Neural Comput & Applic (2011) 20:703–715
DOI 10.1007/s00521-010-0411-7

ORIGINAL ARTICLE

An unsupervised approach to learn the kernel functions: from global influence to local similarity

M. Ehsan Abbasnejad · Dhanesh Ramachandram · Rajeswari Mandava

Received: 1 December 2009 / Accepted: 5 June 2010 / Published online: 20 June 2010

Abstract Recently there has been a steep growth in the development of kernel-based learning algorithms. The intrinsic problem in such algorithms is the selection of the optimal kernel for the learning task of interest. In this paper, we propose an unsupervised approach to learn a linear combination of kernel functions, such that the resulting kernel best serves the objectives of the learning task. This is achieved through measuring the influence of each point on the structure of the dataset. This measure is calculated by constructing a weighted graph on which a random walk is performed. The measure of influence in the feature space is probabilistically related to the input space that yields an optimization problem to be solved. The optimization problem is formulated in two different convex settings, namely linear and semidefinite programming, dependent on the type of kernel combination considered. The contributions of this paper are twofold: first, a novel unsupervised approach to learn the kernel function, and second, a method to infer the local similarity represented by the kernel function by measuring the global influence of each point toward the structure of the dataset. The proposed approach focuses on the kernel selection which is independent of the kernel-based learning algorithm. The empirical evaluation of the proposed approach with various datasets shows the effectiveness of the algorithm in practice.

M. E. Abbasnejad · D. Ramachandram (✉) · R. Mandava
Computer Vision Research Group,
School of Computer Sciences, Universiti Sains Malaysia,
11800 Penang, Malaysia
e-mail: dhaneshr@cs.usm.my

M. E. Abbasnejad
e-mail: ehsan.com08@student.usm.my

R. Mandava
e-mail: mandava@cs.usm.my

Keywords Learning the kernels · Kernel methods · Support vector machine · Kernel-PCA

1 Introduction

Kernel methods in machine learning have gained significant attention in recent years. Supervised algorithms such as support vector machine (SVM) [35] and kernel Fisher discriminant analysis [26] as well as unsupervised algorithms like kernel principle component analysis (kernel-PCA) [30] and support vector clustering [3] have been successfully used in various real-world applications. The efficiency of these methods is highly dependent on the choice of the kernel function. A kernel function performs the *inner product* of two data points in a higher dimensional space. Kernel functions are tools for non-linearization of the algorithms through projection of data from the *input space* to a higher dimensional Euclidean space called the *feature space*. This projection is performed using a non-linear *mapping function*. A kernel function allows a mapping of data to a higher dimensional space in which linear analysis exhibits non-linear behavior in the input space. Consequently, many of the existing learning algorithms that depend only on inner products may be rewritten using kernels, a method referred to as *kernel trick*. Specifically, existing linear algorithms may be easily converted to a kernel-based method using kernel trick. (Interested readers may refer to [27, 32] for a comprehensive introduction to kernel methods).

Formally, A kernel function k is defined as:

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \; k : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \tag{1}$$

where $\langle . \, , . \rangle$ denotes *inner product* operation. *Mapping function* $\phi : \mathbb{R}^d \to \mathcal{H}$ maps data points x_i and x_j

non-linearly to Hilbert space $\mathcal{H}$. The kernel function k could be used in the construction of a $n \times n$ matrix:

$$K = (k(x_i, x_j))_{ij} \quad (2)$$

The matrix K is called *Kernel Matrix* (or *Gram Matrix*) of k with respect to $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$. The kernel matrix K is always symmetric and positive semidefinite.

In a kernel function k, the mapping function ϕ gives rise to the feature space in which the relation between data points is defined as a linear combination of their features. Consequently, it is implied that the kernel function defines the local relation between a pair of points regardless of the others in the dataset under study. An inner product also corresponds to the cosine between two input vectors. The angle between two vectors is used as the reference for their similarity: a larger angle corresponds to smaller similarity value. A good feature space defined by k is the one that best determines the similarity of points for a specific dataset.

According to Mercer's Theorem [4], any matrix may be interpreted as the inner product in the feature space as long as it is positive-definite, or conversely any positive-definite matrix may be used as a kernel matrix. Following the introduction of this theorem, a large number of kernels have been formulated. Among them are variations of parametric and non-parametric kernels. Some of the most popular ones are $k(x, x') = x^\top x'$ (Linear kernel), $k(x, x') = \exp(-||x - x'||^2/2\sigma^2)$ (Gaussian kernel with bandwidth σ) and $k(x, x') = (x^\top x' + 1)^d$ (Polynomial kernel with degree d) for two points x and x'. Each of the positive semidefinite kernels specify a unique *predefined* ϕ and a corresponding feature space $\mathcal{H}$.

