PerTurbo: A New Classification Algorithm Based on the Spectrum Perturbations of the Laplace-Beltrami Operator

Nicolas Courty[1], Thomas Burger[2], and Johann Laurent[2]

[1] Université de Bretagne Sud, Université Européenne de Bretagne, Valoria
nicolas.courty@univ-ubs.fr
http://www-valoria.univ-ubs.fr/Nicolas.Courty/

[2] Université de Bretagne Sud, Université Européenne de Bretagne, CNRS, Lab-STICC
thomas.burger@univ-ubs.fr
http://www-labsticc.univ-ubs.fr/~burger/

Abstract. PerTurbo, an original, non-parametric and efficient classification method is presented here. In our framework, the manifold of each class is characterized by its Laplace-Beltrami operator, which is evaluated with classical methods involving the graph Laplacian. The classification criterion is established thanks to a measure of the magnitude of the spectrum perturbation of this operator. The first experiments show good performances against classical algorithms of the state-of-the-art. Moreover, from this measure is derived an efficient policy to design sampling queries in a context of active learning. Performances collected over toy examples and real world datasets assess the qualities of this strategy.

1 Introduction

Let us consider a vector space $\mathcal{X}$ (classically, a subspace of $\mathbb{R}^n, n \in \mathbb{N}$), called the **input space**, and a set of m labels $\mathcal{U} = \{u_1, \dots, u_\ell, \dots, u_m\}$. Moreover, we assume that there is an unknown function $h : \mathcal{X} \mapsto \mathcal{U}$, which maps any item $x \in \mathcal{X}$ onto a label $u = h(x) \in \mathcal{U}$ which identifies a class. The purpose of training a classifier on $\mathcal{X} \times \mathcal{U}$ is to find a "good" estimation $\hat{h}$ of h. To do so, we consider a training set $\mathcal{T} = \{(x_1, h(x_1)), \dots, (x_N, h(x_N))\}$, made of N items, as well as their corresponding labels. Any such $(x_i, h(x_i)) \in \mathcal{X} \times \mathcal{U}$ is called a **training example**. Then, $\hat{h}$ is derived so that it provides a mapping which is as correct as possible on $\mathcal{T}$, and following the *Empirical Risk Minimization principle*, it is expected that $\hat{h}$ will correctly map onto $\mathcal{U}$ other items $\tilde{x}$ of $\mathcal{X}$ for which $h(\tilde{x})$ is unknown (called **test samples**).

Most of the time, $\mathcal{T}$ is assumed to have some coherence from a geometric point of view in $\mathcal{X}$ and one seeks to characterize where items of $\mathcal{T}$ live [1]. Classically, one defines for each class ℓ, a latent space $\mathcal{Y}_\ell$ spanned by $\mathcal{T}_\ell$, the set of all the training examples with label u_ℓ. Then, when a new test sample $\tilde{x}$ is considered, it is projected onto $\mathcal{Y}_\ell$, $\forall \ell$ in order to estimate the similarity between $\tilde{x}$ and the

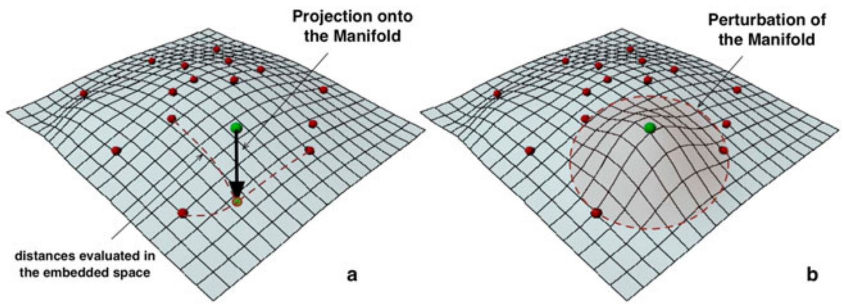
Projection onto the Manifold
distances evaluated in the embedded space
a
Perturbation of the Manifold
b

Fig. 1. The training examples (points in red) are used to evaluate a low-dimensional space where the data live (in light blue). Then, classicaly, once a new sample (in green) has to be labelled, it can be projected in this embedding, where an embedded metric is used (a). In the case of Perturbo (b), the deformation of the manifold is evaluated, thanks to the Laplace-Beltrami operator.

$\mathcal{T}_\ell$, as is illustrated in Figure 1(a). Finally, $\tilde{x}$ is classified according to the class to which it is the most similar.

In the literature [1], there are many such classification algorithms. Nonetheless, the main criticism is that these methods are parametric, or that strong assumptions are made, such as gaussianity and linearity (PCA+mahalanobis distance), or gaussianity and countablity (Hidden Markov Models). In this paper, we propose a non-parametric algorithm, efficient on datasets which may be embedded in non-linear manifolds. It is based on the following principle:

1- Each $\mathcal{T}_\ell$ is embedded in a dedicated manifold $\mathscr{M}_\ell$, which is characterized by means of tools derived from Riemannian geometry: an approximation of the Laplace-Beltrami operator. The latter is used to define the $\mathcal{Y}_\ell$.

2- The test sample $\tilde{x}$ is considered. Its similarity with each class u_ℓ is computed as follows: We consider the manifold $\tilde{\mathscr{M}}_\ell$ which embeds $\{\mathcal{T}_\ell, \tilde{x}\}$, i.e. the original training set $\mathcal{T}_\ell$ to which the test sample $\tilde{x}$ has been added, and we quantify the differences between the spectrum of $\tilde{\mathscr{M}}_\ell$ and $\mathscr{M}_\ell$, as is illustrated in Figure 1(b). In other words, we quantify the perturbation of the manifold $\mathscr{M}_\ell$ when $\tilde{x}$ is added to class u_ℓ.

3- $\tilde{x}$ is classified into the class, the manifold of which was the least perturbated by its adjunction.

