Learning Anisotropic RBF Kernels

Fabio Aiolli and Michele Donini

University of Padova - Department of Mathematics
Via Trieste, 63, 35121 Padova - Italy
{aiolli,mdonini}@math.unipd.it

Abstract. We present an approach for learning an anisotropic RBF kernel in a game theoretical setting where the value of the game is the degree of separation between positive and negative training examples. The method extends a previously proposed method (KOMD) to perform feature re-weighting and distance metric learning in a kernel-based classification setting. Experiments on several benchmark datasets demonstrate that our method generally outperforms state-of-the-art distance metric learning methods, including the Large Margin Nearest Neighbor Classification family of methods.

1 Introduction

Kernel machines have gained great popularity in the last decades. Their fortune is greatly due to the possibility to plug general kernels into them. The kernel function represents a priori knowledge about similarities between pairs of examples in a domain.

The most popular kernel is undoubtedly the RBF kernel, which is a general purpose kernel that is based on the Euclidean distance between examples. Similarly to the Euclidean distance, the RBF kernel gives an equal weight to different features and the strength of this weight depends on a single external parameter that needs to be tuned against validation data. However, it is well known that different features typically have unequal impact and importance in solving a given classification task.

This issue has motivated several feature selection methods to select or weight different features in different ways. While feature selection is generally very difficult to perform with nonlinear kernels, one can learn the metric directly from data more easily. This task is known as distance metric learning (DML). For example, many researchers (see [1], [2], [3], [4]) have proposed a number of algorithms for the optimization of the Mahalanobis distance. Specifically, they replace the common Euclidean metric with the more powerful distance $(\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{M}(\mathbf{x}_i - \mathbf{x}_j)$ and try to learn the combination matrix $\mathbf{M}$. The learned distance in DML is typically optimized for (and used in) a nearest neighbors setting. Given the high number of free parameters to learn together with the fact that these methods are used with nearest neighbors, these approaches can be prone to overfitting, in particular when the training sample is small.

Recently, there have been also attempts to learn the kernel directly from data. In this setting, called kernel learning (KL), one looks for a kernel matrix which maximizes a measure of agreement between training labels and the similarity induced by the learned kernel matrix. This has been done either by optimizing with respect to the notion of

alignment ([5],[6]) or minimizing the value of the dual of the objective function of the SVM constructed on the kernel itself ([7]).

In this paper, we propose to combine ideas from DML and KL. Specifically, we focus on the family of anisotropic RBF kernels, that is kernels in the form $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M}(\mathbf{x}_i - \mathbf{x}_j))$ where $\mathbf{M} = \mathbf{diag}(\boldsymbol{\beta})$ is the diagonal matrix created using the vector $\boldsymbol{\beta} \in \mathbb{R}^m$ of parameters (one value for each feature) to learn. This form generalizes the RBF kernel for which we have $\boldsymbol{\beta} = \beta_0 \mathbf{1}$ being β_0 the external RBF shape parameter and $\mathbf{1}$ the vector with all entries equal to 1. The method proposed extends a recent kernel based algorithm, namely the Kernel Optimization of Margin Distribution (KOMD) method, to learn an *anisotropic RBF* from data. We maintain the same game theoretical setting where two players compete and the value of the game consists of the separation between positive and negative training data.

Definitions and Notation. We consider a classification problem with training examples $\{(\mathbf{x}_1, y_i), \ldots, (\mathbf{x}_l, y_l)\}$, and test examples $\{(\mathbf{x}_{l+1}, y_{l+1}), \ldots, (\mathbf{x}_L, y_L)\}$, $\mathbf{x}_i \in \mathbb{R}^m$, $y_i \in \{-1, +1\}$. We use $\mathbf{X} \in \mathbb{R}^{L \times m}$ to denote the matrix where examples are arranged in rows and $\mathbf{y} \in \mathbb{R}^L$ is the vector of labels. The matrix $\mathbf{K} \in \mathbb{R}^{L \times L}$ denotes the complete kernel matrix containing the kernel values of each data pair. Further, we indicate with an hat, like for example $\hat{\mathbf{X}} \in \mathbb{R}^{l \times m}$, $\hat{\mathbf{y}} \in \mathbb{R}^l$, and $\hat{\mathbf{K}} \in \mathbb{R}^{l \times l}$, the submatrices (or subvectors) obtained considering training examples only. We let $\mathbb{R}_+$ the set of non-negative real numbers. Given a training set, we consider the domain Γ of probability distributions $\boldsymbol{\gamma} \in \mathbb{R}^l_+$ defined over the sets of positive and negative examples. More formally, $\Gamma = \{\boldsymbol{\gamma} \in \mathbb{R}^l_+ \mid \sum_{i \in \oplus} \gamma^{(i)} = 1, \sum_{i \in \ominus} \gamma^{(i)} = 1\}$, where $\oplus$ and $\ominus$ are the sets of the indices of positive and negative examples respectively. Finally, we define the submatrix of positive (negative) examples of the matrix $\hat{\mathbf{X}}$ as $\hat{\mathbf{X}}^+$ ($\hat{\mathbf{X}}^-$).