Generally, the selection of an appropriate kernel for a given dataset is a tedious. In addition to the kernel selection problem, the determination of optimal parameters (referred to as hyper-parameters) for a given parametric kernel is also challenging (e.g. Gaussian and Polynomial kernel introduced earlier). These two issues are among the most crucial aspects of model selection in kernel-based algorithms. Recently, a new trend of research in machine learning called *learning the kernel* has emerged that aims to solve the aforementioned problems. This class of methods seeks to infer the kernel function/matrix from the data.

In this paper, we propose a solution for the kernel selection problem. The proposed approach produces a positive-definite kernel such that the feature space it generates best specifies similarity of the points for the dataset at hand. In an unsupervised setting, where no additional information about the dataset in consideration is unavailable, it may be beneficial to consider the structure of the entire dataset to derive a similarity measure based on the influence of a given point to all the other points. This is because similarity is a relative measure and is context-dependent. As an example in real-world, similarity in a group of students may have more emphasis on the gender while in other group may refer to the field of study.

The organization of this paper is as follows: In the proceeding section, the problem of kernel selection and its related works will be presented. The proposed approach will be detailed in Sect. 3. The discussions about the approach are presented in Sect. 4. The empirical evaluation of the algorithm on several datasets is provided in Sect. 5. Lastly, Sect. 6 presents the conclusion.

2 Learning the kernel

The most important aspect of model selection in kernel methods is the formulation of the kernel function. The kernel function is used as the generalized inner product of the points in a higher dimensional space which amounts to the similarity between those points in the input space. One of the promising methods of determining the optimal kernel for a given dataset is by learning it. This notion of learning the kernel formulation has spawned a new line of research in kernel-based machine learning. Finding such kernel is a challenging problem, because, the kernel implicitly represents the inner product in the feature space; hence, the mapping function that projects data to the feature space cannot be directly defined or accessed for analysis. However, it is possible to evaluate the kernel function based on other criteria such as defining the optimality conditions over the structural representation of data in the feature space or assessing the performance of the chosen kernel-based learning algorithm with the selected kernel.

The problem of learning has been traditionally solved using cross-validation [9]. It is the simplest and most commonly used method for learning the kernel. Cross-validation generally consists of testing a supervised algorithm with a range of possible kernels and hyper-parameters on a subset of training examples. Subsequently, the kernel and its corresponding hyper-parameters that produced the best average results are selected. Several runs of training and testing cycles are required to obtain a good result, which may not be viable for large datasets. Needless to say, cross-validation is limited to just a few kernels and hyper-parameters, and most notably it is not applicable to unsupervised algorithms.

The shortcomings of cross-validation in the context of learning the kernel have motivated researchers to propose various methods to infer kernel function/matrix from training data. Various approaches have been proposed to address the related issues in learning the kernels such as hyper-parameter selection [41] and construction of an appropriate kernel matrix [22, 38] to name a few. If adequate labeled examples are available, then one may use the

notion of kernel alignment [8] that measures the similarity between two semidefinite matrices. The kernel alignment measure can be used to assess the efficiency of the constructed kernel in an iterative method as in [8, 15]. A similar method has also been used in [16], in which a kernel matrix is optimized using the Gram-Schmidt optimization method.

As discussed earlier, kernel function is a representation of the inner product of mapped data points in the feature space. The mapped points in the feature space form a Riemannian manifold. Using the Riemannian metric, a conformal transformation of the kernel has been proposed by Amari [1]. Amari's approach is one of the earliest attempts to improve the kernels for SVM via direct modification of the mapping function. Conformal transformation of the kernels, as proposed by Amari, was adapted in [39] to improve the kernel functions through kernel discriminant analysis criterion in the feature space.

Apart from the aforementioned methods, several other approaches formulated the objective of learning the kernel as a convex optimization problem. Regularization of the kernel functions inferred from the *Reproduction Kernel Hilbert Space* (RKHS) results in the objective functions in the convex optimization form [17, 25]. The pivotal aspect in regularization is the choice of the loss function which identifies the convexity of the objective function. Lanckriet et al. [22] proposed the use of a linear combination of base kernels to construct the optimal kernel matrix in soft/hard margin classifiers. The kernel matrix of the combination of base kernels is constructed from training and test data in a transductive setting using semidefinite programming. Semidefinite programming [20] is a newly proposed class of convex optimization methods, which is specifically designed to solve problems with constraints on positive definiteness of the results. Another method that used semidefinite programming to learn the kernel was proposed by Weinberger et al. [36, 38]. Unlike Lanckriet's method that necessitates the availability of labeled examples, Weinberger's method is unsupervised and specifically aimed for dimensionality reduction. In conformance with other non-linear dimensionality reduction algorithms that seek to preserve a specific aspect of dataset, Weinberger's algorithm maintains the local distance between the data points to learning a kernel in the kernel-PCA. There are other similar unsupervised methods for kernel learning typically concentrated on manifold learning and dimensionality reduction [13, 31].

