.83±.01	.83±.02	.83±.03	.84±.01	.83±.03	.84±.02	**.85±.02**
hutsof99	.74±.07	.69±.06	.65±.09	.73±.07	.60±.10	.66±.11	.66±.14	.67±.05	.59±.07	.70±.05	**.77±.04**
humandev	.88±.03	.86±.02	.85±.03	.89±.04	.85±.03	.87±.04	.88±.03	.86±.03	.88±.03	.88±.02	**.89±.04**
elusage	.90±.05	.84±.06	.84±.06	.89±.04	.84±.06	.87±.05	.89±.04	.84±.06	.90±.04	.89±.04	**.92±.04**
baskball	.70±.02	.65±.04	.68±.02	**.71±.03**	.71±.03	.63±.02	.66±.05	.69±.04	.69±.04	.68±.02	**.71±.03**
michiganacc	.72±.06	.60±.09	.71±.05	.71±.04	.67±.06	.68±.05	.71±.05	.69±.04	.72±.04	.71±.05	**.73±.05**
election2000	.92±.04	.91±.04	.91±.03	.92±.02	.91±.03	.92±.01	.90±.07	.92±.02	.72±.06	.92±.03	**.93±.02**
cyyoung9302	.80±.04	.83±.05	.83±.02	.86±.04	.75±.10	.83±.02	.84±.02	.83±.05	.83±.07	.84±.03	**.87±.04**
bankruptcy	.84±.07	.84±.06	.82±.04	.90±.05	.80±.05	.78±.07	.89±.06	.81±.05	.78±.15	.90±.05	**.95±.02**
asbestos	.65±.04	.60±.05	.60±.03	.65±.07	.60±.05	.59±.04	.56±.06	.60±.06	.65±.05	.60±.06	**.68±.06**
MindCave2	.70±.04	.65±.06	.63±.05	.72±.04	.66±.04	.64±.05	.67±.06	.69±.06	.65±.03	.71±.04	**.73±.06**

Algorithm 2 Sparse Multi-Prototype Classification

Choose parameters λ, η, μ and τ, σ
Initialize $W^0 = \{W_y^0\}_{y\in\mathcal{Y}}$ and $V^0 = \{V_y^0\}_{y\in\mathcal{Y}}$
Populate $B = \{B_y\}_{y\in\mathcal{Y}}$
$\bar{W}^0 = W^0$
for $t = 0, 1, \ldots, T$ **do**
  Update V^{t+1} using ℓ_1 projections
  $Z_y^t = V_y^t + \lambda\sigma B_y \bar{W}_y^t$
  $V_y^{t+1} = \arg\min_{||Z||_{1,\infty}\leq 1} (1/2)||Z - Z_y^t||_F^2$
  Update W^{t+1} using soft thresholding
  $U_y^t = W_y^t - \tau\left(\nabla f_y(W_y^t, W_{\setminus y}^t) + \lambda B_y^\top V_y^{t+1}\right)$
  $Q_y^t(i,j) = \dfrac{\text{sign}(U_y^t(i,j))}{\mu\tau\eta + 1}$
  $W_y^{t+1}(i,j) = Q_y^t(i,j)\ \ \left(|U_y^t(i,j)| - \mu\tau\right)_+$
  Update $\bar{W}^{t+1}$
  $\bar{W}_y^{t+1} = 2W_y^{t+1} - W_y^t$

(non-)linearity, and include models from both the generative and the discriminative families.

Fig. 1 shows the decision boundaries obtained by the different classifiers on the vineyard data. The two classes are depicted in red and blue. The test data have been encircled to distinguish them from the training instances. We observe that, on this problem, the different instantiations of our model provide a better separation of the two classes compared to the other models. For instance, both the linear classifiers, i.e. Logistic Regression and SVM with linear kernel, seem to underfit the data. On the other hand, the SMP models are able to carve out good decision boundaries. We further observe that our model trained with five prototypes performs better than that trained with two on this data. However, this phenomenon does not hold in general, since having multiple prototypes might lead to overfitting, especially in small datasets, for low values of λ.

