Metric Embedding for Kernel Classification Rules

Bharath K. Sriperumbudur BHARATHSV@UCSD.EDU
Omer A. Lang OLANG@UCSD.EDU
Gert R. G. Lanckriet GERT@ECE.UCSD.EDU
Department of Electrical and Computer Engineering, University of California, San Diego, CA 92093 USA.

Abstract

In this paper, we consider a smoothing kernel based classification rule and propose an algorithm for optimizing the performance of the rule by learning the bandwidth of the smoothing kernel along with a data-dependent distance metric. The data-dependent distance metric is obtained by learning a function that embeds an arbitrary metric space into a Euclidean space while minimizing an upper bound on the resubstitution estimate of the error probability of the kernel classification rule. By restricting this embedding function to a reproducing kernel Hilbert space, we reduce the problem to solving a semidefinite program and show the resulting kernel classification rule to be a variation of the k-nearest neighbor rule. We compare the performance of the kernel rule (using the learned data-dependent distance metric) to state-of-the-art distance metric learning algorithms (designed for k-nearest neighbor classification) on some benchmark datasets. The results show that the proposed rule has either better or as good classification accuracy as the other metric learning algorithms.

1. Introduction

Parzen window methods, also called smoothing kernel rules are widely used in nonparametric density estimation and function estimation, and are popularly known as kernel density and kernel regression estimates, respectively. In this paper, we consider these rules for classification. To this end, let us consider the binary classification problem of classifying $x \in \mathbb{R}^D$, given an i.i.d. training sample $\{(X_i, Y_i)\}_{i=1}^n$ drawn from some unknown distribution $\mathcal{D}$, where $X_i \in \mathbb{R}^D$ and $Y_i \in \{0,1\}, \forall\, i \in [n] := \{1,\dots,n\}$. The *kernel classification rule* (Devroye et al., 1996, Chapter 10) is given by

$$g_n(x) = \begin{cases} 0 & \text{if } \sum_{i=1}^n \mathbb{1}_{\{Y_i=0\}} K\left(\frac{x-X_i}{h}\right) \\ & \quad \geq \sum_{i=1}^n \mathbb{1}_{\{Y_i=1\}} K\left(\frac{x-X_i}{h}\right) \\ 1 & \text{otherwise,} \end{cases} \qquad (1)$$

where $K : \mathbb{R}^D \to \mathbb{R}$ is a *kernel function*, which is usually nonnegative and monotone decreasing along rays starting from the origin. The number $h > 0$ is called the *smoothing factor*, or *bandwidth*, of the kernel function, which provides some form of distance weighting. We warn the reader not to confuse the kernel function, K, with the reproducing kernel (Schölkopf & Smola, 2002) of a reproducing kernel Hilbert space (RKHS), which we will denote with $\mathfrak{K}$.[1] When $K(x) = \mathbb{1}_{\{\|x\|_2 \leq 1\}}(x)$ (sometimes called the *naïve kernel*), the rule is similar to the k-nearest neighbor (k-NN) rule except that k is different for each X_i in the training set. The k-NN rule classifies each unlabeled example by the majority label among its k-nearest neighbors in the training set, whereas the kernel rule with the naïve kernel classifies each unlabeled example by the majority label among its neighbors that lie within a radius of h. Devroye and Krzyżak (1989) proved that for regular kernels (see Devroye et al., (1996, Definition 10.1)), if the smoothing parameter $h \to 0$ such that $nh^D \to \infty$ as $n \to \infty$, then the kernel classification rule is *universally consistent*. But, for a particular n, asymptotic results provide little guidance in the selection of h. On the other hand, selecting the wrong value of h may lead to very poor error rates. In fact, the crux of every nonparametric estimation problem is the choice of an appropriate smoothing factor. This is one of the questions that we address in this paper by proposing an algorithm to learn an optimal h.

The second question that we address is learning an optimal distance metric. For $x \in \mathbb{R}^D$, K is usually a function of $\|x\|_2$. Some popular kernels include the Gaussian kernel, $K(x) = e^{-\|x\|_2^2}$; the Cauchy kernel, $K(x) =$

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008. Copyright 2008 by the author(s)/owner(s).

[1]Unlike $\mathfrak{K}$, K is not required to be a positive definite function. If K is a positive definite function, then it corresponds to a translation-invariant kernel of some RKHS defined on $\mathbb{R}^D$. In such a case, the classification rule in Eq. (1) is similar to the ones that appear in kernel machines literature.

$1/(1 + \|x\|_2^{D+1})$; and the Epanechnikov kernel $K(x) = (1 - \|x\|_2^2)\mathbb{1}_{\{\|x\|_2 \le 1\}}$.[2] Snapp and Venkatesh (1998) have shown that the finite-sample risk of the k-NN rule may be reduced, for large values of n, by using a weighted Euclidean metric, even though the infinite sample risk is independent of the metric used. This has been experimentally confirmed by Xing et al. (2003); Shalev-Shwartz et al. (2004); Goldberger et al. (2005); Globerson and Roweis (2006); Weinberger et al. (2006). They all assume the metric to be $\rho(x, y) = \sqrt{(x-y)^T \mathbf{\Sigma}(x-y)} = \|\mathbf{L}(x-y)\|_2$ for $x, y \in \mathbb{R}^D$, where $\mathbf{\Sigma} = \mathbf{L}^T\mathbf{L}$ is the weighting matrix, and optimize over $\mathbf{\Sigma}$ to improve the performance of the k-NN rule. Since the kernel rule is similar to the k-NN rule, one would expect that its performance can be improved by making K a function of $\|\mathbf{L}x\|_2$. Another way to interpret this is to find a linear transformation $\mathbf{L} \in \mathbb{R}^{d \times D}$ so that the transformed data lie in a Euclidean metric space.

Some applications call for natural distance measures that reflect the underlying structure of the data at hand. For example, when computing the distance between two images, *tangent distance* would be more appropriate than the Euclidean distance. Similarly, while computing the distance between points that lie on a low-dimensional manifold in $\mathbb{R}^D$, *geodesic distance* is a more natural distance measure than the Euclidean distance. Most of the time, since the true distance metric is either unknown or difficult to compute, Euclidean or weighted Euclidean distance is used as a surrogate. In the absence of prior knowledge, the data may be used to select a suitable metric, which can lead to better classification performance. In addition, instead of $x \in \mathbb{R}^D$, suppose $x \in (\mathcal{X}, \rho)$, where $\mathcal{X}$ is a metric space with ρ as its metric. One would like to extend the kernel classification rule to such $\mathcal{X}$. In this paper, we address these issues by learning a transformation that embeds the data from $\mathcal{X}$ into a Euclidean metric space while improving the performance of the kernel classification rule.

The rest of the paper is organized as follows. In §2, we formulate the multi-class kernel classification rule and propose learning a transformation, φ, (that embeds the training data into a Euclidean space) and the bandwidth parameter, h, by minimizing an upper bound on the resubstitution estimate of the error probability. To achieve this, in §3, we restrict φ to an RKHS and derive a representation for it by invoking the generalized representer theorem. Since the resulting optimization problem is non-convex, in §4, we approximate it with a semidefinite program when K is a naïve kernel. We present experimental results in §5, wherein we show on benchmark datasets that the proposed algorithm performs better than k-NN and state-of-the-art metric learning algorithms developed for the k-NN rule.

[2]The Gaussian kernel is a positive definite function on $\mathbb{R}^D$ while the Epanechnikov and naïve kernels are not.