The characterization of a Riemannian manifold by means of the Laplace-Beltrani operator is known from a long time [2], and was exploited thoroughly in the machine learning community [3,4]. In the context of computer vision and graphics, this theoretical framework has been used for 3D mesh manipulation [5,6] or, as an example, for large graph matching of 3D meshes [7]. Among other, it has recently been proposed to quantify the interest of a sample in a 3D mesh resampling task by use of matrix pertubation theory [8]. The main contributions of this paper are the following:

1- The adaptation of the perturbation measure used in mesh resampling to define a dissimilarity measure between a manifold and a sample, so that a new classification algorithm, called PerTurbo, can be derived;

2- Links between these computer graphics tools and classical methods in machine learning are provided;

3- Finally, we show the interest of PerTurbo for active learning problems.

The paper is organized as follows: Section 2 goes over the state-of-the-art. Section 3 is the core of the paper, in which the PerTurbo algorithm is derived and formal justifications are given. Finally, experimental assessments are described in Section 4.

2 State-of-the Art

Let us consider a Riemannian manifold $\mathscr{M}$, (i.e. differentiable everywhere). Its geometric structure can be expressed by defining the Laplace-Beltrami operator $\triangle(.)$, which can be seen as a generalization of the Laplacian operator over Riemannian manifolds. It completely defines the manifold up to an isometry [5]. Performing an eigenanalysis of this operator makes it possible to conduct various tasks on the manifold. Let $\mathbf{u}_i(\mathbf{x})$ be the eigenfunctions and their corresponding eigenvalues λ_i of this operator. They are the solutions to the equation:

$$\triangle(\mathscr{M}) \cdot \mathbf{u}_i(\mathbf{x}) = \lambda_i \cdot \mathbf{u}_i(\mathbf{x}). \tag{1}$$

Unfortunately, finding an analytic expression of this operator is generally not possible. However, it is established in [9], that the kernel function in the integral transform which models the propagation of the heat on a graph along time, noted $\mathbf{H}^t(.)$, and called the Heat Kernel, shares some properties with the Laplace-Beltrami operator. Notably, we have $\mathbf{H}^t(\mathscr{M}) = e^{-t\triangle(\mathscr{M})}$. For a short propagation time period, the exponential converges to its first order terms and we have [9]:

$$\triangle(\mathscr{M}) = \lim_{t\to 0} \frac{\mathbf{I} - \mathbf{H}^t(\mathscr{M})}{t}. \tag{2}$$

Hence the Heat Kernel can be used as a good approximation for the Laplace-Beltrami operator $\triangle(\mathscr{M})$. Unfortunately, the Heat Kernel may also not be known on $\mathscr{M}$. However, the latter can be robustly approximated by the Gaussian kernel applied to a sample $\mathcal{T}$ of $\mathscr{M}$. Finally, in [8,9], the characterization of a manifold $\mathscr{M}$ by means of the spectrum of its Laplace-Beltrami operator is approximated by the spectrum of $\mathbf{K}(\mathcal{T})$, the Gram matrix of $\mathcal{T}$, where the Gaussian kernel $k(.,.)$ is used as a dot product. Hence, the (ith, jth) term of $\mathbf{K}(\mathcal{T})$ is:

$$\mathbf{K}_{ij}(\mathcal{T}) = k(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right) \tag{3}$$

For the sake of compact notation, we simply write $\mathbf{K}$ instead of $\mathbf{K}(\mathcal{T})$, and we consider the spectrum of $\mathbf{K}$ as an approximation of the spectrum of $\triangle(\mathscr{M})$ that should be used for the characterization of $\mathscr{M}$.

Besides, as explained in the spectral graph theory framework [10,4,3,11], the Laplace-Beltrami operator shares a lot of properties with operators defined on the weighted graph of a manifold (called graph matrices). Notably, it is well-known [12] that $\triangle$ shares a common spectrum with the *generalized graph Laplacian* L. Let us note $\mathcal{G}$ the fully connected graph $(\mathcal{T}, E)$, the vertice $\mathcal{T}$ of which are sampled elements of $\mathscr{M}$, and the edges E of which are fitted with weights given by a symetric matrix W. The latter encodes a similarity metric on $\mathscr{M}$, and $W_{ij} \geq 0$. Then, L is defined such that $L_{ij} = W_{ij}$ if $i \neq j$, and $L_{ii} = -\sum_{j \neq i} W_{ij}$ otherwise. The spectrum of L is particularly interesting to efficiently capture the geometry of $\mathcal{G}$: The multiplicity of eigenvalue 0 provides the number of connected components of $\mathcal{G}$, and the corresponding eigenfunctions are (1) indicator functions of these connected components, and (2) invariant mesures of random walks evolving on these connected components.

In case of no eigenfunctions with eigenvalue 0, the k eigenfunctions with the smallest eigenvalues are indicator functions of a clustering of $\mathcal{G}$ into k clusters [13]. This result is derived from the previous one: the edges of $\mathcal{G}$ for which the smallest perturbation would lead to a 0 weight are highlighted by the procedure, so that separeted connected components appear. This perturbation of the edges can also be seen as a perturbation of the spectrum of L: We look for the graph with k connected components, the graph Laplacian spectrum of which is the closest to that of $\mathcal{G}$.

More generally, the eigenfunctions with the smallest eigenvalues capture the main part of the geometry of $\mathcal{G}$. In other words, if the largest eigenvalues are dropped, $\mathcal{G}$ is simplified (in the sense that fewer eigenfunctions are necessary to described it) in a manner which best preserves its geometry. If the edges of $\mathcal{G}$ only encode local neighborhood, only the local geometry is preserved, while more global patterns are regularized, such as in Graph Laplacian Eigenmap [14] (GLE).

GLE and similar techniques based on simplifying the geometry of a graph thanks to spectral analysis are really useful to estimate a "smooth" function on the graph, i.e. a function the behavior of which is rather homogeonous with respect to the simplified geometry, such as, for instance, the fonction h which maps the training examples and test samples to the labels. This is why they are rather similar to matrix regularization methods [15]. On the other hand, they can also be seen as dimensionality reduction methods [16]. More specifically, it appears that GLE is a particular case of Kernel-PCA [17], as is established in [10].

