A Metric-learning based framework for Support Vector Machines and Multiple Kernel Learning

Huyen Do*, Alexandros Kalousis†

*Computer Science Department, University of Geneva, Switzerland
Email: huyen.doth@gmail.com
†Business Informatics, University of Applied Sciences Western Switzerland
Email: Alexandros.Kalousis@hesge.ch

Abstract—Most metric learning algorithms, as well as Fisher’s Discriminant Analysis (FDA), optimize some cost function of different measures of within-and between-class distances. On the other hand, Support Vector Machines (SVMs) and several Multiple Kernel Learning (MKL) algorithms are based on the SVM large margin theory. Recently, SVMs have been analyzed from a metric learning perspective, and formulated as a Mahalanobis metric learning problem. This new perspective allows us to combine ideas from both SVM and metric learning, and to develop new algorithms that build on the strengths of each. Inspired by the metric learning interpretation of SVM, we develop here a new metric-learning based SVM framework in which we incorporate metric learning concepts within SVM. We extend the optimization problem of SVM to include some measure of the within-class distance and along the way we develop a new within-class distance measure which is appropriate for SVM. In addition, we adopt the same approach for MKL and show that it can be also formulated as a Mahalanobis metric learning problem. Our end result is a number of SVM/MKL algorithms that incorporate metric learning concepts. We experiment with them on a set of benchmark datasets and observe important predictive performance improvements.

Index Terms—Multiple Kernel Learning, Support Vector Machines, Metric Learning, unified view, convex optimization

I. INTRODUCTION

Support Vector Machines [5] have been an active research area for more than two decades. They have been widely used not only because of their excellent predictive performance but also because their generalization ability is supported by solid generalization error bounds defined over the radius-margin ratio. Mahalanobis metric learning has started attracting significant attention rather recently [3], [12], [13], [16], [18], [20], [22], [23]. However, relying mostly on intuition, it still lacks theoretical support. Very recently SVM has been reformulated in the metric learning context and has been shown to be equivalent to a Mahalanobis metric learning problem [9]. This new interpretation of SVM brings the worlds of SVM and metric learning together into a single unified view. This allows us to exploit the advantages of each one to develop for example hybrid algorithms or to derive theoretical error bounds for metric learning problems exploiting the SVM error bounds.

In this paper we build on the ideas presented in [9] to develop a novel metric-learning-based SVM framework and equip SVM with a metric learning bias. More precisely we will define new SVM optimization problems that will make use of both the between- and within-class distances. Under the metric learning view of SVM, the margin plays the role of the between-class distance. However, SVM ignores the within-class distance. In the metric-learning-based SVM framework that we present here, we maximize the SVM margin and minimize some measure of the within-class distance. We can use different measures of the within-class distance with SVM and we will define a new such measure that is more appropriate for SVM. We will give a new SVM algorithm that optimizes both the margin and the new within-class distance measure that we define. The resulting optimization problem is convex and can be directly kernelized. Moreover we will follow the same approach with MKL and show that it, also, can be formulated as a Mahalanobis metric learning problem. As a result we develop a novel family of MKL methods that incorporate the metric learning bias. We experiment with the developed algorithms on a number of benchmark datasets and saw that the incorporation of the within-class distance measures in the SVM learning problem brings significant performance improvements.

Finally, we give a unified view of SVM, metric-learning-based SVM, metric learning algorithms and Fisher Discriminant Analysis (FDA) using the concepts of between-class and within-class distances. This view provides new insights to the existing algorithms and unveils some unexpected relations.

The rest of the paper is organized as follows. In the next section we briefly describe the basic concepts of SVM, FDA and Mahalanobis metric learning. In section III we summarize the metric learning view of SVM. In section IV we propose the metric-learning-based SVM framework which relies on the use of the between-class and within-class distance measures. In section V we provide a common view of SVM, metric-learning based SVM, FDA and metric learning algorithms. In Section 5 we employ the metric learning perspective in the context of MKL and develop the metric-learning-based MKL framework. We report experimental results in section VII and conclude with section VIII.

II. PRELIMINARY

Consider a binary classification problem in which we are given a set of learning instances $S = \{x_1, x_2, ..., x_n\}$...
\{(x_1, y_1), ..., (x_n, y_n)\}, x_i \in \mathcal{R}^d\), where \(y_i\) is the class label of the \(x_i\) instance and \(y_i \in \{+1, -1\}\). Let \(D_i = \{x_j | y_j = C_i\}\) be the samples of class \(C_i\).

A. Support Vector Machines and MKL

SVMs learn a hyperplane \(H_w : w^T x + b = 0\) which maximizes the margin between the two classes. The SVM margin is defined informally as the distance of the nearest instances from \(H_w\) [5]. The SVM optimization problem is:

\[
\min_{w, \xi, b} \|w\|_2^2 + C \sum_i \xi_i \\
\text{s.t.} \quad y_i (w^T x_i + b) \geq 1 - \xi_i, \forall i
\]  

(1)

The performance of SVM strongly depends on the choice of kernel. MKL addresses that problem by selecting or learning the appropriate kernel(s) for a given problem [1], [4], [6], [10], [18], [20], [22]. Given a set of basis kernel functions, \(Z = \{K_k(x, x')|k := 1 ... m\}\), MKL learns a kernel combination, usually convex, so that some cost function, e.g margin, \(Z = \text{kernel}\). MKL SVM maximizes the margin between the two classes. The constraints respectively, parametrized either by \(M\) or \(A\). The

\[
d_M^2(x_i, x_j) \leq f_1, \text{ if } y_i = y_j
\]

(4)

\[
d_M^2(x_i, x_j) \geq f_2, \text{ if } y_i \neq y_j
\]

for some constants \(f_1, f_2\). These constraints reflect the primal bias of most metric learning algorithms, maximizing the between-class distance and minimizing the within-class distance, or in other words, instances of the same class should have small distances while those of different classes should have large distances. If we take two simple measures of the within and between-class distances such as the sum of the pairwise distances of the same-class instances and the sum of the pairwise distances of the different-class instances respectively, the pairwise constraints in (4) will ensure that our between-class distance, denoted by \(d_B\), is bigger than \(f_2 \times t_1\), and our within-class distance, denoted by \(d_W\), is smaller than \(f_1 \times t_2\), where \(t_1, t_2\) are the numbers of instance pairs of the same and different class respectively over which we impose the constrains..