The combination of kernels used in [22] was extended by Bach et al. [2] to an algorithm called multiple kernel learning (MKL). The formulation of MKL is embedded into SVM such that kernel selection and learning the model are jointly performed. The integration of MKL within SVM leads to a mew supervised learning algorithm that consequently cannot easily be utilized for other problems. In MKL, each kernel learns from a distinct data source, which makes this approach more suitable for cases where fusion of data is required. This method has been extended in [29] to the cases where new formulation of learning algorithm (SVM) is not required. In this method, authors considered a gradient-based wrapper on the well-known implementation of SVM. On the contrary, a variation of MKL, referred to as composite kernel learning [23], focuses on learning the kernel function from a single data source.

In addition to the methods that solely consider labeled information, preserving the intrinsic geometric structure of data has also been recently used to improve the performance of the kernels. Among these methods, heat or diffusion kernel [18] and graph Laplacian kernel [14] are the data-dependent kernels built from scratch using training data. Other methods [19, 31, 42] also select an optimal kernel, such that it has the highest compatibility with the structure of the dataset. The significance of these methods, as discussed earlier, is that the optimal kernel obtained is specialized in distinguishing the similarities particularly for the dataset at hand.

Although each of the aforementioned methods has reported success in the specific problem for which they have been designed for, they may not be easily adaptable to other problems. In addition, in most of the aforementioned methods, there are parameters to be determined in prior. Therefore, we propose an approach to learn the kernel that is not dependent on a specific problem or learning algorithm. Our approach is an unsupervised algorithm that is formulated and explored in two different settings of convex optimization obtained from two methods that we examine for combining the base kernels. The details of the proposed approach will be discussed in the subsequent section.

3 Methodology

In general, to learn the kernel, we seek to find the optimal feature space such that the inner product of the points in that space best defines the similarity between the pairs of points in the input space. In order to find such feature space, one has to define the mapping function that is dependent on the nature of the problem and the characteristics of the underlying data (that may not be known a priori). This mapping function is implicitly defined in a kernel. In addition, a given kernel function defines the relationships between two points within a unique feature space. Therefore, instead of finding the corresponding mapping function, one may find the optimal positive-definite kernel to define the feature space of choice. Formally, given an unlabeled dataset $\mathcal{X} = \{x_1, x_2, \ldots, x_n \in \mathbb{R}^d\}$, we search for $\mathcal{H}^*$ as the optimal feature space:

$$\phi^* : \mathcal{X} \to \mathcal{H}^*, \kappa^*(x_i, x_j) = \langle \phi^*(x_i), \phi^*(x_j) \rangle \quad (3)$$

where κ^* is the optimal kernel function (or matrix $\mathcal{K}^*$ accordingly). To obtain κ^*, we propose an unsupervised approach that uses the intrinsic characteristics of the dataset to learn the optimal kernel. In other words, the optimality of the ultimate feature space is determined based on its intrinsic structure. These intrinsic characteristics of the dataset are captured by measuring the influence of each point in the input space and reflecting it in the feature space. In this case, if the influence of data points on the structure of the dataset is calculated, it is used to determine the relationship between data points. The influence of points in the input space and the feature space are thereafter probabilistically determined, i.e. the intrinsic structure of the dataset in the feature space is dependent on the structure of the input space determined by the measure that we will introduce. Consequently, the linear combination of the data points in the feature space is inferred from the input space which intuitively amounts to inferring the local relation from a global perspective.

Therefore, in the proposed algorithm, we initially calculate the influence of each point on the dataset, which we use subsequently to guide the learning of κ^*. We model the dataset as a graph and perform *random walks* [24, 33] to calculate the influence of each point in the dataset as detailed in Sect. 3.1. To perform random walks, the distance metric is defined in the input and feature space. In our case, the Euclidean distance is used, and its definition in the feature space is described in Sect. 3.2. After obtaining the measure of influence, the optimal kernel is obtained through a linear combination of the base kernel functions. The justification is provided in Sect. 3.3. As it will be shown, the linear combination of kernel functions could be regarded as a concatenation of various feature spaces that ultimately produces $\mathcal{H}^*$. Finally, the optimal combination of kernels is determined through an optimization problem that will be discussed in Sect. 3.4.

3.1 Determining structure of a dataset through random walks on the graph

In order to determine the influence of each point in the dataset, we define a probability space $(\mathcal{D}, \mathcal{C}, \mu)$. In this space, μ is the probability measure defined on the dataset $\mathcal{D}$ and $\mathcal{C}$ is the sets of events, i.e. the connectivity between the points. The probability measure μ is the notion that we interchangeably use to indicate the influence of each point in the dataset $\mathcal{D}$.