The core idea of Kernel PCA [17] is to perform a principal component analysis (PCA) in a feature space $\mathcal{Z}$ rather than in the input space $\mathcal{X}$. If one wants to embed a set of points $\mathcal{T}$ in a space $\mathcal{Y}$ of smaller dimensionality, the classical PCA corresponds to finding the eigenvectors of the $\dim(\mathcal{Y})$ highest eigenvalues of the following covariance matrix,

$$\mathbb{C}\mathrm{ov}(\mathcal{T}) = \sum_{x_i \in \mathscr{M}} x_i \cdot x_i^T, \tag{4}$$

whereas the Kernel PCA considers the Gram matrix $\mathbb{G}(\mathcal{T})$, whose (ith, jth) element is $q(x_i, x_j)$, where q is a kernel function. If the kernel is the euclidian dot product, i.e. if $q_{i,j} =< x_i, x_j >$, then $\mathbb{G}(\mathcal{T})$ turns out to have the same eigenvectors and eigenvalues as $\mathbb{C}\mathrm{ov}(\mathcal{T})$ (up to a square root). Finally, the projection $\tilde{x}_{\mathcal{Y}}$ of any test sample $\tilde{x}$ in the latent space $\mathcal{Y}$ is given by the following linear combination:

$$\tilde{x}_{\mathcal{Y}} = \sum_{\ell=1}^{\dim(\mathcal{Y})} \left(\sum_{i=1}^{|\mathcal{T}|} \alpha_i^\ell \cdot q(\tilde{x}, x_i) \right) \cdot \boldsymbol{e}_\ell \tag{5}$$

where $\alpha^\ell = \{\alpha_1^\ell, \ldots, \alpha_{|\mathcal{T}|}^\ell\}$ is the eigenvector of $\mathbb{G}(\mathcal{T})$ associated to the ℓth eigenvalue, and $\boldsymbol{e}_\ell$ the unit vector colinear to α^ℓ.

In [10], it is established that GLE [14] can be re-interpretated as particular cases of Kernel PCA, with a data-designed kernel. Let us consider the preivous graph $\mathcal{G}$, but we fit its edges with the following weights: $W_{ij} = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right)$, i.e. $W = \mathbf{K}$. Roughly, L can be interpreted as the transition rate matrix of a random walk (a continuous time markov chain) on $\mathcal{G}$. Then, from L, let us define another matrix T such that T_{ij} corresponds to the expectation of commutation time of the random walk between x_i and x_j. We have

$$T = \frac{1}{2} \int_0^\infty \left(e^{Lt} - \frac{\mathbb{1}}{|\mathcal{T}|^2} \right) dt, \tag{6}$$

where $\mathbb{1}_{ij} = 1$. As can be seen in [10], finding the space spanned by the eigenvectors of the $\dim(\mathcal{Y})$ smallest eigenvalues of L is equivalent to find the space spanned by the eigenvectors of the $\dim(\mathcal{Y})$ largest eigenvalues of T. Then, performing a Kernel PCA with kernel $q(x_i, x_j) = T_{ij}$ is equivalent to applying GLE.

The main interest of GLE with respect to generic kernel PCA, is to provide a more interpretable feature space. On the other hand, as pointed out in [10], the projection of a test sample onto $\mathcal{Y}$ is not possible, as there is no analytical form for the kernel: The consideration of a new sample would mean redefining $\mathcal{G}$ as well as q, and re-performing the kernel PCA. Thus, GLE can not be used for classification issue.

In [8], the manifold $\mathscr{M}$ practically corresponds to the 3D surface of an object that one wants to reconstruct from a mesh $\mathcal{T}$ which samples $\mathscr{M}$. To make sure that the reconstruction is the most accurate possible while minimizing the number of points in the mesh, it is necessary to add or remove samples to $\mathcal{T}$. To evaluate the interest of a point in the mesh, the authors propose to estimate the modification of the Laplace-Beltrami spectrum induced by the adjunction of this point. This work is based on the interpretation of this pertubation estimation in a kernel machine learning setting. In this paper, we propose to interpret the perturbation measure from [8] as a dissimilarity criterion. We show that the latter allows to conduct classification according to the dimensionality reduction performed in GLE. We also establish that this measure provides an efficient tool to design queries in an active learning scheme.

3 A New Classification Method

3.1 A Kernel Machine View on the Perturbation Measure

The perturbation measure proposed in [8] is easily understadable in the framework of the kernel trick [18]. First, as the Gram matrix of the Gaussian kernel can be used as a good approximation for the Laplace-Beltrami operator, let us consider the feature space corresponding to the Gaussian kernel. Let ϕ be the mapping from the input space $\mathcal{X}$ to the feature space $\mathcal{Z}$. In [8], it is established that the perturbation involved by the projection $r(\tilde{x})$ of $\phi(\tilde{x})$ on $\mathcal{Y}$ (the subspace spanned by $\phi(\mathcal{T})$) can be neglected, and that the perturbation is mainly the result of $o(\tilde{x})$, the component of $\phi(\tilde{x})$ which is orthogonal to $\mathcal{Y}$. As a consequence, a fair and normalized measure of the perturbation is given by $\frac{||o(\tilde{x})||^2}{||\phi(\tilde{x})||^2} = 1 - \frac{||r(\tilde{x})||^2}{||\phi(\tilde{x})||^2}$. The norm of $r(\tilde{x})$ remains to be computed. The projection over the space spanned by $\phi(\mathcal{T})$ can be written as the following operator: $\Phi(\Phi^T\Phi)^{-1}\Phi^T$ [19] if Φ is the matrix whose columns are the elements of $\phi(\mathcal{T})$. We get:

$$\begin{aligned} ||r(\tilde{\mathbf{x}})||^2 = ||\Phi(\Phi^T\Phi)^{-1}\Phi^T\phi(\tilde{\mathbf{x}})||^2 &= (\Phi(\Phi^T\Phi)^{-1}\Phi^T\phi(\tilde{\mathbf{x}}))^T \cdot \Phi(\Phi^T\Phi)^{-1}\Phi^T\phi(\tilde{\mathbf{x}}) \\ &= \phi(\tilde{\mathbf{x}})^T\Phi((\Phi^T\Phi)^{-1})^T\Phi^T\Phi(\Phi^T\Phi)^{-1}\Phi^T\phi(\tilde{\mathbf{x}}) \\ &= (\phi(\tilde{\mathbf{x}})^T\Phi)((\Phi^T\Phi)^T)^{-1}(\Phi^T\phi(\tilde{\mathbf{x}})) \qquad (7) \end{aligned}$$

