A metric learning perspective of SVM: on the relation of LMNN and SVM

Huyen Do
Computer Science Dept.
University of Geneva
Switzerland
Huyen.Do@unige.ch

Alexandros Kalousis
Business Informatics
University of Applied Science
West. Switzerland
Alexandros.Kalousis@hesge.ch

Jun Wang
Computer Science Dept.
University of Geneva
Switzerland
Jun.Wang@unige.ch

Adam Woznica
Computer Science Dept.
University of Geneva
Switzerland
Adam.Woznica@unige.ch

Abstract

Support Vector Machines, *SVM*s, and the Large Margin Nearest Neighbor algorithm, *LMNN*, are two very popular learning algorithms with quite different learning biases. In this paper we bring them into a unified view and show that they have a much stronger relation than what is commonly thought. We analyze *SVM*s from a metric learning perspective and cast them as a metric learning problem, a view which helps us uncover the relations of the two algorithms. We show that *LMNN* can be seen as learning a set of local *SVM*-like models in a quadratic space. Along the way and inspired by the metric-based interpretation of *SVM*s we derive a novel variant of *SVM*s, ϵ-*SVM*, to which *LMNN* is even more similar. We give a unified view of *LMNN* and the different *SVM* variants. Finally we provide some preliminary experiments on a number of benchmark datasets in which show that ϵ-*SVM* compares favorably both with respect to *LMNN* and *SVM*.

1 Introduction

Support Vector Machines, [2], and metric learning algorithms, [15, 8, 9, 16], are two very popular learning paradigms with quite distinct learning biases. In this paper we focus on *SVM*s and *LMNN*, one of the most prominent metric learning algorithms, [15]; we bring them into a unified view and show that they have a much stronger relation than what is commonly accepted.

We show that *SVM* can be formulated as a metric learning problem, a fact which provides new insights to it. Based on these insights and employing learning biases typically used in metric learning we propose ϵ-*SVM*, a novel *SVM* variant which is shown to empirically outperform *SVM*. ϵ-*SVM* relates, but is simpler, to the radius-margin ratio error bound optimization that is often used when *SVM* is coupled with feature selection and weighting, [12, 5], and multiple kernel learning [1, 4, 7]. More importantly we demonstrate a very strong and previously unknown connection between *LMNN* and *SVM*. Until now *LMNN* has been considered as the distance-based counterpart of *SVM*, in the somehow shallow sense that both use some concepts of margin, even though their respective margin concepts are defined differently, and the hinge loss function, within a convex optimization problem. We show that the relation between *LMNN* and *SVM* is much deeper and demonstrate that *LMNN* can be seen as a set of local *SVM*-like classifiers in a quadratic space. In fact we show that *LMNN* is even more similar to ϵ-*SVM* than *SVM*. This strong connection has the potential to lead to more efficient *LMNN* implementations, especially for large scale problems and vice versa, to lead to more efficient schema of multiclass *SVM*. Moreover the result is also valid for other large margin metric learning algorithms. Finally we use the metric-based view to provide a unified view of *SVM*, ϵ-*SVM* and *LMNN*.

Overall the main contribution of the paper is the unified view of *LMNN*, *SVM* and the variants of the latter. Along the way we also devise a new algorithm, ϵ-*SVM*, that combines ideas from both *SVM* and *LMNN* and finds its support in the *SVM* error bound.

In the next section we will briefly describe the basic concepts of *SVM* and *LMNN*. In section 2 we describe *SVM* in the metric learning view and the new insights that this view brings. We also discuss some invariant properties of *SVM* and how metric learning may or

Appearing in Proceedings of the 15^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2012, La Palma, Canary Islands. Volume XX of JMLR: W&CP XX. Copyright 2012 by the authors.

may not help to improve *SVM*. Section 3 describes ϵ-*SVM*, a metric-learning and *SVM* inspired algorithm. Section 4 discusses the relation of *LMNN* and *SVM*, and provides a unified view of the different *SVM* and *LMNN* variants. In section 6 we give some experimental results for ϵ-*SVM* and compare it with *SVM* and *LMNN*. Finally we conclude in section 7.

1.1 Basic *SVM* and *LMNN* concepts

We consider a binary classification problem in which we are given a set of n learning instances $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}, \mathbf{x}_i \in \mathcal{R}^d$, where y_i is the class label of the $\mathbf{x}_i$ instance and $y_i \in \{+1, -1\}$. We denote by $\|\mathbf{x}\|_p$ the l_p norm of the $\mathbf{x}$ vector. Let $H_{\mathbf{w}}$ be a hyperplane given by $\mathbf{w}^T\mathbf{x} + b = 0$; the signed distance, $d(\mathbf{x}_i, H_{\mathbf{w}})$, of some point $\mathbf{x}_i$ from the $H_{\mathbf{w}}$ hyperplane is given by: $d(\mathbf{x}_i, H_{\mathbf{w}}) = \frac{\mathbf{w}^T\mathbf{x_i}+b}{\|\mathbf{w}\|_p}$. *SVM*s learn a hyperplane $H_{\mathbf{w}}$ which separates the two classes and has maximum margin γ, which is, informally, the distance of the nearest instances from $H_{\mathbf{w}}$: $\gamma = \min_i[y_i d(\mathbf{x}_i, H_{\mathbf{w}})]_+$ [2]. The optimization problem that *SVM*s solve is:

$$\max_{\mathbf{w},b,\gamma} \ \gamma \qquad s.t. \quad \frac{y_i(\mathbf{w}^T\mathbf{x}_i + b)}{\|\mathbf{w}\|_2} \geq \gamma, \ \forall i \tag{1}$$

which is usually rewritten to avoid the uniform scaling problem with $\mathbf{w}, b$, as:

$$\min_{\mathbf{w},b} \ \|\mathbf{w}\|_2^2 \qquad s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \forall i \tag{2}$$

The margin maximization is motivated by the *SVM* error bound which is a function of the R^2/γ^2 ratio; R is the radius of the smallest sphere that contains the learning instances. Standard *SVM*s only focus on the margin and ignore the radius because for a given feature space this is fixed. However the R^2/γ^2 ratio has been used as an optimization criterion in several cases, for feature selection, feature weighting and multiple kernel learning [1, 12, 4, 5, 7]. The biggest challenge when using the radius-margin bound is that it leads to non convex optimization problems.

In metric learning we learn a Mahalanobis metric parametrized by a Positive Semi-Definite (PSD) matrix $\mathbf{M}$ under some cost function and some constraints on the Mahalanobis distances of same and different class instances. The squared Mahalanobis distance has the following form $d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T\mathbf{M}(\mathbf{x}_i - \mathbf{x}_j)$. $\mathbf{M}$ can be rewritten as $\mathbf{M} = \mathbf{L}^T\mathbf{L}$, i.e the Mahalanobis distance computation can be considered as a two-step procedure, that first computes a linear transformation of the instances given by the matrix $\mathbf{L}$ and then the Euclidean distance in the transformed space, i.e $d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = d^2(\mathbf{L}\mathbf{x}_i, \mathbf{L}\mathbf{x}_j)$.