2. Problem Formulation

Let $\{(X_i, Y_i)\}_{i=1}^n$ denote an i.i.d. training set drawn from some unknown distribution $\mathcal{D}$ where $X_i \in (\mathcal{X}, \rho)$ and $Y_i \in [L]$, with L being the number of classes. The multi-class kernel classification rule is given by

$$g_n(x) = \arg\max_{l \in [L]} \sum_{i=1}^n \mathbb{1}_{\{Y_i = l\}} K_{X_i,h}(x), \qquad (2)$$

where $K : \mathcal{X} \to \mathbb{R}_+$ and $K_{x_0,h}(x) = \chi\left(\frac{\rho(x,x_0)}{h}\right)$ for some nonnegative function, χ, with $\chi(0) = 1$. The probability of error associated with the above rule is $L(g_n) := \Pr_{(X,Y)\sim\mathcal{D}}(g_n(X) \ne Y)$ where Y is the true label associated with X. Since $\mathcal{D}$ is unknown, $L(g_n)$ cannot be computed directly but can only be estimated from the training set. The *resubstitution estimate*,[3] $\widehat{L}(g_n)$, which counts the number of errors committed on the training set by the classification rule, is given by $\widehat{L}(g_n) := \frac{1}{n}\sum_{i=1}^n \mathbb{1}_{\{g_n(X_i) \ne Y_i\}}$. As aforementioned, when $\mathcal{X} = \mathbb{R}^D$, ρ is usually chosen to be $\|.\|_2$. Previous works in distance metric learning learn a linear transformation $\mathbf{L} : \mathbb{R}^D \to \mathbb{R}^d$ leading to the distance metric, $\rho_{\mathbf{L}}(X_i, X_j) := \|\mathbf{L}X_i - \mathbf{L}X_j\|_2 = \sqrt{(X_i - X_j)^T \mathbf{\Sigma}(X_i - X_j)}$, where $\mathbf{\Sigma}$ captures the variance-covariance structure of the data. In this work, our goal is to jointly learn h and a measurable function, $\varphi \in \mathcal{C} := \{\varphi : \mathcal{X} \to \mathbb{R}^d\}$, so that the resubstitution estimate of the error probability is minimized with $\rho_\varphi(X_i, X_j) := \|\varphi(X_i) - \varphi(X_j)\|_2$. Once h and φ are known, the kernel classification rule is completely specified by Eq. (2).

Devroye et al., (1996, Section 25.6) show that kernel rules of the form in Eq. (1) picked by minimizing $\widehat{L}(g_n)$ with smoothing factor $h > 0$ are generally inconsistent if X is nonatomic. The same argument can be extended to the multi-class rule given by Eq. (2). To learn φ, simply minimizing $\widehat{L}(g_n)$ without any smoothness conditions on φ can lead to kernel rules that overfit the training set. Such a φ can be constructed as follows. Let n_l be the number of points that belong to l^{th} class. Suppose $n_1 = n_2 = \cdots = n_L$. Then for any $h \ge 1$, choosing $\varphi(X) = Y_i$ when $X = X_i$ and $\varphi(X) = 0$ when $X \notin \{X_i\}_{i=1}^n$ clearly yields zero resubstitution error. However, such a choice of φ leads to a kernel rule that always assigns the unseen data to class 1, leading to very poor performance. Therefore, to avoid overfitting to the training set, the function class $\mathcal{C}$ should satisfy some smoothness properties so that highly non-smooth functions like the one we defined above are not chosen while minimizing $\widehat{L}(g_n)$. To this end, we introduce a penalty functional, $\Omega : \mathcal{C} \to \mathbb{R}_+$, which penalizes

[3]Apart from the resubstitution estimate, holdout and deleted estimates can also be used to estimate the error probability. These estimates are usually more reliable but more involved than the resubstitution estimate.

non-smooth functions in $\mathcal{C}$ so that they are not selected.[4] Therefore, our goal is to learn φ and h by minimizing the regularized error functional given as

$$L_{reg}(\varphi, h) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{g_n(X_i)\neq Y_i\}} + \lambda'\,\Omega[\varphi], \tag{3}$$

where $\varphi \in \mathcal{C}$, $h > 0$ and the regularization parameter, $\lambda' > 0$. g_n in Eq. (3) is given by Eq. (2), with ρ replaced by ρ_φ. Minimizing $L_{reg}(\varphi, h)$ is equivalent to minimizing the number of training instances for which $g_n(X) \neq Y$, over the function class, $\{\varphi : \Omega[\varphi] \leq s\}$, for some appropriately chosen s.

Consider $g_n(x)$ defined in Eq. (2). Suppose $Y_i = k$ for some X_i. Then $g_n(X_i) = k$ if and only if

$$\sum_{\{j:Y_j=k\}} K^{\varphi}_{X_j,h}(X_i) \geq \max_{\substack{l\in[L]\\ l\neq k}} \sum_{\{j:Y_j=l\}} K^{\varphi}_{X_j,h}(X_i), \tag{4}$$

where the superscript φ is used to indicate the dependence of K on φ.[5] Since the right hand side of Eq. (4) involves the max function which is not differentiable, we use the inequality $\max\{a_1,\dots,a_m\} \leq \sum_{i=1}^m a_i$ to upper bound[6] it with $\sum_{\substack{l\in[L]\\ l\neq k}} \sum_{j=1}^n \mathbb{1}_{\{Y_j=l\}} K_{X_j}(X_i)$. Thus, to maximize $\sum_{i=1}^n \mathbb{1}_{\{g_n(X_i)=Y_i\}}$, we maximize its lower bound given by
$\sum_{i=1}^n \mathbb{1}_{\left\{\sum_{\substack{j=1\\ j\neq i}}^n \mathbb{1}_{\{Y_j=Y_i\}} K_{X_j}(X_i) \geq \sum_{j=1}^n \mathbb{1}_{\{Y_j\neq Y_i\}} K_{X_j}(X_i)\right\}}$, resulting in a conservative rule.[7] In the above bound, we use $j \neq i$ just to make sure that $\varphi(X_i)$ is not the only point within its neighborhood of radius h. Define $\tau_{ij} := 2\delta_{Y_i,Y_j} - 1$ where δ represents the Kronecker delta. Then, the problem of learning φ and h by minimizing $L_{reg}(\varphi, h)$ in Eq. (3) reduces to solving the following optimization problem,

$$\min_{\varphi,\,h}\left\{\sum_{i=1}^n \psi_i(\varphi, h) + \lambda\,\Omega[\varphi] \,:\, \varphi \in \mathcal{C},\, h > 0\right\}, \tag{5}$$

where $\lambda = n\lambda'$ and $\psi_i(\varphi, h)$ given by

$$\mathbb{1}_{\left\{\sum_{\substack{j=1\\ j\neq i}}^n \mathbb{1}_{\{\tau_{ij}=1\}} K_{X_j}(X_i) < \sum_{j=1}^n \mathbb{1}_{\{\tau_{ij}=-1\}} K_{X_j}(X_i)\right\}}$$

is an upper bound on $\mathbb{1}_{\{g_n(X_i)\neq Y_i\}}$ for $i \in [n]$. Solving the above non-convex optimization problem is NP-hard. The gradient optimization is difficult because the gradients are zero almost everywhere. In addition to the computational hardness, the problem in Eq. (5) is not theoretically solvable unless some assumptions about $\mathcal{C}$ are made. In the following section, we assume $\mathcal{C}$ to be an RKHS with the reproducing kernel $\mathfrak{K}$ and provide a representation for the optimum φ that minimizes Eq. (5). We remind the reader that K is a smoothing kernel which is not required to be a positive definite function but takes on positive values, while $\mathfrak{K}$ is a reproducing kernel which is positive definite and can take negative values.

