Online and Batch Learning of Pseudo-Metrics

Shai Shalev-Shwartz SHAIS@CS.HUJI.AC.IL
School of Computer Science & Engineering, The Hebrew University

Yoram Singer SINGER@CS.HUJI.AC.IL
School of Computer Science & Engineering, The Hebrew University

Andrew Y. Ng ANG@CS.STANFORD.EDU
Computer Science Department, Stanford University

Abstract

We describe and analyze an online algorithm for supervised learning of pseudo-metrics. The algorithm receives pairs of instances and predicts their similarity according to a pseudo-metric. The pseudo-metrics we use are quadratic forms parameterized by positive semi-definite matrices. The core of the algorithm is an update rule that is based on successive projections onto the positive semi-definite cone and onto half-space constraints imposed by the examples. We describe an efficient procedure for performing these projections, derive a worst case mistake bound on the similarity predictions, and discuss a dual version of the algorithm in which it is simple to incorporate kernel operators. The online algorithm also serves as a building block for deriving a large-margin batch algorithm. We demonstrate the merits of the proposed approach by conducting experiments on MNIST dataset and on document filtering.

1. Introduction

Many problems in machine learning and statistics require the access to a metric over instances. For example, the performance of the nearest neighbor algorithm (Cover & Hart, 1967), multi-dimensional scaling (Cox & Cox, 1994) and clustering algorithms such as K-means (MacQueen, 1965), all depend critically on whether the metric they are given truly reflects the underlying relationships between the input instances. Several recent papers have focused on the problem of automatically learning a distance function from examples (Xing et al., 2003; Shental et al., 2002). These papers have focused on batch learning algorithms. A batch algorithm for learning a distance function is provided with a predefined set of examples. Each example consists of two instances and a binary label indicating whether the two instances are similar or dissimilar. The work of (Xing et al., 2003; Shental et al., 2002) used various techniques that are effective in batch settings, but do not have natural, computationally-efficient online versions. Furthermore, these algorithms did not come with any theoretical error guarantees. In this paper, we discuss, analyze, and experiment with an *online* algorithm for learning pseudo-metrics. As in a batch setting, we receive pairs of instances which may be similar or dissimilar. But in contrast to batch learning, in the online setting we need to extend a prediction on each pair as it is received. After predicting whether the current pair of instances is similar, we receive the correct feedback on the instances' similarity or dissimilarity. Informally, the goal of the online algorithm is to minimize the number of prediction errors. Online learning algorithms enjoy several practical and theoretical advantages: They are often simple to implement; they are typically both memory and run-time efficient; they often come with formal guarantees in the form of worst case bounds on their performance; there exist several methods for converting from online to batch learning, which come with formal guarantees on the batch algorithm obtained through the conversion. Moreover, there are applications such as text filtering in which the set of examples is indeed not given all at once, but instead revealed in a sequential manner while predictions are requested on-the-fly.

The online algorithm we suggest incrementally learns a pseudo-metric and a threshold. As in (Xing et al., 2003), the pseudo-metrics we use are quadratic forms parametrized by positive semi-definite (PSD) matrices. At each time step, we get a pair of instances and calculate the distance between them according to our current pseudo-metric. We decide that the instances are similar if this distance is less than the current threshold and otherwise we say that the instances are dissimilar. After extending our prediction, we get the true similarity label of the pair of in-

Appearing in *Proceedings of the 21^{st} International Conference on Machine Learning*, Banff, Canada, 2004.

stances and update our pseudo-metric and threshold. Our update rule is based on the projection operation. Intuitively, we look for a new pseudo-metric and threshold that on the one hand will predict correctly the last example we have just received and on the other hand will be as close as possible to the previous pseudo-metric and threshold. The idea of using the projection operation for online algorithms was first introduced by (Herbster, 2001), and was further developed by (Crammer et al., 2003). The resulting update rule enjoys some nice properties. First, the PSD matrix we learn is a linear combination of rank-one matrices defined by vectors in the span of the instances. This allows us to develop a dual version of the algorithm that employs kernels. Further, we show that all the PSD matrices obtained by the online algorithm are norm bounded. We use this property to prove an online error bound, and to design a large-margin batch algorithm based on the online algorithm.

This paper is organized as follows. Sec. 2 formally introduces the problem of online learning of pseudo-metrics and sets the notation used throughout the paper. In Sec. 3, we describe our pseudo-metric learning algorithm for the separable case and show that the resulting online algorithm can be implicitly implemented using kernel operators. In Sec. 4, we derive a worst-case loss bound for the online algorithm. A modification of the online algorithm to the inseparable case and a corresponding loss bound is briefly discussed in Sec. 5. In Sec. 6, we describe a simple online to batch learning conversion, and discuss the generalization properties of resulting batch algorithm. Experimental results are provided in Sec. 7. The experiments apply our algorithm to the tasks of digit recognition and online document filtering. We compare the performance of our algorithm to both other batch similarity learning algorithms and online algorithms for classification.

2. Problem Setting

Let $\mathcal{X}$ denote our feature space. For concreteness we assume that $\mathcal{X} = \mathbb{R}^n$. Our goal is to learn a pseudo-metric over $\mathcal{X}$. A pseudo-metric is a function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which needs to satisfy three requirements, (i) $d(\mathbf{x}, \mathbf{x}') \geq 0$, (ii) $d(\mathbf{x}, \mathbf{x}') = d(\mathbf{x}', \mathbf{x})$, and (iii) $d(\mathbf{x}_1, \mathbf{x}_2) + d(\mathbf{x}_2, \mathbf{x}_3) \leq d(\mathbf{x}_1, \mathbf{x}_3)$. While the instances may belong to a well defined partition of $\mathcal{X}$ into classes, we do not receive direct supervision in the form of class labels. Instead, we get similarity and dissimilarity feedback. Therefore, we assume that we receive examples of the form $\mathbf{z} = (\mathbf{x}, \mathbf{x}', y) \in (\mathcal{X} \times \mathcal{X} \times \{+1, -1\})$. Each example is composed of an instance pair $(\mathbf{x}, \mathbf{x}')$ and a label y which equals $+1$ if $\mathbf{x}$ and $\mathbf{x}'$ are considered similar and -1 otherwise. As in (Xing et al., 2003), we restrict ourselves to pseudo-metrics of the form

$$d_A(\mathbf{x}, \mathbf{x}') \equiv \sqrt{(\mathbf{x} - \mathbf{x}')^t A (\mathbf{x} - \mathbf{x}')} \ ,$$

where $A \succeq 0$ is a symmetric positive semi-definite matrix. It is easy to verify that if $A \succeq 0$ then d_A is indeed a pseudo-metric. Furthermore, there exists a matrix W such that

$$(\mathbf{x} - \mathbf{x}')^t A (\mathbf{x} - \mathbf{x}') = \|W\mathbf{x} - W\mathbf{x}'\|_2^2 \ .$$

Therefore, $d_A(\mathbf{x}, \mathbf{x}')$ is the Euclidean distance between the image of $\mathbf{x}$ and $\mathbf{x}'$ due to a linear transformation W.