To determine the measure μ for each point in the dataset $\mathcal{D}$, we represent our dataset as a weighted graph. In this representation, each data point corresponds to a vertex $\{v_1, v_2, \ldots, v_n | v_i \in \mathcal{D}\}$ and each edge between v_i and v_j in the graph is weighted with a function $w(v_i, v_j)$ that satisfies [21]:

1. Matrix $W = (w(v_i, v_j))_{ij}$ is symmetric, i.e. $W = W^\top$.
2. Function w satisfies $w(v_i, v_j) \geq 0$.

We also introduce $d(v_i)$ to denote the degree of a node in the graph as:

$$d(v_i) = \sum_{j=1}^{n} w(v_i, v_j). \quad (4)$$

Having defined w and d, the probability of transition between node i and node j at timestamp τ is denoted by p_τ. That is, we define how probable it is to select node j while at node i. Thus, p_1 is calculated as:

$$p_1(v_i, v_j) = \frac{w(v_i, v_j)}{d(v_i)}. \quad (5)$$

It is clear that the following holds:

$$\sum_{j=1}^{n} p_\tau(v_i, v_j) = 1. \quad (6)$$

By considering P as the transition matrix constructed from p_1 for all vertices, one may calculate the transition matrix in successive timestamps (P is a Markov matrix). By treating this model as *Markov Chain* of the consequent probabilities of walking from one vertex to another in Eq. (5), the matrix of proximities P^τ at τ could be obtained. As τ increases, the local geometry information is propagated and accumulated to obtain the global characterization of a graph.

Furthermore, a similar formulation has been used in the diffusion maps [7, 21, 28]. It is shown that the probability p_τ induced from each timestamp t may be used to define a *diffusion distance* between the points:

$$D_\tau^2(v_i, v_j) = ||p_\tau(v_i, .) - p_\tau(v_j, .)||^2_{1/\mu(.)} \quad (7)$$

Diffusion distance between two points reflects the connectivity in the graphs, where larger distances imply higher probability of transition between those points.

Since increasing the value of τ reveals the global characteristics of the dataset, we are specifically interested in the probability values where $\tau \to \infty$. To calculate this probability, we define the vector u such that $u = uP$. The vector u is, in fact the left eigenvector of the transition matrix P, where the eigenvalue is 1. As the graph is connected, from the definition of u, it is clear that:

$$\lim_{\tau \to \infty} P^\tau = \mathbf{1}u \quad (8)$$

where $\mathbf{1}$ is the column vector of length n with all the entries equal to 1. The following also holds for each entry of u:

$$\lim_{\tau\to\infty} p_\tau(v_j, v_i) = u_i. \tag{9}$$

In this equation, u_i is called the *stationary distribution* of v_i mainly in the parlance of random walks. The value of u_i is given by:

$$u_i = \frac{d(v_i)}{\sum_{j=1}^{n} d(v_j)}. \tag{10}$$

Hence, u_i is dependent on the degree of v_i. If w is a shift invariant kernel function referring to the similarity of points, u_i is equal to the *Parzen Window* which estimates the density of the respective point [28].

Stationary distribution of each point has the highest potential to reveal the intrinsic structure of a dataset. Hence, we use the stationary distribution of each point to denote the influence of that point in the dataset. In case w is a similarity function, a larger u_i yields greater influence on the structure of the graph. In the proposed approach, we use the Euclidean distance between two points as function w. When the Euclidean distance is used for w, p_τ shows the probability of node i *not* transiting to node j. In other words, omitting a point with smaller value of u_i influences the connectivity, and consequently, the structure of the graph more strongly than the points with larger value of u_i. A point with smaller u_i is in close proximity to other points in the graph; therefore, it could be inferred that it is positioned in a dense area. In the definition of μ, we use u_i subject to the use of distance function in calculation of d. Therefore, μ is defined as:

$$\mu(v_i) = 1 - u_i = 1 - \frac{d(v_i)}{\sum_{j=1}^{n} d(v_j)}. \tag{11}$$

3.2 Euclidean distances and kernels

Kernel functions could be used to define distances between two points. As described, $\phi(x)$ is a nonlinear mapping to the feature space. Therefore, the Euclidean distance between two points could be defined between the mapped points in the feature space, i.e.:

$$\Delta_{ij} = ||\phi(x_i) - \phi(x_j)|| \tag{12}$$

which could easily be shown to be equal to:

$$\Delta_{ij}^2 = K_{ii} + K_{jj} - 2K_{ij} \tag{13}$$

where Δ_{ij} is the $n \times n$ matrix of the distances between the points i and j in the feature space. This equation implies that kernel functions may be used to infer the distance between points.

Specifically, the Euclidean distance is considered in this paper because it produces a simple representation in the feature space. As it is shown in Eq. (13), this distance in the feature space is easily calculated based on the kernel value obtained from the respective feature map. Furthermore, the Euclidean distance induced from the feature space is defined linearly on the kernel function which leads to a desirable formulation of the ultimate optimization objective of the proposed approach. Additionally, the Euclidean distance is a nonparametric shift invariant metric which does not need any value to be determined before being used.