[4]This is equivalent to restricting the size of the function class $\mathcal{C}$ from which φ has to be selected.

[5]To simplify the notation, from now onwards, we write $K^{\varphi}_{X_j,h}(X_i)$ as $K_{X_j}(X_i)$ where the dependence of K on φ and h is implicit.

[6]Another upper bound that can be used for the max function is $\max\{a_1,\dots,a_m\} \leq \log\left(\sum_{i=1}^m e^{a_i}\right)$.

[7]Using the upper bound of max function in Eq. (4) makes the resulting kernel rule conservative as there can be samples from the training set that do not satisfy this inequality but get correctly classified according to Eq. (2).

3. Regularization in Reproducing Kernel Hilbert Space

Many machine learning algorithms like SVMs, regularization networks, and logistic regression can be derived within the framework of regularization in RKHS by choosing an appropriate empirical risk functional with the penalizer being the squared RKHS norm (see Evgeniou et al. (2000)). In Eq. (5), we have extended this framework to kernel classification rules, wherein we compute the $\varphi \in \mathcal{C}$ and $h > 0$ that minimize an upper bound on the resubstitution estimate of the error probability. To this end, we choose $\mathcal{C}$ to be an RKHS with the penalty functional being the squared RKHS norm,[8] i.e., $\Omega[\varphi] = \|\varphi\|^2_{\mathcal{C}}$. By fixing h, the objective function $\sum_{i=1}^n \psi_i(\varphi, h)$ in Eq. (5) depends on φ only through $\{\|\varphi(X_i) - \varphi(X_j)\|_2\}_{i,j=1}^n$. By letting $\sum_{i=1}^n \psi_i(\varphi, h) = \theta_h\left(\{\|\varphi(X_i) - \varphi(X_j)\|_2\}_{i,j=1}^n\right)$ where $\theta_h : \mathbb{R}^{n^2} \to \mathbb{R}_+$, Eq. (5) can be written as

$$\min_{h>0}\min_{\varphi\in\mathcal{C}} \theta_h\left(\{\|\varphi(X_i) - \varphi(X_j)\|_2\}_{i,j=1}^n\right) + \lambda\,\|\varphi\|^2_{\mathcal{C}}. \tag{6}$$

The following result provides a representation for the minimizer of Eq. (6), and is proved in Appendix A. We remind the reader that φ is a vector-valued mapping from $\mathcal{X}$ to $\mathbb{R}^d$.

Theorem 1 (Multi-output regularization)**.** *Suppose $\mathcal{C} = \{\varphi : \mathcal{X} \to \mathbb{R}^d\} = \mathcal{H}_1 \times \ldots \times \mathcal{H}_d$ where $\mathcal{H}_i$ is an RKHS with reproducing kernel $\mathfrak{K}_i : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ and $\varphi = (\varphi_1,\dots,\varphi_d)$ with $\mathcal{H}_i \ni \varphi_i : \mathcal{X} \to \mathbb{R}$. Then each minimizer $\varphi \in \mathcal{C}$ of Eq. (6) admits a representation of the form*

$$\varphi_j = \sum_{i=1}^n c_{ij}\mathfrak{K}_j(.,X_i),\ \forall\, j \in [d] \tag{7}$$

where $c_{ij} \in \mathbb{R}$ and $\sum_{i=1}^n c_{ij} = 0$, $\forall\, i \in [n]$, $\forall\, j \in [d]$.

[8]Another choice for $\mathcal{C}$ could be the space of bounded Lipschitz functions with the penalty functional, $\Omega[\varphi] = \|\varphi\|_L$, where $\|\varphi\|_L$ is the Lipschitz semi-norm of φ. With this choice of $\mathcal{C}$ and Ω, von Luxburg and Bousquet (2004) studied large margin classification in metric spaces. One more interesting choice for $\mathcal{C}$ could be the space of Mercer kernel maps. It can be shown that solving for φ in Eq. (5) with such a choice for $\mathcal{C}$ is equivalent to learning the kernel matrix associated with φ and $\{X_i\}_{i=1}^n$. However, this approach is not useful as it does not allow for an out-of-sample extension.

Remark 2. *(a) By Eq. (7), φ is completely determined by $\{c_{ij} : i \in [n], j \in [d]\}$. Therefore, the problem of learning φ reduces to learning $n \cdot d$ scalars, $\{c_{ij} : \sum_{i=1}^n c_{ij} = 0\}$.*

(b) θ_h in Eq. (6) depends on φ through $\|\varphi(.) - \varphi(.)\|_2$. Therefore, for any $z, w \in \mathcal{X}$, we have $\|\varphi(z) - \varphi(w)\|_2^2 = \sum_{m=1}^d \left[\mathbf{c}_m^T (\mathbf{k}_m^z - \mathbf{k}_m^w)\right]^2 = \sum_{m=1}^d tr(\mathbf{\Sigma}_m(\mathbf{k}_m^z - \mathbf{k}_m^w)(\mathbf{k}_m^z - \mathbf{k}_m^w)^T)$ where $\mathbf{c}_m := (c_{1m}, \dots, c_{nm})^T$, $\mathbf{k}_m^z := (\mathfrak{K}_m(z, X_1), \dots, \mathfrak{K}_m(z, X_n))^T$, $\mathbf{\Sigma}_m := \mathbf{c}_m\mathbf{c}_m^T$, $\forall\, m \in [d]$ and $tr(.)$ represents the trace.

(c) The regularizer, $\|\varphi\|_{\mathcal{C}}^2$ in Eq. (6) is given by $\|\varphi\|_{\mathcal{C}}^2 = \sum_{m=1}^d \|\varphi_m\|_{\mathcal{H}_m}^2 = \sum_{m=1}^d \sum_{i,j=1}^n c_{im}c_{jm}\mathfrak{K}_m(X_i, X_j) = \sum_{m=1}^d \mathbf{c}_m^T\mathbf{K}_m\mathbf{c}_m = \sum_{m=1}^d tr(\mathbf{K}_m\mathbf{\Sigma}_m)$ where $\mathbf{K}_m := (\mathbf{k}_m^{X_1}, \dots, \mathbf{k}_m^{X_n})$.

(d) Since φ appears in the form of ρ_φ and $\|\varphi\|_{\mathcal{C}}^2$ in Eq. (6), learning φ is equivalent to learning $\{\mathbf{\Sigma}_m \succeq 0 : rank(\mathbf{\Sigma}_m) = 1,\ \mathbf{1}^T\mathbf{\Sigma}_m\mathbf{1} = 0\}_{m=1}^d$.

In the above remark, we have shown that θ_h and $\|\varphi\|_{\mathcal{C}}$ in Eq. (6) depend only on the entries in d kernel matrices (associated with d kernel functions) and $n \cdot d$ scalars, $\{c_{ij}\}$. In addition, we also reduced the representation of φ from $\{\mathbf{c}_m\}_{m=1}^d$ to $\{\mathbf{\Sigma}_m\}_{m=1}^d$. It can be seen that ρ_φ^2 and $\|\varphi\|_{\mathcal{C}}^2$ are convex quadratic functions of $\{\mathbf{c}_m\}_{m=1}^d$, while they are linear functions of $\{\mathbf{\Sigma}_m\}_{m=1}^d$. Depending on the nature of K, one representation would be more useful than the other.