The margin of a sample S, denoted γ, is defined to be the minimum separation between all pairs of similar and dissimilar examples. Let $(\mathbf{x}_1, \mathbf{x}'_1, +1)$ and $(\mathbf{x}_2, \mathbf{x}'_2, -1)$ be such a pair. Then, the margin requirement translates to

$$(d_A(\mathbf{x}_1, \mathbf{x}'_1))^2 \leq (d_A(\mathbf{x}_2, \mathbf{x}'_2))^2 - \gamma \ . \tag{1}$$

Note that we can scale A and γ by any positive constant factor without essentially modifying the properties of the solution (as in the case of many classification problems). We therefore set γ to be 2 and later on look for a matrix A which has a small norm. If we get a sample S of m tuples of the form $(\mathbf{x}, \mathbf{x}', y)$ there are, however, $O(m^2)$ constraints of the form described by Eq. (1). Thus, we introduce a threshold $b \in \mathbb{R}$ and replace the above constraints with the following set of constraints,

$$\begin{aligned} \forall (\mathbf{x}, \mathbf{x}', y) : y = +1 \quad &\Rightarrow \quad (d(\mathbf{x}, \mathbf{x}'))^2 \leq b - 1 \ , \\ \forall (\mathbf{x}, \mathbf{x}', y) : y = -1 \quad &\Rightarrow \quad (d(\mathbf{x}, \mathbf{x}'))^2 \geq b + 1 \ , \end{aligned}$$

which can be written as a single linear constraint as follows,

$$y \left(b - (d_A(\mathbf{x}, \mathbf{x}'))^2 \right) \geq 1 \ . \tag{2}$$

Given a set of examples we can now define a constrained optimization problem to find A. Note that in addition to the constraint defined in Eq. (2), we also need to impose the constraint that A must be positive semi-definite (PSD). Solving this constrained optimization problem can be performed by standard methods, such as interior-point algorithms for solving semi-definite programs. In this paper we focus instead on a simple and efficient online approach, and later use well-studied techniques for converting from online to batch learning algorithms. We thus obtain the best of both worlds: a loss bound for an efficient online algorithm, and a generalization bound for the resulting batch algorithm.

In the online setting we observe tuples $(\mathbf{x}_\tau, \mathbf{x}'_\tau, y_\tau)$ in a sequential manner. On time step τ we first observe $(\mathbf{x}_\tau, \mathbf{x}'_\tau)$, and calculate $d_A(\mathbf{x}_\tau, \mathbf{x}'_\tau)$. If the square of $d_A(\mathbf{x}_\tau, \mathbf{x}'_\tau)$ is greater than the threshold b we predict that the pair is dissimilar. Otherwise, we say that the pair is similar. After extending the prediction, we receive the true label y_τ and may suffer a loss if there is a discrepancy between our prediction and y_τ. The loss we discuss in this paper is an adaptation of the hinge loss,

$$\ell_\tau(A, b) \doteq \max \left\{ 0 \ , \ y_\tau \left((d_A(\mathbf{x}_\tau, \mathbf{x}'_\tau))^2 - b \right) + 1 \right\} \ .$$

Thus, if we satisfy the inequality in Eq. (2) we suffer no loss. Otherwise, we pay a cost that grows linearly with the amount the inequality is violated. The goal of the online algorithm is to minimize the *cumulative* loss it suffers. As in other online algorithms the matrix A and the threshold b are updated after receiving the feedback y_τ. Therefore, we denote by (A_τ, b_τ) the matrix-threshold pair used for prediction on round τ.

3. An Online Algorithm

We now present our first algorithm, which assumes that there exists a matrix $A^\star \succeq 0$ and a scalar $b^\star \geq 1$ that perfectly separates the data. Namely, we assume that $\ell_\tau(A^\star, b^\star) = 0$ for all τ. A modification of the algorithm for the inseparable case is given in Sec. 5.

The general method we use for deriving our on-line update rule is based on the orthogonal projection operation. Formally, given a vector $\mathbf{x} \in \mathbb{R}^k$ and a closed convex set $C \subset \mathbb{R}^k$, the orthogonal projection of $\mathbf{x}$ onto C is defined by

$$\mathcal{P}_C(\mathbf{x}) = \operatorname*{argmin}_{\mathbf{x}' \in C} \|\mathbf{x} - \mathbf{x}'\|_2^2 \ .$$

In words, $\mathcal{P}_C(\mathbf{x})$ is the vector in C that is closest to $\mathbf{x}$.

For simplicity of presentation, we refer to (A, b) both as a matrix-scalar pair and as a vector in $\mathbb{R}^{n^2+1}$ where the first n^2 elements of the vector are the elements of A (listed column-wise) and the last entry of the vector is b. For each time step τ, we define the set $C_\tau \subset \mathbb{R}^{n^2+1}$ as

$$C_\tau = \left\{ (A, b) \in \mathbb{R}^{n^2+1} \ : \ \ell_\tau(A, b) = 0 \right\} \ . \qquad (3)$$

Thus, C_τ is the set of all matrix-threshold pairs which attain zero loss on the example $(\mathbf{x}_\tau, \mathbf{x}'_\tau, y_\tau)$. Recall that a necessary condition imposed on a matrix A used as a pseudo-metric is that $A \succeq 0$. In addition, the threshold must be at least 1 (otherwise the loss on any similar points will be non-zero). Thus, we denote by C_a the set of all admissible matrix-threshold pairs,

$$C_a = \{(A, b) \in \mathbb{R}^{n^2+1} \ : \ A \succeq 0, \ b \geq 1\} \ .$$

Equipped with the above definitions, we now describe the update step of the online algorithm. The update is comprised of two projections. First we project the current matrix-threshold pair (A_τ, b_τ) onto C_τ. Let $(A_{\hat{\tau}}, b_{\hat{\tau}}) = \mathcal{P}_{C_\tau}(A_\tau, b_\tau)$ be the resulting matrix-threshold pair. In words, we attempt to keep $(A_{\hat{\tau}}, b_{\hat{\tau}})$ as close to (A_τ, b_τ) as possible, while forcing $(A_{\hat{\tau}}, b_{\hat{\tau}})$ to achieve a zero loss on the most recent example. We then define the new matrix-threshold pair $(A_{\tau+1}, b_{\tau+1})$ as the projection of $(A_{\hat{\tau}}, b_{\hat{\tau}})$ onto the set C_a, thus ensuring that $(A_{\tau+1}, b_{\tau+1})$ is admissible for deciding whether two instances $\mathbf{x}, \mathbf{x}'$ are similar or dissimilar. In summary, the update rule of our online algorithm is composed of two successive projections,