2 Distance Metric Learning

Distance metric learning (DML) methods try to learn the best metric for a specific input space and dataset. The performance of a learning algorithm (nearest-neighbors classifiers, kernel algorithms etc.) mostly depends on the metric used. Many DML algorithms have been proposed. All of them try to find a positive semi-definite (PSD) matrix $\mathbf{M} \in \mathbb{R}^{m \times m}$ such that the induced metric $d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{M}(\mathbf{x}_i - \mathbf{x}_j)$ is optimal for the task at hand. For example, the Euclidian distance is a special case where $\mathbf{M} = \mathbf{I}$. There are three principal families of DML algorithms [8]: *eigenvector methods*, *convex optimization* and *neighborhood component analysis*.

In the eigenvector methods, the matrix $\mathbf{M}$ is parameterized by the product of a real valued matrix with its transposed, namely $\mathbf{M} = \mathbf{L}^\top \mathbf{L}$, in order to maintain the matrix positive semi-definite. In this case, the matrix $\mathbf{M}$ is called *Mahalanobis metric*. These methods use the covariance matrix to optimize the linear transformation $\mathbf{x}_i \rightarrow \mathbf{L}\mathbf{x}_i$ that projects the training inputs. Finding the optimal projection is the task of eigenvector methods with a constraint that defines $\mathbf{L}$ as a projection matrix: $\mathbf{L}\mathbf{L}^\top = \mathbf{I}$. These algorithms don't use the training labels and then they are totally unsupervised.

Convex optimization algorithms represent another family of DML algorithms. It is possible to formulate a DML as a convex optimization problem over the cone of

correct matrices $\mathbf{M}$. This cone is the cone of positive semi-definite matrices, namely $\mathfrak{M} = \{\mathbf{M} \in \mathbb{R}^{m \times m} : \forall \tau \in eig(\mathbf{M}), \tau \geq 0\}$. Algorithms in this family are supervised and optimal positive semi-definite matrix $\mathbf{M}$ is obtained optimizing the square root of the *Mahalanobis metric* and enforcing the SDP constraint $\mathbf{M} \succeq 0$. There are also online versions of convex optimization algorithms for DML, like *POLA* [3] for example.

Another family of algorithms for DML is called neighborhood component analysis. In [2], for example, the authors try to learn a *Mahalanobis metric* from the expected leave-one-out classification error. In this case they use a stochastic variant of k-nearest neighbor with *Mahalanobis metric*. This algorithm has an objective function that is not convex and can suffer from local minima. Metric Learning by Collapsing Classes (MLCC) [1] is an evolution of the above mentioned method that can be formulated by a convex problem but with the hypothesis that the examples in each class have only one mode. Another important algorithm in this family is the Large Margin Nearest Neighbor Classification (LMNNC) [8] that learns a *Mahalanobis distance metric* with a k-nearest neighbor by semi-definite programming and also in this case we have the semi-positive constraint for $\mathbf{M}$ in the optimization problem. Finally, a generalization of the LMNNC is the Gradient Boosted LMNNC (GB-LMNNC) [9] that learns a non-linear transformation directly in the function space. Specifically, it extends the *Mahalanobis metric* between two examples (e.g.: $\|\mathbf{L}\mathbf{x}_i - \mathbf{L}\mathbf{x}_j\|_2$) by using a non linear transformation ϕ to define the new Euclidian distance $\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|_2$. Given the non linearity of ϕ, GB-LMNNC uses the gradient boosted regression tree in order to change the metric (GBRT) [10]. So, the algorithm learns and combines an ensemble of multivariate regression trees (that are weak learners) using gradient boosting that minimizes the original LMNN objective function in the function space.