Corollary 3. *Suppose $\mathfrak{K}_1 = \ldots = \mathfrak{K}_d = \mathfrak{K}$. Then, for any $z, w \in \mathcal{X}$, $\rho_\varphi^2(z, w)$ is the Mahalanobis distance between $\mathbf{k}^z$ and $\mathbf{k}^w$, with $\sum_{m=1}^d \mathbf{\Sigma}_m$ as its metric.*

Proof. By Remark 2, we have $\rho_\varphi^2(z, w) = \|\varphi(z) - \varphi(w)\|_2^2 = \sum_{m=1}^d (\mathbf{k}_m^z - \mathbf{k}_m^w)^T \mathbf{\Sigma}_m (\mathbf{k}_m^z - \mathbf{k}_m^w)$. Since $\mathfrak{K}_1 = \ldots = \mathfrak{K}_d = \mathfrak{K}$, we have $\mathbf{k}_1^z = \ldots = \mathbf{k}_d^z = \mathbf{k}^z$. Therefore, $\rho_\varphi^2(z, w) = (\mathbf{k}^z - \mathbf{k}^w)^T\mathbf{\Sigma}(\mathbf{k}^z - \mathbf{k}^w)$ where $\mathbf{\Sigma} := \sum_{m=1}^d \mathbf{\Sigma}_m$. $\square$

The above result reduces the problem of learning φ to learning a matrix, $\mathbf{\Sigma} \succeq 0$, such that $\text{rank}(\mathbf{\Sigma}) \leq d$ and $\mathbf{1}^T\mathbf{\Sigma}\mathbf{1} = 0$. We now study the above result for linear kernels. The following corollary shows that applying a linear kernel is equivalent to assuming the underlying distance metric in $\mathcal{X}$ to be the Mahalanobis distance.

Corollary 4 (Linear kernel). *Let $\mathcal{X} = \mathbb{R}^D$ and $z, w \in \mathcal{X}$. If $\mathfrak{K}(z, w) = \langle z, w\rangle_2 = z^T w$, then $\varphi(z) = \mathbf{L}z \in \mathbb{R}^d$ and $\|\varphi(z) - \varphi(w)\|_2^2 = (z - w)^T\mathbf{A}(z - w)$ with $\mathbf{A} := \mathbf{L}^T\mathbf{L}$.*

Proof. By Remark 2 and Corollary 3, we have $\varphi_m(z) = \sum_{i=1}^n c_{im}\mathfrak{K}(z, X_i) = (\sum_{i=1}^n c_{im}X_i)^T z =: \boldsymbol{\ell}_m^T z$. Therefore, $\varphi(z) = \mathbf{L}z$, where $\mathbf{L} := (\boldsymbol{\ell}_1, \dots, \boldsymbol{\ell}_d)^T$. In addition, $\|\varphi(z) - \varphi(w)\|_2^2 = (z - w)^T\mathbf{A}(z - w)$ with $\mathbf{A} := \mathbf{L}^T\mathbf{L}$. $\square$

In the following section, we use these results to derive an algorithm that jointly learns φ and h by solving Eq. (5).

4. Convex Relaxations & Semidefinite Program

Having addressed the theoretical issue of making assumptions about $\mathcal{C}$ to solve Eq. (5), we return to address the computational issue pointed out in §2. The program in Eq. (5) is NP-hard because of the nature of $\{\psi_i\}_{i=1}^n$. This issue can be alleviated by minimizing a convex upper bound of ψ_i, instead of ψ_i. Some of the convex upper bounds for the function $\psi(x) = \mathbb{1}_{\{x>0\}}$ are $\Psi(x) = \max(0, 1 + x) := [1 + x]_+$, $\Psi(x) = \log(1 + e^x)$ etc. Replacing ψ_i by Ψ_i in Eq. (5) results in the following program,

$$\min_{\substack{\varphi \in \mathcal{C} \\ h>0}} \sum_{i=1}^n \Psi_i\left(\gamma_i^-(\varphi, h) - \gamma_i^+(\varphi, h)\right) + \lambda\,\|\varphi\|_{\mathcal{C}}^2, \qquad (8)$$

where $\gamma_i^+(\varphi, h) := \sum_{\substack{j\neq i \\ \tau_{ij}=1}} K_{X_j}(X_i)$ and $\gamma_i^-(\varphi, h) := \sum_{\{j:\tau_{ij}=-1\}} K_{X_j}(X_i)$. Eq. (8) can still be computationally hard to solve depending on the choice of the smoothing kernel, K. Even if we choose K such that γ^+ and γ^- are jointly convex in φ and h for some representation of φ (see Remark 2), Eq. (8) is still non-convex as the argument of Ψ_i is a difference of two convex functions.[9] In addition, if $\Psi(x) = [1 + x]_+$, then Eq. (8) is a d.c. (difference of convex functions) program (Horst & Thoai, 1999), which is NP-hard to solve. So, even for the nicest of cases, one has to resort to local optimization methods or computationally intensive global optimization methods. Nevertheless, if one does not worry about this disadvantage, then solving Eq. (8) yields φ (in terms of $\{\mathbf{c}_m\}_{m=1}^d$ or $\{\mathbf{\Sigma}_m\}_{m=1}^d$, depending on the chosen representation) and h that can be used in Eq. (2) to classify unseen data. However, in the following, we show that Eq. (8) can be turned into a convex program for the naïve kernel. As mentioned in §1, this choice of kernel leads to a classification rule that is similar in principle to the k-NN rule.

4.1. Naïve kernel: Semidefinite relaxation

The naïve kernel, $K_{x_0}(x) = \mathbb{1}_{\{\rho_\varphi(x, x_0)\leq h\}}$, indicates that the points, $\varphi(x)$, that lie within a ball of radius h centered at $\varphi(x_0)$ have a weighting factor of 1, while the remaining points have zero weight. Using this in Eq. (8), we have $\gamma_i^-(\varphi, h) - \gamma_i^+(\varphi, h) = \sum_{\{j:\tau_{ij}=-1\}} \mathbb{1}_{\{\rho_\varphi(X_i, X_j)\leq h\}} - \sum_{\{j:\tau_{ij}=1\}} \mathbb{1}_{\{\rho_\varphi(X_i, X_j)\leq h\}} + 1$, which represents the difference between number of points with label different from Y_i that lie within the ball of radius of h centered at $\varphi(X_i)$ and the number of points with the same label as X_i (excluding X_i) that lie within the same ball. If this difference is

[9]For example, let K be a Gaussian kernel, $K_y(x) = \exp(-\rho_\varphi^2(x, y)/h)$. Using the $\{\mathbf{\Sigma}_m\}_{m=1}^d$ representation for φ, we have $\rho_\varphi^2(x, y)$ is linear in $\{\mathbf{\Sigma}_m\}_{m=1}^d$ and therefore, $K_y(x)$ is convex in $\{\mathbf{\Sigma}_m\}_{m=1}^d$. Here, we assume h to be fixed. This means γ_i^+ and γ_i^- are convex in φ.

positive, then the classification rule in Eq. (2) makes an error in classifying X_i. Therefore, φ and h should be chosen such that this misclassification rate is minimized. Suppose that $\{\varphi(X_i)\}_{i=1}^n$ is given. Then, h determines the misclassification rate like k in k-NN. It can be seen that the kernel classification rule and k-NN rule are similar when K is a naïve kernel. In the case of k-NN, the number of nearest neighbors are fixed for any point, whereas with the kernel rule, it varies for every point. On the other hand, the radius of the ball containing the nearest neighbors of a point varies with every point in the k-NN setting while it is the same for every point in the kernel rule.