LMNN, one of the most popular metric learning methods, also works based on the concept of margin which is nevertheless different from that of *SVM*. While the *SVM* margin is defined *globally* with respect to a hyperplane, the *LMNN* margin is defined *locally* with respect to center points. The *LMNN* margin, $\gamma_{\mathbf{x}_0}$, of a center point, instance $\mathbf{x}_0$, is given by:

$$\begin{aligned} \gamma_{\mathbf{x}_0} &= \min_{i,j}[d^2_{\mathbf{M}}(\mathbf{x}_0, \mathbf{x}_j) - d^2_{\mathbf{M}}(\mathbf{x}_0, \mathbf{x}_i)]_+ \\ & \quad\ y_0 \neq y_j, \mathbf{x}_i \in targ(\mathbf{x}_0) \end{aligned} \tag{3}$$

where $targ(\mathbf{x}_0)$ is the set of the *target neighbors* of $\mathbf{x}_0$, which is defined as $targ(\mathbf{x}_0) = \{\mathbf{x}_i | \mathbf{x}_i \in \text{neighborhood of } \mathbf{x}_0 \wedge y_0 == y_i\}$. The neighborhood can be defined either as a k nearest neighbors or as a sphere of some radius.

LMNN maximizes the sum of the margins of all instances. The underlying idea is that it learns a metric $\mathbf{M}$ or a linear transformation $\mathbf{L}$ which brings instances close to their same class center point while it moves away from it different class instances with a margin of one. This learning bias is commonly referred as large margin metric learning [13, 15, 14, 10]. *LMNN* optimization problem is [15]:

$$\begin{aligned} \min_{\mathbf{M},\boldsymbol{\xi}} \quad & \sum_i \sum_{\mathbf{x}_j \in targ(\mathbf{x}_i)} (d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) + C \sum_l (1 - y_i y_l)\xi_{ijl}) \\ s.t. \quad & d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l) - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \geq 1 - \xi_{ijl}, \\ & \forall (j | \mathbf{x}_j \in targ(\mathbf{x}_i)), \forall i, l; \ \boldsymbol{\xi} \geq 0, \ \ \mathbf{M} \succeq 0 \end{aligned} \tag{4}$$

2 *SVM* under a metric learning view

We can formulate the *SVM* learning problem as a metric learning problem in which the learned transformation matrix is diagonal. To do so we will first define a new metric learning problem and then we will show its equivalence to *SVM*.

We start by introducing the fixed hyperplane $H_{\mathbf{1}}$, the normal vector of which is $\mathbf{1}$ (a d-dimensional vector of ones), i.e. $H_1 : \mathbf{1}^T\mathbf{x} + b = 0$. Consider the following linear transformation $\tilde{\mathbf{x}} = \mathbf{W}\mathbf{x}$, where $\mathbf{W}$ is a $d \times d$ diagonal matrix with $diag(\mathbf{W}) = \mathbf{w} = (w_1, \dots, w_d)^T$.

We now define the margin γ of two classes with respect to the hyperplane H_1 as the minimum absolute difference of the distances of any two, different-class instances, $\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}_j}$, projected to the norm vector of $H_{\mathbf{1}}$ hyperplane, which can be finally written as: $\gamma = \min_{i,j,y_i \neq y_j} \frac{|\mathbf{1}^T\mathbf{W}(\mathbf{x}_i - \mathbf{x}_j)|}{\sqrt{d}} = \min_{i,j,y_i \neq y_j} |d(\tilde{\mathbf{x}}_i, H_{\mathbf{1}}) - d(\tilde{\mathbf{x}_j}, H_{\mathbf{1}})|$. In fact this is the *SVM* margin, see equation (17) in Appendix.

We are interested in that linear transformation, $diag(\mathbf{W}) = \mathbf{w}$, for which it holds $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 0$, i.e. the instances of the two classes lie on two different sides of the hyperplane $H_\mathbf{1}$, and which has a maximum margin. This is achieved by the following metric-learning optimization problem [1]:

$$\begin{aligned}
\max_{\mathbf{w}} \min_{i,j,y_i \neq y_j} \quad & (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{w}\mathbf{w}^T (\mathbf{x}_i - \mathbf{x}_j) && (5) \\
s.t. \quad & y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 0, \forall i && \\
\Leftrightarrow \max_{\mathbf{w},\gamma} \quad & \gamma^2 && (6) \\
s.t. \quad & (\mathbf{w}^T(\mathbf{x}_i - \mathbf{x}_j))^2 \geq \gamma^2, \forall (i, j, y_i \neq y_j) && \\
& y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 0, \forall i &&
\end{aligned}$$

Additionally we prefer the $\mathbf{W}$ transformation that places the two classes "symmetrically" with respect to the $H_\mathbf{1}$ hyperplane, i.e. the separating hyperplane is at the middle of the margin instances of the two classes: $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq \gamma/2, \forall i$. Notice that this can always be achieved by adjusting the translation parameter b by adding to it the $\frac{\gamma_1 - \gamma_2}{2}$ value, where γ_1 and γ_2 are respectively the margins of the y_1 and y_2 classes to the $H_\mathbf{1}$ hyperplane. Hence the parameter b of the H_1 hyperplane can be removed and replaced by an equivalent translation transformation b. From now on, we assume $H_\mathbf{1} : \mathbf{1}^T \mathbf{x} + 0 = 0$ and we add a translation b after the linear transformation $\mathbf{W}$. With this 'symmetry' preference, (6) can be reformulated as (see the detailed proof in Appendix):

$$\max_{\mathbf{w},b,\gamma} \gamma \qquad s.t. \;\; y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq \gamma, \forall i \qquad (7)$$

This optimization problem learns a diagonal transformation $\mathbf{W}$ and a translation b which maximize the margin and place the classes 'symmetrically' to the $H_1 : \mathbf{1}^T \mathbf{x} = 0$ hyperplane. However this optimization problem scales with the $\mathbf{w}$ (as is the case with the SVM formulation (1)). Therefore we need a way to control the uniform scaling of $\mathbf{w}$. One way is to fix some norm of $\mathbf{w}$ e.g. $\|\mathbf{w}\|_p = 1$, and the optimization problem becomes:

$$\max_{\mathbf{w},\gamma,b} \gamma \qquad s.t. \;\; y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq \gamma, \forall i, \|\mathbf{w}\|_p = 1 (8)$$

Notice that both problems (7) and (8) are still different from the standard SVM formulation given in (1) or (2). However it can be shown that they are in fact equivalent to SVM; for the detailed proof see in the Appendix.

Thus we see that starting from a Mahalanobis metric learning problem (5), i.e learning a linear transformation $\mathbf{W}$, and with the appropriate cost function and constraints on pairwise distances, we arrive to a standard SVM learning problem. We can describe SVM in the metric learning jargon as follows: it learns a diagonal linear transformation $\mathbf{W}$ and a translation b which maximize the margin and place the two classes symmetrically in the opposite sides of the hyperplane $H_\mathbf{1} : \mathbf{1}^T \mathbf{x} = 0$. 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 $H_\mathbf{1}$ and the space is scaled, $\mathbf{W}$, and then translated, b, so that the instances are placed optimally around $H_\mathbf{1}$. Introducing a fixed hyperplane $H_\mathbf{1}$ will provide various advantages in relation to the different radius-margin SVM versions and $LMNN$ as we will show soon. It is also worth to note that the SVM metric view holds true for any kernel space, since its final optimization problem is the same as that of standard SVM, i.e we can kernelize it directly as SVM.

From the metric learning perspective, one may think that learning a full linear transformation instead of a diagonal could bring more advantage, however this is not true for the case of SVM, see the detailed proof in Appendix.

3 ϵ-*SVM*, an alternative to the radius-margin approach

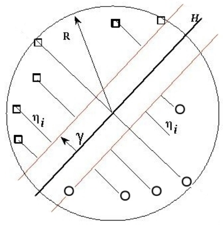


Figure 1: Within class distance: controlling the radius R or the instance-hyperplane distances ϵ_i.