It is also worth stressing that inferring distances from kernel functions may generally be seen as introducing a nonlinear metric. However, the problem of defining a kernel function is more generic than finding an appropriate distance metric, as different kernel functions may produce similar metrics [5]. Therefore, the proposed approach in this paper may be applied in the learning algorithms that depend heavily on the distance metrics between data points for their computation.

3.3 Combination of kernels

Linear combinations of kernel functions have recently been used as the basis for the introduction of new kernels. Each kernel in the combination has a specific mapping and a geometric representation in the Hilbert space yielding various characteristics. For example, the linear kernel is considered to assess the similarity between the points, even if they are located far from each other, while the Gaussian kernel is well-suited for assessing points in proximity. In a combination, each kernel could represent a set of features in which summation corresponds to their fusion. Therefore, by combining several mapping functions a feature space that best serves one's goals could be found.

Formally, the combination of kernels is a set of linear summation of b base kernel matrices defined as:

$$G(\mathcal{K}) = \left\{ \mathcal{K} | \mathcal{K} = \sum_{\ell=1}^{b} \alpha_\ell K_\ell \quad \forall \alpha \in \mathbb{R} \right\} \tag{14}$$

Correspondingly, kernel function κ is obtained from a linear combination of base kernel functions k_ℓ (from which kernel matrix K_ℓ is obtained) as

$$\kappa(x_i, x_j) = \sum_{\ell=1}^{b} \alpha_\ell k_\ell(x_i, x_j) \quad \forall \alpha \in \mathbb{R} \tag{15}$$

Each kernel function κ could be interpreted as concatenation of various feature spaces defined by base kernels:

$$\begin{aligned} \kappa(x_i, x_j) &= \overbrace{\langle \alpha_1\phi_1(x_i), \phi_1(x_j)\rangle}^{\alpha_1 k_1} + \overbrace{\langle \alpha_2\phi_2(x_i), \phi_2(x_j)\rangle}^{\alpha_2 k_2} + \cdots \\ &= \left\langle \begin{pmatrix} \alpha_1\phi_1(x_i) \\ \alpha_2\phi_2(x_i) \\ \vdots \end{pmatrix}, \begin{pmatrix} \phi_1(x_j) \\ \phi_2(x_j) \\ \vdots \end{pmatrix} \right\rangle \end{aligned} \tag{16}$$

In the set $G(\mathcal{K}), \alpha_\ell$ weighs each of the base kernel matrices K_ℓ with respect to the others. In general, two

assumptions could be made for the base kernels. In the first case, each kernel is dedicated to a set of distinct feature sets obtained from heterogeneous data sources. Thus, each base kernel is assumed to be suitable for a specific data source. In the second case, the base kernels are presumed to be suitable for the problem at hand, i.e. these kernels could be among those that have given good results for similar problems. Hence, all base kernels utilize a single data source. Our approach is in line with the latter case. It should be noted that we do not impose any constraints on the type of base kernels; thus, they could consist of the same kernels with varying parameters. Specifically, in this paper, we focus on the following two subsets of $G(\mathcal{K})$:

– *Affine combination of kernels*: the affine combination of the base kernels is defined as:

$$\mathcal{K} = \left\{ \mathcal{K} \in G(\mathcal{K}), \sum_{\ell=1}^{b} \alpha_\ell = 1 \right\}. \tag{17}$$

In general, the subspace obtained from the affine combination of positive-definite matrices in $G(\mathcal{K})$ is symmetric; therefore, it is required to constrain the resulting optimal kernel to satisfy the positive-definiteness condition of a kernel matrix.

– *Convex combination of kernels*: the convex combination of the base kernels is:

$$\mathcal{K} = \left\{ \mathcal{K} \in G(\mathcal{K}), \sum_{\ell=1}^{b} \alpha_\ell = 1, \alpha_\ell \geq 0 \right\}. \tag{18}$$

In this case, the set $G(\mathcal{K})$ is always positive-definite; therefore, the search space to find the optimal kernel is smaller.

In the kernel combinations, we consider in this paper, each of the base kernel matrices K_ℓ is normalized according to the normalization formula [10]:

$$(K'_\ell)_{ij} = \frac{(K_\ell)_{ij}}{\sqrt{(K_\ell)_{ii}(K_\ell)_{jj}}}. \tag{19}$$

Normalization of base kernels deters one kernel from dominating the final result in the optimal solution.

3.4 Learning the kernel through influence determination

The proposed approach is formulated as a parametric algorithm that seeks to determine the optimal combination of base kernels. In other words, the objective is to find the vector $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_m]^\top$ which are the coefficients of the linear combination of base kernels.