1. $(A_{\hat{\tau}}, b_{\hat{\tau}}) = \mathcal{P}_{C_\tau}(A_\tau, b_\tau)$,
2. $(A_{\tau+1}, b_{\tau+1}) = \mathcal{P}_{C_a}(A_{\hat{\tau}}, b_{\hat{\tau}})$.

In the following, we show how to efficiently perform these projections.

3.1. Projecting onto C_τ

Recall that we refer to (A, b) both as a matrix-scalar pair and as a vector in $\mathbb{R}^{n^2+1}$. For the simplicity of representation, we denote by $\mathbf{w} \in \mathbb{R}^{n^2+1}$ the vector representation of (A, b). Analogously, $\mathbf{w}_\tau, \mathbf{w}_{\hat{\tau}}, \mathbf{w}_{\tau+1}$ denote the vectors corresponding to $(A_\tau, b_\tau), (A_{\hat{\tau}}, b_{\hat{\tau}}), (A_{\tau+1}, b_{\tau+1})$. In addition, let $\boldsymbol{\chi}_\tau \in \mathbb{R}^{n^2+1}$ be the vector corresponding to the matrix-scalar pair $(-y_\tau \mathbf{v}_\tau \mathbf{v}_\tau^t, y_\tau)$, where $\mathbf{v}_\tau = \mathbf{x}_\tau - \mathbf{x}'_\tau$. Using the above terminology along with simple algebraic manipulations, we can rewrite the definition of C_τ from Eq. (3) as $C_\tau = \{\mathbf{w} \in \mathbb{R}^{n^2+1} \ : \ \mathbf{w} \cdot \boldsymbol{\chi}_\tau \geq 1\}$. It is easy to verify that the projection of $\mathbf{w}_\tau$ onto C_τ is given by $\mathcal{P}_{C_\tau}(\mathbf{w}_\tau) = \mathbf{w}_\tau + \alpha_\tau \boldsymbol{\chi}_\tau$ where $\alpha_\tau = 0$ if $\mathbf{w}_\tau \cdot \boldsymbol{\chi}_\tau \geq 1$ and otherwise $\alpha_\tau = (1 - \mathbf{w}_\tau \cdot \boldsymbol{\chi}_\tau) / \|\boldsymbol{\chi}_\tau\|_2^2$.

We now use the fact that $\mathbf{w}_\tau$ and $\boldsymbol{\chi}_\tau$ are the vectors corresponding to the matrix-scalar pairs (A_τ, b_τ) and $(-y_\tau \mathbf{v}_\tau \mathbf{v}_\tau^t, y_\tau)$. Therefore, we get that,

$$\alpha_\tau = \frac{\ell_\tau(A_\tau, b_\tau)}{\|\boldsymbol{\chi}_\tau\|_2^2} = \frac{\ell_\tau(A_\tau, b_\tau)}{\|\mathbf{v}_\tau\|_2^4 + 1} \ ,$$

and the update becomes

$$A_{\hat{\tau}} = A_\tau - y_\tau \alpha_t \mathbf{v}_\tau \mathbf{v}_\tau^t \quad , \quad b_{\hat{\tau}} = b_\tau + \alpha_t y_\tau \ . \qquad (4)$$

3.2. Projecting onto C_a

We now describe an efficient method for projecting $(A_{\hat{\tau}}, b_{\hat{\tau}})$ onto C_a. First note that if $(A_{\tau+1}, b_{\tau+1}) = \mathcal{P}_{C_a}(A_{\hat{\tau}}, b_{\hat{\tau}})$ then $A_{\tau+1}$ is the projection of $A_{\hat{\tau}}$ onto the set of all PSD matrices and $b_{\tau+1}$ is the projection of $b_{\hat{\tau}}$ onto the set $\{b \in \mathbb{R} : b \geq 1\}$. The projection of $b_{\hat{\tau}}$ onto the above set is $\max\{1, b_{\hat{\tau}}\}$. It remains to show how to project $A_{\hat{\tau}}$ onto the set of all PSD matrices.

We start with the case $y_\tau = -1$. In this case $A_{\hat{\tau}} = A_\tau + \alpha_t \mathbf{v}_\tau \mathbf{v}_\tau^t$ where $\alpha_t \geq 0$ and hence $A_{\hat{\tau}} \succeq 0$. Therefore, the projection of $A_{\hat{\tau}}$ onto the set of the PSD matrices is $A_{\hat{\tau}}$ itself. However, if $y_\tau = 1$ $A_{\hat{\tau}}$ might not be positive semi-definite. Since $A_{\hat{\tau}}$ is symmetric, we can rewrite $A_{\hat{\tau}}$ as $A_{\hat{\tau}} = \sum_{i=1}^n \lambda_i \mathbf{u}_i \mathbf{u}_i^t$, where λ_i is the i'th eigenvalue of $A_{\hat{\tau}}$ and $\mathbf{u}_i$ is its corresponding eigenvector. Without loss of generality, we assume that $\lambda_1 \geq \ldots \geq \lambda_n$ and that the eigenvectors $\{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$ form an orthonormal basis of $\mathbb{R}^n$. The matrix $A_{\tau+1}$ is the projection of $A_{\hat{\tau}}$ onto the