which, with the kernel notation $\mathbf{k}_{\tilde{\mathbf{x}}} = \Phi^T\phi(\tilde{\mathbf{x}})$ and noting that $\mathbf{K} = \Phi^T\Phi$, gives $||r(\tilde{\mathbf{x}})||^2 = \mathbf{k}_{\tilde{\mathbf{x}}}^T\mathbf{K}^{-1}\mathbf{k}_{\tilde{\mathbf{x}}}$. Finally, remarking that for a Gaussian kernel $||\phi(\tilde{\mathbf{x}})||^2 = k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}) = 1$, the final perturbation measure reads [8]:

$$\tau(\tilde{\mathbf{x}}, \mathscr{M}) = 1 - \mathbf{k}_{\tilde{\mathbf{x}}}^T\mathbf{K}^{-1}\mathbf{k}_{\tilde{\mathbf{x}}}. \qquad (8)$$

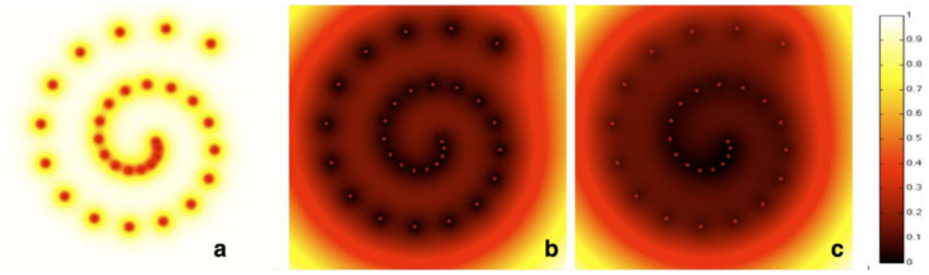

Fig. 2. Illustration of the measure on a spiral toy dataset: (a-b) The measure τ is shown with respectively two different sigma parameters $\sigma = 0.1$ and $\sigma = 1$. (c) The regularized τ measure according to equation (11) with $\sigma = 1$ and $\alpha = 0.1$.

Practically, this measure belongs to $[0, 1]$. If $\tau(\tilde{\mathbf{x}}, \mathscr{M}) = 0$, then, the new point does not modify the manifold, whereas if $\tau(\tilde{\mathbf{x}}, \mathscr{M}) = 1$, the modification is maximum, as is illustrated in Figure 3. From this measure, a natural class-wise measure arises, where a dissimilarity to each class is derived from the perturbation of the manifold $\mathscr{M}_\ell$ associated to class ℓ:

$$\tau(\tilde{\mathbf{x}}, \mathscr{M}_\ell) = 1 - \mathbf{k}_{\tilde{\mathbf{x}}}^T \mathbf{K}_\ell^{-\mathbf{1}} \mathbf{k}_{\tilde{\mathbf{x}}}. \tag{9}$$

Then, each new test sample $\tilde{\mathbf{x}}$, is associated to the class with the least induced perturbation, which reads as:

$$\arg\min_\ell \tau(\tilde{\mathbf{x}}, \mathscr{M}_\ell), \tag{10}$$

Finally, the PerTurbo algorithm simply relies on:

(1) Training step: The computation of the inverse of $\mathbf{K}_\ell$, $\forall \ell \leq m$. This corresponds to a computational complexity of $o(m \times \left(\frac{N}{m}\right)^3)$, $\frac{N}{m}$ being the mean number of training examples per classes.

(2) Testing step: The application of Eq. 9 and 10 which only involves basic vector-matrix multiplications to evaluate the perturbation measure m times.

Figures 3(a) and 3(d) illustrate the behavior of the estimated classification function ($\hat{h}$, according to the notation defined in the introduction) in different toy examples.

3.2 Perturbation Measure and Regularization Techniques

PerTurbo works as long as the perturbation measure τ is defined for all the classes. Hence, PerTurbo is always defined as long as $\mathbf{K}_\ell$ is invertible $\forall \ell \leq m$. If any of these matrices is not invertible, then, it is always possible to consider the pseudo-inverse: Roughly, it corresponds to only inverting the eigenvalues which are strictly greater than zero, and dropping the others. This logic can be extended to any cut in the spectrum of $\mathbf{K}^{-1}$, by dropping some eigenvalues of too small norm, as is explained in Section 2. Hence, if we consider only the greatest eigenvalues of $\mathbf{K}^{-1}$, i.e. (only the smallest none zero eigenvalues of $\mathbf{K}$), then, we practically consider for each class ℓ the subspace $\mathcal{Y}_\ell$ corresponding to the application of the GLE algorithm for dimensionality reduction to each set of example $\mathcal{T}_\ell$.

It is also possible to consider regularization techniques to find a close invertible matrix: For instance, in the case of Tikhonov regularization, one considers

$$\tilde{\mathbf{K}} = \mathbf{K} + \alpha I, \tag{11}$$

where I is the identity matrix, and $\alpha \in \mathbb{R}_+^*$. In such a case, it appears that the perturbation measure $\tau(.)$ shares important similarities with the kernel Mahalanobis distance derived from the regularized covariance operator [20]. The main difference derives from the fact that the measure τ is normalized and thus allows for comparisons between several classes. Hence, in spite of lack of centered data, the perturbation measure can be pictured as a Mahalanobis distance in the feature space. Let us note that, as indicated in [20], the kernel Mahalanobis distance is not affected by kernel centering.