C. Fisher Discriminant Analysis

FDA, although usually not considered a metric learning method, it also learns linear projections in the same manner as metric learning methods. It uses a similar learning bias as metric learning algorithms: the samples are well separated if their between-class distance is large and their within-class distance is small. These quantities are defined as follows [17]: let the sample mean of class \(C_i\) be \(m_i = \frac{1}{n_i} \sum_{x \in D_i} x\), then the within-class distance (or within-class scatter) of two classes \(C_1, C_2\), is defined as \(s_w = \sum_{i=1,2} \sum_{x \notin D_i} \|x - m_i\|^2\), and the between-class distance (or between-class scatter) is defined as the squared distance between the means of the two classes, \(s_B = \|m_1 - m_2\|^2\). In the case of two-class problems FDA seeks for a projection line with a direction vector \(w\) which optimizes the ratio of the between-class over the within-class distances in the projected space, its cost function is:

\[
\max_w J(w) = \frac{s_{w,w}}{s_{w,w}} = \frac{(w^T (m_1 - m_2))^2}{\sum_{i=1,2} \sum_{x \in D_i} (w^T (x - m_i))^2}
\]

(5)

where \(S_w = \sum_{i=1,2} \sum_{x \in D_i} (x - m_i)(x - m_i)^T\) and \(S_B = (m_1 - m_2)(m_1 - m_2)^T\) are the within-class and between-class scatter matrices, respectively. Another measure often used in the different variants of FDA is the total scatter matrix, \(S_T = S_w + S_B\), i.e a multiply of the covariance matrix which quantifies the total data spread.

III. RELATED WORK

Do et al [9] recently show SVM can be formulated as a Mahalanobis metric learning problem in which the transformation matrix is diagonal \(W\), \(\text{diag}(W) = w = \ldots\)
\((w_1, \ldots, w_d)^T, \|w\|_2 = 1\). In the metric learning jargon SVM learns a diagonal linear transformation \(W\) and a translation \(b\) which maximize the margin and place the two classes symmetrically in the two different sides of the hyperplane \(H_1: x^T w = 0\). In the standard view of SVM, the space is fixed and the hyperplane is moved around to achieve the optimal margin. In the metric view of SVM, the hyperplane is fixed to \(H_1\) and the space is scaled, \(W\), and then translated, \(b\), so that the instances are placed optimally around \(H_1\) \([2]\).

\([9]\) proposed a measure of the within-class distance for SVM. This measure is inspired by the relation, developed in that paper, between SVM and LMNN—Large Margin Nearest Neighbor \([23]\)—a popular metric learning algorithm. It is defined as the sum of the distances of the instances from the margin hyperplane and for the class \(C_i\), it is given by:

\[
d^w_i = \sum_{x \in C_i} \|d(x, H_w)\| - \gamma.
\]

The authors then proposed an SVM variant, called \(\epsilon\)-SVM, which optimizes the margin and the above within-class distance measure, essentially combining both the SVM and the LMNN learning biases. As we will see below \(\epsilon\)-SVM turns out to be a special case of the \(SVM\), which will be described in section \([10]\). The optimization problem of \(\epsilon\)-SVM is:

\[
\min_{w,b} \quad w^T w + \lambda \sum_i \max(0, y_i(w^T x_i + b) - 1)
\]

\[+ C \sum_i \max(0, 1 - y_i(w^T x_i + b))
\]

which is equivalent to:

\[
\min_{w,b,\xi,\xi_n} \quad w^T w + C \sum_i \xi_i + C_2 \sum_i \eta_i
\]

\[s.t. \quad 1 - \xi_i \leq y_i(w^T x_i + b) \leq 1 + \eta_i\]

\[\xi_i \geq 0, \eta_i \geq 0, \forall i\]

where \(\eta_i\) is the distance of the \(i\)th instance from its margin hyperplane, and \(\xi_i\) are the SVM slack variables which allow for the soft margin. This problem is convex and can be kernelized directly as a standard SVM.

\([21]\) proposed to maximize the margin and to constrain the outputs of SVM, their optimization problem thus optimizes the margin and some measure of the data spread. This approach also falls within our general metric-learning-based SVM framework that we will present right away.

IV. A METRIC-LEARNING-BASED SVM FRAMEWORK

Since SVM can be seen as a metric learning algorithm, we can interpret it in terms of the within- and between-class distances, as is done with typical metric learning algorithms using \([3]\), \([4]\). The SVM margin can be seen as a measure of the between-class distance. Unlike FDA where the between-class distance is defined via the distance of the class means, in SVMs, it is defined as the minimum distance of different-class instances from the hyperplane, i.e. twice the SVM margin. Unlike other metric learning algorithms, where the between-class distance takes into account all pairs of instances of different classes, the SVM between-class distance only focuses on pairs of instances of different classes which lie on the margin (in \([9]\), the SVM margin is reformulated as: \(\min_{i,j, y_i \neq y_j} (x_i - x_j)^T w^T w(x_i - x_j)\)). In other words, SVM between-class distance, i.e the margin, is a simple example of \([4]\).

Similar to most metric learning algorithms, SVM maximizes the \textit{between-class distance}; however, unlike them, it ignores the \textit{within-class distance}. In other words, instances of different classes are pushed far away by the margin, but there is no constraint on instances of the same class.