4.1 LOW-DIMENSIONAL REGIME (NO SPARSITY)

We found that the SMP performed very well on several low-dimensional (i.e. $n \leq 20$) OpenML datasets.[2] We now describe the results of our experiments with these datasets. We preprocessed all the data to normalize each feature to have zero mean and unit variance. We split each dataset evenly into train and test sets using random partitioning. For SMP, we clustered the training examples in each class into $p = 2$ clusters using k-means, and initial-

[2]Available at https://www.openml.org/

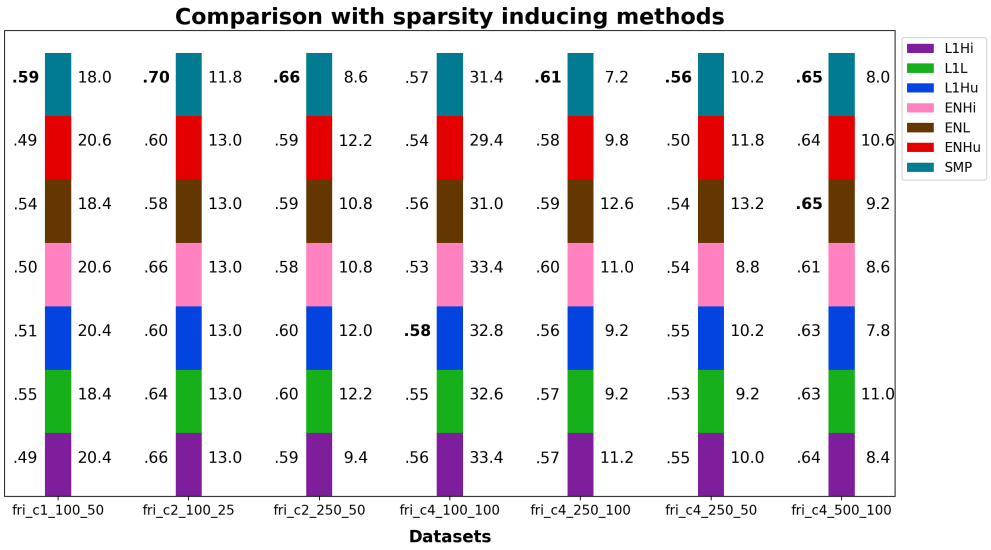

Figure 2: **Comparison on high dimensional OpenML datasets.** Each stacked bar shows two numbers: average test accuracy on the left, and total number of selected features (including multiplicities) on the right.

ized the class prototypes with the cluster centers. We followed 5-fold cross-validation (CV) for each method to obtain a good setting of hyperparameters. The details are given in the Supplementary. We report the average test accuracy and standard deviation over five random partitions per dataset.

The results on test accuracy are documented in Table 1. Evidently, SMP is seen to perform very well on many of these datasets. Note that these results should not be misconstrued as implying that SMP would generally work well with arbitrary data. Indeed, we discovered that the performance of SMP was suboptimal on many other datasets, where algorithms like RSVM and GB performed much better due to highly non-linear structure in the data.

4.2 HIGH-DIMENSIONAL REGIME

In this section, we explicate the results of our experiments on high dimensional data, where feature selection becomes especially critical. Our objective is to demonstrate the efficacy of SMP in recovering discriminative sparse features.

We first describe the experimental setup. We compare SMP with six baselines that induce sparsity by minimizing an ℓ_1-regularized loss function (Bach *et al.* , 2012). These baselines minimize a regularized empirical loss function, namely hinge loss, log loss, or the binary-classification Huber loss, where the regularization consisted of either ℓ_1 or elastic net penalty (i.e. both ℓ_1 *and* ℓ_2 terms). We call these six baselines as L1Hi (ℓ_1, hinge), L1L (ℓ_1, log), L1Hu (ℓ_1, Huber), EnHi(elastic net, hinge), ENL(elastic net, log) and ENHu(elastic net, huber) respectively.

The amount of sparsity achieved by different baselines at any fixed penalty is method specific. Therefore, we first observed the sparsity obtained with SMP on each method, and then modulated the ℓ_1 penalty for other methods to have roughly the same number of selected features. Then, we retrained these classifier using only the selected features, using the same loss (hinge, loss, or log) and an additional ℓ_2 penalty. Our procedure ensured that each baseline benefited, in effect, from an *elastic net*-like regularization while having the most important features at its disposal. We followed 5-fold CV to find a good setting of hyperparameters for each method.

Our results on several high dimensional OpenML datasets are summarized in Fig. 2. The first number in the name of a dataset represents the number of instances in the dataset, while the second term represents dimensionality. In SMP, since some features might be selected in more than one prototype, for fairness of evaluation, we included the multiplicity while computing the selected feature count. These results underscore the merits in combining the power of multiple prototypes with sparse representations.

References

Aiolli, F., & Sperduti, A. 2005. Multiclass Classification with Multi-Prototype Support Vector Machines. *Journal of Machine Learning Research (JMLR)*, **6**, 817–850.

Bach, F., Jenatton, R., Mairal, J., & Obozinski, G. 2012. Optimization with Sparsity-Inducing Penalties. *Foundations and Trends in Machine Learning*, **4**(1), 1–106.