3 The KOMD Algorithm

The KOMD [11] algorithm is a kernel machine that optimizes the margin distribution in a game theoretic setting allowing the user to specify a trade-off between the minimal and the average value of the margin over the training set. Specifically, the classification task is posed as a two-player zero-sum game. The classification task requires to learn a unitary norm vector $\mathbf{w}$ such that $\mathbf{w}^\top(\phi(\mathbf{x}_p) - \phi(\mathbf{x}_n)) > 0$ for most of positive-negative instance pairs in the training data. The scenario of the game consists of one player that choose the vector of unitary norm $\mathbf{w}$ and the other that picks pairs of positive-negative examples according to distributions $\boldsymbol{\gamma}^+$ and $\boldsymbol{\gamma}^-$ over the positive and negative examples, respectively. The value of the game is the expected margin obtained, that is $\mathbf{w}^\top(\phi(\mathbf{x}_p) - \phi(\mathbf{x}_n)), \mathbf{x}_p \sim \boldsymbol{\gamma}^+, \mathbf{x}_n \sim \boldsymbol{\gamma}^-$. The first player wants to maximize this value while the second one wants to minimize it. This setting generalizes the hard SVM and can be solved efficiently by optimizing a simple regularized and linearly constrained convex function defined on variables $\boldsymbol{\gamma}$, namely,

$$\min_{\boldsymbol{\gamma} \in \Gamma} (1 - \lambda) \underbrace{\boldsymbol{\gamma}^\top \mathbf{Y}\hat{\mathbf{K}}\mathbf{Y}\boldsymbol{\gamma}}_{\mathcal{Q}(\boldsymbol{\gamma})} + \lambda \underbrace{\boldsymbol{\gamma}^\top \boldsymbol{\gamma}}_{\mathcal{R}(\boldsymbol{\gamma})}.$$

with $\mathbf{Y} = \mathbf{diag}(\hat{\mathbf{y}})$. The regularization parameter λ has two critical points: $\lambda = 0$ and $\lambda = 1$. When $\lambda = 0$, the solution is the hard SVM. In fact, let $\boldsymbol{\gamma}^* \in \Gamma$ the vector

that minimizes $\mathcal{Q}(\boldsymbol{\gamma})$, value of $\mathcal{Q}(\boldsymbol{\gamma}^*)$ in this case is the squared distance between the convex hull enclosing positive points $\phi(\mathbf{x}_p), \mathbf{x}_p \in \hat{\mathbf{X}}^+$, and the convex hull enclosing negative points $\phi(\mathbf{x}_n), \mathbf{x}_n \in \hat{\mathbf{X}}^-$, in the features space induced by the kernel $\mathbf{K}$. When $\lambda = 1$ the optimal solution is analytically defined by the vector of uniform distributions over positive and negative examples, that is, $\gamma^{(i)}_{unif} = 1/|\hat{\mathbf{X}}^+|$ when $y_i = +1$, and $\gamma^{(i)}_{unif} = 1/|\hat{\mathbf{X}}^-|$ when $y_i = -1$. In this case, the optimal objective value is the squared distance from the positive and negative centroids in feature space. The external parameter $\lambda \in (0, 1)$ allows to select the correct trade-off between the two extreme cases above. Clearly, a correct selection of this parameter is fundamental if we are interested in finding the best performance for a classification task and this is usually made by validating on training data. In Figure 1 an example of the solutions found by the above algorithm for a toy problem varying the value of λ is depicted.

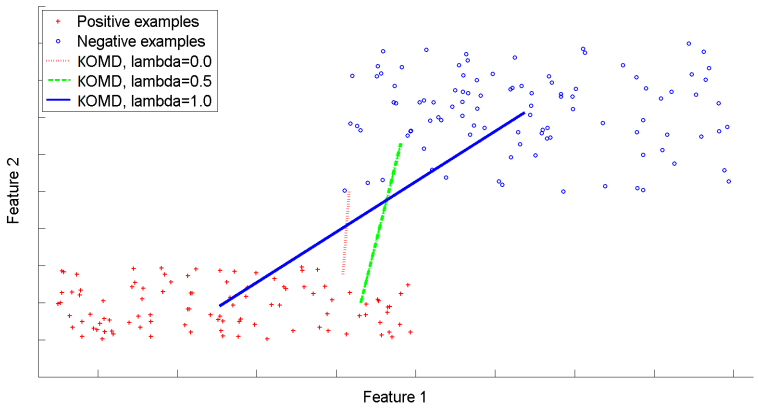

Fig. 1. KOMD solutions found using different λ in a simple toy classification problem.