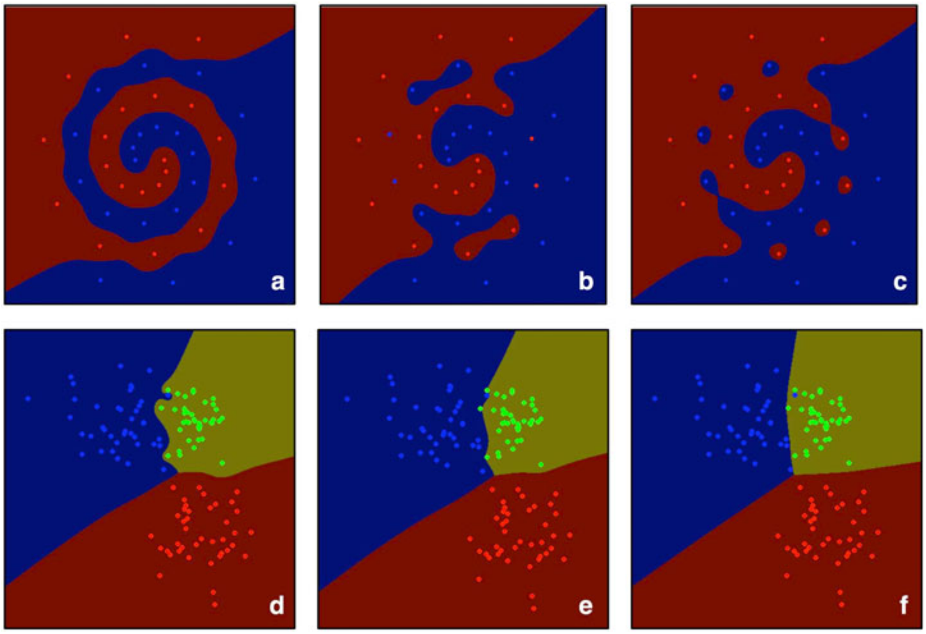

Fig. 3. Illustration of the classification procedure on two different datasets: (a-b-c) Two imbricated spirals and (d-e-f) a mixture of 3 Gaussians. In both cases, the lefthand images (a-d) present the classification with the original Perturbo measure, whereas center images (b-e) present sprectrum cut (95% of the eigenvalues were kept) and the righthand images (c-f) present the classification with the regularized Perturbo measure. In the two examples, we have $\sigma = 1$ and $\alpha = 0.1$.

Hence, beyond alternatives to define an approximation for $\mathbf{K}^{-1}$ when $\mathbf{K}$ is not invertible, these strategies are interesting to regularize the graph and consequently, the function that has to be trained on it. Finally, they can be used even in case of invertible matrices, in order to improve the efficiency of the classification, by an adapted dimentionality reduction strategy, such as illustrated on Figure 3, which exhibits the classification behavior of two toy examples, under different regularization constraints. This issue will be quantitatively discussed in the experimental section.

3.3 Active Learning

In most problems, unlabeled data (test samples) are numerous and free, whereas labelled data (training examples) are usually not, as the labeling task is expensive. Thus, the purpose of active learning is to improve the training by extracting some knowledge from the huge amount of test samples available. To do so, the training algorithm is allowed to query an oracle (classically, a human operator) to get the label of particular test samples, the knowledge of which would greatly improve the capability of the classfier. Thus, the key issue in active learning is to derive methods to detect these particular test examples [21]. To do so, rather intuitive and efficient strategies are based on defining particular regions of interest in $\mathcal{X}$, and to query the test samples which lives in them. More precisely, it is particularly efficient to query near the expected boundaries of the classes. Following the same idea, several other strategies can be defined, as is described in [21].

Depending on the classification algorithm, defining a region of interest around the expected borders of the classes may be rather complicated. To this extent, PerTurbo is particularly interesting, as such a region of interest can easily be defined according to the level sets of the implicit surfaces derived from the potentials of the $\tau_\ell(\tilde{\mathbf{x}})$, $\forall \ell$. In the case there are only two classes, the border corresponds to the zeros of $|\tau_1(.) - \tau_2(.)|$. Then, the region of the query should be defined around the border:

$$B = \{\mathbf{x} \in \mathcal{X} | \; |\tau_1(\mathbf{x}) - \tau_2(\mathbf{x})| < \gamma\} \tag{12}$$

and it can easily be represented, as depicted in Fig. 4. Similarly, in the case of more than two classes, the region to query corresponds to the neighborhood of the border between any pair of classes. Let us consider the two classes which are the least perturbated by the sample $\mathbf{x}$, and let us consider the corresponding pertubation functions $\tau_{r(1)}(\mathbf{x})$ ($r(1)$ being the least perturbated class) and $\tau_{r(2)}(\mathbf{x})$ ($r(2)$ being the second least perturbated class):

$$B' = \{\mathbf{x} \in \mathcal{X} | \; |\tau_{r(1)}(\mathbf{x}) - \tau_{r(2)}(\mathbf{x})| < \gamma\} \tag{13}$$

As pictured in Fig. 4, this region corresponds to the neighborhood of the expected borders of the classes. In the experimental section, we illustrate the efficiency of this query strategy for active learning with PerTurbo.

4 Experimental Assesment

In this part, we provide experimental comparisons between PerTurbo and the state-of-the-art. First we only consider the off-line classification, whithout any active learning method. Second, we focus on the relevance of the query strategies.

4.1 Classification Performances

In this section, we apply PerTurbo to various datasets, and we compare the resulting performances to the state-of-the-art.

We consider two kinds of datasets. First, there are three synthetic datasets that can be easily reproduced. Second, there are ten real datasets that are publicly available on the UCI Machine Learning Repository [22]. They are described in Table 1. In the case of simulated datasets, we use a Gaussian Mixture Model (GMM). For most of the datasets, no predefined training and test sets exists, so we randomly define them, so that 20% of the datasets are used for training, and the classification process is repeated ten times so that means and variances of