Initialize: Set $A_1 = 0$; $b_1 \in \mathbb{R}$
For $\tau = 1, 2, \ldots$

Get a pair of instances: $(\mathbf{x}_\tau, \mathbf{x}'_\tau) \in \mathbb{R}^n \times \mathbb{R}^n$
Predict: $\mathbf{x}_\tau, \mathbf{x}'_\tau$ are similar iff $(d_{A_\tau}(\mathbf{x}_\tau, \mathbf{x}'_\tau))^2 \leq b_\tau$
Get the true target $y_\tau \in \{+1, -1\}$
Suffer loss: $\ell_\tau(A_\tau, b_\tau) =$
$\max\left\{0\ ,\ y_\tau\left((d_{A_\tau}(\mathbf{x}_\tau, \mathbf{x}'_\tau))^2 - b_\tau\right) + 1\right\}$
If $(\ell_\tau(A_\tau, b_\tau) > 0)$:
Set $\mathbf{v}_\tau = (\mathbf{x}_\tau - \mathbf{x}'_\tau)$
Set $\alpha_\tau = \frac{\ell_\tau(A_\tau, b_\tau)}{1 + \|\mathbf{v}_\tau\|^4}$
Define $A_{\hat{\tau}} = A_\tau - y_\tau \alpha_\tau \mathbf{v}_\tau \mathbf{v}_\tau^t$; $b_{\hat{\tau}} = b_\tau + y_\tau \alpha_\tau$
If $(y_\tau = 1)$,
Update: $b_{\tau+1} = b_{\hat{\tau}}$
Find $(\lambda_n, \mathbf{u}_n)$ - the minimal eigenvalue of $A_{\hat{\tau}}$ and its corresponding eigenvector
If $(\lambda_n < 0)$,
Update: $A_{\tau+1} = A_{\hat{\tau}} - \lambda_n \mathbf{u}_n \mathbf{u}_n^t$
Else
Update: $A_{\tau+1} = A_{\hat{\tau}}$
Else $[y_\tau = -1]$
Update: $A_{\tau+1} = A_{\hat{\tau}}$; $b_{\hat{\tau}} = \max\{b_{\hat{\tau}}, 1\}$
Else $[\ell_\tau(A_\tau, b_\tau) = 0]$
Update: $A_{\tau+1} = A_\tau$; $b_{\tau+1} = b_\tau$

Figure 1. The pseudo-metric online learning algorithm (POLA).

positive semi-definite cone. Given the set of eigenvectors and eigenvalues of $A_{\hat{\tau}}$, the projection yields that $A_{\tau+1}$ can be written as,

$$A_{\tau+1} = \sum_{i:\lambda_i > 0} \lambda_i \mathbf{u}_i \mathbf{u}_i^t \ .$$

(See for instance Golub & Van Loan, 1989.) In addition, from the (eigenvalue) Interlacing Theorem we have that $A_{\hat{\tau}}$ has at most a *single* negative eigenvalue (cf. Wilkinson, 1965, pp. 94-97 and Golub and Van Loan, 1989, page 412). Therefore, we get that $A_{\tau+1} = A_{\hat{\tau}} - \lambda_n \mathbf{u}_n \mathbf{u}_n^t$. Here, λ_n and $\mathbf{u}_n$ can be calculated efficiently using the Lanczos method (see, e.g., Golub and Van Loan, 1989). We name the resulting algorithm POLA as an abbreviation for Pseudo-metric Online Learning Algorithm. The pseudo-code of POLA is given in Fig. 1.

3.3. Kernel-based Implementation

The pseudo-metrics we have used so far take the form

$$(d_A(\mathbf{x}, \mathbf{x}'))^2 = (\mathbf{x} - \mathbf{x}')^t A (\mathbf{x} - \mathbf{x}') = \|W\mathbf{x} - W\mathbf{x}'\|_2^2 \ ,$$

where $W = \sqrt{A}$ exists since $A \succeq 0$. Therefore, $d_A(\mathbf{x}, \mathbf{x}')$ is the Euclidean distance between the image of $\mathbf{x}$ and $\mathbf{x}'$ due to a *linear* transformation W. In real-world applications, similarity and dissimilarity constraints over instances might not be satisfied by such simple distance functions. A common preprocessing strategy is to use a non-linear mapping function $\phi : \mathcal{X} \to F$ that maps the data into some high dimensional feature space F and then learn in F (Vapnik, 1998). Since F is high-dimensional, we need an efficient way to access the data in F. In this section we present a dual version of the algorithm in Fig. 1, where interface to the data is limited to inner products. Thus, if we have a kernel function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that efficiently computes the inner products in F, $K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}')$, we can efficiently learn a pseudo-metric over F.

To derive a dual version for the online algorithm, we first show that for any time τ, the matrix A_τ can be written as

$$A_\tau = \sum_{i=1}^{m} \beta_i \mathbf{r}_i \mathbf{r}_i^t \ , \tag{5}$$

where $m \leq 2\tau$ and all the vectors $\mathbf{r}_i$ are in the span of the vectors $\{\mathbf{v}_1 = (\phi(\mathbf{x}_1) - \phi(\mathbf{x}'_1)), \ldots, \mathbf{v}_\tau = (\phi(\mathbf{x}_\tau) - \phi(\mathbf{x}'_\tau))\}$, namely,

$$\mathbf{r}_i \ = \ \sum_{j=1}^{\tau} \rho_{j,i} (\phi(\mathbf{x}_j) - \phi(\mathbf{x}'_j)) \ = \ \sum_{j=1}^{\tau} \rho_{j,i} \mathbf{v}_j \ .$$

The above representation of A_τ enables us to efficiently calculate the distance between a new pair of instances using the kernel function, because using Eq. (5), we have:

$$\begin{aligned}
(d_A(\mathbf{x}_{\tau+1}, \mathbf{x}'_{\tau+1}))^2 &= {\mathbf{v}_{\tau+1}}^t A_\tau \mathbf{v}_{\tau+1} \\
&= \sum_{i=1}^{m} \beta_i \left(\mathbf{r}_i \cdot \mathbf{v}_{\tau+1}\right)^2 \\
&= \sum_{i=1}^{m} \beta_i \left(\sum_{j=1}^{\tau} \rho_{j,i} \mathbf{v}_j \cdot \mathbf{v}_{\tau+1}\right)^2 \ .
\end{aligned}$$

In addition, we have that $\mathbf{v}_j \cdot \mathbf{v}_{\tau+1} = K(\mathbf{x}_j, \mathbf{x}_{\tau+1}) - K(\mathbf{x}_j, \mathbf{x}'_{\tau+1}) - K(\mathbf{x}'_j, \mathbf{x}_{\tau+1}) + K(\mathbf{x}'_j, \mathbf{x}'_{\tau+1})$.