4 Extending the game to features

In this paper, we propose to extend the game illustrated in Section 3 by considering an additional player which selects the kernel matrix $\mathbf{K}$ from the family of anisotropic *(Gaussian) Radial Basis Function* kernel (RBF). The RBF kernel is defined by

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\beta_0 \|\mathbf{x}_i - \mathbf{x}_j\|_2^2) = \exp(-(\mathbf{x}_i - \mathbf{x}_j)^\top \beta_0 \mathbf{I} (\mathbf{x}_i - \mathbf{x}_j))$$

where $\beta_0 \in \mathbb{R}_+$ is an external parameter. The RBF kernel can also be seen as using the trivial metric $\mathbf{M} = \beta_0 \mathbf{I} = \mathbf{diag}(\beta_0, ..., \beta_0)$. In the anisotropic RBF we have a generalized metric $\mathbf{M} = \mathbf{diag}(\beta^{(1)}, ..., \beta^{(m)})$ and we can write the anisotropic RBF as:

$$K_\beta(\mathbf{x}_i, \mathbf{x}_j) = \prod_{r=1}^{m} \exp(-\beta^{(r)}(\mathbf{x}_i^{(r)} - \mathbf{x}_j^{(r)})^2), \quad \boldsymbol{\beta} \in \mathbb{R}_+^m$$

where $\mathbf{x}_i^{(r)}$ is the r^{th} feature of the i^{th} example and $\beta^{(r)} \in \mathbb{R}_+$.

This new formulation has a greater number of degrees of freedom than the classical RBF kernel. A useful observation is that $K_\beta(\cdot,\cdot)$ can be seen as a element-wise product of kernels evaluated on a single feature. More formally:

$$K_\beta = \bigotimes_{r=1}^{m} K_{\beta^{(r)}}$$

with $K_{\beta^{(r)}}$ the RBF kernel defined on the r^{th} feature only with parameter $\beta^{(r)}$. From this point of view, finding the best parameters for an *anisotropic* RBF is a DML problem and we need to optimize the kernel representation by finding a trade-off between the components of $\boldsymbol{\beta}$.

We are now interested in an extension of the game presented in a previous section. For this, we define an additional player that sets the parameters of the anisotropic RBF. This player will prefer uncorrelated features so to avoid redundancies. For this reason we define a redundancy (or correlation matrix) $\mathbf{C}$ among the features $f_1, ..., f_m$, defined using an RBF kernel with parameter τ and normalized with respect to the number of features. Basically, each feature is considered as an example in order to generate the correlation matrix $\mathbf{C} \in \mathbb{R}_+^{m\times m}$ such that:

$$\mathbf{C}_{ij} = \exp(-\frac{\tau}{m}\|f_i - f_j\|_2^2) \;\; \forall i,j = 1, ..., m.$$

Finally, we propose to use the following regularized optimization problem as objective for the player $\boldsymbol{\beta}$:

$$\max_{\boldsymbol{\beta}\in\mathbb{R}_+^m} \;\; \mathcal{Q}(\boldsymbol{\beta},\boldsymbol{\gamma}) - \mu\mathcal{C}(\boldsymbol{\beta}) \tag{1}$$

where $\mathcal{Q}(\boldsymbol{\beta},\boldsymbol{\gamma}) = \boldsymbol{\gamma}^\top \mathbf{Y}\mathbf{K}(\boldsymbol{\beta})\mathbf{Y}\boldsymbol{\gamma}$ and $\mathcal{C}(\boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{\beta}^t\mathbf{C}\boldsymbol{\beta}$.