A common bias in metric learning is to learn a metric which keeps instances of the same class close while pushing instances of different classes far away. This bias is often implemented through local or global constraints on the pair-wise distances of same-class and different-class instances which make the *between class distance* large and the *within class distance* small. Under the metric view of SVM we see that the learned linear transformation does control the between class distance by maximizing the margin, but it ignores the within class distance. Inspired by the metric learning biases we will extend the SVM under the metric view and incorporate constraints on the within-class distance which we will minimize.

We can quantify the within class distance with a number of different ways such as the sum of instance distances from the centers of their class, as in Fisher Discriminant Analysis (FDA) [11], or by the pairwise distances of the instances of each class, as it is done in several metric learning algorithms [16, 3, 15]. Yet another

[1] Notice that $(\mathbf{w}^T(\mathbf{x}_i - \mathbf{x}_j))^2 = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{w}\mathbf{w}^T (\mathbf{x}_i - \mathbf{x}_j)$ is the Mahalanobis distance associated with the rank 1 matrix $\mathbf{M} = \mathbf{w}\mathbf{w}^T$

way to indirectly minimize the within class distance is by minimizing the total data spread while maximizing the between class distance. One measure of the total data spread is the radius, R, of the sphere that contains the data; so by minimizing the radius-margin ratio, R^2/γ^2, we can naturally minimize the within class distance while maximizing the between. Interestingly the *SVM* theoretical error bound points towards the minimization of the same quantity, i.e. the radius-margin ratio, thus the proposed metric learning bias finds its theoretical support in the *SVM* error bound. However optimizing over the margin and the radius poses several difficulties since the radius is computed in a complex form, [12, 4, 7].

We can avoid the problems that are caused by the radius by using a different quantity to indirectly control the within class distance. We propose to minimize instead of the radius, the sum of the instance distances from the margin hyperplane, this sum is yet another way to control the data spread. We will call the resulting algorithm, which in addition to the margin maximization also minimizes the within-class-distance through the sum of the instance distances from the margin hyperplane, *ϵ-SVM*. The minimization of the sum has a similar effect to the minimization of the *SVM* radius, Figure 1. In a later section we will show that *ϵ-SVM* can be seen as a bridge between *LMNN* and *SVM*.

We define the optimization problem of *ϵ-SVM* as follows: select the transformations, one linear diagonal, $diag(\mathbf{W}) = \mathbf{w}$, and one translation, b, which maximize the margin and keep the instances symmetrically and within a small distance from the $H_\mathbf{1}$ hyperplane. This is achieved by the following optimization problem:

$$\begin{aligned}\min_{\mathbf{w},b} \quad & \mathbf{w}^T\mathbf{w} + \lambda\sum_i max(0, y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1) \qquad (9)\\ & + C\sum_i max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))\end{aligned}$$

$max(0, y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1)$ penalizes instances which lie on the correct side of the hyperplane but are far from the margin. $max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)$ is the *SVM* hinge loss which penalize instances violating the margin. (9) is equivalent to:

$$\begin{aligned}\min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\epsilon}} \quad & \mathbf{w}^T\mathbf{w} + \lambda\sum_i \epsilon_i + C\sum_i \xi_i \qquad (10)\\ s.t. \quad & 1 - \xi_i \leq y_i(\mathbf{w}^T\mathbf{x}_i + b) \leq 1 + \epsilon_i, \forall i, \boldsymbol{\xi}, \boldsymbol{\epsilon} \geq 0\end{aligned}$$

where ξ_i is the standard *SVM* slack variable for instance i and ϵ_i the distance of that instance from the margin hyperplane. This is a quadratic programing problem which can be solved in a similar manner to standard *SVM*, although it has double number of linear constraints, compared to that of *SVM*. Its dual form is as follows:

$$\begin{aligned}\max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \quad & \tfrac{-1}{2}\textstyle\sum_{ij}(\alpha_i - \beta_i)(\alpha_j - \beta_j)y_iy_j\mathbf{x}_i\mathbf{x}_j + \sum_i(\alpha_i - \beta_i)\\ s.t. \quad & \textstyle\sum_i(\alpha_i - \beta_i)y_i = 0;\ 0 \leq \alpha_i \leq C;\ 0 \leq \beta_i \leq \lambda, \forall i (11)\end{aligned}$$

Note that we have two hyper-parameters C and λ, typically we will assign a higher value to the C since we tolerate less the violations of the margin compared to a larger data spread. The two parameters control the trade off between the importance of the margin and the within data spread.

4 On the relation of *LMNN* and *SVM*

In this section we demonstrate the relation between *LMNN* and *SVM*. Lets define a quadratic mapping $\mathbf{\Phi}$ that maps $\mathbf{x}$ to a quadratic space where $\mathbf{\Phi}(\mathbf{x}) = (x_1^2, x_2^2, \ldots, x_d^2, x_ix_{j\{d\geq i>j\geq 1\}}, x_1, x_2, \ldots, x_d) \in \mathcal{R}^{d'}$, $d' = \frac{d(d+3)}{2}$. We will show that the *LMNN* margin in the original space is the *SVM* margin in this quadratic space.

Concretely, the squared distance of an instance $\mathbf{x}$ from a center point instance $\mathbf{x}_l$ in the linearly transformed space that corresponds to the Mahalanobis distance $\mathbf{M}$ learned by *LMNN* can be expressed in a linear form of $\mathbf{\Phi}(\mathbf{x})$. Let $\mathbf{L}$ be the linear transformation associated with the learned Mahalanobis distance $\mathbf{M}$. We have:

$$\begin{aligned}d^2_{\mathbf{M}}(\mathbf{x}, \mathbf{x}_l) &= d^2(\mathbf{L}\mathbf{x}, \mathbf{L}\mathbf{x}_l) \qquad (12)\\ &= \mathbf{x}^T\mathbf{L}^T\mathbf{L}\mathbf{x} - 2\mathbf{x}_l^T\mathbf{L}^T\mathbf{L}\mathbf{x} + \mathbf{x}_l^T\mathbf{L}^T\mathbf{L}\mathbf{x}_l\\ &= \mathbf{w}_l^T\mathbf{\Phi}(\mathbf{x}) + b_l\end{aligned}$$

where $b_l = \mathbf{x}_l^T\mathbf{L}^T\mathbf{L}\mathbf{x}_l$ and $\mathbf{w}_l$ has two parts, quadratic $\mathbf{w}^{quad}$ and linear $\mathbf{w}_l^{lin}$: $\mathbf{w}_l = (\mathbf{w}^{quad}, \mathbf{w}_l^{lin})$ where $\mathbf{w}^{quad}$ is the vector of the coefficients of the quadratic components of $\mathbf{\Phi}(\mathbf{x})$, with $d' - d$ elements given by the elements of $\mathbf{L}^T\mathbf{L}$, and $\mathbf{w}_l^{lin}$ is equal to $-2\mathbf{L}^T\mathbf{L}\mathbf{x}_l$—the vector of coefficients of the linear components of $\mathbf{\Phi}(\mathbf{x})$. $d^2(\mathbf{L}\mathbf{x}, \mathbf{L}\mathbf{x}_l)$ is proportional to the distance of $\mathbf{\Phi}(\mathbf{x})$ from the hyperplane $H_l : \mathbf{w}_l^T\mathbf{\Phi}(\mathbf{x}) + b_l = 0$ since $d^2(\mathbf{L}\mathbf{x}, \mathbf{L}\mathbf{x}_l) = \|\mathbf{w}_l\|d(\mathbf{\Phi}(\mathbf{x}), H_l)$. Notice that this value is always non negative, so in the quadratic space $\mathbf{\Phi}(\mathbf{x})$ always lies on the positive side of H_l.

