Nonlinear Feature Extraction with Max-Margin Data Shifting

Jianqiao Wangni[†], **Ning Chen**[‡*]

[‡]MOE Key lab of Bioinformatics, Bioinformatics Division and Center for Synthetic & Systems Biology,
TNList, [†]Department of Electronic Engineering, Tsinghua University, Beijing, 100084, China
zjnqha@gmail.com, ningchen@tsinghua.edu.cn

Abstract

Feature extraction is an important task in machine learning. In this paper, we present a simple and efficient method, named max-margin data shifting (MMDS), to process the data before feature extraction. By relying on a large-margin classifier, MMDS is helpful to enhance the discriminative ability of subsequent feature extractors. The kernel trick can be applied to extract nonlinear features from input data. We further analyze in detail the example of principal component analysis (PCA). The empirical results on multiple linear and nonlinear models demonstrate that MMDS can efficiently improve the performance of unsupervised extractors.

Introduction

Feature extractors are important in machine learning and can heavily affect final performance. They transform low-level *input data*, like raw pixels in images and bag-of-words in documents, to high-level *features* like typical visual patterns (Bengio, Courville, and Vincent 2013) and topics (Blei, Ng, and Jordan 2003). Supervised feature extraction methods have attracted much attention because they are able to learn suitable features for specific tasks. For example, supervised LDA (Mcauliffe and Blei 2008; Zhu, Ahmed, and Xing 2012) jointly models words in documents and their responses (e.g., movie ratings); supervised dictionary learning (Mairal et al. 2009) seeks the overcomplete basis for signals that belong to different classes.

In general, the training data are valuable and limited, therefore cannot cover the rich variations. For example, each node of a sensor network may have a probability to mechanically fail, and visual objects have various types of translations and rotations. Therefore, training a good extractor requires a large scale of observations; and how to accomplish such a task by using only a relatively small dataset is an important and challenging goal to pursue. One useful technique to improve the robustness of extractors is corruption, which artificially generates more data by adding noise (e.g., Guassian noise or blankout noise) to the original data. This noising scheme has proven effective on introducing adaptive regularization (Bishop 1995); helping auto-encoder (Vincent et al. 2008) to learn semantic features; and improving the generalization ability of deep networks (Hinton et al. 2012).

The *explicit corruption* method transforms each data instance $\mathbf{x}_n \in \mathbb{R}^D$ $(n = 1, \cdots, N)$ to S versions $\tilde{\mathbf{x}}_{ns}$ through some pre-defined corrupting models $p(\tilde{\mathbf{x}}|\mathbf{x}_n)$, and minimizes the average loss $\frac{1}{NS}\sum_n \sum_s L(\tilde{\mathbf{x}}_{ns}, y_n)$ over the corrupted dataset. Though being simple, the disadvantage of the explicit corruption is also obvious—corrupting every observation to multiple copies will dramatically increase the training size and thereby training time. Recent work on marginalized corrupted features (MCF) (Maaten et al. 2013) provides an *implicit corruption* scheme, which avoids enumerating multiple copies and instead minimizes the expected loss $\frac{1}{N}\mathbb{E}_p[L(\tilde{\mathbf{x}}_n, y_n)]$ under the corrupting model. MCF showed promise in learning autoencoders (Chen et al. 2014) and link predictors (Chen et al. 2015).

In this paper, we present max-margin data shifting (MMDS), a simple and efficient method that processes the input data before extracting features. MMDS relies on a large-margin classifier on the original data. By building a subsequent feature extractor (e.g., PCA), we are able to learn features that are suitable for classification tasks. MMDS can be approximately derived as the mean of a supervised corrupting model, which considers both the standard noising model (e.g., Gaussian noise) on input data and the discriminative ability of a potential corruption according to a pre-learned large-margin classifier. The latter factor encodes our intuition that some corrupted data points may cross the decision boundary (if given) and lead to a difficult case for classification; and such corrupted data points should be discouraged.

Our empirical results on various datasets demonstrate the effectiveness of MMDS in the context of various feature extractors.

Supervised Data Corruption

Considering a fully labeled dataset that includes N data points $\mathbf{x}_n \in \mathbb{R}^D$ and labels $y_n \in \mathcal{Y} = \{1, \ldots, H\}$, where H is the number of categories, we aim to extract the high-level representation, or latent features, $\mathbf{z}_n \in \mathbb{R}^K$ from data $\mathbf{x}_n$. Instead of dealing with the original data, we consider the augmented data $\tilde{\mathbf{x}}_{ns}$, which are sampled from a *supervised* corrupting distribution $p_n(\tilde{\mathbf{x}}|\mathbf{x}_n, y_n, \boldsymbol{\eta})$, parameterized by $\boldsymbol{\eta}$. Assuming each data is corrupted for S times,

[*]Corresponding author.

the fitness of each $\mathbf{z}_n$ can be measured by a loss function $L(\mathbf{x}_n, \mathbf{z}_n, \Theta) = \frac{1}{S}\sum_s L(\tilde{\mathbf{x}}_{ns}, \mathbf{z}_n, \Theta)$ of a feature extractor, whose parameters are denoted by Θ. For various extractor models, we use different loss functions. For example, the loss function $L(\tilde{\mathbf{x}}_{ns}, \mathbf{z}_n, \Theta)$ for Non-negative Matrix Factorization (NMF) (Lee and Seung 2001), Kmeans (KM) and Principal Component Analysis (PCA) (Collins, Dasgupta, and Schapire 2001) are defined as

$$\begin{aligned}
&NMF: \|\tilde{\mathbf{x}}_{ns} - \mathbf{W}^\intercal \mathbf{z}_n\|^2, \quad \mathbf{z}_n \in \mathbb{R}_+^K, \mathbf{W} \in \mathbb{R}_+^{K\times D},\\
&KM: \|\tilde{\mathbf{x}}_{ns} - \mathbf{W}^\intercal \mathbf{z}_n\|^2, \quad \mathbf{z}_n \in \{e_k\}_{k=1}^K, \mathbf{W} \in \mathbb{R}^{K\times D}\\
&PCA: \frac{\tilde{\mathbf{x}}_{ns}^\intercal \tilde{\mathbf{x}}_{ns}}{2\sigma^2} - \frac{\mathbf{z}_n^\intercal \mathbf{W} \tilde{\mathbf{x}}_{ns}}{\sigma^2} + \mathcal{A}(\mathbf{z}_n^\intercal \mathbf{W}), \mathbf{W} \in \mathcal{W}
\end{aligned} \tag{1}$$