Note that, the proposed type of regularizer differs significantly from the usual trace regularizer used in kernel learning. In our opinion, the trace regularizer does not fit the notion of complexity in terms of the space of functions that can be generated using a kernel. For example, all RBF kernels have the same trace independently from the RBF parameter weighting the distance between examples, while the complexity of the resulting kernels can be dramatically different. On the other side, the correlation of the features on the parameters $\boldsymbol{\beta}$ that we propose, well fits the idea that good features are more useful if they represent different points of view of the examples.

Summarizing, the extended game we propose has value $\mathcal{Q}(\boldsymbol{\beta},\boldsymbol{\gamma})$ and the two players individually aim at optimizing their strategies according to the following optimization problems:

$$\mathrm{P}_{\boldsymbol{\gamma}}: \; \min_{\boldsymbol{\gamma}\in\Gamma} \;\; (1-\lambda)\mathcal{Q}(\boldsymbol{\beta},\boldsymbol{\gamma}) + \lambda||\boldsymbol{\gamma}||^2 \tag{2}$$

$$\mathrm{P}_{\boldsymbol{\beta}}: \max_{\boldsymbol{\beta}\in\mathbb{R}_+^d} \;\; \mathcal{Q}(\boldsymbol{\beta},\boldsymbol{\gamma}) - \mu\mathcal{C}(\boldsymbol{\beta}) \tag{3}$$

In the following, we give the simple alternating algorithm we used to solve the multi-objective problem given above.

(0) Find the best β_0 and λ with KOMD validation;
for *t=1,...,T* **do**
  (1) Set $\boldsymbol{\beta} = \boldsymbol{\beta}_{t-1}$ and generate a solution $\boldsymbol{\gamma}_t$ optimizing the problem described in Eq. 2;
  (2) Set $\boldsymbol{\gamma} = \boldsymbol{\gamma}_t$ and generate a solution $\boldsymbol{\beta}_t$ optimizing the problem described in Eq. 3;
end

Algorithm 1: ARBF algorithm

4.1 Gradient based optimization for $P_{\boldsymbol{\beta}}$

The function $\mathcal{Q}(\boldsymbol{\beta})$ in Eq. 3 has the r^{th} component of the gradient equal to:

$$\frac{\partial \mathcal{Q}(\boldsymbol{\beta})}{\partial \beta^{(r)}} = -\sum_{i,j} y_i y_j \gamma^{(i)} \gamma^{(j)} K_\beta(\mathbf{x}_i, \mathbf{x}_j)(\mathbf{x}_i^{(r)} - \mathbf{x}_j^{(r)})^2 = -\boldsymbol{\gamma}^\top \mathbf{Y}(\mathbf{D}_r \otimes \mathbf{K}_\beta)\mathbf{Y}\boldsymbol{\gamma}$$

where $\mathbf{D}_r \in \mathbb{R}^{n \times n}$ is the simmetric matrix of pairwise squared differences of the r^{th} feature, that is, $\mathbf{D}_r(i,j) = (\mathbf{x}_i^{(r)} - \mathbf{x}_j^{(r)})^2$. Then, the partial derivative of Eq. 3 with respect to $\beta^{(r)}$ will be:

$$\frac{\partial \mathcal{Q}(\boldsymbol{\beta})}{\partial \beta^{(r)}} - \mu \mathbf{C}_r \boldsymbol{\beta},$$

where $\mathbf{C}_r$ is the r^{th} row of $\mathbf{C}$.

4.2 Reducing the problem $P_{\boldsymbol{\beta}}$ to an unconstrained optimization problem

It is well known that solving a constrained optimization problem with gradient based optimization techniques is particularly difficult. For this, we reduced the problem to an unconstrained one by performing a simple change of variables, that is $\beta^{(r)} = e^{-\alpha^{(r)}}$. Computing the gradient with respect to variables $\alpha^{(r)}$ we obtain

$$\frac{\partial \mathcal{Q}(\boldsymbol{\alpha})}{\partial \alpha^{(r)}} = \frac{\partial \mathcal{Q}(\boldsymbol{\beta})}{\partial \beta^{(r)}} \frac{\partial \beta^{(r)}(\boldsymbol{\alpha})}{\partial \alpha^{(r)}} = (\boldsymbol{\gamma}^\top \mathbf{Y}(\mathbf{D}_r \otimes \mathbf{K}_\beta)\mathbf{Y}\boldsymbol{\gamma}) e^{-\alpha^{(r)}}$$