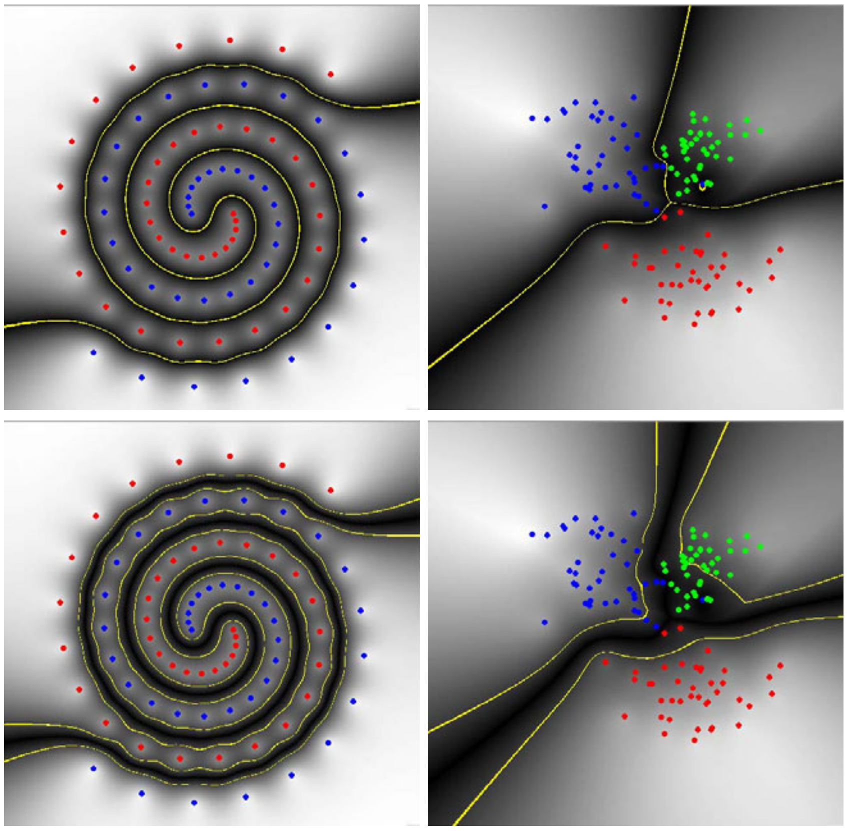

Fig. 4. Illustration of the estimation of the location of the border (top), and selection of the region to query (bottom), for toy examples: A two spiral case (left) and a three Gaussian case (right).

the performances can be computed. On the other hand, for datasets with predefined training and testing set (such as Hill-valley), we keep those provided so that the results are reproducible.

Three versions of PerTurbo are considered. In PerTurbo (full), we do not reduce the dimensionality of $\mathbf{K}^{-1}$. In PerTurbo (gle), on the contrary, we consider only its highest eigenvalues of $\mathbf{K}^{-1}$, so that they sum up to 95% of its trace, in a GLE-inspired way. Although it is possible, here we do not try to optimize the dimensionality reduction according to the dataset. Finally, in PerTurbo (reg), we apply a regularisation to $\mathbf{K}$. $\mathbf{K}^{-1}$ being non-inversible is mentionned in the experiments. Theoretically this should not happen, unless two samples are very close to each other, or due to numerical precision issues . Finally, the tuning of the σ parameter (from the Gaussian kernel) and the α parameter when needed are obtained by a classical cross-validation procedure operated on the learning set. Those parameters were set once for all the classes in a given dataset, although one value per class could have been obtained.

Table 1. Description of the GMM simulated datasets and of the datasets from UCI

Datasets	#Training	#Tests	#Classes	#Variables	Comments
SimData-1	200	800	10	19	64 components
SimData-2	200	800	10	26	64 components
SimData-3	200	800	10	31	75 components
Ionosphere	71	280	2	34	
diabets	154	614	2	8	missing values
Blood-transfusion	150	598	2	4	
Ecoli	67	269	8	7	too small for CV
Glasses	43	171	6	9	
Wines	36	142	3	13	
Parkinsons	39	156	2	22	
Letter-reco	4000	16000	26	16	
Hill-valley1	606	606	2	100	50% unlabeled
Hill-valley2	606	606	2	100	50% unlabeled

For comparisons, we consider Support Vector Machines (SVM) and k Nearest Neighbors (k-NN): SVM provides among the highest classification performances in the state-of-the-art. Moreover, numerous libraries implement it with several additional tools which help to tune the numerous parameters required in the use of SVM, or better, provide an automatic procedure to select the optimum parameters by means of cross-validation. This guaranties that the reference performances are optimized. On the other hand, k-NN is completely intuitive and requires the tuning of a single parameter (k the number of considered neighbors), so that it provides a naive and easy-to-interpret lower bound for any non-parametric model. Moreover, the difference of the performances between k-NN and SVM provides an indication of the sensitivity of the performances to the choice of the algorithm.

For SVM, we use, the R toolbox kernlab [23]. To choose the hyperparameter of the algorithm, we simply test all the available kernel and choose the most efficient one, which appears to be the Gaussian kernel. As a matter of fact, in all the datasets but one, the Gaussian Kernel provides the best results, sometimes with ties, sometimes with a strong difference. In the remaining one, the Laplace kernel performed very slightly better. Thus, the choice of the Gaussian kernel for all the dataset is reasonable. Then, the tuning of the parameters is automatically achieved thanks to a 5-fold cross-validation. For problems with more than two classes, a 1-versus-1 combination scheme is considered.

Concerning k-NN, the algorithm is applied for all the possible values of $k \in \mathcal{I}$ and the highest performance only is considered. Practically, the best value for k is sought in $\mathcal{I} = \left[1, \left\lfloor \sqrt{2 \cdot \frac{N}{m}} \right\rfloor\right]$, where $\frac{N}{m}$ is the means of training examples per classes, and $\lfloor . \rfloor$ represents the integer part. Depending on the number of classes and of k, ties are possible in the k-NN procedure. Then, they are randomly solved. In such a case, the performances vary, and we provide their expectation by computing the means of the repetition of 10 classifications.

Table 2. Comparison of the accuracy rates (percentages) with the various classification algorithms. For simulated datasets, the mean and standard deviation on ten-fold repetition is given. N.I stands for "not inversible", which refers to the inverse of **K**.