where $\mathcal{W} = \{\mathbf{W} \in R^{K\times D} : \mathbf{W}\mathbf{W}^\intercal = \mathbf{I}\}$, e_k is a vector that places 1 in the k-th dimension and 0 in all other dimensions, σ is a noise parameter, and $\mathcal{A}(\mathbf{z}_n^\intercal \mathbf{W}) = \int \exp(\mathbf{z}_n^\intercal \mathbf{W}\mathbf{x}/\sigma^2 - \mathbf{x}^\intercal\mathbf{x}/2\sigma^2)d\mathbf{x}$ is log-partition function. Our optimization objective function is $\mathcal{F}(\Theta) = \frac{1}{S}\sum_n\sum_s L(\tilde{\mathbf{x}}_{ns}, \mathbf{z}_n, \Theta)$. Let $S \to \infty$, we have

$$\mathcal{F}(\Theta) = \sum_n \mathbb{E}_{p(\tilde{\mathbf{x}}|\mathbf{x}_n, y_n, \boldsymbol{\eta})}[L(\tilde{\mathbf{x}}, \mathbf{z}_n, \Theta)]. \tag{2}$$

A standard corrupting model treats all the corrupted versions equally, as indicated by the uniform weight $\frac{1}{S}$ in the average loss. Here, we suggest to distinguish two types of corrupted data points. As illustrated in Figure 1, each of the two data points in different classes (marked as red circles) are corrupted many times, as indicated by the squares and triangles around each data point. Suppose there is a linear decision boundary that well separates the two red points. Then, some of the corrupted data points may cross the decision boundary or be near the boundary and get mixed with the data in a different class. Such points may not be good for extracting features that are suitable for classification, as they make it harder to find a good separating line in the original space. Therefore, we propose a *dropping* step to remove such corrupted points.

The above *corrupting and dropping* design can be formally described by the supervised corrupting distribution $p(\tilde{\mathbf{x}}|\mathbf{x}_n, y_n, \boldsymbol{\eta})$, which considers both the standard noise (e.g., Gaussian noise) on input data $\mathbf{x}_n$ and the discriminative ability of each data. Here, we measure the discriminative ability via data margin. Let $\boldsymbol{\eta}_y$ be the classifier weights associated with class y. Following (Crammer and Singer 2002), we can learn a large-margin classifier by solving the following problem:

$$\min_{\boldsymbol{\eta}} \frac{1}{2}\sum_y \|\boldsymbol{\eta}_y\|^2 + C\sum_n \max_y(\Delta\ell_n(y) + \boldsymbol{\eta}_y^\intercal \mathbf{x}_n) - \boldsymbol{\eta}_{y_n}^\intercal \mathbf{x}_n \tag{3}$$

where $\Delta\ell_n(y) = \ell\mathbb{I}(y \neq y_n)$ measures the cost of making a wrong prediction, and for simplicity we have omitted the offset parameters. As we need to improve the separability of the training set, besides dropping the samples that cross the decision boundary, we also drop the ones that are within a certain distance from the boundary (marked as crosses in Figure 1). To improve the robustness, we further relax the deterministic dropping as a soft assignment. That is, we put a weight that decreases exponentially w.r.t. to the hinge-loss (i.e., the gap between their margins and the threshold):

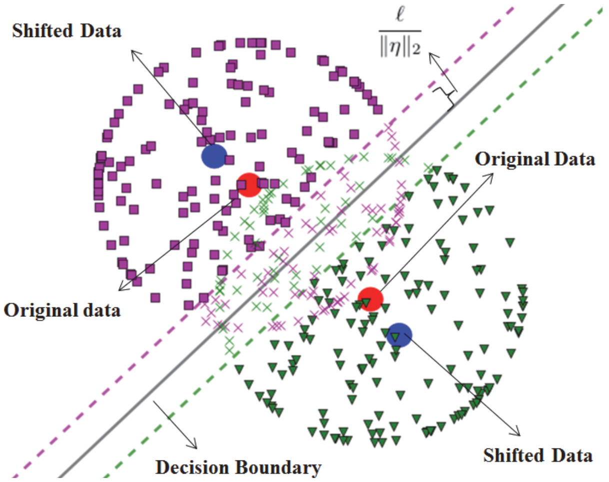

Figure 1: The original data and corrupted data. (Best viewed in color)

$$\phi(y_n|\tilde{\mathbf{x}}, \boldsymbol{\eta}) = \exp\{-C\max_y(\Delta\ell_n(y) + \boldsymbol{\eta}_y^\intercal\tilde{\mathbf{x}} - \boldsymbol{\eta}_{y_n}^\intercal\tilde{\mathbf{x}})\}, \tag{4}$$

whose minus logarithm is the multiclass hinge loss by considering the margin favored by the true label over an alternative label. C is a constant. For real-valued data, we further consider the Gaussian noising model, and define our supervised corrupting distribution p as

$$p(\tilde{\mathbf{x}}|\mathbf{x}_n, y_n, \boldsymbol{\eta}) \propto \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}_n, \sigma^2\mathbf{I})\phi(y_n|\tilde{\mathbf{x}}, \boldsymbol{\eta}), \tag{5}$$

which is a combination of standard data noising and the discriminative ability. In the sequel, we denote $p_n(\tilde{\mathbf{x}}) = p(\tilde{\mathbf{x}}|\mathbf{x}_n, y_n, \boldsymbol{\eta})$.

We derive an approximate objective to the intractable expectation in Eq. (2):

$$\mathcal{F}(\Theta) \approx \sum_n L(\mathbb{E}_{p_n}[\tilde{\mathbf{x}}], \mathbf{z}_n, \Theta), \tag{6}$$

where the approximation is due to the movement of the expectation over p_n from outside to inside of the loss function. The objective only depends on data expectation and thereby is efficient without explicitly considering a large amount of corrupted samples. Figure 1 gives a geometrical view: As a proportion of the corrupted data are deleted according to the cutting planes (i.e., the two dotted lines), their mean $\mathbb{E}_{p_n}[\tilde{\mathbf{x}}]$ (blue circles) are shifted from original data $\mathbf{x}_n$ (red circles). Besides, the shifting direction is perpendicular to the cutting planes due to the geometric symmetry.