Interpreting SVM as a metric learning problem allows us to fully equip it with the metric learning primal bias, i.e to maximize some between-class distance measure and minimize some within-class distance measure. We propose a learning framework in which in addition to the standard SVM margin maximization, we also minimize some measure of the within-class distance. We will call the resulting learning algorithms metric-learning-based SVM. In the next sections we will give different general functions of the within and between class distances which can be the target of optimization. We will then propose SVM specific within-class distance measures, and finally formulated the full learning problem which now will also include some measure of the within class distance.

A. Within- and between-class distances cost functions

There are several ways to maximize the between-class distances \(d_B\) while minimizing the within-class distances \(d_W\). Below we give some simple and widely used cost functions that include these two terms:

\[
\max \quad F_1 = \frac{d_B}{d_W}
\]

\[
\max \quad F_2 = d_B + \lambda \frac{1}{d_W}
\]

\[
\max \quad F_3 = d_B - \lambda d_W
\]

\[
\min \quad F_4 = \frac{1}{d_B} + \lambda d_W
\]

Depending on the exact measures of the within- and between-class distances, these cost functions will lead to convex or non-convex optimization problems. An example of the first cost function \(F_1\) of \([8]\) is FDA \([5]\) where the resulting optimization problem is convex and easy to solve. \(F_1\) places equal importance on the between- and within-class distances. On the other hand \(F_2, F_3,\) and \(F_4\) allow us to better control their trade-off through the introduction of an additional hyperparameter \(\lambda\). Within our metric-learning-based SVM framework, \(d_B\) will denote the margin, and \(d_W\) will denote some measure of the within-class distance.

B. Metric-learning based SVM

In this section we start by presenting a new measure of the within-class distance \(d_W\) which is appropriate for SVM. We will then use this measure to define one instantiation of our metric-learning-based SVM framework. In addition, we discuss a number of other within-class distance measures which can be used to define other instantiations of our framework.
1) *A bandwidth-based within-class distance measure:* One way to control the within-class distance is by forcing the learning instances to stay close to their class margin hyperplane, confining them within a band defined by their margin hyperplane and a hyperplane parallel to it. The width of this band can be seen as a measure of the within-class distance. This bandwidth is equal to the maximum distance of the instances to their class margin given by \( d_{W_2} = \max_{x \in C_i} |d(x, H_w)| - \gamma \). To avoid the effect of outlier instances we add slack variables that allow some of them to lie outside the band.

We introduce now a new SVM variant, which optimizes both the margin and the measure of the within-class distance described above. Its cost function trades-off the margin max-
both the margin and the measure of the within-class distance
can be seen as a measure of the within-class distance. This
confining them within a band defined by their margin hyper-
to control the within-class distance is by forcing the learn-

To avoid the effect of outlier instances we add slack variables

\[
\max_{w,b,\gamma,\epsilon} \gamma - \lambda \epsilon \gamma \\
\text{s.t.} \quad y_i(w^T x_i + b) \geq \gamma, \forall i \\
\frac{y_i(w^T x_i + b)}{\|w\|_2} \leq \gamma + \epsilon \gamma, \forall i \\
\gamma \|w\|_2 = 1, \epsilon \geq 0 \\
\Leftrightarrow \max_{w,b,\epsilon} (1 - \lambda \epsilon)/\|w\|_2 \\
\text{s.t.} \quad y_i(w^T x_i + b) \geq 1 \\
\frac{y_i(w^T x_i + b)}{\|w\|_2} \leq 1 + \epsilon, \forall i, \epsilon \geq 0
\]

Problem (13) is not convex, however, it can be easily ker-
nelized as standard SVM. If we fix \( \epsilon \) then it becomes convex.
Fixing \( \epsilon \) is equivalent to fixing a specific value for the margin-
bandwidth ratio and then maximizing the margin; this is de-
scribed by the following optimization problem:

\[
\min_{w,b,\xi,\eta} \|w\|_2^2 + C_1 \sum \xi_i + C_2 \sum \eta_i \\
\text{s.t.} \quad y_i(w^T x_i + b) \geq 1 - \xi_i \\
\frac{y_i(w^T x_i + b)}{\|w\|_2} \leq 1 + \epsilon + \eta_i, \forall i \\
\xi, \eta \geq 0
\]

where \( \xi \) and \( \eta \) are the slack variables that allow for the margin
and bandwidth violations, the latter is needed to alleviate the
effects of outlier instances. Intuitively the hyperparameter \( C_2 \),
which controls the bandwidth slack variable, should be bigger
than \( C_1 \) which controls the margin slack variable, since we
can tolerate more instances outside the band than inside the
margin.

Problem (14) is convex, quadratic and can be solved simi-
larly to standard SVM. We will call it \( SVM_m \). Its dual form
is:

\[
\max_{\alpha,\beta} \sum_{i} (\alpha_i - (1 + \epsilon)\beta_i) \\
\text{s.t.} \quad \frac{1}{2} \sum_{ij} (\alpha_i - \beta_i)(\alpha_j - \beta_j)y_iy_jx_i^T x_j \\
\sum_{i} (\alpha_i - \beta_i)y_i = 0 \\
0 \leq \alpha_i \leq C_1, \forall i \\
0 \leq \beta_i \leq C_2, \forall i
\]

It can be also kernelized directly as SVM, as the term \( x_i^T x_j \) that appears in the dual form (15) can be replaced by a kernel function.