$\gamma_i^-(\varphi,h) - \gamma_i^+(\varphi,h)$ can be further reduced to a more amenable form by the following algebra. Using $\sum_{\{j:\tau_{ij}=1\}} \mathbb{1}_{\{\rho_\varphi(X_i,X_j)\le h\}} = \sum_{j=1}^n \mathbb{1}_{\{\tau_{ij}=1\}} - \sum_{\{j:\tau_{ij}=1\}} \mathbb{1}_{\{\rho_\varphi(X_i,X_j)>h\}}$, we get $\gamma_i^-(\varphi,h) - \gamma_i^+(\varphi,h) = 1 - n_i^+ + \sum_{j=1}^n \mathbb{1}_{\{\tau_{ij}\rho_\varphi^2(X_i,X_j)>\tau_{ij}\tilde{h}\}}$ where $n_i^+ := \sum_{j=1}^n \mathbb{1}_{\{\tau_{ij}=1\}}$ and $\tilde{h} := h^2$. Note that we have neglected the set $\{j : \tau_{ij} = -1; \rho_\varphi(X_i,X_j) = h\}$ in the above calculation for simplicity. Using $\Psi(x) = [1+x]_+$, the first half of the objective function in Eq. (8) reduces to $\sum_{i=1}^n \Big[2 - n_i^+ + \sum_{j=1}^n \mathbb{1}_{\{\tau_{ij}\rho_\varphi^2(X_i,X_j)>\tau_{ij}\tilde{h}\}}\Big]_+$. Applying the convex relaxation one more time to the step function, we get $\sum_{i=1}^n \Big[2 - n_i^+ + \sum_{j=1}^n \Big[1 + \tau_{ij}\rho_\varphi^2(X_i,X_j) - \tau_{ij}\tilde{h}\Big]_+\Big]_+$ as an upper bound on the first half of the objective function in Eq. (8). Since ρ_φ^2 is a quadratic function of $\{\mathbf{c}_m\}_{m=1}^d$, it can be shown that representing φ in terms of $\{\mathbf{c}_m\}_{m=1}^d$ results in a d.c. program, whereas its representation in terms of $\{\mathbf{\Sigma}_m\}_{m=1}^d$ results in a semidefinite program (SDP) (except for the rank constraints), since ρ_φ^2 is linear in $\{\mathbf{\Sigma}_m\}_{m=1}^d$. Assuming for simplicity that $\mathfrak{K}_1 = \ldots = \mathfrak{K}_d = \mathfrak{K}$ and neglecting the constraint $\text{rank}(\mathbf{\Sigma}) \le d$, we obtain the following SDP,

$$\begin{aligned} \min_{\mathbf{\Sigma},\tilde{h}} \quad & \sum_{i=1}^n \Big[2 - n_i^+ + \sum_{j=1}^n \Big[1 + \tau_{ij}\text{tr}(\mathbf{M}_{ij}\mathbf{\Sigma}) - \tau_{ij}\tilde{h}\Big]_+\Big]_+ \\ & + \lambda\,\text{tr}(\mathbf{K}\mathbf{\Sigma}) \\ \text{s.t.} \quad & \mathbf{\Sigma} \succeq 0,\ \mathbf{1}^T\mathbf{\Sigma}\mathbf{1} = 0,\ \tilde{h} > 0, \end{aligned} \tag{9}$$

where $\mathbf{M}_{ij} := (\mathbf{k}^{X_i} - \mathbf{k}^{X_j})(\mathbf{k}^{X_i} - \mathbf{k}^{X_j})^T$. For notational details, refer to Remark 2 and Corollary 3. Since one does not usually know the optimal embedding dimension, d, the $\mathbf{\Sigma}$ representation is advantageous as it is independent of d (as we neglected the rank constraint) and depends only on n. On the other hand, it is a disadvantage as the algorithm does not scale well to large datasets.

Although the program in Eq. (9) is convex, solving it by general purpose solvers that use interior point methods scales as $O(n^6)$, which is prohibitive. Instead, following the ideas of Weinberger et al. (2006), we used a first order gradient method (which scales as $O(n^2)$ per iteration) and an alternating projections method (which scales as $O(n^3)$ per iteration). At each iteration, we take a small step in the direction of the negative gradient of the objective function, followed by a projection onto the set $\mathcal{N} = \{\mathbf{\Sigma} : \mathbf{\Sigma} \succeq 0, \mathbf{1}^T\mathbf{\Sigma}\mathbf{1} = 0\}$ and $\{\tilde{h} > 0\}$. The projection onto $\mathcal{N}$ is performed by an alternating projections method which involves projecting a symmetric matrix alternately between the convex sets, $\mathcal{A} = \{\mathbf{\Sigma} : \mathbf{\Sigma} \succeq 0\}$ and $\mathcal{B} = \{\mathbf{\Sigma} : \mathbf{1}^T\mathbf{\Sigma}\mathbf{1} = 0\}$. Since $\mathcal{A} \cap \mathcal{B} \neq \emptyset$, this alternating projections method is guaranteed to find a point in $\mathcal{A} \cap \mathcal{B}$. Given any $\mathbf{A}_0 \in \mathcal{A}$, the alternating projections algorithm computes $\mathbf{B}_m = P_\mathcal{B}(\mathbf{A}_m) \in \mathcal{B}$, $\mathbf{A}_{m+1} = P_\mathcal{A}(\mathbf{B}_m) \in \mathcal{A}$, $m = 0,1,2,\ldots$, where $P_\mathcal{A}$ and $P_\mathcal{B}$ are the projection on $\mathcal{A}$ and $\mathcal{B}$, respectively. In summary, the update rule can be given as $\mathbf{B}_m = \mathbf{A}_m - \frac{\mathbf{1}^T\mathbf{A}_m\mathbf{1}}{n^2}\mathbf{1}\mathbf{1}^T$ and $\mathbf{A}_{m+1} = \sum_{i=1}^n [\lambda_i]_+\mathbf{u}_i\mathbf{u}_i^T$ where $\{\mathbf{u}_i\}_{i=1}^n$ and $\{\lambda_i\}_{i=1}^n$ are the eigenvectors and eigenvalues of $\mathbf{B}_m$.[10] A pseudocode of the gradient projection algorithm to solve Eq. (9) is shown in Algorithm 1.

Algorithm 1 Gradient Projection Algorithm

```
Require: $\{\mathbf{M}_{ij}\}_{i,j=1}^n$, $\mathbf{K}$, $\{\tau_{ij}\}_{i,j=1}^n$, $\{n_i^+\}_{i=1}^n$, $\lambda > 0$,
    $\epsilon > 0$ and $\{\alpha_i, \beta_i\} > 0$ (see Eq. (9))
 1: Set $t = 0$. Choose $\mathbf{\Sigma}_0 \in \mathcal{A}$ and $\tilde{h}_0 > 0$.
 2: repeat
 3:     $A_t = \{i : \sum_{j=1}^n \Big[1 + \tau_{ij}\text{tr}(\mathbf{M}_{ij}\mathbf{\Sigma}_t) - \tau_{ij}\tilde{h}_t\Big]_+ + 2 \le n_i^+\} \times \{j : j \in [n]\}$
 4:     $B_t = \{(i,j) : 1 + \tau_{ij}\text{tr}(\mathbf{M}_{ij}\mathbf{\Sigma}_t) > \tau_{ij}\tilde{h}_t\}$
 5:     $N_t = B_t \backslash A_t$
 6:     $\mathbf{\Sigma}_{t+1} = P_\mathcal{N}(\mathbf{\Sigma}_t - \alpha_t \sum_{(i,j)\in N_t} \tau_{ij}\mathbf{M}_{ij} - \alpha_t\lambda\mathbf{K})$
 7:     $\tilde{h}_{t+1} = \max(\epsilon, \tilde{h}_t + \beta_t \sum_{(i,j)\in N_t} \tau_{ij})$
 8:     $t = t + 1$
 9: until convergence
10: return $\mathbf{\Sigma}_t$, $\tilde{h}_t$
```