In order to make the above procedure practical, we still need to approximate the expectation $\mathbb{E}_{p_n}[\tilde{\mathbf{x}}]$. Below, we present a simple procedure, which works well in practice. First, we can show that the corrupting distribution $p_n(\tilde{\mathbf{x}})$ can be obtained by minimizing the following objective

$$\mathrm{KL}\left(q_n(\tilde{\mathbf{x}})\|q_n^0(\tilde{\mathbf{x}})\right) + C\mathbb{E}_{q_n}[\max_y(\Delta\ell_n(y) + \boldsymbol{\eta}_y^\intercal\tilde{\mathbf{x}} - \boldsymbol{\eta}_{y_n}^\intercal\tilde{\mathbf{x}})],$$

where $q_n^0(\tilde{\mathbf{x}}) = \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}_n, \sigma^2\mathbf{I})$ and the *KL* function is Kullback−Leibler divergence. Then, we move the expectation inside the *max* function and deal with the following objective (denoted by $\mathcal{L}(q_n(\tilde{\mathbf{x}}))$):

$$\mathrm{KL}\left(q_n(\tilde{\mathbf{x}})||q_n^0(\tilde{\mathbf{x}})\right) + C\max_y(\Delta\ell_n(y) - \mathbb{E}_{q_n}[\boldsymbol{\eta}_{y_n}^\intercal\tilde{\mathbf{x}} - \boldsymbol{\eta}_y^\intercal\tilde{\mathbf{x}}]),$$

which is in fact a lower bound due to the Jensen's inequality and the convexity of *max* function. Considering the following optimization problem

$$\min_{q(\tilde{\mathbf{x}}),\boldsymbol{\eta}} \sum_n \mathcal{L}(q_n(\tilde{\mathbf{x}})) + \frac{1}{2}\|\boldsymbol{\eta}\|^2. \tag{7}$$

which can be written in a constrained form:

$$\min_{\{q_n(\tilde{\mathbf{x}})\},\boldsymbol{\eta},\boldsymbol{\xi}} \quad \sum_n \mathrm{KL}\left(q_n(\tilde{\mathbf{x}})||q_n^0(\tilde{\mathbf{x}})\right) + \frac{1}{2}\|\boldsymbol{\eta}\|^2 + C\sum_{n,y}\xi_n^y$$
$$\text{s.t.}: \quad \forall n, y \in \mathcal{Y}, \quad \mathbb{E}_{q_n}[\boldsymbol{\eta}_{y_n}^\intercal\tilde{\mathbf{x}} - \boldsymbol{\eta}_y^\intercal\tilde{\mathbf{x}}] \geq \Delta\ell_n(y) - \xi_n^y,$$

with the normalization constraint that $\int q_n(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} = 1$. The Lagrangian function is $L(\{q_n(\tilde{\mathbf{x}})\}, \boldsymbol{\eta}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \lambda) = \sum_n \mathrm{KL}\left(q_n(\tilde{\mathbf{x}})||q_n^0(\tilde{\mathbf{x}})\right) + \lambda\int q_n(\tilde{\mathbf{x}})d\tilde{\mathbf{x}} - \lambda + C\sum_{n,y}\xi_n^y - \sum_{n,y}\alpha_n^y(\mathbb{E}_{q_n}[\boldsymbol{\eta}_{y_n}^\intercal\tilde{\mathbf{x}} - \boldsymbol{\eta}_y^\intercal\tilde{\mathbf{x}} - \Delta\ell_n(y)] + \xi_n^y)$. Though finding a rigorous solution needs to iterate over $q(\tilde{\mathbf{x}})$ and $\boldsymbol{\eta}$, we consider a simple procedure without iteration and derive a simple and efficient operator. Specifically, we start with setting $q_n(\tilde{\mathbf{x}})$ at the prior $q_n^0(\tilde{\mathbf{x}})$, and solve for the classifier weights $\boldsymbol{\eta}$. For this step, we get the solution $\boldsymbol{\eta}_y = \sum_n \alpha_n^y\mathbf{x}_n$ and the dual parameters $\{\boldsymbol{\alpha}\}$ are obtained by optimizing the dual form of a multiclass SVM (Crammer and Singer 2002). This step can be efficiently done by an existing solver. Then by setting the derivative of the objective w.r.t $\{q_n(x)\}_{n=1}^N$ to zero while fixing $\boldsymbol{\alpha}$ and $\boldsymbol{\eta}$, we get

$$q_n(\tilde{\mathbf{x}}) \sim \mathcal{N}(\mathbf{x}_n + \sigma^2\Sigma_y\alpha_n^y[\boldsymbol{\eta}_{y_n} - \boldsymbol{\eta}_y], \sigma^2\mathbf{I}). \tag{8}$$

Since we use $q_n(\tilde{\mathbf{x}})$ to approximate $p_n(\tilde{\mathbf{x}})$, we have a rule of calculating $\mathbb{E}_{p_n}[\mathbf{x}_n]$, as $\mathbb{E}_{p_n}[\mathbf{x}_n] \approx \mathbb{E}_{q_n}[\mathbf{x}_n]$.

Max-Margin Data Shifting

With the above approximation, we get a simple and efficient rule to shift the input data:

$$\text{(MMDS)} \;: \mathcal{T}[\mathbf{x}_n] = \mathbf{x}_n + \sigma^2\sum_y \alpha_n^y[\boldsymbol{\eta}_{y_n} - \boldsymbol{\eta}_y], \tag{9}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\eta}$ are the dual and primal solutions of SVM in Eq. (3). We will refer to the operation $\mathcal{T}$ as *max-margin data shifting* (*MMDS*). This operation satisfies our intuition—the simple multiplication with SVM coefficients suggests that the MMDS data are more likely to be predicted as true labels, in contrast with all other labels. Then we train an unsupervised extractor with the MMDS data, by optimizing

$$F(\Theta) = \sum_n L(\mathcal{T}[\mathbf{x}_n], \mathbf{z}_n, \Theta).$$

Note that we only use the MMDS data in learning the feature extractor. The procedure of training an extractor on the MMDS data is summarized as

$$\text{Training Data}\{\mathbf{x}\} \xrightarrow{\text{Shift}} \mathcal{T}[\mathbf{x}] \xrightarrow{\text{Train}} \text{Extractor}.$$

After learning the feature extractor, we use it to extract features from the original data and learn a classifier. The procedure for training a classifier is summarized as

$$\text{Training Data}\{\mathbf{x}, y\} \xrightarrow{\text{Extractor}} \{\mathbf{z}, y\} \xrightarrow{\text{Train}} \text{Classifier}.$$

Finally, for testing data, we extract the latent features $\mathbf{z}$ and apply the classifier to make the prediction.