We now use an inductive argument to show that A_τ can indeed be written as in Eq. (5). The initial matrix A_1 is the zero matrix and clearly fits the form of Eq. (5). Assume that A_τ is of the form in Eq. (5). The first step of the online update rule is to define $A_{\hat{\tau}} = A_\tau - y_\tau \alpha_\tau \mathbf{v}_\tau \mathbf{v}_\tau^t$. Thus, $A_{\hat{\tau}}$ can also be written as in Eq. (5). If the resulting matrix $A_{\hat{\tau}}$ is positive semi-definite we do not have to do anything. If it does have a (single) negative eigenvalue, we find $(\lambda, \mathbf{u})$, the minimal eigenvalue of $A_{\hat{\tau}}$ and its corresponding eigenvector. We then set $A_{\tau+1} = A_{\hat{\tau}} - \lambda \mathbf{u}\mathbf{u}^t$. It remains to show that $\mathbf{u}$ is also in the span of $\{\mathbf{v}_1, \ldots, \mathbf{v}_\tau\}$. Since $\mathbf{u}$ is an eigenvector of $A_{\hat{\tau}}$ we have that $A_{\hat{\tau}}\mathbf{u} = \lambda \mathbf{u}$. Using the inductive assumption, we rewrite $A_{\hat{\tau}}$ as in Eq. (5) and get that, $(\sum_{i=1}^{m} \beta_i \mathbf{r}_i \mathbf{r}_i^t)\, \mathbf{u} = \lambda \mathbf{u}$, which yields,

$$\mathbf{u} = \sum_{i=1}^{m} \left(\frac{\beta_i (\mathbf{r}_i \cdot \mathbf{u})}{\lambda}\right) \mathbf{r}_i \ . \tag{6}$$

Therefore, $\mathbf{u}$ is in the subspace spanned by $\{\mathbf{r}_1, \ldots, \mathbf{r}_m\}$ and thus it is also in the span of $\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$, which concludes the derivation.

In the rest of this section we explain how to find, via inner-products, the minimal eigenvalue, λ, and its corresponding eigenvector, $\mathbf{u}$, of the matrix $A_{\hat{\tau}}$. Let $Q \in \mathbb{R}^{n \times d}$ be a matrix whose columns form an orthonormal basis for the subspace spanned by $\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ and let $\mathbf{q}_i$ denote the i'th column of Q. From Eq. (6) we get that each eigenvector $\mathbf{u}$ of $A_{\hat{\tau}}$ can be written as a linear combination of the columns of Q and thus there exists a vector $\boldsymbol{\kappa} \in \mathbb{R}^d$ such that $\mathbf{u} = Q\boldsymbol{\kappa}$. Since $\mathbf{u}$ is an eigenvector of $A_{\hat{\tau}}$, we get that

$$A_{\hat{\tau}}\mathbf{u} = \lambda\mathbf{u} \implies A_{\hat{\tau}}Q\boldsymbol{\kappa} = \lambda Q\boldsymbol{\kappa} \ .$$

Multiplying both sides by Q^t we get that $Q^t A_{\hat{\tau}} Q\boldsymbol{\kappa} = \lambda\boldsymbol{\kappa}$. The reverse direction is also correct. Namely, if $\boldsymbol{\kappa}$ is an eigenvector of $Q^t A_{\hat{\tau}} Q$ then $\mathbf{u}$ is an eigenvector of $A_{\hat{\tau}}$ with the same eigenvalue. We have thus shown that $\mathbf{u}$ is an eigenvector of $A_{\hat{\tau}}$ with an eigenvalue λ iff $\mathbf{u} = Q\boldsymbol{\kappa}$ where $\boldsymbol{\kappa}$ is an eigenvector of $Q^t A_{\hat{\tau}} Q$ with the same eigenvalue λ. The matrix $Q^t AQ$ can be computed using inner-products since

$$(Q^t AQ)_{k,j} = \sum_{i=1}^{m} \beta_i (\mathbf{q}_k \cdot \mathbf{r}_i)(\mathbf{q}_j \cdot \mathbf{r}_i) \ .$$

Finally note that $\{\mathbf{q}_1, \ldots, \mathbf{q}_d\}$ can be found implicitly using the kernel Gram-Schmidt procedure. In summary, we have shown that all the steps of the online algorithm in Fig. 1 can be implemented via a kernel function.

4. Analysis

The following theorem provides a loss bound for the algorithm in Fig. 1. After proving the theorem we discuss a few of its implications.

Theorem 1 *Let $(\mathbf{x}_1, \mathbf{x}'_1, y_1), \ldots, (\mathbf{x}_\tau, \mathbf{x}'_\tau, y_\tau), \ldots$ be a sequence of examples and let R be an upper bound such that $\forall\tau : R \geq \|\mathbf{x}_\tau - \mathbf{x}'_\tau\|_2^4 + 1$. Assume that there exist $A^\star \succeq 0$ and $b^\star \geq 1$ for which $\forall\tau \geq 1$, $\ell_\tau(A^\star, b^\star) = 0$. Then the following bound holds for any $T \geq 1$*

$$\sum_{\tau=1}^{T} \left(\ell_\tau(A_\tau, b_\tau)\right)^2 \ \leq \ R \left(\|A^\star\|_F^2 + (b^\star - b_1)^2\right) \ . \quad (7)$$

The proof of the theorem is based on the following lemma

Lemma 2 *Let $\mathbf{w} \in \mathbb{R}^n$ be any vector and let $C \subset \mathbb{R}^n$ be a closed convex set. Then for any $\mathbf{w}^\star \in C$ we have*

$$\|\mathbf{w} - \mathbf{w}^\star\|_2^2 - \|\mathcal{P}_C(\mathbf{w}) - \mathbf{w}^\star\|_2^2 \geq \|\mathbf{w} - \mathcal{P}_C(\mathbf{w})\|_2^2 \ . \quad (8)$$

For a proof see for instance (Censor & Zenios, 1997), Thm. 2.4.1.

Proof of Theorem 1:
For simplicity, we use the definitions of $\mathbf{w}, \mathbf{w}_\tau, \mathbf{w}_{\tau+1}, \mathbf{w}_{\hat{\tau}}$ and $\boldsymbol{\chi}_\tau$ from Sec. 3.1. We also denote by $\mathbf{w}^\star \in \mathbb{R}^{n^2+1}$ the vector corresponding to the matrix-scalar pair $(A^\star, b^\star)$.