A more general formulation of the *LMNN* optimization problem (4) which reflects the same learning bias of *LMNN*, i.e for each center point, keeps the same-class instances close to the center and pushes different-class instances outside the margin, is:

$$\begin{aligned}\min_{\mathbf{M},\boldsymbol{\xi},\gamma_{\mathbf{x}_l},\forall l} \quad & \sum_l \frac{1}{\gamma^2_{\mathbf{x}_l}} + \lambda \sum_l \sum_{\mathbf{x}_i \in targ(\mathbf{x}_l)} (d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l) \\ + \quad & C \sum_j (1 - y_j y_l) \xi_{ijl}) \qquad (13) \\ s.t. \quad & d^2_{\mathbf{M}}(\mathbf{x}_j, \mathbf{x}_l) - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l) \geq \gamma_{\mathbf{x}_l} - \xi_{ijl}, \\ & \forall (i|\mathbf{x}_i \in targ(\mathbf{x}_l)), \forall j, l; \\ & \gamma_{\mathbf{x}_l} \geq 0, \forall l, \quad \xi \geq 0, \quad \mathbf{M} \succeq 0\end{aligned}$$

Standard *LMNN* puts one extra constraint on each $\gamma_{\mathbf{x}_l}$, it sets each of them to one. With this constraint problem (13) reduces to (4). We can rewrite problem (13) as:

$$\begin{aligned}\min_{\mathbf{L},\boldsymbol{\xi},R_{l,\forall l},\gamma_{\mathbf{x}_l},\forall l} \quad & \sum_l \frac{1}{\gamma^2_{\mathbf{x}_l}} + C \sum_{li} \xi_{li} \qquad (14) \\ + \quad & \lambda \sum_l \sum_{\mathbf{x}_i \in targ(\mathbf{x}_l)} d^2(\mathbf{L}\mathbf{x}_i, \mathbf{L}\mathbf{x}_l) \\ s.t.\ d^2(\mathbf{L}\mathbf{x}_i, \mathbf{L}\mathbf{x}_l) \quad \leq \quad & R_l^2 + \xi_{li}, \forall \mathbf{x}_i \in targ(\mathbf{x}_l); \forall \mathbf{x}_l \\ d^2(\mathbf{L}\mathbf{x}_j, \mathbf{L}\mathbf{x}_l) \quad \geq \quad & R_l^2 + \gamma_{\mathbf{x}_l} - \xi_{lj}, \forall (\mathbf{x}_j | y_j \neq y_l); \forall \mathbf{x}_l, \\ & \gamma_{\mathbf{x}_l} \geq 0, \forall l, \boldsymbol{\xi} \geq 0\end{aligned}$$

In this formulation, visualized in fig. 2(a), the target neighbors (marked with ×) of the $\mathbf{x}_l$ center point are constrained within the C_l circle with radius R_l and center $\mathbf{L}\mathbf{x}_l$ while the instances that have a label which is different from that of $\mathbf{x}_l$ (marked with □) are placed outside the circle C''_l with center also $\mathbf{L}\mathbf{x}_l$ and radius $R''_l = \sqrt{R_l^2 + \gamma_{\mathbf{x}_l}}$. We denote by β_l the difference $R''_l - R_l$ of the radii of the two circles.

If we replace all the $d^2(\mathbf{L}\mathbf{x}, \mathbf{L}\mathbf{z})$ terms in problem (14) with their linear equivalent in the quadratic space, problem (12), and break up the optimization problem to a series of n optimization problems one for each instance $\mathbf{x}_l$ then we get for each $\mathbf{x}_l$ the following optimization problem:

$$\begin{aligned}\min_{\mathbf{w}_l,b'_l,\boldsymbol{\xi}_l,\gamma_{\mathbf{x}_l}} \quad & \frac{1}{\gamma^2_{\mathbf{x}_l}} + C \sum_{\mathbf{x}_i \in B(\mathbf{x}_l)} \xi_{li} \qquad (15) \\ + \quad & \lambda \sum_{\mathbf{x}_i \in targ(\mathbf{x}_l)} (\mathbf{w}_l^T(\boldsymbol{\Phi}(\mathbf{x}_i) - \boldsymbol{\Phi}(\mathbf{x}_l))) \\ s.t.\ \mathbf{w}_l^T \boldsymbol{\Phi}(\mathbf{x}_i) + b'_l \quad \leq \quad & -\gamma_{\mathbf{x}_l}/2 + \xi_{li}, \forall \mathbf{x}_i \in targ(\mathbf{x}_l) \\ \mathbf{w}_l^T \boldsymbol{\Phi}(\mathbf{x}_j) + b'_l \quad \geq \quad & \gamma_{\mathbf{x}_l}/2 - \xi_{lj}, \forall \mathbf{x}_j, y_j \neq y_l \\ \mathbf{w}_l^T(\boldsymbol{\Phi}(\mathbf{x}_i) - \boldsymbol{\Phi}(\mathbf{x}_l)) \quad \geq \quad & 0, \forall \mathbf{x}_i \in B(\mathbf{x}_l), \ \gamma_{\mathbf{x}_l} \geq 0, \boldsymbol{\xi}_l \geq 0\end{aligned}$$

where $\mathbf{w}_l$ is not an independent variable but its quadratic and linear components are related as described above (formula (12)). The instance set $B(\mathbf{x}_l) = \{\mathbf{x}_i \in targ(\mathbf{x}_l)\} \cup \{\mathbf{x}_i | y_i \neq y_l\}$ is the set of all target neighbors and different class instances. In the formula (15) we replace b_l by $-\mathbf{w}_l^T \boldsymbol{\Phi}(\mathbf{x}_l)$ since $\mathbf{w}_l^T \boldsymbol{\Phi}(\mathbf{x}_l) + b_l = 0$, and $b'_l = b_l - (R_l^2 + \gamma_{\mathbf{x}_l}/2)$. We denote the hyperplane $\mathbf{w}_l^T \boldsymbol{\Phi}(\mathbf{x}) + b'_l = 0$ by H'_l, as in fig. 3.