The above shifting process can be naturally extended to consider nonlinear mapping on the input data, e.g., via a kernel mapping in a Reproducing Kernel Hilbert Space (RKHS). *Random Fourier Features* (*RFF*) (Rahimi and Recht 2009) provide a simple and scalable method to embed the original data to RKHS defined by shift-invariant kernels like radical basis function (RBF). (Lopez-Paz et al. 2014) proposed Randomized Kernel PCA, which uses PCA to fit the *RFF* embedding. Suppose a kernel satisfying $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \mathcal{K}(\mathbf{x} - \mathbf{x}', \mathbf{0})$, and $p_f(\boldsymbol{\omega})$ is a distribution that normalizes the Fourier transformation of this kernel $p_f(\boldsymbol{\omega}) \propto \int \exp(i\boldsymbol{\omega}^\intercal\mathbf{x})\mathcal{K}(\mathbf{x}, \mathbf{0})d\mathbf{x}$, where$\int p_f(\boldsymbol{\omega})d\boldsymbol{\omega} = 1$. The T-dimensional *RFF* embeddings are generated as

$$\mathcal{F}[\mathbf{x}] \triangleq [\cos(\boldsymbol{\omega}_1^\intercal\mathbf{x} + b_1), \cdots, \cos(\boldsymbol{\omega}_T^\intercal\mathbf{x} + b_T)]^\intercal, \tag{10}$$

where $\boldsymbol{\omega}_t \sim p_f(\boldsymbol{\omega})$ and $b_t \sim \text{Uniform}(-\pi, \pi)$. We can apply any extractors to fit the MMDS-processed versions of the *RFF* embeddings, to learn discriminative nonlinear features. The training procedure of kernelized extractors is

$$\{\mathbf{x}\} \xrightarrow{\textit{RFF}} \mathcal{F}[\mathbf{x}] \xrightarrow{\text{MMDS}} \mathcal{T}[\mathcal{F}[\mathbf{x}]] \xrightarrow{\text{Train}} \text{Extractor}. \tag{11}$$

Notice that our framework can be easily generalized to semi-supervised learning. All we need to do is to shift the labeled data while keeping the unlabeled ones not changed.

Application in Principal Component Analysis

Now, we use principal component analysis (PCA) to explain the effect of MMDS in detail. The vanilla PCA seeks the dominant components underlying data. To consider labels, which provide side information with the potential to select components suitable for specific tasks, various extensions have been developed. For example, Supervised PCA (Yu et al. 2006; Rish et al. 2008; Du et al. 2015) combines probabilistic PCA with a regression task under the assumption that both data and side information are generated from a common latent space through linear mapping. The SVDM (Pereira and Gordon 2006) seeks a low dimensional linear embedding of training data using singular vector decomposition (SVD). (Rish et al. 2008) tackled the above problem using a closed-form update rule with provable convergence. Though these methods were proven to be effective, they deal with label information in a regression manner. Furthermore, most of them are based on EM solvers, which can get trapped at local minimum, and suffer from time complexity issues. In this section, we first define a max-margin PCA (MMPCA). Then we show that MMPCA can be approximately solved by learning an unsupervised PCA on the MMDS data, and we will refer to this method as MMDS-PCA.

A PCA model assumes that zero-centered data are generated from features through linear mapping that $\mathbf{x} = \mathbf{W}^\intercal\mathbf{z} + \boldsymbol{\epsilon}$, where $\mathbf{z}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\boldsymbol{\epsilon}$ is Gaussian noise. Following the notation of (Guo 2009), we use the formulation of Exponential Family PCA (EFPCA) for further description due to simplicity. We also use capital letters to represent matrices stacking variables. For instance, $\mathbf{Z} \in \mathbb{R}^{K \times N}$ is a matrix whose n-th column is $\mathbf{z}_n$.

Our objective function on the original data matrix $\mathbf{X}$ without corruption is described as in Eq. (1):

$$L(\mathbf{X}, \mathbf{Z}, \mathbf{W}) \triangleq \mathcal{A}(\mathbf{Z}^\intercal \mathbf{W}) - \operatorname{tr}(\mathbf{Z}^\intercal \mathbf{W} \mathbf{X}) + \frac{1}{2} \operatorname{tr}(\mathbf{Z}^\intercal \mathbf{Z}),$$

where $\mathbf{W} \in \mathcal{W}$, $\mathcal{A}(\mathbf{Z}^\intercal \mathbf{W}) = \sum_n \mathcal{A}(\mathbf{z}_n^\intercal \mathbf{W})$, and we omit the term that only depends on $\mathbf{X}$. This model is difficult to optimize because the orthogonal constraint $\mathcal{W}$ is non-convex, and the objective is not jointly convex on $\mathbf{Z}$ and $\mathbf{W}$, but it is convex on $\mathbf{Z}$ for fixed $\mathbf{W}$ in a good way. Denoting $\mathcal{A}^*$ as the Fenchel conjugate of $\mathcal{A}$, we have $\mathcal{A}(\mathbf{z}_n^\intercal \mathbf{W}) = \max_{\mathbf{U}_n} \operatorname{tr}(\mathbf{z}_n^\intercal \mathbf{W} \mathbf{U}_n) - \mathcal{A}^*(\mathbf{U}_n)$. Notice that $\mathcal{A}^*(\mathbf{U}_n)$ is convex. Denoting $\mathcal{A}^*(\mathbf{U}) = \sum_n \mathcal{A}^*(\mathbf{U}_n)$ and setting $\partial L / \partial \mathbf{z}_n = 0$, we get $\mathbf{z}_n = \mathbf{W}(\mathbf{x}_n - \mathbf{U}_n)$. Substituting it back, the dual form of PCA is:

$$L(\mathbf{X}, \mathbf{M}, \mathbf{U}) \triangleq -\mathcal{A}^*(\mathbf{U}) - \frac{1}{2} \operatorname{tr}((\mathbf{U} - \mathbf{X})(\mathbf{U} - \mathbf{X})^\intercal \mathbf{M}),$$

where $\mathbf{M} \in \mathcal{M} \triangleq \{\mathbf{W}^\intercal \mathbf{W} : \mathbf{W} \in \mathcal{W}\}$. (Guo 2009) suggested that the domain $\mathcal{M}$ can be relaxed to $\mathcal{M}' = \{\mathbf{M} : \mathbf{I} \succeq \mathbf{M} \succeq 0, \operatorname{tr}(\mathbf{M}) = K\}$. After this step the objective satisfies strong minmax property (Borwein and Lewis 2010), which implies that the *min* and *max* operators are exchangeable in coordinate descent optimization. Then we get a closed-form solution as $\mathbf{W} = Q^K((\mathbf{X} - \mathbf{U})(\mathbf{X} - \mathbf{U})^\intercal)$, where the operator $Q^K(\mathbf{A})$ represents the matrix stacking the top-K eigenvectors of matrix $\mathbf{A}$. It is ensured that the solution $\mathbf{W}$ is orthogonal and satisfies our premise. This indicates that the relaxation of domain from $\mathcal{M}$ to $\mathcal{M}'$ is tight.