Datasets	PerTurbo (full)	PerTurbo (gle)	PerTurbo (reg)	SVM	k-NN
SimData-1	78.7(1.9)	79.0 (1.8)	78.7 (1.9)	76.7 (1.3)	75.7 (1.6)
SimData-2	54.0 (2.4)	54.7 (2.5)	54.0 (2.4)	45.0 (1.8)	40.5 (1.9)
SimData-3	19.5 (1.2)	19.7 (1.2)	19.5 (1.2)	16.2 (1.3)	16.7 (1.7)
Ionosphere	91.9 (2.5)	91.5 (2.0)	92.1 (1.6)	92.5 (1.8)	84.2 (1.3)
diabets	71.0 (3.0)	71.6 (3.2)	72.6 (2.2)	74.0 (1.7)	71.2 (1.9)
Blood-transfusion	N.I.	74.5 (2.3)	76.9 (1.0)	77.6 (1.5)	75.1 (0.9)
Ecoli	82.4 (2.8)	82.7 (2.45)	83.7 (2.5)	83.2 (2.2)	82.5 (1.8)
Glass	65.4 (2.9)	64.5 (3.9)	65.4 (2.9)	60.6 (4.4)	63.5 (2.4)
Wines	70.9 (3.3)	72.60 (1.3)	70.5 (2.8)	96.1 (1.7)	70.5 (2.1)
Parkinsons	82.8 (1.9)	82.8 (1.8)	83.8 (1.3)	85.0 (3.3)	82.5 (1.1)
Letter-reco	N.I.	92.7 (0.2)	92.5 (0.3)	91.9 (0.3)	90.0 (0.4)
Hill-valley1	60.5	53.8	60.7	56.4 (2.4)	62.0
Hill-valley2	59.6	54.1	59.9	55.3 (1.0)	56.3

Accuracy rates are given in Table 2. Let us note that, in this table, k-NN accuracies are over-estimated, as the k parameter is not estimated during a cross validation, but directly on the test set. This explains why its performances are sometimes higher than with SVM. On the contrary, SVM performances are optimized with cross-validations, as SVM is much too sensitive to (hyper-)parameter tuning. Finally, PerTurbo evaluations are loosly optimized, and a common tuning for each class is used. Hence, the comparison is sharp with respect to PerTurbo, and loose with respect to reference methods. From the accuracy rates, it appears that, except for the Wine datasets, where SVM completely outperforms both PerTurbo and k-NN, there is no strict and major dominance of a method over the others. Moreover, the relative interest of each method strongly depends on the datasets, such as suggested by the No Free Lunch theorem [24]. There are numerous datasets promoting PerTurbo (SimData-1, SimData-2, SimData-3, Glass, hill-valley1, hill-valley2), whereas some other promote SVM (Diabete, Blood-transfusion, Wine, Parkinsons), while the remaining are not significant to determine a best method (Ionosphere, Ecoli, Letter-recognition), as the performances are similar. Hence, even if from these tests, it is impossible to assess the superiority of PerTurbo, it appears that this framework is able to provide classification methods comparable to that of the state-of-the-art, and we are confident that, in the future, more elaborated and tuned version of PerTurbo will provide even more accurate results.

Concerning, the variations of PerTurbo (full, gle, reg), it appears that the best method is not always the same, and that accuracies strongly depend on the datasets. Hence, further investigations are required to find the most adapted variation, as well as a methodology to tune the corresponding parameters.

Beyond these tests, we have observed that, in general, PerTurbo is less robust/accurate than SVM when:

(1) values of some variables are missing or when there are integer-valued variables (binary variables being completely unadapted),

(2) the number of example is very small, as the manifold is difficult to characterize (However, the cross-validation for the tuning of the numerous parameters of SVM is also not possible with too few examples).

On the other hand, we have observed that when the number of classes is important, PerTurbo is more efficient than SVM. Moreover, in presence of numerous noisy variables, dimensionality reduction or regularization techniques provide a fundamental advantage to PerTurbo with respect to SVM. Finally, several other elements promotes PerTurbo: First, its extreme algorithmic simplicity, comparable to that of a PCA; Second, its simplicity of use and the restricted number of parameters; Third, its high computational efficiency, for both training and testing. As a matter of fact, the training is so efficient that the active learning procedures requiring several iterations of the training are easily manageable.

4.2 Active Learning Evaluation

Now, we focus on the active part of the training algorithm, and more specifically, we validate that the pertubation measure used in PerTurbo is adapted to derive and implement one of the most classical strategies to define queries (i.e. where queries are performed around the borders of the classes).

We both consider PerTurbo and k-NN. For these two algorithms, we construct the queries according to the perturbation measure, such as described in Section 3.3. Hence, we show that the validity of the latter for active learning is independent of the classification algorithm, also it naturally fits best with PerTurbo. Practically, between each training, the training set is enriched with a single sample, which has been chosen randomly amongst the samples which corresponds to 10% of the test set for which the criterion of Eq. 13 is the smallest. These results are compared to a reference strategy, where the queries are randomly chosen from the test dataset, according to a uniform distribution. Such a reference strategy is equivalent to compute the off-line performances of the classification algorithm for various sizes of the training dataset. Due to the random selection of the samples, the evolution of the accuracy of the classifiers may be rather noisy. Thus, the process is repeated 10 times, and we only provide means and variances. The results are given in Fig. 4.2 for three different datasets: Ecoli, Wine, and SimData-1. It clearly illustrates the efficiency of the algorithm, as the active learning strategy always outperforms the reference one: At the end of the process, the differences of performances correspond to 14.2, 27.8 and 12.9 points with the k-NN (respectively for Ecoli, Wine and SimData-1), and to 12.9, 20.9 and 10.5 with PerTurbo.