Brucker, P. 1984. An $O(n)$ algorithm for quadratic Knapsack problems. *Operations Research Letters*, **3**(3), 163–166.

Bucila, B., Caruana, R., & Niculescu-Mizil, A. 2006. Model Compression. *Pages 535–541 of: The Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Chambolle, A., & Pock, T. 2011. A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging. *Journal of Mathematical Imaging and Vision*, **40**(1), 120–145.

Chambolle, A., & Pock, T. 2016. On the ergodic convergence rates of a first-order primal-dual algorithm. *Math. Program.*, **159**(1-2), 253–287.

Chen, W., Wilson, J., Tyree, S., Weinberger, K., & Chen, Y. 2015. Compressing neural networks with the hashing trick. *Pages 2285–2294 of: ICML*.

Crammer, K., & Singer, Y. 2001. On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines. *JMLR*, **2**, 265–292.

Dekel, O., & Singer, Y. 2007. Support vector machines on a budget. *Pages 345–352 of: NIPS*.

Dekel, O., Shalev-Shwartz, S., & Singer, Y. 2008. The Forgetron: A kernel-based perceptron on a budget. *SIAM Journal on Computing*, **37**(5), 1342–1372.

Duchi, J., Shalev-Shwartz, S., Singer, Y., & Chandra, T. 2008. Efficient projection onto the ℓ_1-ball for learning in high dimensions. *Pages 272–279 of: ICML*.

Feng, J., Yuan, X., Wang, Z., Xu, H., & Yan, S. 2012. Auto-Grouped Sparse Representation for Visual Analysis. **23**, 640–653.

Gupta, C., Suggala, A. S., Goyal, A., Simhadri, H. V., Paranjape, B., Kumar, A., Goyal, S., Udupa, R., Varma, M., & Jain, P. 2017. ProtoNN: Compressed and Accurate kNN for Resource-scarce Devices. *Pages 1331–1340 of: ICML*.

Han, S., Mao, H., & Dally, W. J. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *In: ICLR*.

He, B., Ma, F., & Yuan, X. 2017. An Algorithmic Framework of Generalized Primal–Dual Hybrid Gradient Methods for Saddle Point Problems. *Journal of Mathematical Imaging and Vision*, **58**(2), 279–293.

Hocking, T. D., Joulin, A., Bach, F., & Vert, J-P. 2011. Clusterpath an algorithm for clustering using convex fusion penalties. *In: ICML*.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., & Bengio, Y. 2016. Binarized Neural Networks. *Pages 4107–4115 of: NIPS*.

Kumar, A., Goyal, S., & Varma, M. 2017. Resource-efficient Machine Learning in 2 KB RAM for the Internet of Things. *Pages 1935–1944 of: ICML*.

Kusner, M., Tyree, S., Weinberger, K. Q., & Agrawal, K. 2014. Stochastic neighbor compression. *Pages 622–630 of: ICML*.

Luo, J.-H., Wu, J., & Lin, W. 2017. ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression. *Pages 5068–5076 of: ICCV*.

Nakkiran, P., Alvarez, R., Prabhavalkar, R., & Parada, C. 2015. Compressing deep neural networks using a rank-constrained topology. *In: Sixteenth Annual Conference of the International Speech Communication Association*.

Nan, F., Wang, J., & Saligrama, V. 2016. Pruning Random Forests for Prediction on a Budget. *Pages 2334–2342 of: NIPS*.

Pardalos, P. M., & Kovoor, N. 1990. An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds. *Mathematical Programming*, **46**, 321–328.

Sainath, T., Kingsbury, B., Sindhwani, V., Arisoy, E., & Ramabhadran, B. 2013. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. *Pages 6655–6659 of: ICASSP*.

Tibshirani, R. 1996. Regression Shrinkage and Selection via the lasso. *Journal of the Royal Statistical Society. Series B (methodological)*, **58**(1), 267–288.

Weston, J., & Watkins, C. 1999. Support vector machines for multi-class pattern recognition. *Pages 219–224 of: Proceedings of the 6th European Symposium on Artificial Neural Networks (ESANN)*.

Yu, Y., Liu, S., & Pan, S. J. 2017. Communication-Efficient Distributed Primal-Dual Algorithm for Saddle Point Problems. *In: UAI*.

Zhang, Y., & Xiao, L. 2015. Stochastic primal-dual coordinate method for regularized empirical risk minimization. *Pages 353–361 of: ICML*.

Zhong, K., Guo, R., Kumar, S., Yan, B., Simcha, D., & Dhillon, I. 2017. Fast Classification with Binary Prototypes. *Pages 1255–1263 of: AISTATS*.

Zou, H., & Hastie, T. 2005. Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society. Series B (methodological)*, **67**(2), 301–320.