A supervised version of PCA is max-margin PCA (*MM-PCA*), which uses SVM to fit the latent variables $\mathbf{z}$ while extracting components:

$$\begin{aligned} &\min_{\mathbf{W} \in \mathcal{W}} \min_{\mathbf{Z}} \min_{\boldsymbol{\nu}} \frac{1}{2} \|\boldsymbol{\nu}\|^2 + L(\mathbf{X}, \mathbf{Z}, \mathbf{W}) + C \sum_n \xi_n \\ &\quad \text{s.t.} : \forall n, \quad y_n(\boldsymbol{\nu}^\intercal \mathbf{z}_n - b) \geq 1 - \xi_n, \quad (\xi_n \geq 0) \end{aligned} \tag{12}$$

where we have restricted us to consider the binary classification (i.e., $y_n \in \{+1, -1\}$) for simplicity. Extension to the multi-class case can be done similarly as in the MMDS formulation. We define the Lagrangian function of the soft-margin SVM $F(\boldsymbol{\alpha}, \boldsymbol{\nu}, \mathbf{Z})$ as

$$\frac{1}{2} \|\boldsymbol{\nu}\|^2 - \sum_n \alpha_n [y_n(\boldsymbol{\nu}^\intercal \mathbf{z}_n - b) - 1 + \xi_n] + C \sum_n \xi_n, \tag{13}$$

where $\forall n, \alpha_n \in [0, C]$ and C is a positive constant. MM-PCA is reformulated as

$$\min_{\mathbf{W} \in \mathcal{W}} \min_{\mathbf{Z}} \max_{\boldsymbol{\alpha}} \min_{\boldsymbol{\nu}} \; F(\boldsymbol{\alpha}, \boldsymbol{\nu}, \mathbf{Z}) + L(\mathbf{X}, \mathbf{Z}, \mathbf{W}). \tag{14}$$

Notice that the *min* and *max* operators in $F(\boldsymbol{\alpha}, \boldsymbol{\nu}, \mathbf{Z})$ are exchangeable due to linearity. Setting the derivative w.r.t $\mathbf{Z}$ and we get $\mathbf{z}_n = \mathbf{W}(\mathbf{x}_n - \mathbf{U}_n) + \alpha_n y_n \boldsymbol{\nu}$.

Now we consider a strongly related SVM model whose Lagrangian function is $F(\boldsymbol{\alpha}, \boldsymbol{\eta}, \mathbf{X}) = \frac{1}{2} \|\boldsymbol{\eta}\|^2 - \sum_n \alpha_n [y_n(\boldsymbol{\eta}^\intercal \mathbf{x}_n - b) - 1 + \xi_n] + C \sum_n \xi_n$, where $(\forall n, \alpha_n \in [0, C])$. This objective function is independent of features. For further derivation, we will make an assumption that the equality $\mathbf{z}_n = \mathbf{W}\mathbf{x}_n$ holds, which means that the learned features approximate the projection of data by $\mathbf{W}$, and this approximation becomes tighter when $Rank(\mathbf{X})$ approaches K. Recalling that $\mathbf{W}\mathbf{W}^\intercal = \mathbf{I}$, the dual forms of the two different SVMs share an equal term $\sum_n \alpha_n - \frac{1}{2} \| \sum_n \alpha_n y_n \mathbf{x}_n \|^2 = \sum_n \alpha_n - \frac{1}{2} \| \sum_n \alpha_n y_n \mathbf{z}_n \|^2$, so that the two dual solutions are actually the same. The primal solutions of two SVMs $\mathbf{W}\boldsymbol{\eta} = \sum_n \alpha_n y_n \mathbf{W}\mathbf{x}_n = \sum_n \alpha_n y_n \mathbf{z}_n = \boldsymbol{\nu}$ are related through $\mathbf{W}$. Recall that $\mathbf{z}_n = \mathbf{W}(\mathbf{x}_n - \mathbf{U}_n) + \alpha_n y_n \boldsymbol{\nu}$, we get $\mathbf{z}_n = \mathbf{W}(\mathcal{T}[\mathbf{x}_n] - \mathbf{U}_n)$, where $\mathcal{T}[\mathbf{x}_n] = \mathbf{x}_n + \alpha_n y_n \boldsymbol{\eta}$ is a binary-labeled and $\sigma^2 = 1$ case of MMDS. According to our previous proof, we have $\max_{\boldsymbol{\alpha}} \min_{\boldsymbol{\nu}} F(\boldsymbol{\alpha}, \boldsymbol{\nu}, \mathbf{Z}) = \max_{\boldsymbol{\alpha}} \min_{\boldsymbol{\eta}} F(\boldsymbol{\alpha}, \boldsymbol{\eta}, \mathbf{X})$ under the premise of $\mathbf{z}_n = \mathbf{W}\mathbf{x}_n$. Substituting it back, we get the dual form of the objective

$$\mathrm{D}[1] : \min_{\mathbf{M} \in \mathcal{M}} \max_{\mathbf{U}, \boldsymbol{\alpha}} \min_{\boldsymbol{\eta}} F(\boldsymbol{\alpha}, \boldsymbol{\eta}, \mathbf{X}) + L(\mathcal{T}[\mathbf{X}], \mathbf{M}, \mathbf{U}).$$

Since the SVM part $F(\boldsymbol{\alpha}, \boldsymbol{\eta}, X)$ does not affect the optimum of other parameters, $\mathbf{M}$ and $\mathbf{W}$ are obtained in a similar manner as before.

For MMDS-PCA, we have the following learning problem:

$$\mathrm{D}[2] : \min_{\mathbf{M} \in \mathcal{M}} \max_{\mathbf{U}} L(\mathcal{T}[\mathbf{X}], \mathbf{M}, \mathbf{U}). \tag{15}$$

By comparing with D[1], the only modification that affects $\mathbf{Z}$ and $\mathbf{W}$ is changing $\mathbf{x}_n$ to $\mathcal{T}[\mathbf{x}_n]$, which implies that the MMPCA is approximated to MMDS-PCA, where the approximation precision depends on the low rank property of $\mathbf{X}$ and the scale of data shifting.