$$\frac{\partial \mathcal{C}(\boldsymbol{\alpha})}{\partial \alpha^{(r)}} = \frac{\partial \mathcal{C}(\boldsymbol{\beta})}{\partial \beta^{(r)}} \frac{\partial \beta^{(r)}(\boldsymbol{\alpha})}{\partial \alpha^{(r)}} = \mathbf{C}_r \boldsymbol{\beta} e^{-\alpha^{(r)}}$$

which leads to the following update

$$\beta^{(r)} \leftarrow e^{-\alpha^{(r)} - \mu(\frac{\partial \mathcal{Q}(\boldsymbol{\alpha})}{\partial \alpha^{(r)}} - \frac{\partial \mathcal{C}(\boldsymbol{\alpha})}{\partial \alpha^{(r)}})} = \beta^{(r)} \Delta_{\beta^{(r)}}$$

where we set $\Delta_{\beta^{(r)}} = e^{-\mu(\frac{\partial \mathcal{Q}(\boldsymbol{\alpha})}{\partial \alpha^{(r)}} - \frac{\partial \mathcal{C}(\boldsymbol{\alpha})}{\partial \alpha^{(r)}})}$.

The simple update above leads to an easy update for the kernel as in the following,

$$\mathbf{K}_\beta \leftarrow \mathbf{K}_\beta \otimes \exp((1 - \Delta_{\beta^{(r)}})\mathbf{D}_r)$$

where $\exp(\mathbf{M})$ denotes the element-wise exponential of a matrix $\mathbf{M}$.

5 Experiments and Results

We have performed the evaluation of our algorithm against six benchmark datasets of varying size, typology and complexity, and we have compared our performances with the same experiments performed using other techniques. The datasets used are *splice*, *ionosphere*, and *diabet* from UCI; *german*, *australian* and *heart* from Statlog (obtained from LIBSVM website[1]). The datasets have all the features scaled to the interval $[-1, 1]$. For each dataset, we constructed several splits containing 70% of the examples for the training set, 10% of the examples for the validation set and used the remaining 20% of the examples as the test set.

We compared our algorithm **ARBF** at different number of steps T, against the following baselines and state-of-the-art techniques:

- **KOMD**: in this case, model selection has been used to find the best parameters $\lambda \in \{0, 0.1, 0.5, 0.9\}$ and $\beta_0 \in \{0.01, 0.1, 0.5, 1.0\}$. A KOMD with standard RBF (shape parameter β_0) has been trained.
- **K-Raw**: this is kNN without learning any new metric, with validation and model selection to find the best k.
- **K-LMNN** and **K-GB-LMNN**: we used the implementation made by the authors[2] and we performed a model selection in order to find the best k for kNN.

Concerning our method, KOMD validation has been used to obtain the initial parameters (β_0 and λ, see Algorithm 1). The parameters $\mu \in \{1, 10, 100\}$ and $\tau \in \{1, 10, 100, 1000\}$ as been selected by model selection. For each technique a ranking over the examples in the test set is obtained (a function from the test set to $\mathbb{R}$). The Area Under Curve (*AUC*) metric is used to measure the performance of such a ranking function. AUC represents an estimation of the probability that a rank of a positive example is bigger than a rank of a negative one (both picked randomly). We evaluated AUC metric for each data set with different techniques and we have obtained the results in Table 1 (using $T = 20$ and $T = 50$, called respectively $\mathbf{ARBF}_{20}$ and $\mathbf{ARBF}_{50}$) and the convergence curves in Figure 2 with the AUC values for each iteration of our algorithm up to $T = 150$.