As described earlier, the optimal mapping function ϕ^* obtained from the combination of base kernels gives rise to the optimal kernel function κ^* and vice versa. We denote two sets $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ and $\Phi = \{\phi^*(x_1), \phi^*(x_2), \ldots, \phi^*(x_n)\}$ as the sets of points in the input and feature space, respectively. The probability of each point $\phi^*(x_i)$ being placed in the correct position in the feature space could be written as a mixture of m distributions, i.e.:

$$p(\phi^*(x_i)) = \sum_{c=1}^{m} p(c) p(\phi^*(x_i)|c) \tag{20}$$

in which $p(\phi^*(x_i)|c)$ is an independent distribution. In this case, the probability density is marginalized over parameter c. The probability of c denoted by $p(c)$ is called the mixing coefficient and can be interpreted as the prior probability of points in Φ given c. In general, Eq. (20) always holds for a given m. In this paper, we consider $m = n$ denoting the assumption that each point is dependent on parameter c. By using Bayes theorem, for a given $\phi^*(x_i)$, we have:

$$p(c|\phi^*(x_i)) = \frac{p(c)p(\phi^*(x_i)|c)}{\sum_{j=1}^{n} p(j)p(\phi^*(x_i)|j)}. \tag{21}$$

We specify $p(\phi^*(x_i)|c)$ with respect to the probability space $(\Phi, \mathcal{C}, \mu)$ as:

$$p(\phi^*(x_i)|c) = \begin{cases} 1 - \frac{d(\phi^*(x_i))}{\sum_{j=1}^{n} d(\phi^*(x_j))} & \text{if } i = c \\ 0 & \text{if } i \neq c. \end{cases} \tag{22}$$

This equation could be interpreted as an extreme case of Gaussian distribution that could be further simplified as:

$$p(\phi^*(x_i)|c) = \delta_{ci}\mu(\phi^*(x_i)) \tag{23}$$

where δ_{ci} is the Kronecker delta. This value determines the probability of parameter c for a given point in the feature space. Moreover, from Eqs. (21) and (23) we have:

$$p(c|\phi^*(x_i)) \propto p(c)\delta_{ci}\mu(\phi^*(x_i)) \tag{24}$$

In order to determine the utmost probability for each point, we maximize the conditional probability $p(c|\phi^*(x_i))$ as:

$$\text{maximize} \sum_{i=1}^{n}\sum_{c=1}^{n} p(c)\delta_{ci}\mu(\phi^*(x_i)) \tag{25}$$

It should be noted that the proposed approach is in the contrary to the well-known maximum likelihood paradigm where the learning algorithm is formulated as a single probability density function, and the objective is to maximize the joint probabilities of each feature in that distribution. In Eq. (25), we formulate the objective of the proposed approach to include several independent probability distributions whose union we intend to maximize. In other words, the probability that each point is placed in its correct position is identified as an event whose occurrence we seek to maximize.

To be able to maximize (25), we set $p(c)$ to be the stationary distribution of x_c in the input space, i.e. $p(c) = \mu(x_c)$. Consequently, a point that has a larger influence in the input space will be given higher chance to be assigned to its correct position in the feature space, or conversely, it needs to be moved lesser.

Therefore, using Eqs. (24) and (11), the objective of the proposed approach is formulated in an optimization that searches for an optimal α:

$$\underset{\alpha}{\text{maximize}}\ (\mathbf{1}-u)^\top(\mathbf{1}-u^*) \tag{26}$$

where u^* is the vector constructed from stationary distributions of each point $\phi^*(x_i)$ in the feature space according to Eq. (10). Thus, we can write each entry u_i^* as:

$$u_i^* \propto d(\phi^*(x_i)) \tag{27}$$

where d from Eqs. (4) and (13) in the feature space is:

$$\begin{aligned} d(\phi^*(x_i)) &= \sum_{j=1}^{n}\Delta_{ij} = \sum_{j=1}^{n}\mathcal{K}_{ii}^* + \mathcal{K}_{jj}^* - 2\mathcal{K}_{ij}^* \\ &= \sum_{\ell=1}^{b}\alpha_\ell \sum_{j=1}^{n}(K'_\ell)_{ii} + (K'_\ell)_{jj} - 2(K'_\ell)_{ij} \\ &= \sum_{\ell=1}^{b}\alpha_\ell \sum_{j=1}^{n}(B_\ell)_{ij} \end{aligned} \tag{28}$$

where B_ℓ is the matrix of distances of ℓth normalized base kernel.