Having computed $\mathbf{\Sigma}$ and $\tilde{h}$ that minimize Eq. (9), a test point, $x \in \mathcal{X}$, can be classified by using the kernel rule in Eq. (2), where $K_{X_i}(x) = \mathbb{1}_{\{\rho_\varphi(x,X_i)\le h\}}$ with $\rho_\varphi^2(x,X_i) = (\mathbf{k}^x - \mathbf{k}^{X_i})^T\mathbf{\Sigma}(\mathbf{k}^x - \mathbf{k}^{X_i})$. Therefore, $\mathbf{\Sigma}$ and h completely specify the classification rule.

5. Experiments & Results

In this section, we compare the performance of our method (referred to as kernel classification rule (KCR)) to several metric learning algorithms on a supervised classification task in terms of the training and test errors. The training phase of KCR involves solving the SDP in Eq. (9) to learn optimal $\mathbf{\Sigma}$ and h from the data, which are then used in Eq. (2) to classify the test data. Note that the SDP in Eq. (9)

[10] Given $\mathbf{A}_m \in \mathcal{A}$, $\mathbf{B}_m$ is obtained by solving $\min\{\|\mathbf{B}_m - \mathbf{A}_m\|_F^2 : \mathbf{1}^T\mathbf{B}_m\mathbf{1} = 0\}$. Similarly, for a given $\mathbf{B}_m \in \mathcal{B}$, $\mathbf{A}_{m+1}$ is obtained by solving $\min\{\|\mathbf{A}_{m+1} - \mathbf{B}_m\|_F^2 : \mathbf{A}_{m+1} \succeq 0\}$.

Table 1. k-NN classification accuracy on UCI datasets. The algorithms compared are k-NN (with Euclidean distance metric), LMNN (large margin NN by Weinberger et al. (2006)), *Kernel*-NN (see footnote 11), KMLCC (kernel version of metric learning by collapsing classes by Globerson and Roweis (2006)), KLMCA (kernel version of LMNN by Torresani and Lee (2007)), and KCR (proposed method). Mean (μ) and standard deviation (σ) of the train and test (generalization) errors (in %) are reported.

Dataset (n, D, l)	Algorithm/ Error	k-NN $\mu \pm \sigma$	LMNN $\mu \pm \sigma$	*Kernel*-NN $\mu \pm \sigma$	KMLCC $\mu \pm \sigma$	KLMCA $\mu \pm \sigma$	KCR $\mu \pm \sigma$
Balance	Train	17.81 ± 1.86	11.40 ± 2.89	10.73 ± 1.32	10.27 ± 2.01	9.93 ± 1.86	10.47 ± 2.11
$(625, 4, 3)$	Test	18.18 ± 1.88	11.49 ± 2.57	17.46 ± 2.13	9.75 ± 1.92	10.54 ± 1.46	$\mathbf{8.94 \pm 3.12}$
Ionosphere	Train	15.89 ± 1.43	3.50 ± 1.18	2.84 ± 0.80	7.05 ± 1.31	3.98 ± 1.94	2.73 ± 1.03
$(351, 34, 2)$	Test	15.95 ± 3.03	12.14 ± 2.92	5.81 ± 2.25	6.54 ± 2.18	$\mathbf{5.19 \pm 2.09}$	5.71 ± 2.60
Iris	Train	4.30 ± 1.55	3.25 ± 1.15	3.60 ± 1.33	3.61 ± 1.59	3.27 ± 1.63	2.29 ± 1.62
$(150, 4, 3)$	Test	4.02 ± 2.22	4.11 ± 2.26	4.83 ± 2.47	3.89 ± 1.55	3.74 ± 2.21	$\mathbf{3.27 \pm 1.87}$
Wine	Train	5.89 ± 1.35	0.90 ± 2.80	4.95 ± 1.35	4.48 ± 1.21	2.18 ± 2.58	1.01 ± 0.73
$(178, 13, 3)$	Test	6.22 ± 2.70	3.41 ± 2.10	7.37 ± 2.82	4.84 ± 2.47	5.17 ± 1.91	$\mathbf{2.13 \pm 1.24}$

is obtained by using the naïve kernel for K in Eq. (2). For other smoothing kernels, one has to solve the program in Eq. (8) to learn optimal $\mathbf{\Sigma}$ and h. Therefore, the results reported in this section under KCR refer to those obtained by using the naïve kernel.

The algorithms used in the comparative evaluation are:

- The k-NN rule with the Euclidean distance metric.
- The LMNN (large margin nearest neighbor) method proposed by Weinberger et al. (2006), which learns a Mahalanobis distance metric by minimizing the distance between predefined target neighbors and separating them by a large margin from the examples with non-matching labels.
- The *Kernel*-NN rule, which uses the empirical kernel maps[11] as training data and performs k-NN classification on this data using the Euclidean distance metric.
- The KMLCC (kernel version of metric learning by collapsing classes) method proposed by Globerson and Roweis (2006), which learns a Mahalanobis distance metric in the kernel space by trying to collapse all examples in the same class to a single point while pushing examples in other classes infinitely far away.
- The KLMCA (kernel version of large margin component analysis) method proposed by Torresani and Lee (2007), which is a non-convex, kernelized version of LMNN.

Four benchmark datasets from the UCI machine learning repository were considered for experimentation. Since the proposed method and KMLCC solve an SDP that scales poorly with n, we did not consider large problem sizes for experimentation.[12] The results shown in Table 1 are the average performance over 20 random splits of the data with 50% for training, 20% for validation and 30% for testing. The Gaussian kernel, $\mathfrak{K}(x, y) = e^{-\upsilon\|x-y\|_2^2}$ was used for the kernel based methods, i.e., *Kernel*-NN, KMLCC, KLMCA and KCR. The parameters υ and λ (only υ for *Kernel*-NN) were set with cross-validation by searching over $\upsilon \in \{2^i\}_{-4}^{4}$ and $\lambda \in \{10^i\}_{-3}^{3}$. While testing, KCR uses the rule in Eq. (2), whereas the k-NN rule was used for all the other methods.[13] It is clear from Table 1 that KCR almost always performs as well as or significantly better than all other methods. However, on the timing front (which we do not report here), KLMCA, which solves a non-convex program for $n \cdot d$ variables, is much faster than KMLCC and KCR, which solve SDPs involving n^2 variables. The role of empirical kernel maps is not clear as there is no consistent behavior between the performance accuracy achieved with k-NN and *Kernel*-NN.