Now let $y_{li} := y_l y_i$, so $y_{li} = 1, \forall \mathbf{x}_i \in targ(\mathbf{x}_l)$ and $y_{lj} = -1, \forall (\mathbf{x}_j | y_i \neq y_l)$. Therefore $-y_{li}(\mathbf{w}_l^T \boldsymbol{\Phi}(\mathbf{x}_i) + b'_l) \geq \gamma_{\mathbf{x}_l}/2 - \xi_{li}, \forall \mathbf{x}_i \in B(\mathbf{x}_l)$. Without loss of generality we can assume that $y_l = -1$ and problem (15) becomes:

$$\begin{aligned}\min_{\mathbf{w}_l,b'_l,\boldsymbol{\xi}_l,\gamma_{\mathbf{x}_l}} \quad & \frac{1}{\gamma^2_{\mathbf{x}_l}} + C \sum_{\mathbf{x}_i \in B(\mathbf{x}_l)} \xi_{li} \qquad (16) \\ + \quad & \lambda \sum_{\mathbf{x}_i \in targ(\mathbf{x}_l)} (\mathbf{w}_l^T(\boldsymbol{\Phi}(\mathbf{x}_i) - \boldsymbol{\Phi}(\mathbf{x}_l))) \\ s.t.\ y_i(\mathbf{w}_l^T \boldsymbol{\Phi}(\mathbf{x}_i) + b'_l) \quad \geq \quad & \gamma_{\mathbf{x}_l}/2 - \xi_{li}, \forall \mathbf{x}_i \in B(\mathbf{x}_l) \\ \mathbf{w}_l^T(\boldsymbol{\Phi}(\mathbf{x}_i) - \boldsymbol{\Phi}(\mathbf{x}_l)) \quad \geq \quad & 0, \forall \mathbf{x}_i \in B(\mathbf{x}_l), \ \gamma_{\mathbf{x}_l} \geq 0, \boldsymbol{\xi}_l \geq 0\end{aligned}$$

where $\mathbf{w}_l$ is still constrained as in (15). It is worth to note that the second constraint of (16) will ensure that $d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l)$ is bigger than or equal to 0, i.e it will *almost* ensure the PSD of the matrix $\mathbf{M}$ [2]. However due to the specific structure of $\mathbf{w}_l$, i.e. quadratic and linear part, the PSD constraint of $\mathbf{M}$ is guaranteed for any $\mathbf{x}$.

We can think of problem (16) as an extension of an *SVM* optimization problem. Its training set is the $B(\mathbf{x}_l)$ set, i.e. the target neighbors of $\mathbf{x}_l$ and all instances with different class label from $\mathbf{x}_l$. Its cost function includes, in addition to the term that maximizes the margin, also a sum term which forces the target neighbors of $\mathbf{x}_l$ to have small $\mathbf{w}_l^T \boldsymbol{\Phi}(\mathbf{x}_i) + b_l$ values, i.e. be at small distance from the H_l hyperplane.

Minimizing the target neighbor distances from the H_l hyperplane makes the distance between support vectors and H_l small. It therefore has the effect of bringing the negative margin hyperplane of H'_l close to H_l, bringing thus also the target neighbors close to the negative margin hyperplane. In other words the optimization problem favors a small width for the band that is defined by H_l and the negative margin hyperplane described above which contains the target neighbors.

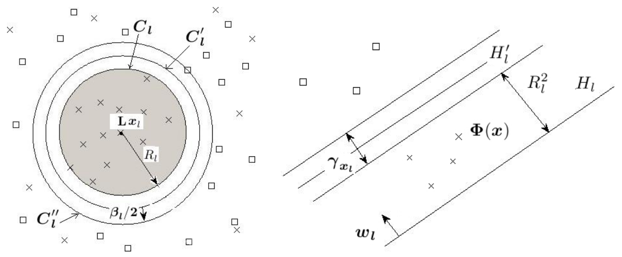
There is an even closer relation of *LMNN* with a local ϵ-*SVM* applied in the quadratic space that we will describe by comparing the optimization problems given in (9) and (16). ϵ-*SVM* has almost the same learning bias as *LMNN*, the former maximizes the margin and brings all instances close to the margin hyperplanes, the latter also maximizes the margin but only brings the target neighbors close to the margin hyperplane. For each center point $\mathbf{x}_l$ the *LMNN* formulation, problem (16), is very similar to that of ϵ-*SVM*, problem (9), applied in the quadratic space. The second term of the cost function of ϵ-*SVM* together with its constraints force all instances to be close to their

[2] *Almost* in the sense that the constraint ensures that $(\mathbf{x} - \mathbf{x}_l)^T \mathbf{M} (\mathbf{x} - \mathbf{x}_l) \geq 0$ for all $\mathbf{x}, \mathbf{x}_l$ in the training dataset, but can not guarantee the same for a new instance $\mathbf{x}$; to ensure $\mathbf{M}$ is PSD, we need $\mathbf{z}^T \mathbf{M} \mathbf{z} \geq 0$ for *all* $\mathbf{z}$.

respective margin hyperplane, this can be seen more clear in the minimization of the $\sum \epsilon_i$ in problem (10). In *LMNN*, problem (16), the third term of the cost function plays a similar role by forcing only the target neighbors to be close to the H_l hyperplane. This in turn has the effect, as mentioned above, to reduce the width of the band containing them and bringing them closer to their margin hyperplane; unlike ϵ-*SVM* no such constraint is imposed on the remaining instances. So *LMNN* is equivalent to n local, modified, ϵ-*SVM*s, one for each instance. ϵ-*SVM* controls the within class distance using the distance to the margin hyperplane and *LMNN* using the distance to the H_l hyperplane.

Note that the n optimization problems are not independent. The different solutions $\mathbf{w}_l$ have a common component which corresponds to the $d(d+1)/2$ quadratic features and is given by the $\mathbf{w}^{quad}$ vector. The linear component of $\mathbf{w}_l$, $\mathbf{w}_l^{lin}$, is a function of the specific $\mathbf{x}_l$, as described above. Overall learning a transformation $\mathbf{L}$ with *LMNN*, eq. (4), is equivalent to learning a local *SVM*-like model, given by H'_l, for *each* $\mathbf{x}_l$ center point in the quadratic space according to problems (15,16). Remember that the $\mathbf{w}_l, b_l$ of the different center points are related, i.e. their common quadratic part is $\mathbf{w}^{quad}$. If we constrain $\mathbf{w}_l, b_l$ to be the same for all $\mathbf{x}_l$ and drop the PSD constraints on $\mathbf{w}_l$ then we get the (global) ϵ-*SVM*-like solution.

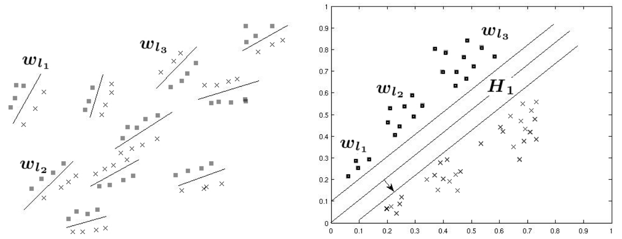
Visualization: In fig. 2(b) we give a visualization of problem (15) in the quadratic space $\mathbf{\Phi}(\mathbf{x})$. Figure 2(b) gives the equivalent linear perspective in the quadratic space of the *LMNN* model in the original space given in fig. 2(a). The center $\mathbf{Lx}_l$ of the C_l circle in fig. 2(a) corresponds to the H_l hyperplane in fig. 2(b); the C'_l circle with center $\mathbf{Lx}_l$ and radius $R'_l = R_l + \beta_l/2$ corresponds to the H'_l separating hyperplane in fig. 2(b). Figure 3(a) illustrates the different local linear models in the quadratic space. We can combine these different local models by employing the metric learning view of *SVM* and make the relation of *LMNN* and *SVM* even more clear. Instead of having many local *SVM*-like hyperplanes we bring each point $\mathbf{\Phi}(\mathbf{x}_l)$ around the $H_\mathbf{1} : \mathbf{1x} + 0 = 0$ hyperplane by applying to it first a $\mathbf{W}_l$ diagonal transformation, $\mathbf{W}_l = \mathbf{w}_l$, and then a translation (fig. 3(b)). As before with $\mathbf{w}_l$ the different $\mathbf{W}_l$ transformations have a common component, which corresponds to the first $d(d+1)/2$ elements of the diagonal associated with the quadratic features, given by the $\mathbf{w}^{quad}$ vector, and an instance dependent component that corresponds to the linear features which is given by $\mathbf{w}_l^{lin} = -2\mathbf{L}^T\mathbf{Lx}_l$; thus the translation transformation is also a function of the specific point $\mathbf{x}_l$. Notice that the common component has many more elements than the instance specific component. There is an analogy to multitask learning where models are learned over different tasks—datasets are forced to have parts of their models the same.