Define $\Delta_\tau = \|\mathbf{w}_\tau - \mathbf{w}^\star\|_2^2 - \|\mathbf{w}_{\tau+1} - \mathbf{w}^\star\|_2^2$. We prove the theorem by bounding $\sum_{\tau=1}^T \Delta_\tau$ from above and below. First note that $\sum_{\tau=1}^T \Delta_\tau$ is a telescopic sum and therefore

$$\begin{aligned} \sum_{\tau=1}^{T} \Delta_\tau &= \|\mathbf{w}_1 - \mathbf{w}^\star\|_2^2 - \|\mathbf{w}_{T+1} - \mathbf{w}^\star\|_2^2 \\ &\leq \|\mathbf{w}_1 - \mathbf{w}^\star\|_2^2 \ . \end{aligned} \quad (9)$$

This provides an upper bound on $\sum_\tau \Delta_\tau$. In the following we prove the lower bound $\Delta_\tau \geq \left(\ell_\tau(A_\tau, b_\tau)\right)^2 / R$. We can subtract and add the term $\|\mathbf{w}_{\hat{\tau}} - \mathbf{w}^\star\|_2^2$ from Δ_τ to get

$$\begin{aligned} \Delta_\tau &= \left(\|\mathbf{w}_\tau - \mathbf{w}^\star\|_2^2 - \|\mathbf{w}_{\hat{\tau}} - \mathbf{w}^\star\|_2^2\right) \\ &+ \left(\|\mathbf{w}_{\hat{\tau}} - \mathbf{w}^\star\|_2^2 - \|\mathbf{w}_{\tau+1} - \mathbf{w}^\star\|_2^2\right) \ . \end{aligned}$$

Recall that $\mathbf{w}_{\hat{\tau}}$ is the projection of $\mathbf{w}_\tau$ onto C_τ and that $\mathbf{w}_{\tau+1}$ is the projection of $\mathbf{w}_{\hat{\tau}}$ onto C_a. By assumption, $\mathbf{w}^\star \in C_a$ and $\mathbf{w}^\star \in C_\tau$. Therefore, we get from Lemma 2 that

$$\begin{aligned} \Delta_\tau &\geq \|\mathbf{w}_{\hat{\tau}} - \mathbf{w}_\tau\|_2^2 + \|\mathbf{w}_{\tau+1} - \mathbf{w}_{\hat{\tau}}\|_2^2 \\ &\geq \|\mathbf{w}_{\hat{\tau}} - \mathbf{w}_\tau\|_2^2 \ . \end{aligned} \quad (10)$$

We now use the fact that $\mathbf{w}_{\hat{\tau}} = \mathbf{w}_\tau + \alpha_\tau \boldsymbol{\chi}_\tau$ where $\alpha_\tau = \ell_\tau(A_\tau, b_\tau) / \|\boldsymbol{\chi}_\tau\|_2^2$ to get that

$$\|\mathbf{w}_{\hat{\tau}} - \mathbf{w}_\tau\|_2^2 = \frac{\left(\ell_\tau(A_\tau, b_\tau)\right)^2}{\|\boldsymbol{\chi}_\tau\|_2^2} \geq \frac{\left(\ell_\tau(A_\tau, b_\tau)\right)^2}{R} \ , \quad (11)$$

where the last inequality is due to the fact that $\|\boldsymbol{\chi}_\tau\|_2^2 = \|\mathbf{x}_\tau - \mathbf{x}'_\tau\|_2^4 + 1 \leq R$. Combining Eq. (11) with Eq. (10) we get $\Delta_\tau \geq \left(\ell_\tau(A_\tau, b_\tau)\right)^2 / R$. Comparing the above lower bound with the upper bound in Eq. (9) we get $\sum_{\tau=1}^T \left(\ell_\tau(A_\tau, b_\tau)\right)^2 \leq R \|\mathbf{w}^\star - \mathbf{w}_1\|_2^2$, which gives the bound in Eq. (7) since $\|\mathbf{w}^\star - \mathbf{w}_1\|_2^2 = \|A^\star\|_F^2 + (b^\star - b_1)^2$. ■

Note that the loss bound of Thm. 1 does not depend on the dimension of the instance space. Therefore, the bound does not change if we employ kernels which map the instances to high dimensional spaces. The sole difference in the bound when using kernels is that the norm of $A^\star$ and the norm of the instances are assumed to be small in the mapped space. Note also that we make a similarity prediction mistake iff $y_\tau \left(b_\tau - (d_{A_\tau}(\mathbf{x}_\tau, \mathbf{x}'_\tau))^2\right) \leq 0$. Thus, if on round τ the predicted similarity is incorrect, then $\left(\ell_\tau(A_\tau, b_\tau)\right)^2 \geq 1$. Therefore, the number of prediction mistakes cannot exceed $R \left(\|A^\star\|_F^2 + (b^\star - b_1)^2\right)$. Finally we would like to

note that while Thm. 1 provides a loss bound on the sum of *squares* of hinge losses, it is possible to derive a bound which is a mere sum of losses. The proof however is more complicated, and is omitted due to lack of space.

5. A Modification for the Inseparable Case

So far, we have assumed that there exists a pseudo-metric that perfectly matches the similarity and dissimilarity relations between instances. In this section, we relax this assumption and describe a modification for the algorithm in Fig. 1 for the inseparable case. Since there is no perfect pseudo-metric that explains the data even from hindsight, we do not expect our online algorithm to attain a fixed amount of loss. Instead, we measure the loss of the online algorithm *relative* to the loss of any other fixed pseudo-metric parametrized by $(A^\star, b^\star)$. The algorithm employs a relaxation parameter, denoted by $\gamma > 0$. The only modification to the algorithm in Fig. 1 is to define

$$\alpha_\tau = \frac{\ell_\tau(A_\tau, b_\tau)}{\|\mathbf{x}_\tau - \mathbf{x}'_\tau\|^4 + 1 + \gamma} \quad . \tag{12}$$

It is possible to derive a loss bound for POLA with the above modification *relative* to the loss of any other fixed pseudo-metric using the same techniques as in (Crammer et al., 2003). The full details are omitted due to the lack of space.

6. Using POLA in Batch Settings

We have focused thus far on online algorithms. However, in many machine learning tasks the entire training data is given to the learning algorithm in advance. Such settings are typically referred to as batch learning. In batch learning the goal is to find an hypothesis which exhibits small empirical error or loss on the training data and generalizes well by obtaining similar low loss on unseen examples. In this section we build upon the fact that we have devised an online learning algorithm with a loss bound on its performance. Specifically, we use POLA as a building block to devise a batch procedure which returns a pseudo-metric that is guaranteed to generalize well.