KMLCC, KLMCA, and KCR learn the Mahalanobis distance metric in $\mathbb{R}^n$ which makes it difficult to visualize the class separability achieved by these methods. To visually appreciate their behavior, we generated a synthetic two dimensional dataset of 3 classes with each class being sampled from a Gaussian distribution with different mean and covariance. Figure 1(a) shows this dataset where the three classes are shown in different colors. Using this as training data, distance metrics were learned using KMLCC, KLMCA and KCR. If $\mathbf{\Sigma}$ is the learned metric, then the two dimensional projection of $x \in \mathbb{R}^n$ is obtained as $\widehat{x} = \mathbf{L}x$ where $\mathbf{L} = (\sqrt{\lambda_1}\mathbf{u}_1, \sqrt{\lambda_2}\mathbf{u}_2)^T$, with $\mathbf{\Sigma} = \sum_{i=1}^{n} \lambda_i \mathbf{u}_i \mathbf{u}_i^T$, and $\lambda_1 \geq \lambda_2 > \cdots \lambda_n$. Figure 1(b-d) show the two di-

[11] *Kernel*-NN is computed as follows. For each training point, X_i, the empirical map w.r.t. $\{X_j\}_{j=1}^n$ defined as $\mathbf{k}^{X_i} := (\mathfrak{K}(X_1, X_i), \ldots, \mathfrak{K}(X_n, X_i))^T$ is computed. Then, $\{\mathbf{k}^{X_i}\}_{i=1}^n$ is considered to be the training set for the NN classification of empirical maps of the test data using the Euclidean distance metric.

[12] To extend KCR to large datasets, one can represent φ in terms of $\{\mathbf{c}_m\}$, which leads to a non-convex program as in KLMCA.

[13] Although KCR, LMNN, and KMLCC solve SDPs to compute the optimal distance metric, KCR has fewer number of parameters to be tuned compared to these other methods. LMNN requires cross-validation over k (in k-NN) and the regularization parameter along with the knowledge about target neighbors. KMLCC requires cross-validation over k, the kernel parameter, υ and the regularization parameter. In KCR, we only need to cross-validate over υ and λ. In addition, if $\mathcal{X} = \mathbb{R}^D$ and $\mathfrak{K}$ is a linear kernel, then KCR only requires cross-validation over λ while computing the optimal Mahalanobis distance metric.

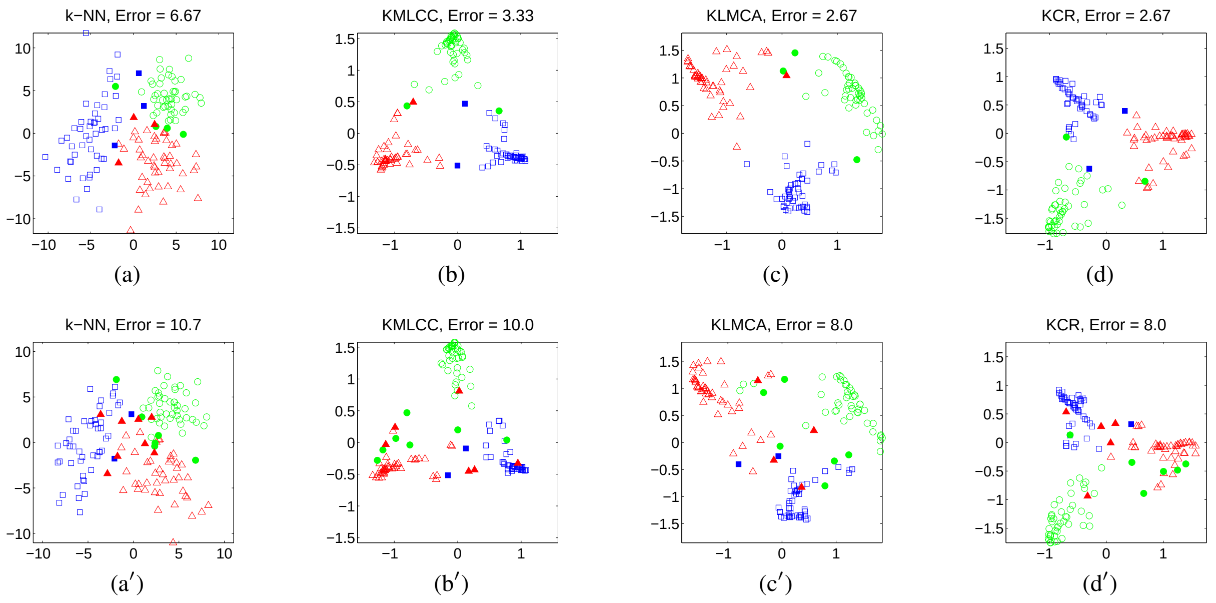

Figure 1. Dataset visualization results of k-NN, KMLCC, KLMCA and KCR applied to a two-dimensional synthetic dataset of three classes with each class being modeled as a Gaussian. (a,a′) denote two independent random draws from this distribution whereas (b-d, b′-d′) represent the two-dimensional projections of these data using the metric learned from KMLCC, KLMCA and KCR. The points in bold represent the misclassified points. It is interesting to note that KLMCA and KCR generate completely different embeddings but have similar error rates. See §5 for more details.

mensional projections of the training set using KMLCC, KLMCA and KCR. The projected points were classified using k-NN if $\mathbf{\Sigma}$ was obtained from KMLCC/KLMCA and using Eq. (2) if $\mathbf{\Sigma}$ was obtained from KCR. The misclassified points are shown in bold. Since the classification is done on the training points, one would expect better error rate and separability between the classes. To understand the generalization performance, a new data sample shown in Figure 1(a′) was generated from the same distribution as the training set. The learned $\mathbf{\Sigma}$ was used to obtain the two dimensional projections of the new data sample which are shown in Figure 1(b′-d′). It is interesting to note that KLMCA and KCR generate completely different projections but have similar error rates.

6. Related Work

We briefly review some relevant work and point out similarities and differences with our method. In our work, we have addressed the problem of extending kernel classification rules to arbitrary metric spaces by learning an embedding function that embeds data into Euclidean space while minimizing an upper bound on the resubstitution estimate of the error probability. The method that is closest in spirit (kernel rules) to ours is the recent work by Weinberger and Tesauro (2007) who learn a Mahalanobis distance metric for kernel regression estimates by minimizing the leave-one-out quadratic regression error of the training set. With the problem being non-convex, they resort to gradient descent techniques. Except for this work, we are not aware of any method related to kernel rules in the context of distance metric learning or learning the bandwidth of the kernel.

There has been lot of work in the area of distance metric learning for k-NN classification, some of which are briefly discussed in §5. The central idea in all these methods is that similarly labeled examples should cluster together and be far away from differently labeled examples. Shalev-Shwartz et al. (2004) proposed an online algorithm for learning a Mahalanobis distance metric with the constraint that any training example is closer to all the examples that share its label than to any other example of different label. In addition, examples from different classes are constrained to be separated by a large margin. Though Shalev-Shwartz et al. (2004) do not solve this as a batch optimization problem, it can be shown that it reduces to an SDP (after rank relaxation) and is in fact the same as Eq. (9) except for the outer $[.]_+$ function and the constraint $\mathbf{1}^T\mathbf{\Sigma}\mathbf{1} = 0$.

7. Concluding Remarks

In this paper, two questions related to the smoothing kernel based classification rule have been addressed. One is related to learning the bandwidth of the smoothing kernel, while the other is to extending the classification rule to arbitrary domains. We jointly addressed them by learning a function in a reproducing kernel Hilbert space while minimizing an upper bound on the resubstitution estimate of the error probability of the kernel rule. For a particular choice of the smoothing kernel, called the naïve kernel, we showed that the resulting rule is related to the k-NN rule. Because of this relation, the kernel rule was compared to k-NN and its state-of-the-art distance metric learning algorithms on a supervised classification task and was shown to have comparable performance to these methods. In the future, we would like to develop some theoretical guarantees for the proposed method along with extending it to large-scale applications.