Consequently, Eq. (26) could be simplified as:

$$\begin{aligned} \underset{\alpha}{\text{maximize}}\ (\mathbf{1}-u)^\top(\mathbf{1}-u^*) &\Rightarrow \text{maximize}_\alpha (u-\mathbf{1})^\top u^* \\ &\Rightarrow \text{maximize}_\alpha (u-\mathbf{1})^\top \Delta \end{aligned} \tag{29}$$

where it can be further simplified as:

$$\underset{\alpha}{\text{maximize}} -\sum_{\ell=1}^{b}\alpha_\ell \sum_{i=1}^{n}\sum_{j=1}^{n}(1-u_j)(B_\ell)_{ij} = -\alpha^\top s \tag{30}$$

In this equation, s can be interpreted as a column vector with size b of weighted sum of the distances obtained from each of the base kernels. Consequently, the value that has to be optimized is in fact a linear combination over α. To solve this optimization problem, we consider two cases based on the types of kernel combinations that we discussed earlier:

1. Affine combination of base kernels: The main issue in this case is the constraints that must be considered on the kernel matrix to be positive definite. Therefore, the objective of the proposed approach in affine combination of kernels is formulated as an instance of semidefinite programming. Semidefinite programming searches for the optimal solution in the intersection of the semidefinite cone and the affine constraints. In other words, the goal is to optimize a linear function of positive semidefinite matrices subject to linear constraints. In this way, the optimization is convex; therefore, it does not suffer from local optima.

$$\begin{aligned} \underset{\alpha}{\text{minimize}}\quad & \alpha^\top s \\ \text{subject to}\quad & \left(\sum_{\ell=1}^{b}\alpha_\ell K'_\ell\right) \succeq 0 \\ & \mathbf{1}^\top\alpha = 1 \end{aligned} \tag{31}$$

2. Convex combination of base kernels: In this case, the search space is limited to the positive-definite matrices; thus, the optimization could be solved easier than semidefinite programming. Hence, the optimal kernel could be obtained from the following linear programming problem:

$$\begin{aligned} \underset{\alpha}{\text{minimize}}\quad & \alpha^\top s \\ \text{subject to}\quad & \mathbf{1}^\top\alpha = 1, \alpha \geq 0 \end{aligned} \tag{32}$$

The solution to the above-mentioned optimization, in fact, lies in the intersection of the conic combination of α (in the objective function) and the hyperplane obtained from the affine combination of α.

It should be noted that one could describe the proposed approach as determining a weighted consensus of the base kernels. The consensus in the feature space is achieved using the guidance of the influence of points in the input space.

4 Discussion

The intuition of the approach, described in the previous section, could be expressed differently. We compute the stationary distribution of each point u_i (which corresponds to the influence of each point) in the input space and reflect this value in the feature space such that those with smaller u_i remain in dense areas. This indicates increasing the density of such points in the feature space which have higher influence on the structure of the dataset. That is, maximization of $\alpha^\top s$ increases the distances between each pair of the data points according to their contribution to the structure of the dataset in the input space. It should also be noted that as indicated in Eq. (6), $\sum_{j=1}^{n} u_j^* = 1$, thus, maximizing Eq. (26) naturally decrease u_i^* in the dense areas. This could also be justified by the diffusion distance in Eq. (7) where the point with smaller differences in u_i in the graph, are more strongly connected. Keeping the points with small u_i in proximity creates a dense area through which the other points are connected.

As an example consider two distinct points x_1 and x_2 in the dataset, each with stationary distributions u_1 and u_2, respectively. Assuming $u_1 > u_2$ indicates that x_2 is

contributing more to the dataset than x_1. However, maximizing their distances in the feature space would increase the distances between $\phi^*(x_1)$ and other points more than $\phi^*(x_2)$ i.e. $\mathcal{K}^*_{1j} > \mathcal{K}^*_{2j}$. Furthermore, if u_1 and u_2 are small with respect to other points, they will be less stretched apart while the points with larger u_i will be pushed away. Thus, points with less influence on the mapped dataset would scatter in less dense areas that have less effect on the decision-making processes.

Having the influential points concentrated in the dense areas, it is expected that the proposed approach outperforms the existing kernels specifically in maximum margin algorithms and dimensionality reduction. Variations in the maximum margin algorithms, either for classification or clustering, find a separating hyperplane in the area with lower density. Our approach seeks to find the influential points in the dataset and keep them in the dense areas, while allowing the rest of the points to scatter around according to their probability of transition. Therefore, similarity between the points is highly dependent on those with smaller stationary distribution. Similarly, this approach is justified for kernel-based dimensionality reduction algorithms like kernel-PCA. Kernel-PCA seeks to extract nonlinear principle components in the dataset. Hence, concentrating on the points with higher influence on the dataset will increase the possibility of finding an intrinsic pattern in feature space. Our empirical evaluation on SVM and kernel-PCA proved the effectiveness of the derived kernel from the proposed method.

5 Experiments

In this section, we present the empirical results obtained from applying the proposed approach to various datasets. In general, one must go through the following stages to obtain the optimal kernel: first, the normalized base kernel matrices $K_c^{'}$, the vector u and correspondingly s from the dataset are constructed. In the next stage, the optimization problem in Eq. (31) or (32) is solved. The optimization is performed using CVX. CVX is a package for specifying and solving convex programs implemented in Matlab [11, 12]. Finally, the coefficients of the optimal kernel obtained from the optimization process, is tested in the target kernel-based algorithms.