(a) Standard *LMNN* model view

(b) *LMNN* model view under an *SVM*-like interpretation

Figure 2: Alternative views on *LMNN*



(a) *LMNN* in a local *SVM*-like view

(b) *LMNN* in an *SVM* metric learning view

Figure 3: On the relation of *LMNN* and *SVM*

Prediction phase Typically after learning a metric through *LMNN* a k-NN algorithm is used to predict the class of a new instance. Under the interpretation of *LMNN* as a set of local ϵ-*SVM* linear classifiers this is equivalent to choosing the k local hyperplanes $H'_l : \mathbf{w}_l^T\mathbf{\Phi}(\mathbf{x}) + b'_l = 0$ which are farther away from the new instance and which leave both the new instance and their respective center points on the same side. The farther away the new instance is from an H'_l local hyperplane the closer it is to the center point associated to H'_l. In effect this means that we simply choose those hyperplanes—classifiers which are more confident on the label prediction of the new instance, and then we have them vote. In a sense this is similar to an ensemble learning schema in which each time we want to classify a new instance, we select those classifiers that are most confident. Then we have them vote in order to determine the final prediction.

5 A unified view of *LMNN*, *SVM* and its variants

The standard *SVM* learning problem only optimizes the margin. On the other hand, *LMNN*, as well as the different variants of radius-margin based *SVM* under the metric learning view (briefly mentioned in section 3), and ϵ-*SVM*, have additional regularizers that

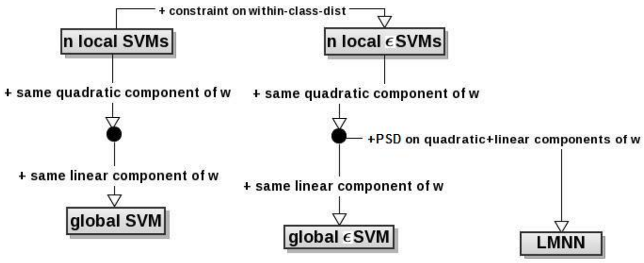

+ constraint on within-class-dist
n local SVMs
n local ϵSVMs
+ same quadratic component of w
+ same quadratic component of w
+PSD on quadratic+linear components of w
+ same linear component of w
+ same linear component of w
global SVM
global ϵSVM
LMNN

Figure 4: Relation among local SVM, local ϵ-SVM, global SVM, global ϵ-SVM and $LMNN$ in the quadratic space.

control directly or indirectly the within class distance. $LMNN$ is much closer to a collection of local ϵ-SVMs than to local SVMs, since both $LMNN$ and ϵ-SVM have that additional regularizer of the within class distance. We briefly summarize the different methods that we discussed:

- In standard SVM the space is fixed and the separating hyperplane is moved to find an optimal position in which the margin is maximized.
- In standard $LMNN$ we find the linear transformation that maximizes the sum of the local margins of all instances while keeping the neighboring instances of the same class close.
- In the metric learning view of SVM we find a linear diagonal transformation $\mathbf{W}, diag(\mathbf{W}) = \mathbf{w}$ plus a translation b that maximize the margin to the fixed hyperplane $H_\mathbf{1}$, clamping the norm of $\mathbf{w}$ to 1, or maximize the norm of $\mathbf{w}$ while clamping the margin to 1.
- In ϵ-SVM we find a linear diagonal transformation and a translation that maximize the margin with respect to the fixed hyperplane $H_\mathbf{1}$ and keep all instances close to $H_\mathbf{1}$.
- Finally in the interpretation of $LMNN$ as a collection of local SVM-like classifiers we find for *each* instance $\mathbf{x}$ a linear transformation and a translation that clamp the margin to one and keep the target neighbors of $\mathbf{x}$ within a band the width of which is minimized. These transformations have common components as described in the previous section. $LMNN$ is set of local related ϵ-SVMs in the quadratic space.

We will now give a high level picture of the relations of the different methods in the quadratic space $\mathbf{\Phi}(\mathbf{x})$ by showing how from one method we can move to another by adding or dropping constraints; the complete set of relations is given in Figure 4.

We start with n local non-related SVMs, if we add a constraint on the within-class distance to each of them we get n non-related ϵ-SVMs. If we add additional constrains that relate the different local ϵ-SVMs, namely by constraining their quadratic components $\mathbf{w}^{quad}$ to be the same and PSD, and their linear component $\mathbf{w}_l^{lin}$ to be a function of $\mathbf{w}^{quad}$ and the center point $\mathbf{x}_l$ then we get $LMNN$.

If we go back to the original set of non-related SVMs and add a constraint that forces all of them to have the same $\mathbf{w}_l$ and b_l' we get the standard global SVM. Similarly if in the collection of the local ϵ-SVMs we add a constraint that forces all of them to have the same optimal hyperplane then we also get the global ϵ-SVM. In that case all the component classifiers of the ensemble reduce to one classifier and no voting is necessary.

$LMNN$ constrains the n local ϵ-SVMs to have the same quadratic component, $\mathbf{w}^{quad}$, makes the linear components dependent on $\mathbf{w}^{quad}$ and their respective center points, accommodating like that the local information, and constrains the quadratic components $\mathbf{w}^{quad}$ to reflect the $\mathbf{L}^T\mathbf{L}$ PSD matrix, equation (12). On the other hand both the global SVM and ϵ-SVM also constrain their quadratic components to be the same but remove the constraint on the PSD property of $\mathbf{w}^{quad}$; both constrain all their linear components to be the same, thus they do not incorporate local information. The last constraint is much more restrictive than the additional constrains of $LMNN$, as a result the global SVM and ϵ-SVM models are much more constrained than that of $LMNN$. Although we demonstrate the relations only for the case of SVM and $LMNN$ they also hold for all other large margin based metric learning algorithms [13, 15, 14, 10].

ϵ-SVM builds upon two popular learning biases, namely large margin learning and metric learning and exploits the strengths of both. In addition it finds theoretical support in the radius-margin SVM error bound. ϵ-SVM can be seen as a bridge which connects SVM and existing large margin based metric learning algorithms, such as $LMNN$.

6 Experiments and results

In addition to studying in detail the relation between the different variants of SVM and metric learning we also examine the performance of the ϵ-SVM algorithm. We experiment with ϵ-SVM, eq. (10), and compare its performance to that of a standard l_1 soft margin SVM, $LMNN$ and FDA, with the following kernels: linear, polynomial with degree 2 and 3, Gaussian with $\sigma = 1$. C is chosen with 10-fold inner Cross-Validation from the set $\{0.1, 1, 10, 100, 1000\}$. For ϵ-SVM we choose to set λ to $C/3$ reflecting the fact that we tolerate less the margin violations than a larger distance form the margin. The full sensitivity analysis and cross valida-

Table 1: Classification error. D gives the number of features of each dataset and N the number of examples. A **bold entry** indicates that the respective method had the lowest error. The number in parentheses indicates the score of the respective algorithm for a given dataset and kernel based on the pairwise comparisons of the McNemar's statistical test.