Using the \( F_4 \) cost function (11) we get the following
non-convex optimization problem which also “maximizes the
margin while keeping the bandwidth \( d_{W_2} \) small”:

\[
\min_{w,b,\gamma,\epsilon} \frac{1}{\gamma^2} + \lambda\epsilon^2\gamma^2 \\
\text{s.t.} \quad y_i(w^T x_i + b) \geq \gamma \\
\frac{y_i(w^T x_i + b)}{\|w\|_2} \leq \gamma + \epsilon\gamma, \forall i \\
\gamma\|w\|_2 = 1, \epsilon \geq 0 \\
\Leftrightarrow \min_{w,b,\epsilon} \|w\|_2^2 + \frac{\lambda\epsilon^2}{\|w\|_2^2} \\
\text{s.t.} \quad 1 \leq y_i(w^T x_i + b) \leq 1 + \epsilon, \forall i, \epsilon \geq 0
\]

However, this problem is non convex even for a fixed \( \epsilon \); therefore, we do not explore it further in this paper and we leave it for future work.

Interestingly, we note that if in the optimization prob-
lem (13) we set \( \epsilon = 0 \) then this problem reduces to the
optimization problem (7), therefore, we can also solve \( \epsilon \)-SVM
of (9) as a special case of \( SVM_m \) using the \( SVM_m \) solver.

We note that \( SVM_m \) is a general formulation for both
SVM and \( \epsilon \)-SVM. In the limit when \( \epsilon \rightarrow \infty \), \( SVM_m \) reduces
to standard SVM, and when \( \epsilon \rightarrow 0 \), \( SVM_m \) becomes \( \epsilon \)-
SVM. From the optimization point of view we remark that
the bigger value of \( \epsilon \) is, the smaller the number of the
active constraints will be (see the second set of constraints in
(14)). Thus in terms of running time, \( \epsilon \)-SVM is the slowest,
followed by \( SVM_m \) and then SVM. However, all are quadratic
optimization problems and can be solved efficiently.

One may also think of the FDA within-class distance as an
appropriate measure, i.e. \( d_{W_2} = w^T S w \). Xiong et al.
(25) combined SVM and FDA, optimizing like that the margin
and the FDA within-class distance. However, we note that
they simply introduced this combination without putting it
in the metric learning context that we described here, and
they provided no interpretation on the use of the within-
and between-class distances. Still their work falls into our
metric-learning-based SVM framework as a special case. Using
the cost function of (9) to optimize the margin and the
FDA within-class distance, we can formulate the following optimization problem:

\[
\min_{\mathbf{w},b} \quad \mathbf{w}^T (\lambda \mathbf{S}_\mathbf{W} + I) \mathbf{w} \\
\text{s.t.} \quad y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \forall i
\]

(18) is convex and is equivalent to the standard SVM problem in the transformed space \( \tilde{\mathbf{x}} = \mathbf{A} \mathbf{x} = (\mathbf{S}_\mathbf{W} + \lambda \mathbf{I})^{-1} \mathbf{x} \), where \( \mathbf{S}_\mathbf{W} \) is the FDA within-class scatter matrix and \( \mathbf{I} \) is the identity matrix [25]. However, it is not straightforward to kernelize and solved it only in the original feature space.

We note that it is an interesting question to develop more measures for the within- and between-class distances, or the data spread. For example the radius of the smallest sphere containing the data is also a measure of the data spread. Therefore, the different variants of the radius-margin based SVMs [8], [14], [19], which maximize the margin and minimize the radius, also fall to our metric-learning-based SVM framework. A more challenging problem is to determine which measure is the best in some specific situations.

V. A UNIFIED VIEW OF FDA, SVM, METRIC-LEARNING-BASED SVM AND METRIC LEARNING

We first analyze FDA from the metric learning perspective as it is done with SVM in [9]. We focus only on binary classification problems. Similar to SVM, we can also formulate FDA as a Mahalanobis metric learning problem, where the transformation matrix is diagonal \( \mathbf{W} \), \( \text{diag}(\mathbf{W}) = \mathbf{w} \). We reformulate the FDA between-class, \( \tilde{s}_b \), and within-class, \( \tilde{s}_w \), distances with respect to the \( H_1 \) hyperplane as the standard FDA between and within-class distances in the transformed space \( \mathbf{W} \mathbf{x} \), projected to the norm vector of \( H_1 \):

\[
\tilde{s}_b = \frac{1}{d} \left\| \mathbf{1}^T (\mathbf{Wm}_i - \mathbf{Wm}_j) \right\|^2
\]

\[
\tilde{s}_w = \frac{1}{d} \sum_{i=1,2} \sum_{\mathbf{x} \in \mathcal{D}_i} \left\| \mathbf{1}^T (\mathbf{Wx} - \mathbf{Wm}_i) \right\|^2
\]

The FDA learning problem then can be stated in the metric learning jargon as follows: we learn a diagonal transformation \( \mathbf{W} \) so that in the transformed space \( \tilde{\mathbf{x}} = \mathbf{W} \mathbf{x} \), the FDA between-class distance, with respect to the \( H_1 \) hyperplane, is maximized and the FDA within-class distances, with respect the same hyperplane, is minimized. This learning problem leads to the standard FDA given in [5]. Thus, we see that from the metric learning perspective both FDA and SVM learn a diagonal transformation \( \mathbf{W} \). However, the way they define their between and within-class distances is different, although all these distances are based on distances from the fixed hyperplane \( H_1 \) and are defined globally. The FDA between-class distance is the squared distance between the two class means, with respect to the hyperplane \( H_1 : \mathbf{1}^T \mathbf{x} = 0 \), while the SVM between-class distance is the minimum distance between two instances of the two classes, with respect to \( H_1 \). On the other hand, FDA defines its within-class distances as the total sum of distances between each instance and its corresponding class mean, while standard SVM ignores the

![Figure 1](image-url)
VI. A METRIC-LEARNING BASED MKL FRAMEWORK

In this section we will show how to exploit the metric-learning approach in the context of MKL. We first show how to formulate \( MKL_{\gamma} \) as a metric learning problem with the help of two linear transformations. We proceed by developing a metric-learning-based MKL framework which optimizes both the margin and some measure of the within-class distance using multiple kernels.