There are various techniques to convert from online to batch learning which come with some formal guarantees, quite a few can be used in our setting. We present here one of the simplest conversion techniques. The conversion procedure uses a convergence parameter, denoted β. It invokes POLA multiple times so long as there exists an example in the training set whose hinge loss exceeds β. If no such example exists the procedure stops and returns the final matrix obtained by POLA. The loss bound of Sec. 3 guarantees that at most $\lceil R \left(\|A^\star\|_{\mathrm{F}}^2 + (b^\star - b_1)^2 \right) / \beta^2 \rceil$ invocations of POLA will be required. By construction, the loss of the final pseudo-metric on *any* example from the training set is at most β. Furthermore, combining the inequality of Eq. (9) with the fact that Δ_τ is non-negative we obtain Lemma 3 below (see also Crammer et al., 2003). This lemma assures that the norm of the result is bounded.

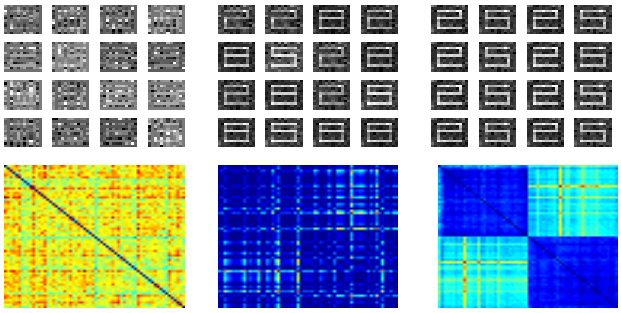

Figure 2. Results of dimensionality reduction using distance learning. Top: noisy images of the digits "2" and "5" (left) reconstruction of the images using PCA (middle) and reconstruction using POLA (right). Bottom: A corresponding color-coded representation of the distances between each pair of images.

Lemma 3 *Under the same conditions of Thm. 1, the following bound holds for any* $\tau \geq 1$

$$\|A_\tau\|_F^2 + (b_\tau - 1)^2 \;\leq\; 4 \left(\|A^\star\|_F^2 + (b^\star)^2 \right) \quad .$$

Combining the bound on the norm with the fact that its empirical loss on the training set is small implies that the resulting pseudo-metric has good generalization properties. That is, assuming that the training set and the test set are i.i.d samples from the same source then with high probability the loss on the test set is also small. The formal derivation uses standard learning theoretic tools and is omitted due to the lack of space.

7. Experimental Results

In this section we present experimental results with synthetic and natural data that demonstrate different merits of POLA. In the first experiment we created two synthetic images of the digits "5" and "2". Each image is composed of 12×12 pixels. We then created 64 noisy versions of the two original images by adding biased noise as follows. We defined two noise patterns: the first pattern was generated by adding a zero mean Gaussian noise to all the odd columns of the digit image; the second pattern was generated in a similar manner by adding noise to the odd rows. The variance of the noise was set such that the signal to noise ratio is 1 (0dB SNR). The noisy images are depicted on the top left part of Fig. 2. The noise degraded the original images up to the point where it is almost impossible to recognize whether the original digit is "2" or "5". The

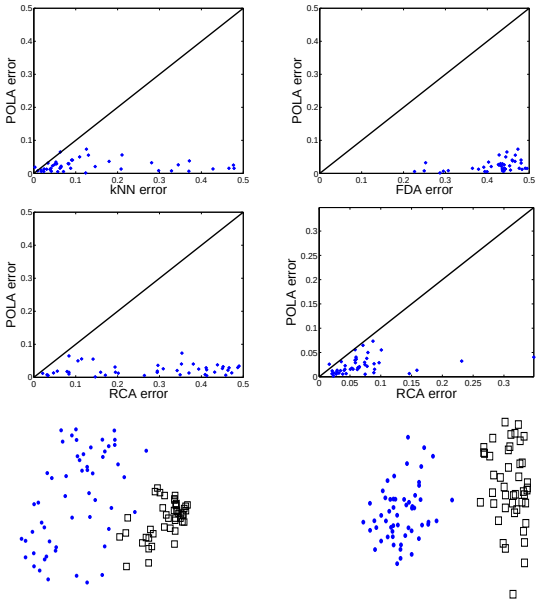

Figure 3. Top and middle rows: scatter plots of error rates for all pairs of digits from the MNIST dataset. Top Left: POLA vs. Euclidean distance. Top Right: POLA vs FDA. Middle Left: POLA vs. RCA without PCA as a preprocessing step. Middle Right: POLA vs. RCA. Bottom row: the digits 0 and 8 after dimensionality reduction using PCA (left) and POLA (right).

plot on the bottom left of Fig. 2 is a color coded representation of the distance between each pair of original noisy images. It is also clear that the distances are rather random and do not reflect the identity of the underlying prototypes. We next performed principal component analysis (PCA) on the images and reconstructed the images using the largest eigenvector. The corresponding reconstructed images are shown on the middle of the top row of Fig. 2. The distances between each two reconstructed images are given on the middle of the bottom row. While some of the images become more intelligible, it is still not possible in most cases to reveal whether an image represents the digit "2" or "5". Similarly, the corresponding distances do not clearly show the underlying two-class structure. Last, we applied POLA to all pairs of noisy images, and reconstructed each image using the largest eigenvector of the learned matrix A. The reconstructed images and the distances are depicted on the right hand side of Fig. 2. Most, if not all, of the reconstructed images look intelligible and we can easily reveal the original digit prototype for each image. Indeed, the distances matrix depicted on the right exhibits a clear block structure corresponding to the partition of the data into two classes. This experiment demonstrates the power of supervised learning of pseudo-metrics for extracting relevant information.