Kernel	Dataset	*SVM*	ϵ-*SVM*	*LMNN*	*FDA*	Dataset	*SVM*	ϵ-*SVM*	*LMNN*	*FDA*
linear	N=62	**17.74** (2)	**17.74** (2)	20.97 (2)	50.00 (0)	N=198	25.25 (1.5)	**21.21** (2)	31.31 (1)	26.77 (1.5)
poly2	D=2000	**35.48** (1.5)	**35.48** (1.5)	**35.48** (1.5)	45.16 (1.5)	D=34	24.24 (2)	**23.74** (2)	29.80 (2)	41.41 (0)
poly3		35.48 (1.5)	**24.19** (2)	27.42 (2)	50.00 (0.5)		**20.20** (2.5)	20.71 (2.5)	31.31 (1)	48.48 (0)
gauss1	colonTumor	35.48 (1.5)	35.48 (1.5)	35.48 (1.5)	35.48 (1.5)	wpbc	23.74 (1.5)	23.74 (1.5)	23.74 (1.5)	23.74 (1.5)
linear	N=60	40.00 (1.5)	**35.00** (1.5)	45.00 (1.5)	**51.67** (1.5)	N=351	**7.98** (2.5)	11.97 (1.5)	8.26 (2)	64.10 (0)
poly2	D=7129	35.00 (2)	35.00 (2)	35.00 (2)	65.00 (0)	D=34	**5.98** (2)	**5.98** (2)	7.12 (2)	32.76 (0)
poly3		36.67 (1.5)	36.67 (1.5)	**33.33** (1.5)	50.00 (1.5)		6.27 (2)	6.27 (2)	**5.98** (2)	30.48(0)
gauss1	centralNS	35.00 (1.5)	35.00 (1.5)	35.00 (1.5)	35.00 (1.5)	ionosphere	10.82 (1)	**5.41** (2.5)	15.38 (0)	8.26 (2.5)
linear	N=134	10.45 (1.5)	**6.72** (2)	11.94 (1)	7.46 (1.5)	N=569	**2.28** (1.5)	3.69 (1.5)	4.04 (1.5)	3.87 (1.5)
poly2	D=1524	60.45 (1.5)	**53.73** (1.5)	54.48 (1.5)	52.24 (1.5)	D=30	**3.51** (2)	4.75 (2)	4.22 (2)	58.52 (0)
poly3		29.85 (1.5)	**22.39** (1.5)	29.85 (1.5)	22.39 (1.5)		2.64(2)	**2.28** (2)	3.34 (2)	630.23 (0)
gauss1	femaleVsMale	60.45 (1.5)	60.45 (1.5)	60.45 (1.5)	60.45 (1.5)	wdbc	16.34 (0)	**6.85** (3)	11.25 (1.5)	10.54 (1.5)
linear	N=72	**1.39** (2)	**1.39** (2)	**1.39** (2)	12.5 (0)	N=354	**31.01** (2)	31.30 (2)	36.23 (1.5)	39.13 (0.5)
poly2	D=7129	34.72 (1.5)	**31.94** (1.5)	33.33 (1.5)	43.06 (1.5)	D=6	30.72 (2)	**29.86** (2.5)	36.23 (1)	39.98 (0.5)
poly3		34.72 (0.5)	**12.50** (2.5)	33.33 (0.5)	13.89 (2.5)		**29.57** (2)	30.43 (2)	31.59 (2)	40.29 (0)
gauss1	Leukemia	34.72 (1.5)	34.72 (1.5)	34.72 (1.5)	34.72 (1.5)	liver	**32.17** (2.5)	32.46 (2.5)	38.26 (0.5)	40.29 (0.5)
linear	N=208	32.69 (0.5)	23.08 (1.5)	**12.02** (3)	27.88 (1)	N=476	17.02 (1)	15.76 (1.5)	**4.83** (3)	20.80 (0.5)
poly2	D=60	19.23 (2)	17.31 (2)	**13.94** (2)	41.83 (0)	D=166	6.30 (2)	6.93 (1.5)	**3.99** (2.5)	50.84(0)
poly3		13.46 (2)	12.98 (2)	**12.50** (2)	26.44 (0)		**4.20** (2)	4.83 (2)	5.25 (2)	36.36 (0)
gauss1	sonar	42.79 (1)	**34.62** (3)	42.79 (1)	42.79 (1)	musk1	43.49 (1)	**39.92** (3)	43.07 (1)	43.07 (1)

Table 2: McNemar score. loose: 0, win: 1, equal 0.5

Kernel	*SVM*	ϵ-*SVM*	*LMNN*	*FDA*
linear	16	17.5	18.5	8
poly2	18.5	18.5	18	5
poly3	17.5	20	16.5	6
gauss1	13	21.5	11.5	14
Total	65	77.5	64.5	33

tion on both C and λ are left for future work. We used ten datasets mainly from the UCI repository [6]. Attributes are standardized to have zero mean and one variance; kernels are normalized to have a trace of one. The number of target neighbors for *LMNN* is set to 5 and its γ parameter is set to 0.5, following the default settings suggested in [15]. We estimated the classification error using 10-fold CV. The results are given in Table 1. Overall each algorithm is applied 40 times (four kernels × ten datasets). Comparing ϵ-SVM with SVM we see that the former has a lower error than SVM in 19 out of the 40 applications and a higher error in only nine. A similar picture appears in the ϵ-*SVM*, *LMNN*, pair where the former has a lower error in 20 out of the 40 applications and a higher in only eight. If we break down the comparison per kernel type, we also see that, i.e. ϵ-SVM has a systematic advantage over the other two algorithms no matter which kernel we use.

To examine the statistical significance of the above results we use the McNemar's test of significance, with a significance level of 0.05. Comparing algorithms A and B on a fixed dataset and a fixed kernel algorithm A is credited with one point if it was significantly better than algorithm B, 0.5 points if there was no significance difference between the two algorithms, and zero points otherwise. In the last rows of Table 1 we report the overall points that each algorithms got and in addition we break them down over the different kernels. ϵ-*SVM* has the highest score with 77.5 points followed by SVM with 65 and *LMNN* with 64.5. *FDA* is far worse with only 33 points. The advantage of ϵ-SVM is much more pronounced in the case of the Gaussian kernel. This could be attributed to its additional regularization on the within-class distance which makes it more appropriate for very high dimensional spaces.

7 Conclusion

In this paper, we have shown how SVM learning can be reformulated as a metric learning problem. Inspired by this reformulation and the metric learning biases, we proposed ϵ-*SVM*, a new *SVM*-based algorithm in which, in addition to the standard SVM constraints we also minimize a measure of the within class distance. More importantly the metric learning view of SVM helped us uncover a so far unknown connection between the two seemingly very different learning paradigms of SVM and *LMNN*. *LMNN* can be seen as a set of local *SVM*-like classifiers in a quadratic space, and more precisely, as a set of local ϵ-*SVM*-like classifiers. Finally preliminary results show a superior performance of ϵ-SVM compared to both SVM and *LMNN*. Although our discussion was limited to binary classification, it can be extended to the multiclass case. Building on the *SVM*-*LMNN* relation, our current work focuses on the full analysis of the multiclass case, a new schema for multiclass SVM which exploits the advantages of both *LMNN* and kNN in multiclass problems, and finally the exploration of the learning models which are in between the SVM, *LMNN* models.