A. MKL from a metric learning perspective

In MKL we learn linear combinations of kernels of the form \( K_{\mu} = \sum_k \mu_k K_k \). The feature space \( \mathcal{H}_\mu \) that corresponds to the learned kernel \( K_{\mu} \) is given by the mapping:

\[
x \rightarrow \Phi_{\mu}(x) = (\sqrt{\mu_1} \Phi_1^T(x), ..., \sqrt{\mu_m} \Phi_m^T(x))^T \in \mathcal{H}_\mu = D \sqrt{\gamma} \Phi(x)
\]

where \( \Phi_k(x) \) is the mapping to the \( K_k \) feature space associated with the \( K_k \) kernel, \( \Phi(x) = (\Phi_1^T(x), ..., \Phi_m^T(x))^T \in \mathcal{H} \), and \( D \sqrt{\gamma} \) is block diagonal matrix with block diagonal elements \( \sqrt{\mu} \).

Similar to SVM, we can also use the fixed hyperplane \( H_1 \) and view \( MKL_{\gamma} \) as learning a block diagonal linear transformation \( D \sqrt{\gamma} \) in the concatenation feature space \( \mathcal{H} \), followed by a diagonal linear transformation \( W \) and a translation \( b \), such that the margin with respect to the \( H_1 \) hyperplane is maximized and the two classes are placed symmetrically in the two different sides of \( H_1 \). The linear transformation \( A \) associated with \( MKL_{\gamma} \) is given by \( A = WD\sqrt{\gamma} \). So \( MKL_{\gamma} \) is also a Mahalanobis metric learning problem where the transformation matrix is \( A = WD\sqrt{\gamma} \).

From a metric learning perspective SVM uses a single diagonal matrix transformation given by \( W \) and \( MKL_{\gamma} \) uses two diagonal matrix transformations given by \( W' = WD\sqrt{\gamma} \); both optimize the same cost function (i.e. the margin). A formal comparison of the two methods under the metric-learning view can be found in the Appendix.

B. MKL and the optimization of the within-class distance

Similar to SVM, standard \( MKL_{\gamma} \) optimizes only the margin, i.e. the between-class distances but ignores the within-class distances. As before with SVM we will now develop a metric-learning-based MKL framework in which we will optimize both the margin and some measure of the within-class distances. We will give two examples of metric-learning-based MKL algorithms, using the \( SVM_m \), section [IV-B1] and \( \epsilon-SVM \) [9].

The \( SVM_m \) optimization problem, equation (14), in the MKL context becomes:

\[
\begin{array}{l}
\min_{w,b,\xi,\eta} \|w\|^2 + \frac{1}{2} \sum_i \xi_i + \frac{1}{2} \sum_i \eta_i \\
\text{s.t.} \quad 1 - \xi_i \leq y_i (\langle w, \Phi_k(x_i) \rangle + b) \leq 1 + \epsilon + \eta_i, \forall i \\
\|\mu\|_1 = 1, \mu_k \geq 0, \forall k, \epsilon \geq 0
\end{array}
\]

which is equivalent to:

\[
\begin{array}{l}
\min_{w,b,\xi,\eta,\mu} \frac{1}{2} \sum_k \langle w_k, w_k \rangle + \frac{C_1}{2} \sum_i \xi_i + \frac{C_2}{2} \sum_i \eta_i \\
\text{s.t.} \quad 1 - \xi_i \leq y_i \sum_k \langle w_k, \Phi_k(x_i) \rangle + b \leq 1 + \epsilon + \eta_i, \forall i \\
\sum_k \mu_k = 1, \mu_k \geq 0, \forall k, \xi_i \geq 0, \forall i
\end{array}
\]

This optimization problem is the counterpart of \( SVM_m \) in the MKL context; we will call it \( MKL_m \). As it was the case with \( SVM_m \) where setting \( \epsilon \) to zero leads to \( \epsilon\)-SVM, here too, if in the \( MKL_m \) optimization problem (22), we set \( \epsilon = 0 \) the resulting optimization problem, which we denote by \( \epsilon\)-MKL, will correspond to the coupling of \( \epsilon\)-SVM with MKL. \( \epsilon\)-MKL thus learns kernel combinations that maximize the margin and minimize the within-class distance as the latter is measured by the sum of the distances from the margin hyperplanes.

The \( MKL_m \) optimization problem, equation (22), is convex and equivalent to:

\[
\begin{array}{l}
\min_{\mu} J(\mu), \\
\text{s.t.} \quad \sum_k \mu_k = 1, \mu_k \geq 0, \forall k
\end{array}
\]

where:

\[
J(\mu) = \left\{ \begin{array}{ll}
\min_{w,b,\xi,\eta} & \frac{1}{2} \sum_k \langle w_k, w_k \rangle + \frac{C_1}{2} \sum_i \xi_i + \frac{C_2}{2} \sum_i \eta_i \\
\text{s.t.} & 1 - \xi_i \leq y_i \sum_k \langle w_k, \Phi_k(x_i) \rangle + b \leq 1 + \epsilon + \eta_i, \\
& \forall \xi_i \geq 0, \forall \eta_i \geq 0
\end{array} \right.
\]

We can solve it by a two step algorithm, similar to the one used in SimpleMKL (20). At the first step of the algorithm we fix \( \mu \). With fixed \( \mu \), problem (22) becomes a SVM-like optimization problem. In the second step, with the optimal values computed by the SVM-like problem in the first step, we optimize the whole problem by gradient descent with respect to \( \mu \). The gradient of \( J(\mu) \) is:

\[
\frac{\partial J(\mu)}{\partial \mu} = -\frac{1}{2} \sum_i (\alpha_i^* - \beta_i^*) (\alpha_i^* - \beta_i^*) y_i y_j K_k(x_i, x_j) \]

\( \alpha^* \), \( \beta^* \) are the solution of the first step.