Appendix A. Proof of Theorem 1

We need the following result to prove Theorem 1.

Lemma 5. *Let $\mathcal{H} = \{f : \mathcal{X} \rightarrow \mathbb{R}\}$ be an RKHS with $\mathfrak{K} : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ as its reproducing kernel. Let $\theta : \mathbb{R}^{n^2} \rightarrow \mathbb{R}$ be an arbitrary function. Then each minimizer $f \in \mathcal{H}$ of*

$$\theta\left(\{f(x_i) - f(x_j)\}_{i,j=1}^n\right) + \lambda\|f\|_{\mathcal{H}}^2 \qquad (10)$$

admits a representation of the form $f = \sum_{i=1}^n c_i \mathfrak{K}(.,x_i)$, where $\{c_i\}_{i=1}^n \in \mathbb{R}$ and $\sum_{i=1}^n c_i = 0$.

Proof. The proof follows the generalized representer theorem (Schölkopf et al., 2001, Theorem 4). Since $f \in \mathcal{H}$, $f(x) = \langle f, \mathfrak{K}(.,x)\rangle_{\mathcal{H}}$. Therefore, the arguments of θ in Eq. (10) are of the form $\{\langle f, \mathfrak{K}(.,x_i) - \mathfrak{K}(.,x_j)\rangle_{\mathcal{H}}\}_{i,j=1}^n$. We decompose $f = f_\parallel + f_\perp$ so that $f_\parallel \in \text{span}\left(\{\mathfrak{K}(.,x_i) - \mathfrak{K}(.,x_j)\}_{i,j=1}^n\right)$ and $\langle f_\perp, \mathfrak{K}(.,x_i) - \mathfrak{K}(.,x_j)\rangle_{\mathcal{H}} = 0, \forall\, i,j \in [n]$. So, $f = \sum_{i,j=1}^n \alpha_{ij}(\mathfrak{K}(.,x_i) - \mathfrak{K}(.,x_j)) + f_\perp$ where $\{\alpha_{ij}\}_{i,j=1}^n \in \mathbb{R}$. Therefore, $f(x_i) - f(x_j) = \langle f, \mathfrak{K}(.,x_i) - \mathfrak{K}(.,x_j)\rangle_{\mathcal{H}} = \langle f_\parallel, \mathfrak{K}(.,x_i) - \mathfrak{K}(.,x_j)\rangle_{\mathcal{H}} = \sum_{p,m=1}^n \alpha_{pm}(\mathfrak{K}(x_i,x_p) - \mathfrak{K}(x_j,x_p) - \mathfrak{K}(x_i,x_m) + \mathfrak{K}(x_j,x_m))$. Now, consider the penalty functional, $\langle f, f\rangle_{\mathcal{H}}$. For all $f_\perp$, $\langle f, f\rangle_{\mathcal{H}} = \|f_\parallel\|_{\mathcal{H}}^2 + \|f_\perp\|_{\mathcal{H}}^2 \geq \|\sum_{i,j=1}^n \alpha_{ij}(\mathfrak{K}(.,x_i) - \mathfrak{K}(.,x_j))\|_{\mathcal{H}}^2$. Thus for any fixed $\alpha_{ij} \in \mathbb{R}$, Eq. (10) is minimized for $f_\perp = 0$. Therefore, the minimizer of Eq. (10) has the form $f = \sum_{i,j=1}^n \alpha_{ij}(\mathfrak{K}(.,x_i) - \mathfrak{K}(.,x_j))$, which is parameterized by n^2 parameters of $\{\alpha_{ij}\}_{i,j=1}^n$. By simple algebra, f reduces to $f = \sum_{i=1}^n c_i \mathfrak{K}(.,x_i)$, where $c_i = \sum_{j=1}^n(\alpha_{ij} - \alpha_{ji})$ satisfies $\sum_{i=1}^n c_i = 0$. □

We are now ready to prove Theorem 1.

Proof of Theorem 1. The arguments of θ_h in Eq. (6) are of the form $\|\varphi(X_i) - \varphi(X_j)\|_2$. Consider $\|\varphi(X_i) - \varphi(X_j)\|_2^2 = \sum_{m=1}^d (\varphi_m(X_i) - \varphi_m(X_j))^2 = \sum_{m=1}^d (\langle \varphi_m, \mathfrak{K}_m(.,X_i) - \mathfrak{K}_m(.,X_j)\rangle_{\mathcal{H}_m})^2$. The penalizer in Eq. (6) reduces to $\|\varphi\|_{\mathcal{C}}^2 = \sum_{m=1}^d \|\varphi_m\|_{\mathcal{H}_m}^2$. Therefore, applying Lemma 5 to each φ_m, $m \in [d]$ proves the result. □

Acknowledgments

We thank the reviewers for their comments which greatly improved the paper. We wish to acknowledge support from the National Science Foundation (grant DMS-MSPA 0625409), the Fair Isaac Corporation and the University of California MICRO program.

References

Devroye, L., Gyorfi, L., & Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition*. New York: Springer-Verlag.

Devroye, L., & Krzyżak, A. (1989). An equivalence theorem for L_1 convergence of the kernel regression estimate. *Journal of Statistical Planning and Inference*, *23*, 71–82.

Evgeniou, T., Pontil, M., & Poggio, T. (2000). Regularization networks and support vector machines. *Advances in Computational Mathematics*, *13*, 1–50.

Globerson, A., & Roweis, S. (2006). Metric learning by collapsing classes. *Advances in Neural Information Processing Systems 18* (pp. 451–458). Cambridge, MA: MIT Press.

Goldberger, J., Roweis, S., Hinton, G., & Salakhutdinov, R. (2005). Neighbourhood components analysis. *Advances in Neural Information Processing Systems 17* (pp. 513–520). Cambridge, MA: MIT Press.

Horst, R., & Thoai, N. V. (1999). D.c. programming: Overview. *Journal of Optimization Theory and Applications*, *103*, 1–43.

Schölkopf, B., Herbrich, R., & Smola, A. J. (2001). A generalized representer theorem. *Proc. of the Annual Conference on Computational Learning Theory* (pp. 416–426).

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels*. Cambridge, MA: MIT Press.

Shalev-Shwartz, S., Singer, Y., & Ng, A. Y. (2004). Online and batch learning of pseudo-metrics. *Proc. of 21st International Conference on Machine Learning*. Banff, Canada.

Snapp, R. R., & Venkatesh, S. S. (1998). Asymptotic expansions of the k nearest neighbor risk. *The Annals of Statistics*, *26*, 850–878.

Torresani, L., & Lee, K. (2007). Large margin component analysis. *Advances in Neural Information Processing Systems 19* (pp. 1385–1392). Cambridge, MA: MIT Press.

von Luxburg, U., & Bousquet, O. (2004). Distance-based classification with Lipschitz functions. *Journal for Machine Learning Research*, *5*, 669–695.

Weinberger, K. Q., Blitzer, J., & Saul, L. K. (2006). Distance metric learning for large margin nearest neighbor classification. *Advances in Neural Information Processing Systems 18* (pp. 1473–1480). Cambridge, MA: MIT Press.

Weinberger, K. Q., & Tesauro, G. (2007). Metric learning for kernel regression. *Proc. of the 11th International Conference on Artificial Intelligence and Statistics*.

Xing, E. P., Ng, A. Y., Jordan, M. I., & Russell, S. (2003). Distance metric learning with application to clustering with side-information. *Advances in Neural Information Processing Systems 15* (pp. 505–512). Cambridge, MA: MIT Press.