References

[1] Chapelle, O., Vapnik, V., Bousquet, O., Mukherjee, S.: Choosing multiple parameters for support vector machines. In: Machine Learning. vol. 46, pp. 131–159. Kluwer Academic Publishers, Hingham, MA, USA (2002)

[2] Cristianini, N., Shawe-Taylor, J.: An introduction to Support Vector Machines. Cambridge University Press (2000)

[3] 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)

[4] Do, H., Kalousis, A., Hilario, M.: Feature weighting using margin and radius based error bound optimization in svms. In: ECML (2009)

[5] Do, H., Kalousis, A., Wozica, A., Hilario, M.: Margin radius based multiple kernel learning. In: ECML (2009)

[6] Frank, A., Asuncion, A.: UCI machine learning repository (2010), http://archive.ics.uci.edu/ml

[7] Gai, K., Chen, G., Zhang, C.: Learning kernels with radiuses of minimum enclosing balls. In: NIPS (2010)

[8] Globerson, A., Roweis, S.: Metric learning by collapsing classes. In: Advances in Neural Information Processing Systems. vol. 18. MIT Press (2006)

[9] Goldberger, J., Roweis, S., Hinton, G., Salakhutdinov, R.: Neighbourhood components analysis. In: Advances in Neural Information Processing Systems. vol. 17. MIT Press (2005)

[10] Huang, K., Ying, Y., Campbell, C.: Gsml: A unified framework for sparse metric learning. In: Proceedings of the 2009 Ninth IEEE International Conference on Data Mining (2009)

[11] O.Duda, R., E.Hart, P., Stork, D.G.: Pattern Classification. A Wiley Interscience Publication (2001)

[12] Rakotomamonjy, A.: Variable selection using svm-based criteria. Journal of Machine Learning Research 3, 1357–1370 (2003)

[13] Schultz, M., Joachims, T.: Learning a distance metric from relative comparisons. In: NIPS (2004)

[14] Shalev-Shwartz, S., Singer, Y., Y.Ng, A.: Online and batch learning of pseudo metrics. In: ICML (2004)

[15] Weinberger, K., Saul, L.: Distance metric learning for large margin nearest neighbor classification. The Journal of Machine Learning Research 10, 207–244 (2009)

[16] 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. MIT Press (2003)

[17] Zhu, J., Rosset, S., Hastie, T., Tibshirani, R.: 1-norm support vector machine. In: Neural Information Processing Systems. p. 16. MIT Press (2003)

Acknowledgement

This work was funded by the Swiss NSF (Grant 200021-122283/1). The support of the European Commission through EU projects DebugIT (FP7-217139) and e-LICO (FP7-231519) is also gratefully acknowledged.

Appendix

Showing the equivalence of problem (6) and problem (7) (Section 2):

With the 'symmetry' preference as described after problem (6), the second constraint of problem (6) can be replaced by the constraint: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq \gamma/2$. Moreover, we can always replace the squared values in problem (6) by their respective absolute values and get:

$$\max_{\mathbf{w}} \quad \gamma \qquad s.t. \;\; |\mathbf{w}^T(\mathbf{x}_i - \mathbf{x}_j)| \geq \gamma, y_i \neq y_j$$
$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq \gamma/2, \forall i,$$

If the constraint $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq \gamma/2$ is satisfied then $\mathbf{w}^T\mathbf{x}_i + b$ and y_i have the same sign; therefore any two instances $\forall \mathbf{x}_i, \mathbf{x}_j$ which have different labels, $(y_i \neq y_j)$, will lie on the opposite sides of the hyperplane. Hence:

$$\begin{aligned} |\mathbf{w}^T(\mathbf{x}_i - \mathbf{x}_j)| &= |(\mathbf{w}^T\mathbf{x}_i + b) - (\mathbf{w}^T\mathbf{x}_j + b)| \quad (17) \\ &= |\mathbf{w}^T\mathbf{x}_i + b| + |\mathbf{w}^T\mathbf{x}_j + b| \\ &= y_i(\mathbf{w}^T\mathbf{x}_i + b) + y_j(\mathbf{w}^T\mathbf{x}_j + b) \\ &\geq \gamma/2 + \gamma/2 = \gamma \end{aligned}$$

Therefore the first constraint of problem (6) is always satisfied if the constraint $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq \gamma/2$ is satisfied, thus (6) is equivalent to (7).

Equivalence of (8) to standard SVM formulation

We will show that problem (8) is equivalent to standard *SVM*. In fact (1) also scales with uniform scaling of $\mathbf{w}, b$ and to avoid this problem, $\|\mathbf{w}\|\gamma$ is fixed to 1 which lead to the second formula (2). We will show that the two ways of avoiding scaling problem, by forcing $\|\mathbf{w}\|\gamma = 1$ or by forcing $\|\mathbf{w}\| = 1$ are equivalent. Indeed, another way to avoid the scaling problem is to find a quantity which is invariant to the scaling of $\mathbf{w}$. Let $\gamma = t\|\mathbf{w}\|_p$, hence $t : 0 \mapsto \infty$, and lets fix $\|\mathbf{w}\|_p = 1$. Then let P be the feasible set of $\mathbf{w}$ which satisfies $(\|\mathbf{w}\|_p = 1 \textit{ and } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 0, \forall i)$, and $Q = \{\mathbf{w} | \|\mathbf{w}\|_p = 1 \textit{ and } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq t\|\mathbf{w}\|_p), \forall i\}$ Notice that, if $t = 0$ then $Q \equiv P$, if $t > 0$ then $Q \subseteq P$, and if $t > t_{max}$ then Q will be empty. For another value of $\|\mathbf{w}\|_p$, $\|\mathbf{w}\|_p = \lambda$, the corresponding feasible sets are P_λ and Q_λ. There is a one to one mapping from P to P_λ, and from Q to Q_λ, and the t_{max_λ} which makes Q_λ empty is the same as t_{max}. So t_{max} is invariant to the scaling of $\mathbf{w}$. Therefore (8) is equivalent to:

$$\begin{aligned} \max_{\mathbf{w},b,t} \quad & t \\ s.t. \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq t\|\mathbf{w}\|_p, \forall i,\ t\|\mathbf{w}\|_p = 1 \end{aligned} \tag{18}$$

The value of the geometric margin here is fixed to $1/\sqrt{d}$ while in standard *SVM* the geometric margin is $\gamma = 1/\|\mathbf{w}\|_2^2$. Using the l_2 norm of $\mathbf{w}$, we get a formulation which is equivalent to that of the hard margin *SVM* given in (2). Using the l_1 norm of $\mathbf{w}$, we get the 1-norm *SVM* [17].

Full linear transformation does not help *SVM*

For any linear transformation $\mathbf{L}$ (full matrix), the distance of $\mathbf{Lx}$ to the hyperplane $H_\mathbf{1} : \mathbf{1x} + b = 0$ is: $d(\mathbf{Lx}, H_\mathbf{1}) = \frac{\mathbf{1}^T\mathbf{Lx}+b}{\sqrt{d}} = \frac{\mathbf{1}^T\mathbf{D_L}\mathbf{x}+b}{\sqrt{d}}$ where $\mathbf{D_L}$ is a diagonal matrix, in which the k diagonal element corresponds to the sum of the elements of the kth column of $\mathbf{L}$: $\mathbf{D_L}_{kk} = \sum_i L_{ik}$. So for any full transformation matrix $\mathbf{L}$ there exists a diagonal transformation $\mathbf{D_L}$ which has the same signed distance to the hyperplane. This is also true for any hyperplane $\mathbf{w}^T\mathbf{x} + b$ where $\mathbf{w}$ does not contain any zero elements. Thus learning a full matrix does not bring any additional advantage to *SVM*.