Data set	(N_e,N_f)	*KOMD*	*K-Raw*	*K-LMNN*	*K-GB-LMNN*	*ARBF*$_{20}$	*ARBF*$_{50}$
australian	(690,14)	$93.2_{\pm 1.5}$	$79.2_{\pm 4.8}$	$79.1_{\pm 5.3}$	$92.4_{\pm 9.2}$	$93.8_{\pm 2.1}$	$\mathbf{94.1}_{\pm 1.6}$
german	(1000,24)	$79.5_{\pm 2.2}$	$66.3_{\pm 2.6}$	$65.5_{\pm 3.0}$	$78.9_{\pm 5.6}$	$80.5_{\pm 4.1}$	$\mathbf{80.7}_{\pm 2.5}$
splice	(1000,60)	$93.7_{\pm 1.5}$	$68.1_{\pm 4.7}$	$79.7_{\pm 3.5}$	$\mathbf{95.1}_{\pm 2.8}$	$94.2_{\pm 1.5}$	$\mathbf{95.1}_{\pm 1.4}$
heart	(270,13)	$90.6_{\pm 3.4}$	$76.6_{\pm 9.5}$	$74.1_{\pm 11.2}$	$92.6_{\pm 7.5}$	$91.1_{\pm 6.0}$	$\mathbf{93.8}_{\pm 3.7}$
diabet	(768,8)	$84.0_{\pm 1.9}$	$72.0_{\pm 5.5}$	$70.9_{\pm 5.8}$	$86.3_{\pm 4.6}$	$86.9_{\pm 3.2}$	$\mathbf{87.1}_{\pm 3.1}$
ionosphere	(351,34)	$97.5_{\pm 1.4}$	$88.4_{\pm 3.8}$	$89.3_{\pm 4.3}$	$97.3_{\pm 3.8}$	$97.7_{\pm 3.5}$	$\mathbf{98.0}_{\pm 3.5}$

Table 1. AUC % (average$_{\pm std}$) obtained against 6 datasets with N_e examples and N_f features.

According to these results, our method obtains the best performance in all the six datasets and significantly improve on the baseline (KOMD) and other state-of-the-art techniques.

[1] `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

[2] `http://www.cse.wustl.edu/~kilian/code/code`

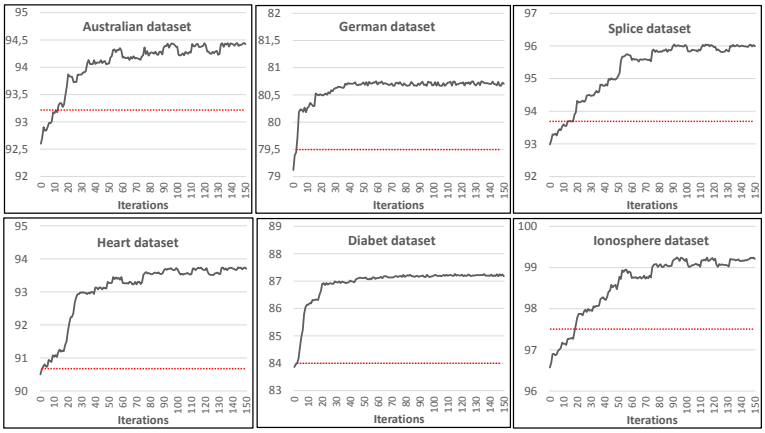

Fig. 2. AUC % values for each iteration of ARBF compared to the KOMD baseline (red dots).

6 Conclusions

We have presented a principled method to learn the parameters of a Anisotropic RBF kernel. We extended an existing kernel based method, namely KOMD, following the same game theoretical ideas used for learning the classifier to learn the kernel. The obtained results seems very promising as most of the times our methods improve the performance of the baseline significantly.

References

1. Amir Globerson and Sam T. Roweis. Metric learning by collapsing classes. In *NIPS*, 2005.
2. Jacob Goldberger, Sam T. Roweis, Geoffrey E. Hinton, and Ruslan Salakhutdinov. Neighbourhood components analysis. In *NIPS*, 2004.
3. Shai Shalev-Shwartz, Yoram Singer, and Andrew Y. Ng. Online and batch learning of pseudo-metrics. In *ICML*, 2004.
4. Carlotta Domeniconi and Dimitrios Gunopulos. Adaptive nearest neighbor classification using support vector machines. In *NIPS*, pages 665–672, 2001.
5. Nello Cristianini, John Shawe-Taylor, André Elisseeff, and Jaz S. Kandola. On kernel-target alignment. In *NIPS*, pages 367–373, 2001.
6. John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.
7. Gert R. G. Lanckriet, Nello Cristianini, Peter L. Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.
8. Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009.
9. Dor Kedem, Stephen Tyree, Kilian Weinberger, Fei Sha, and Gert Lanckriet. Non-linear metric learning. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2582–2590. 2012.
10. Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.
11. Fabio Aiolli, Giovanni Da San Martino, and Alessandro Sperduti. A kernel method for the optimization of the margin distribution. In *ICANN (1)*, pages 305–314, 2008.