VII. EXPERIMENTS

We performed two sets of experiments. In the first we examined the performance of three metric-learning-based SVM algorithms, namely \( SVM_m \) (14), \( \epsilon\)-SVM (6) (9), and \( SVM_{FDA} \) (13) (25). Note that \( \epsilon\)-SVM is a special case of \( SVM_m \) when \( \epsilon \) is set to zero. We used as baseline the performance of standard SVM and FDA. We used the \textit{stprotool} toolbox (11) for FDA. In the second set, we compared the two metric-learning-based MKL algorithms, i.e. \( MKL_m \) and \( \epsilon\)-MKL, with SimpleMKL (20), a state-of-the-art \( MKL_{\gamma} \) algorithm.

We experimented with UCI benchmark datasets. We first standardized the data to a zero mean and one variance. We examine the performance of the different SVM based methods over the four following kernels: linear, polynomial with degree 2 and 3, and a Gaussian kernel with \( \sigma = 1 \). Kernel \( K \) was normalized as follows: \( K_{ij} = K_{ij} / \sqrt{K_{ii} K_{jj}} \). For the MKL
methods, we learned combinations of the following 20 kernels: 10 polynomial with degree from one to ten, ten Gaussian with bandwidth \( \sigma \in \{0.5, 1, 2, 5, 7, 10, 12, 15, 17, 20\} \), the same set of basic kernels as the one used in SimpleMKL \(^{[20]}\).

Note that SVM-FDA can be only applied on the original feature space since it does not have a kernelized version. For SVM\(_m\),\( \epsilon \)-SVM, MKL\(_m\),\( \epsilon \)-MKL, we set \( C \) to \( C_1/3 \), as we can tolerate more bandwidth than margin violations. For SVM\(_m\), \( \epsilon \) is set to three, i.e. the allowed band is three times wider than the margin. The optimal parameter \( C \) or \( C_1 \) of the margin slack variables is chosen by an inner 10-fold cross-validation, from the set of \( C = \{0.1, 1, 10, 100, 1000\} \). For SVM-FDA we choose \( C \) and \( \lambda \) from the same set of \( \{0.1, 1, 10, 100, 1000\} \) using the 10-fold inner cross-validation. We estimated the classification error using 10-fold cross validation. Folds are the same for all algorithms.

The results for the SVM algorithms are given in Table \( \text{II} \) and for the MKL ones in Table \( \text{[I]} \). We can see that SVM\(_m\) has the lowest error in most of the cases. Among the 28 different experiments performed for each algorithm (7 datasets \( \times \) 4 kernels), SVM\(_m\) has the lowest error 19 times, \( \epsilon \)-SVM 10 times, SVM 5 times and FDA never. We should note that the second constraint of both \( \epsilon \)-SVM \((7)\) and SVM\(_m\) \((14)\) will not be triggered if the data points are already very close to the margin; in such case SVM\(_m\) and \( \epsilon \)-SVM have the same performance as SVM. We also compared the statistical significance of the performance difference using a McNemar’s test with the significance level set at 0.05. For each experiment of a given algorithm, i.e. kernel and dataset, the algorithm was credited with one point if it was significantly better than another algorithm for the same experiment, half a point if there was no significance difference between the two algorithms, and no point if it was significantly worse than the other algorithm. Under this ranking schema, in the original feature space, SVM\(_m\) got a total score of 20.5 points over the seven datasets, out of a total possible maximum of 28, followed by SVM-FDA with 17, SVM with 15, \( \epsilon \)-SVM with 14; FDA is by far the worst with only 3.5 points. In the polynomial kernel space of degree two and three, SVM\(_m\) and \( \epsilon \)-SVM rank first with 14.5 points but their advantage over SVM which got 13 points is not so pronounced; FDA got zero points. With the gaussian kernel, both SVM\(_m\) and \( \epsilon \)-SVM have a significant advantage over SVM and FDA; the former two got 14.5 points while SVM and FDA got only six and seven points respectively. Overall SVM\(_m\) has a consistent advantage over the different datasets and the different kernels we experimented with. We should note here that the incorporation of the within-class distance in the optimization problem seems to bring the largest benefit when the kernel that is used is not that appropriate for the given problem, e.g. the Gaussian kernel for the datasets we experimented with here, this advantage is not so pronounced when the chosen kernel is good, e.g. polynomial of degree two and three. In other words incorporating the within-class distance seems to have a corrective effect when a mediocre or poor kernel is used.

The poor performance of FDA can be explained by the between-class distance measure it deploys. Using the class means as a between-class distance measure can lead to class overlapping. \( \epsilon \)-SVM, SVM\(_m\) and SVM-FDA do not have this problem since they use the margin as the between-class distance. Unlike SVM-FDA which can be applied only in the original feature space, since it cannot be kernelized, both \( \epsilon \)-SVM and SVM\(_m\) can be easily kernelized.