Analysis on Principal Components

Based on the above analysis on the relation between MM-PCA and MMDS-PCA, it is obvious that data shifting can enlarge the feature margin. Now we analyze how the principal components are affected. Suppose that input data are zero-centered after preprocessing, their covariance matrix is decomposed as $\mathcal{S}(\mathbf{X}) = \frac{1}{N} \sum_n \mathbf{x}_n \mathbf{x}_n^\intercal = \mathbf{V}\boldsymbol{\Omega}\mathbf{V}^\intercal$, where $\mathbf{V}^\intercal \mathbf{V} = I$, and $\boldsymbol{\Omega} = \operatorname{diag}[l_1, \cdots, l_D]$ arranges the eigenvalues in a descending order. Recall that in binary case we have $\boldsymbol{\eta} = \sum_n \alpha_n y_n \mathbf{x}_n$. The covariance matrix of MMDS data is $\mathcal{S}(\mathcal{T}[\mathbf{X}]) = \mathbf{V}\boldsymbol{\Omega}\mathbf{V}^\intercal + \alpha_0 \boldsymbol{\eta}\boldsymbol{\eta}^\intercal / N$, where $\alpha_0 = \sum_n \alpha_n^2 + 2$. The equation proves that MMDS-PCA equals to using SVM coefficient as one training data. Denoting $\boldsymbol{\gamma} = \mathbf{V}^\intercal \boldsymbol{\eta} \in \mathbb{R}^D$, we have $\boldsymbol{\eta} = \mathbf{V}\mathbf{V}^\intercal \boldsymbol{\eta} = \mathbf{V}\boldsymbol{\gamma}$, so $\boldsymbol{\eta}\boldsymbol{\eta}^\intercal = (\Sigma_d \gamma_d \mathbf{V}_d)(\Sigma_d \gamma_d \mathbf{V}_d^\intercal) = \Sigma_{j,k} \gamma_j \gamma_k \mathbf{V}_j \mathbf{V}_k^\intercal$. According to the property of normal distribution, we have $\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \sum_n \alpha_n^2 \boldsymbol{\Phi})$, where $\boldsymbol{\Phi}$ is the covariance matrix of data distribution. $\mathcal{S}(\mathbf{X})$ can be viewed as a sampling approximation of $\boldsymbol{\Phi}$, and (Zhang et al. 2012) proved the tail bounds on sum of N random matrices, $\forall \epsilon \geq 0$,

$$\Pr\{\|\mathcal{S}(\mathbf{X}) - \boldsymbol{\Phi}\|_2 \geq \mathcal{Q}(N, \zeta, \epsilon) \|\boldsymbol{\Phi}\|_2\} \leq 2De^{-\epsilon}, \tag{16}$$

where $\mathcal{Q}(N, \zeta, \epsilon) = \sqrt{2\epsilon\zeta + 1/N} + 2\epsilon\zeta/N$ and $\zeta = \operatorname{tr}(\boldsymbol{\Phi}) / \|\boldsymbol{\Phi}\|_2$. We denote $\bar{\alpha}_0 = \mathbb{E}[\sum_n \alpha_n^2]$ to represent taking expectation w.r.t data distribution. Suppose there is enough amount of data, $\gamma_j \gamma_k (j \neq k)$ will approach zero,

$$\mathbb{E}[\gamma_j \gamma_k] \approx \bar{\alpha}_0 \mathbb{E}[\mathbf{V}_j^\intercal \mathcal{S}(\mathbf{X}) \mathbf{V}_k] = \alpha_0 \mathbb{E}[\mathbf{V}_j^\intercal \mathbf{V}_k] l_k = 0,$$

where the last step comes from the definition of $\mathbf{V}$. Now we can approximate an eigen-decomposition by $\boldsymbol{\eta}\boldsymbol{\eta}^\intercal = \mathbf{V}\boldsymbol{\gamma}\boldsymbol{\gamma}^\intercal\mathbf{V}^\intercal \approx \sum_d \gamma_d^2 \mathbf{V}_d \mathbf{V}_d^\intercal = \mathbf{V}\,\mathrm{diag}[\gamma_1^2, \cdots, \gamma_D^2]\mathbf{V}^\intercal$. Now we have

$$\mathcal{S}(\mathcal{T}[\mathbf{X}]) \approx \mathbf{V}\boldsymbol{\Omega}\mathbf{V}^\intercal + \alpha_0 \boldsymbol{\eta}\boldsymbol{\eta}^\intercal \triangleq \mathbf{V}\bar{\boldsymbol{\Omega}}\mathbf{V}^\intercal, \qquad (17)$$

where $\bar{\boldsymbol{\Omega}} = \mathrm{diag}[l_1 + \alpha_0 \gamma_1^2, \cdots, l_D + \alpha_0 \gamma_D^2]$. Since $\forall d, \|\mathbf{V}_d\|^2 = 1$, so $\gamma_d = \boldsymbol{\eta}^\intercal \mathbf{V}_d = \rho(\boldsymbol{\eta}, \mathbf{V}_d)\|\boldsymbol{\eta}\|$, where $\rho(\cdot, \cdot)$ is the correlation coefficient. This means that γ_d measures the linear correlation between the eigenvector $\mathbf{V}_d$ and SVM coefficients $\boldsymbol{\eta}$.

Recall that PCA chooses principal components according to eigen-values. As data shifting affects the descending order of eigenvalues of data covariance, the components which are more correlated with $\boldsymbol{\eta}$ will have a higher possibility of being selected as principal ones. Then the projection is influenced, especially when we reduce dimension heavily. This result concords with our intuition, because the multiclass SVM can be viewed as a discriminative projection to a $|\mathcal{Y}|$ dimensional subspace. To conclude, SVM supervises PCA to obtain a balance between data reconstruction and linear separability.

The calculation of covariance matrix needs $O(ND^2)$ computation, and the eigen-decomposition step consumes $O(D^3)$ computation, with stochastic SVM solver taking $O(N)$. When the data size becomes relatively large, the total complexity is approximately $O(ND^2)$. In contrast, all of the previous methods are iterative algorithms, and the per-iteration complexity is $O(NDK^4)$ for BMMPCA (Du et al. 2015), $O(NDK)$ (primal form) or $O(N^2)$ (dual form) time for SPPCA (Yu et al. 2006) and $O(N^2)$ for SEPCA (Guo 2009), which are generally slower than our method.

Experiments

We implement our data shifting phase multiclass SVM using the stochastic minibatch Frank-Wolfe algorithm (Lacoste-Julien et al. 2014) with a fixed maximum number of iterations. For the final classification phase, we use the *LibLinear* package (Fan et al. 2008) to learn multiclass SVMs. To show the effect of data shifting operation, Figure 1 visualizes a 2 dimensional embedding of the Digits-HOG data in 4 classes, which will be introduced later. The two figures show projections by 2-dimensional PCA and MMDS-PCA respectively. We can see that the features extracted from MMDS data are more separable.