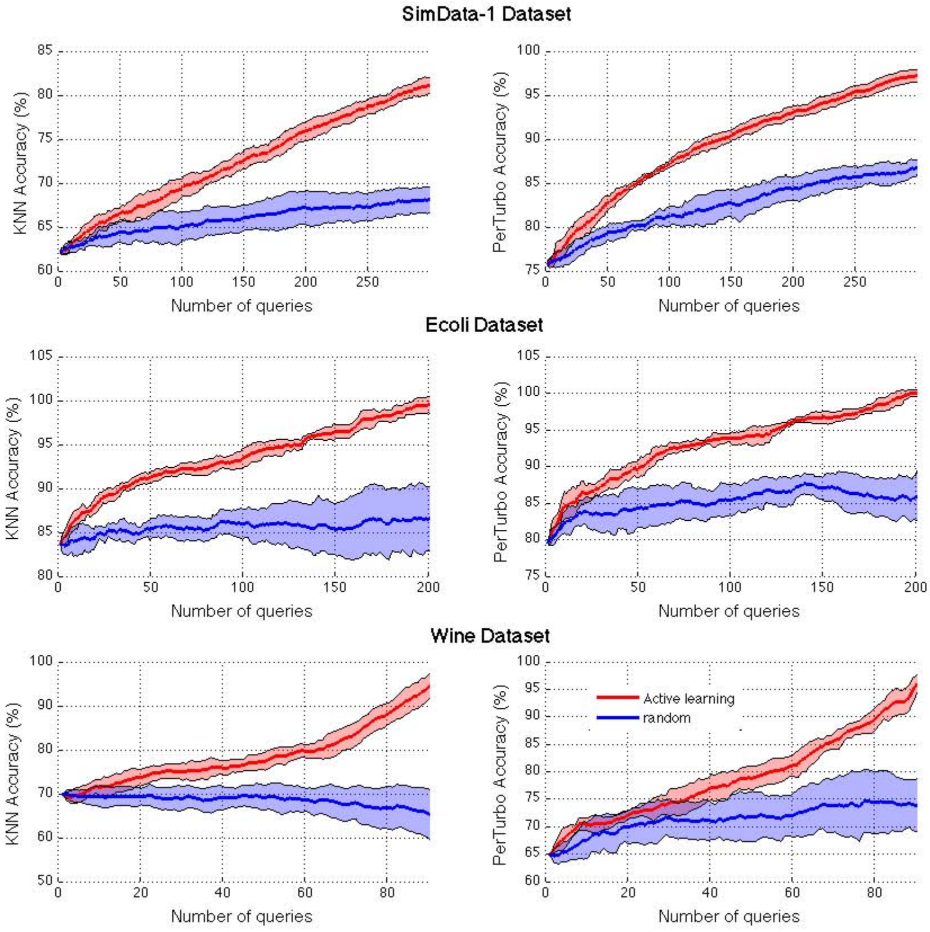

Fig. 5. Comparison of the active learning stragegy (red, upper trajectory) with the reference one (blue lower trajectory), for SimData-1 (Top), Ecoli (middle) and Wine (bottom) datasets. The classification algorithm is either PerTurbo (right column) or k-NN (left column). The thick line represents the mean trajectory, and the shaded strip around the mean represents the volatility.

5 Conclusion

In this paper, we have presented an original classification method based on the analysis of the perturbations of the spectrum of an approximation of the Laplace-Beltrami operator. Provided that the training examples of each class live on a dedicated manifold which dimension is much lower than the input space dimension, we propose to evaluate the modification induced by any test sample when added to the manifold of each class, and to choose the class corresponding to the manifold where the perturbation is the smallest. This makes it possible to derive an interesting strategy for sample queries in an active learning scenario. The method is very simple, easy to implement and involves very few extra parameters to tune. The experiments conducted with toy examples and real world datasets showed performances comparable or slightly better than classical methods of the state-of-the-art, including SVM. Future works will focus on a more systematical consideration of the parameters of the algorithm, as well as comparisons with more powerful kernel-based methods.

References

1. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley, Chichester (2001)
2. Chavel, I.: Eigenvalues in Riemannian geometry. Academic Press, Orlando (1984)
3. Lafon, S., Lee, A.B.: Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. IEEE Transactions on Pattern Analysis and Machine Intelligence 28, 1393–1403 (2006)
4. Nadler, B., Lafon, S., Coifman, R., Kevrekidis, I.: Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. In: NIPS (2005)
5. Reuter, M., Wolter, F.-E., Peinecke, N.: Laplace-beltrami spectra as "shape-dna" of surfaces and solids. Computer-Aided Design 38(4), 342–366 (2006)
6. Rustamov, R.: Laplace-beltrami eigenfunctions for deformation invariant shape representation. In: Proc. of the Fifth Eurographics Symp. on Geometry Processing, pp. 225–233 (2007)
7. Knossow, D., Sharma, A., Mateus, D., Horaud, R.: Inexact matching of large and sparse graphs using laplacian eigenvectors. In: Torsello, A., Escolano, F., Brun, L. (eds.) GbRPR 2009. LNCS, vol. 5534, pp. 144–153. Springer, Heidelberg (2009)
8. Öztireli, C., Alexa, M., Gross, M.: Spectral sampling of manifolds. ACM, New York (2010)
9. Coifman, R.R., Lafon, S.: Diffusion maps. Applied and Computational Harmonic Analysis 21(1), 5–30 (2006)
10. Ham, J., Lee, D., Mika, S., Schölkopf, B.: A kernel view of the dimensionality reduction of manifolds. In: Proc. of the International Conference on Machine learning, ICML 2004, pp. 47–57 (2004)
11. Belkin, M., Sun, J., Wang, Y.: Constructing laplace operator from point clouds in rd. In: Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, pp. 1031–1040. Society for Industrial and Applied Mathematics, Philadelphia (2000)
12. Dey, T., Ranjan, P., Wang, Y.: Convergence, stability, and discrete approximation of laplace spectra. In: Proc. of ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, pp. 650–663 (2010)
13. Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing 17(4), 395–416 (2007)
14. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural computation 15(6), 1373–1396 (2003)
15. Neumaier, A.: Solving ill-conditioned and singular linear systems: A tutorial on regularization. Siam Review 40(3), 636–666 (1998)
16. Lee, J.A., Verleysen, M.: Nonlinear dimensionality reduction. Springer, Heidelberg (2007)

17. Schölkopf, B., Smola, A., Müller, K.-R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computing 10(5), 1299–1319 (1998)
18. Aizerman, M.A., Braverman, E.M., Rozonoèr, L.: Theoretical foundations of the potential function method in pattern recognition learning. Automation and remote control 25(6), 821–837 (1964)
19. Meyer, C.: Matrix Analysis and Applied Linear Algebra. Society for Industrial and Applied Mathematics, Philadelphia (2000)
20. Haasdonk, B., Pękalska, E.: Classification with Kernel Mahalanobis Distance Classifiers. In: Advances in Data Analysis, Data Handling and Business Intelligence, Studies in Classification, Data Analysis, and Knowledge Organization, pp. 351–361 (2008)
21. Settles, B.: Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison (2009)
22. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
23. Karatzoglou, A., Smola, A., Hornik, K., Zeileis, A.: kernlab–an S4 package for kernel methods in R. Journal of Statistical Software 11(9) (2004)
24. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997)