Our next set of experiments compares the performance of the k Nearest Neighbor (kNN) classifier with different (pseudo) metrics on the MNIST dataset. MNIST contains images of the 10 digits each of which is represented by 28×28 pixels. We randomly picked $10,000$ examples from the training set and used all the $10,000$ examples of the test set. Next, $\binom{10}{2} = 45$ binary classification problems were generated by comparing all pairs of digits. In the first experiment we compared the performance of kNN using the Euclidean distance to its performance when using a pseudo-metric obtained by running POLA on the training set. To train POLA we randomly chose $1,000$ pairs of instances and used the last hypothesis generated by POLA for evaluation on the test set (see Sec. 6). A comparison of the error on all 45 binary classification problems is given on the top left scatter plot of Fig. 3. Each point in the plot corresponds to a binary classification problem. The x-axis designates the error of kNN with Euclidean distance while the y-axis is the error of kNN using POLA's pseudo-metric. It is clear that using the learned pseudo-metric greatly reduces the error rate. In fact, the error when using POLA as a pre-processing step is lower than the vanilla kNN in *all* of the 45 binary problems. Next, we compared the RCA algorithm for learning distances (Shental et al., 2002) to POLA. RCA follows the same learning setting as POLA in a batch mode. We compared the performance of kNN using a distance function learned by RCA to its performance using a pseudo-metric learned by POLA. RCA uses PCA as a preprocessing step in order to reduce dimensionality. We thus applied PCA independently to each binary problem and reduced the dimension of each 28^2 image to a 40 dimensional vector. This value of the dimension was chosen by experimentation on the test set. The results, comparing POLA with RCA, are given on the middle right plot of Fig. 3. POLA outperforms RCA on all but one of the 45 binary problems. We also applied RCA without the dimensionality reduction step. The results are given on the middle left plot of Fig. 3. Here, the results of RCA are much worse and POLA outperforms RCA on all of the 45 binary problems. The fact that POLA does not require dimensionality reduction is in accordance with our formal analysis. Indeed, the loss bound of Thm. 1 depends on the Frobenius norm of $A^\star$ and does *not* depend on the actual dimension of the instances.

We also compared POLA to Fisher Discriminant Analysis (FDA) (Duda et al., 2001). FDA can be viewed as a dimensionality reduction method in the presence of supervision. The simplest form of FDA for binary classification problems projects the instances onto a *single* dimension. Thus, to make a fair comparison with FDA, we projected the data of each binary problem onto the largest eigenvector of the matrix found by POLA. We then compared the performance of kNN using the projected data obtained by POLA and FDA. The results are depicted on top right part

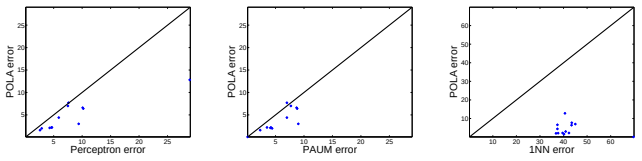

Figure 4. A comparison of the error of various online algorithms for document filtering in the dataset Reuters-21578.

of Fig. 3. Here again kNN with POLA clearly outperforms kNN with FDA on all of the problems.

In the final experiment with the MNIST dataset we randomly selected 100 images corresponding to the digits "0" and "8". We then projected each image onto the two largest eigenvectors obtained by PCA and the two largest eigenvectors of the matrix learned by POLA. The two projections are shown in the bottom row of Fig. 3 The projected points using the eigenvectors obtained by POLA generated two perfectly separable clusters each of which corresponds to a different digit. In contrast the analogous clusters when using PCA are interleaved. This demonstrates the potential power of POLA for dimensionality reduction.

We also compared POLA with other online algorithms on the problem of document filtering. We used the Reuters-21578 dataset. This dataset contains about $10,000$ documents. Each document in the corpus is labeled by zero or more topics from a predefined set. We represented the documents using the standard vector-space model with length-normalized tfidf after selecting 500 words (Singhal et al., 1996). Of the entire set of topics we chose 13 topics to use in our experiments. The topics were chosen such that the number of relevant documents is much smaller than the number of irrelevant documents for the topic. (The ratio between the number of relevant and irrelevant documents was in the range $[0.1, 0.01]$). We then evaluated POLA and various online algorithms on the task of online document filtering as follows. On each time step, we calculated the distance between a new instance to all of the relevant documents observed thus far. If the distance to the closest relevant document was less than the current threshold we predicted that the document is relevant. Otherwise, we predicted that it is irrelevant. To evaluate the performance of the algorithms we calculated both the average number of false positives (relevant) and the average number of false negatives (irrelevant). We then took the average of these two errors. The end results is an *equalized* error estimate that does not depend on the density of relevant document. We compared the results of POLA to the results obtained by the Perceptron algorithm (Rosenblatt, 1958), the PAUM algorithm (Li et al., 2002) (a variant of the Perceptron), and a simple 1NN classifier that uses on each round the documents observed so far. The results are given in Fig. 4. As can be seen from the scatter plot in the right, POLA clearly outperforms the simple 1NN algorithm. However, the performance of PAUM and the standard Perceptron algorithm are often comparable to POLA. Since the PAUM algorithm depends on parameters that drastically effect its performance, it is possible that finer tuning of these parameters will improve its performance.

Acknowledgments Thanks to G. Elidan, M. Fink, E. Egozi, and M. Shalev for discussion and comments. The work of A.N. was supported in part by the Department of the Interior/DARPA under contract number NBCHD030010. The work of Y.S. and S.S. was supported by EU PASCAL Network Of Excellence.

References

Censor, Y., & Zenios, S. (1997). *Parallel optimization: Theory, algorithms, and applications*. Oxford Press, NY, USA.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions in Information Theory, IT-13*, 21–27.

Cox, T., & Cox, M. (1994). *Multidimensional scaling*. Chapman and Hall, London.

Crammer, K., Dekel, O., Shalev-Shwartz, S., & Singer, Y. (2003). Online passive aggressive algorithms. *Advances in Neural Information Processing Systems 16*.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification*. Wiley. 2 edition.

Golub, G., & Loan, C. V. (1989). *Matrix computations*. John Hopkins University Press.

Herbster, M. (2001). Learning additive models online with fast evaluating kernels. *Proc. of the 14th Annual Conf. on Computational Learning Theory*.

Li, Y., Zaragoza, H., He, R., ShaweTaylor, J., & Kandola, J. (2002). The perceptron algorithm with uneven margins. *Proc. of the 19th International Conf. on Machine Learning*.

MacQueen, J. (1965). On convergence of k-means and partitions with minimum average variance. *Ann. Math. Statist.*

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review, 65*, 386–407. (Reprinted in *Neurocomputing* (MIT Press, 1988).).

Shental, N., Hertz, T., Weinshall, D., & Pavel, M. (2002). Adjustment Learning and Relevant Component Analysis. *7th Euro. conf. of Comp. Vision (ECCV)*.

Singhal, A., Buckley, C., & Mitra, M. (1996). Pivoted document length normalization. *R & D in Information Retrieval*.

Vapnik, V. N. (1998). *Statistical learning theory*. Wiley.

Wilkinson, J. (1965). *The algebric eigenvalue problem*. Claderon Press, Oxford.

Xing, E., Ng, A.Y., Jordan, M., & Russell, S. (2003). Distance metric learning, with application to clustering with side-information. *Advances in Neural Information Processing Systems 15*.