Real World Datasets with PCA

We tested MMDS-PCA with eigen-decomposition solvers on multiple real world datasets on various tasks, including face recognition, tumor diagnosis and video retrieval. The **Yale** dataset contains 165 images of 15 individuals. The **YaleB** (the extended Yale Face Database B) dataset includes 38 individuals and about 64 near frontal face images. The **ORL** dataset contains 10 different varying lighting and facial detail images for each of 40 distinct subjects. All the faces are manually aligned, cropped and resized to 32*32 pixels. The **11 Tumor** dataset contains 174 gene samples of 11 different class, documented as 12,522 dimensional vectors. The **TRECVID2003** dataset contains 1078 video shots of 5 categories, documented as 165-dim HSV color histogram, and we evenly split them to training and testing set. The number of training samples per class is 2 for ORL, 3 for Yale, 11 for tumor, and 5 for YaleB.

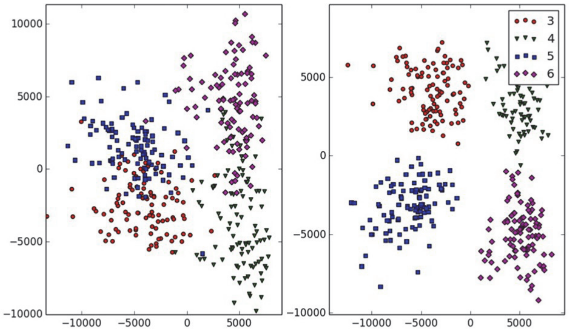

Figure 2: 2D-embedding of the digit images processed by (left) PCA and (right) MMDS-PCA.

Table 1: Classification Accuracy (%) on various datasets.

	Yale	Yale B	ORL	11 tumor
PCA	55.8	12.9	51.6	67.6
LDA	37.1	15.7	28.6	28.6
SPPCA	51.6	9.8	61.7	63.0
SDR-GLM	58.8	19.0	-	63.5
SEPCA	64.4	20.5	-	**88.9**
MMDS-PCA	**70.1**	**33.3**	**73.5**	73.3

We compare with a wide range of state-of-the-art supervised feature extractors, including SPPCA (Yu et al. 2006), SEPCA (Guo 2009), supervised dimensionality reduction with generalized linear models (SDR-GLM) (Rish et al. 2008), large-margin Harmonium (MMH) (Chen et al. 2012), and infinite latent SVM (iLSVM) (Zhu, Chen, and Xing 2014); and two baseline methods—PCA and linear discriminant analysis (LDA). For all models, the data are projected into 10-dimensional space (K=10). We use 5 (or the number of minimal category) folds cross-validation to find proper parameters.

Table 1 and Table 2 present the accuracy of different models on various datasets. We can see that our easily-implemented MMDS-PCA outperforms most state-of-art supervised models on many datasets, except for one case in 11 tumor, which is very suitable for exponential family PCA (Guo 2009) to fit.

Generalization Ability and Parameter Sensitivity

We randomly choose 500 samples from the **MNIST** dataset, and use 10,000 samples for testing. The **Digits** dataset is within OpenCV. We extract 64 dimensional HOG features (Dalal and Triggs 2005) of 5,000 digits images and evenly split them to train/test sets. The **Letters** dataset (Ben, Carlos,

Table 2: Classification Accuracy (%) on the TRECVID

	EFH	MMH	iLSVM	MMDS-PCA
TRECVID	56.5	56.6	56.3	**59.74**

Table 3: Recognition accuracy (%) of different dimension reduction methods using the *RFF* embedding of various datasets and their MMDS versions.

Data\Model	ICA	SPCA	KM	FA	NMF
MNIST	57.8	30.3	48.1	64.9	38.3
MNIST-MMDS	63.7	48.8	61.8	74.4	47.6
Digits	62.8	63.5	63.2	70.0	28.4
Digits-MMDS	81.6	75.5	71.2	86.1	72.1
Letter	31.6	26.9	17.9	38.5	31.0
Letter-MMDS	39.7	38.2	46.3	52.8	38.2

and Daphne 2004) contains 26 kinds of Latin characters, and we use 5,375 samples for training while the other 46,777 for testing. For these data, we calculate their *RFF* embeddings with a RBF kernel $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2/\sigma^2)$, σ^2 is determined during experiments, and the dimensions of *RFF* is set to 500 for Digits and 1,000 for the other datasets.

We test the generalization ability of MMDS using multiple feature extractors, including Independent Components Analysis (ICA), Sparse Principal Components Analysis (SPCA) (Zou, Hastie, and Tibshirani 2006), Kmeans (KM), Factor Analysis (FA), and Non-negative Matrix Factorization (NMF) (Lee and Seung 2001). We kernelize them by using *RFF* embeddings under the previously mentioned settings. We extract 5-dim features using all of these models. Table 3 presents the accuracy using either the original data or the one processed by MMDS. Even these models are not easy to develop supervised versions, our results demonstrate that they can consistently benefit from MMDS, and most improvements are significant.

We tested handwritten symbols recognition tasks using PCA, Autoencoder (AE), Kernel PCA (KPCA) (Lopez-Paz et al. 2014), on the MNIST, Digits, and Letters datasets to study the parameter sensitivity. We use PCA with a standard eigen-decomposition solver. We implemented the autoencoder, which optimizes

$$\frac{1}{N}\sum_n \|\mathbf{x}_n - \varphi(\mathbf{W}^\intercal\varphi(\mathbf{W}\mathbf{x}_n + \mathbf{b}_1) + \mathbf{b}_2))\|^2 \qquad (18)$$

w.r.t. $\mathbf{W}, \mathbf{b}_1, \mathbf{b}_2$, where φ is the sigmoid function. We use an L-BFGS solver with a fixed maximum number of iterations, and normalize the data to $(0.1, 0.9)$ before sending to the autoencoder. We implemented KPCA using *RFF* embeddings in the previous setting. Then we use the same classifier setting for every model pairs that use either the original data or the shifted data.

Figure 3 presents the performance of various extractors under different shifting scales (i.e., σ^2 in MMDS), where the accuracy and reconstruction errors are normalized. The tendency shows that as the shifting scale gets larger, the reconstruction error stably increases and the accuracy will first increase and then decrease. Such results suggest that the discriminative MMDS may reduce the fitness on the observed data, and a suitable scale of shifting operation is helpful for extracting discriminative features; but an arbitrarily large shifting may not well fit the input data.