![Fig. 2. Running time of SVM\(_m\), \( \epsilon \)-SVM, SVM-FDA, SVM and FDA. Datasets are in the same order as in Table II.](image)

![Table II](image)

| Datasets | \# Instances | \# Features | SimpleMKL | MKL\(_m\) | \( \epsilon \)-MKL |
|----------|--------------|-------------|------------|-----------|---------------|
| Ionosphere | 706 | 122 | 13.65 | 13.60 | 13.55 |
| Musk1 | 476 | 166 | 5.25 | 4.41 | 4.20 |
| Wdbc | 569 | 30 | 2.64 | 2.11 | 2.64 |
| Stroke | 208 | 2810 | 23.08 | 22.60 | 22.12 |
| Ovarian | 253 | 771 | 4.35 | 4.35 | 3.95 |
| Liver | 354 | 6 | 34.20 | 34.49 | 32.46 |
| Wpbc | 198 | 34 | 23.23 | 22.73 | 22.73 |

Score McNemar

7.5 | 8.5 | 8
## Table I

| Kernel | Datasets | $SV M_m$ | $\epsilon$-SVM | $SVM$-FDA | $SVM$ | $FDA$ | Kernel | Datasets | $SV M_m$ | $\epsilon$-SVM | $SVM$-FDA | $SVM$ | $FDA$ |
|--------|----------|----------|----------------|-----------|-------|-------|--------|----------|----------|----------------|-----------|-------|-------|
| Linear | sonar    | 21.63    | 23.08          | 20.67     | 32.69 | 27.88 | sonar  | 13.46    | 12.98    | 13.46         | 26.44     |       |       |
|        | wpbc     | 19.7     | 21.21          | 22.73     | 25.25 | 26.77 | wpbc   | 20.2     | 20.71    | 20.2         | 48.48     |       |       |
|        | ionosphere | 9.12    | 11.97          | 10.26     | 7.98  | 64.1  | ionosphere | 5.98     | 6.27     | 6.27         | 30.48     |       |       |
|        | wdbc     | 2.11     | 3.69           | 2.99      | 2.28  | 3.87  | wdbe   | 2.64     | 2.28     | 2.64         | 62.74     |       |       |
|        | liver    | 30.43    | 31.3           | 30.72     | 31.01 | 39.13 | liver   | 28.7     | 30.43    | 29.57        | 40.29     |       |       |
|        | musk1    | 13.03    | 15.76          | 13.24     | 17.02 | 20.8  | musk1   | 4.2      | 4.83     | 4.2         | 36.36     |       |       |
|        | ovariann | 4.35     | 4.35           | 3.95      | 4.35  | 48.22 | ovariann | 9.49     | 9.49     | 19.37        | 64.03     |       |       |
| Score  | McNemar  | 20.5     | 17             | 17        | 15    | 3.5   | Score  | McNemar  | 14.5     | 14.5     | 13            | 0        |       |       |
| Poly2  | sonar    | 17.79    | 17.31          | 19.23     | 41.83 |       | sonar   | 34.62    | 34.62    | 42.79        | 42.79     |       |       |
|        | wpbc     | 23.74    | 23.74          | 24.24     | 41.83 |       | wpbc    | 23.74    | 23.74    | 23.74        | 23.74     |       |       |
|        | ionosphere | 5.41    | 5.98           | 5.98      | 32.76 |       | ionosphere | 5.41     | 5.41     | 10.82        | 8.26      |       |       |
|        | wdbc     | 3.51     | 4.75           | 3.51      | 58.52 |       | wdbe   | 6.5      | 6.85     | 16.34        | 10.54     |       |       |
|        | liver    | 30.14    | 29.86          | 30.72     | 39.98 |       | liver   | 32.17    | 32.46    | 32.17        | 40.29     |       |       |
|        | musk1    | 6.09     | 6.93           | 6.3       | 50.84 |       | musk1   | 39.92    | 39.92    | 43.49        | 43.07     |       |       |
|        | ovariann | 9.09     | 9.09           | 15.81     | 64.03 |       | ovariann | 35.97    | 35.97    | 35.97        | 35.97     |       |       |
| Score  | McNemar  | 14.5     | 14.5           | 13        | 0     |       | Score  | McNemar  | 14.5     | 14.5     | 6            | 7        |       |       |

### VIII. Conclusion

From the results, it is apparent that one does not need only to control for the between-class distance, as standard $SV M$ does, but also for the within-class distance, since the incorporation of some measure of the latter in the optimization problem considerably improves the performance over the standard $SV M$. When it comes to determining which within-class distance measure is more appropriate then the width of the band containing the instances that we deployed in $SV M_m$ has a clear advantage since it does not only lead to the best performance but it results in a convex optimization problem which is easy to kernelize and in addition includes as a special case $\epsilon$-$SV M$.

### Appendix

We here present a formal comparison of $SV M$ and $MKL_{\gamma}$ under the metric-learning view.

From a metric learning perspective $SV M$ uses a single diagonal matrix transformation given by $W$ and $MKL_{\gamma}$ a two diagonal matrix transformation given by $W' = WD\sqrt{\mu}$. Both optimize the same cost function (i.e the margin). We will compare $SV M$ and $MKL_{\gamma}$ by comparing the use of one and two diagonal transformation matrices.

For $MKL_{\gamma}$, the $D\sqrt{\mu}$ matrix is block diagonal, while if we use that decomposition with $SV M$ it is a simple diagonal matrix. We denote by $w = diag(W)$, $w' = diag(W')$ and $\sqrt{\mu} = diag(D\sqrt{\mu})$. The optimization problems of $SV M$ and $MKL_{\gamma}$ can be formulated as follows:

**SVM:**

$$
\max_{w,b} \gamma \quad (24)
$$

subject to

$$
y_i(w^Tx_i + b) \geq \gamma, \forall i \\
\|w\|_p = 1
$$

**MKL:**

$$
\max_{w',b} \gamma \quad (25)
$$

subject to

$$
y_i((w' \circ \mu)^T x_i + b) \geq \gamma, \forall i \\
\|w'\|_p = 1, \|\mu\|_p = 1, \mu_i \geq 0
$$

We now try to highlight relationship of these two optimization problems. Given some $\mu$ vector such that $\mu_i \neq 0, \forall i$, $\|\mu\|_p = 1$, and $w = w' \circ \mu$, we define a new norm $l_\mu(w)$ as follows: $l_\mu(w) = \|w\|_p$. It is easy to prove that $l_\mu(w)$ satisfies all the properties of a norm, which means it is a valid norm.
Using this new norm, we can rewrite problem (25) as:

\[
\max_{w, b} \gamma \quad s.t. \quad y_i (w^T x_i + b) \geq \gamma, \forall i \quad (26)
\]

Let \( T_w \) be the feasible set of \( \|w\|_p = 1 \), obviously \( T_w \) is also the feasible set \( \|w\|_p = 1 \); \( T_{\mu} \) is the feasible set of \( \|\mu\|_p = 1 \). Let \( T_{w, \mu} = \{ w \circ \mu | \mu \in T_{\mu}, w \in T_w \} \). For a given value of \( \mu = c \) there is a one-to-one mapping from \( T_w \) to \( T_{w, \mu} = T_{w, c} \). The cardinality of the \( T_w \) feasible set is much smaller than the cardinality of \( T_{w, \mu} \) therefore using this new norm gives more flexibility in finding a solution of the optimization problem, which could potentially lead to a better solution.

In Figure 3 we illustrate the relations of the different norms just discussed in the two dimensional space. We compare the \( p=1 \) and \( p=2 \) norms of \( w \) to its \( l_\mu \) norm. We fix \( \|w\|_1 = 1 \), its feasible set \( T_w \) is the diamond, \( \|w\|_2 = 1 \), its feasible set \( T_w \) is the outer circle, and \( l_\mu(w) = 1 \), \( \|\mu\|_1 = 1 \), for which we have the set of feasible sets, \( T_{w, \mu} \), that correspond to the different values of \( \mu \) that satisfy \( \|\mu\|_1 = 1 \), which are given by the inner ellipsoids.

REFERENCES

[1] Argyriou, A., Micchelli, C.A., Pontil, M.: Learning convex combinations of continuously parameterized basic kernels. In: COLT 2005 (2005)
[2] Bach, F., Thibaux, R., Jordan, M.: Computing regularization paths for learning multiple kernels (2005)
[3] Cortes, C., Mohri, M., Rostamizadeh, A.: L2 regularization for learning kernels. In: UAI (2010)
[4] Cortes, C., Mohri, M., Rostamizadeh, A.: Two stage learning kernel algorithms. In: International Conference on Machine Learning (2010)
[5] Cristianini, N., Shawe-Taylor, J.: An introduction to Support Vector Machines. Cambridge University Press (2000)
[6] Cristianini, N., Shawe-Taylor, J., Elisseeff, A.: On the extensions of kernel alignment. In: Advances in Neural Information Processing Systems 14. MIT Press (2001)
[7] Davis, J., Kulis, B., Jain, P., Sra, S., Dhillon, I.: Information-theoretic metric learning. In: Proceedings of the 24th international conference on Machine learning. ACM New York, NY, USA (2007)
[8] Do, H., Kalousis, A., Hilario, M.: Feature weighting using margin and radius based error bound optimization in svms. In: European Conference on Machine Learning (2009)
[9] Do, H., Kalousis, A., Wang, J., Woznica, A.: A metric learning perspective of svm - on the relation of svm and lmnns. In: International Conference on Artificial Intelligence and Statistics (2012)
[10] Efron, B., Hastie, T., Johnstone, I., Tibshirani, R.: Least angle regression. Annals of Statistics (2004)
[11] Franc, V., Havac, V.: Statistical Pattern Recognition Toolbox. Czech Technical University Prague, http://cmp.felk.cvut.cz/cmp/software/stprtoolbox/
[12] Globerson, A., Roweis, S.: Metric learning by collapsing classes. In: Advances in Neural Information Processing Systems. vol. 18. MIT Press (2006)
[13] Goldberger, J., Roweis, S., Hinton, G., Salakhutdinov, R.: Neighbourhood components analysis. In: Advances in Neural Information Processing Systems. vol. 17. MIT Press (2005)
[14] J.Weston, S.Mukherjee, O.Chapelle, M.Pontil, J.Poggio, V.Vapnik: Feature selection for svms. vol. 13, pp. 668–674 (2000)
[15] Kloft, M., Brefeld, U., Sonnenburg, S., Zien, A.: lp multiple kernel learning. Journal of Machine Learning Research (2011)
[16] Lanckriet, G., Cristianini, N., Bartlett, P., Ghaoui, L.E.: Learning the kernel matrix with semidefinite programming. Journal of Machine Learning Research 5, 27–72 (2004)
[17] O.Duda, R., E.Hart, P., Stork, D.G.: Pattern Classification. A Wiley Interscience Publication (2001)
[18] Onb, C.S., Smola, A.J., Williamson, R.C.: Learning the kernel with hyperkernels. Journal of Machine Learning Research 6, 1043–1071 (2005)
[19] Rakotomamonjy, A.: Variable selection using svm-based criteria. Journal of Machine Learning Research 3, 1357–1370 (2003)
[20] Rakotomamonjy, A., Bach, F., Canu, S., Grandvalet, Y.: Simplėmlkl. Journal of Machine Learning Research 9, 2491–2521 (2008)
[21] Shivaswamy, P.K., Jebara, T.: Maximum relative margin and data-dependent regularization. Journal of Machine Learning Research 11, 747–788 (2010)
[22] Sonnenburg, S., Raetsch, G., Christin, Schaefer, Schoelkopf, B.: Large scale multiple kernel learning. Journal of Machine Learning Research 11, 1531–1565 (July 2006)
[23] Weinberger, K., Saul, L.: Distance metric learning for large margin nearest neighbor classification. The Journal of Machine Learning Research 10, 207–244 (2009)
[24] Xing, E., Ng, A., Jordan, M., Russell, S.: Distance metric learning with application to clustering with side-information. In: Advances in Neural Information Processing Systems (2003)
[25] Xiong, T., Cherkassky, V.: A combined svm and lda approach for classification. In: IJCNN (2005)