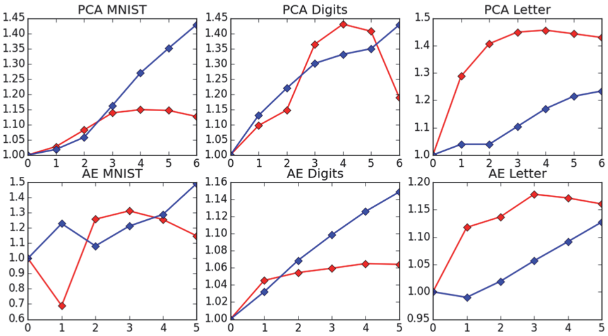

Figure 3: Sensitivity of MM-models: X-axis, shifting scale σ^2; Red, classification accuracy; Blue, reconstruction error. All the statistics are normalized for visualization.

Finally, Table 4 presents the performance under different settings of feature dimensions (K). We can see that the improvement brought by MMDS over the original data is substantial. The selection of discriminative components is very important, especially under low dimensional conditions, whereas the information is more compressed.

Table 4: Accuracy w.r.t changing feature dimensions (%). In each column, the left number is the accuracy of fitting the original data, while the right number is the accuracy of using the MMDS data.

Data (K)\Model	PCA		AE		KPCA	
MNIST(5)	64.4	69.7	43.8	49.7	64.4	73.7
MNIST(10)	75.3	79.2	58.1	76.3	74.7	81.7
MNIST(20)	81.5	83.0	77.7	81.6	81.0	83.1
Digits(5)	58.3	79.4	73.5	79.0	70.8	83.8
Digits(10)	79.6	89.1	87.4	93.0	85.8	94.0
Digits(20)	86.8	90.2	92.7	93.6	91.9	94.3
Letters(5)	35.1	51.1	35.4	42.2	38.4	53.3
Letters(10)	53.6	62.9	50.7	58.8	53.4	65.0
Letters(20)	65.5	69.7	61.2	65.8	62.9	71.9

Conclusions

We present a simple *max-margin data shifting* (MMDS) operator to learn discriminative features using unsupervised extractors. MMDS can be approximately derived as the mean of a supervised corrupting model that considers both the standard data noise and the discriminative ability of a potential corruption according to a pre-trained large-margin classifier. Our experiments on various datasets show that MMDS can help a wide family of models on learning features that are suitable for classification tasks.

Acknowledgments

The work was supported by the National Basic Research Program (973 Program) of China (Nos. 2013CB329403, 2012CB316301), National NSF of China (Nos. 61305066, 61322308, 61332007), TNList Big Data Initiative, and Tsinghua Initiative Scientific Research Program (Nos. 20121088071, 20141080934).

References

Ben, T.; Carlos, G.; and Daphne, K. 2004. Max-margin markov networks. *Advances in Neural Information Processing Systems* 16:25.

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

Bishop, C. M. 1995. Training with noise is equivalent to tikhonov regularization. *Neural computation* 7(1):108–116.

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research* 3:993–1022.

Borwein, J. M., and Lewis, A. S. 2010. *Convex analysis and nonlinear optimization: theory and examples*, volume 3. Springer Science & Business Media.

Chen, N.; Zhu, J.; Sun, F.; and Xing, E. P. 2012. Large-margin predictive latent subspace learning for multiview data analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*.

Chen, M.; Weinberger, K. Q.; Sha, F.; and Bengio, Y. 2014. Marginalized denoising auto-encoders for nonlinear representations. In *ICML*. ACM.

Chen, Z.; Chen, M.; Weinberger, K. Q.; and Zhang, W. 2015. Marginalized denoising for link prediction and multi-label learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Collins, M.; Dasgupta, S.; and Schapire, R. E. 2001. A generalization of principal components analysis to the exponential family. In *Advances in Neural Information Processing Systems*, 617–624.

Crammer, K., and Singer, Y. 2002. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research* 2:265–292.

Dalal, N., and Triggs, B. 2005. Histograms of oriented gradients for human detection. In *CVPR 2005*. IEEE.

Du, C.; Zhe, S.; Zhuang, F.; Qi, Y.; He, Q.; and Shi, Z. 2015. Bayesian maximum margin principal component analysis. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; and Lin, C.-J. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research* 9:1871–1874.

Guo, Y. 2009. Supervised exponential family principal component analysis via convex optimization. In *Advances in Neural Information Processing Systems*, 569–576.

Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. R. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*.

Lacoste-Julien, S.; Jaggi, M.; Schmidt, M.; and Pletscher, P. 2014. Block-coordinate frank-wolfe optimization for structural svms. In *ICML*. ACM.

Lee, D. D., and Seung, H. S. 2001. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, 556–562.

Lopez-Paz, D.; DE, M.; Sra, S.; Ghahramani, Z.; and Schölkopf, B. 2014. Randomized nonlinear component analysis. In *ICML*. ACM.

Maaten, L.; Chen, M.; Tyree, S.; and Weinberger, K. 2013. Learning with marginalized corrupted features. In *ICML*. ACM.

Mairal, J.; Ponce, J.; Sapiro, G.; Zisserman, A.; and Bach, F. R. 2009. Supervised dictionary learning. In *Advances in Neural Information Processing Systems*, 1033–1040.

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

Pereira, F., and Gordon, G. 2006. The support vector decomposition machine. In *ICML*. ACM.

Rahimi, A., and Recht, B. 2009. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*.

Rish, I.; Grabarnik, G.; Cecchi, G.; Pereira, F.; and Gordon, G. J. 2008. Closed-form supervised dimensionality reduction with generalized linear models. In *ICML*. ACM.

Vincent, P.; Larochelle, H.; Bengio, Y.; and Manzagol, P.-A. 2008. Extracting and composing robust features with denoising autoencoders. In *ICML*. ACM.

Yu, S.; Yu, K.; Tresp, V.; Kriegel, H.-P.; and Wu, M. 2006. Supervised probabilistic principal component analysis. In *ACM SIGKDD*, 464–473.

Zhang, L.; Mahdavi, M.; Jin, R.; Yang, T.; and Zhu, S. 2012. Recovering the optimal solution by dual random projection. *arXiv preprint arXiv:1211.3046*.

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

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

Zou, H.; Hastie, T.; and Tibshirani, R. 2006. Sparse principal component analysis. *Journal of computational and graphical statistics* 15(2):265–286.