In order to show the effectiveness of the proposed approach, three experiments are performed. In the first experiment presented in Sect. 5.1, we illustrate the effectiveness of the optimal kernel obtained from the proposed approach in highly similar images. This experiment is performed to examine how the optimal kernel obtained from the proposed approach is able to discriminate the input features that may be highly similar. It is expected that a good kernel returns close values for similar images, while the distinctions between them are well-reflected in a specific dataset. In the second experiment, in Sect. 5.2, we illustrate the results of applying the optimal kernel to an unsupervised algorithm, i.e. kernel-PCA, for dimensionality reduction. A good kernel will capture the most valuable components of the data better, to evaluate the results of kernel-PCA. We assess the data obtained from kernel-PCA in k-nearest neighbor classifier (KNN) to be able to compare the results. In the last experiment in Sect. 5.3, the results of applying the proposed approach in a supervised algorithm, i.e. SVM, are illustrated. The best kernel is considered to be the one with higher accuracy in SVM. As it will be shown, the results of the optimal kernel are comparatively better than each of the base kernels in various cases.

5.1 Optimal kernel from highly similar images

In the first experiment, we test the ability of the optimized kernel to differentiate highly similar images. The teapot dataset from Weinberger and Saul [37] is used to run this experiment. This dataset originally consists of RGB-colored images of size 76 × 101 and consists of images of a teapot captured from various angles as it is rotated through

Fig. 1 This figure illustrates the images of 12 teapots captured consecutively used in the experiment in Sect. 5.1

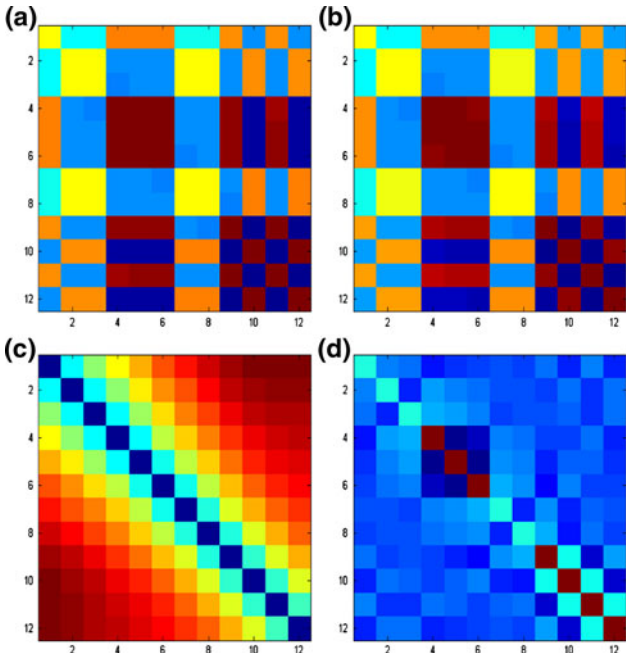

Fig. 2 The matrices constructed from different kernels are illustrated. In (**a**) and (**b**), Linear and Polynomial kernels are illustrated, respectively. The matrix of Euclidean distances is also shown in (**c**). Having the base kernels and the Euclidean distance between points, the proposed algorithm is used to construct the matrix shown in (**d**). The color of each entry in this matrix is based on the kernel value obtained from respective row and column. Similarity in colors shows closeness of their kernel values

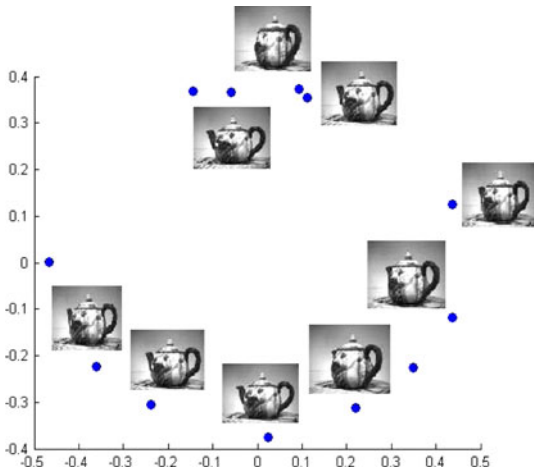

Fig. 3 In this figure, two eigenvectors corresponding to the largest eigenvalues are plotted. Some of the points are accompanied by their corresponding images. As it could be seen, some images remain closer to the center. These are images with higher influence

360 degrees on its base. We select 12 consecutive images from this dataset as illustrated in Fig. 1, and it is quite evident that the images look very similar to each other. Generally speaking, this poses a challenge to any learning algorithm which aims to differentiate each of these images. Subsequently, we construct three different base kernels, namely, the linear, Polynomial and the Gaussian kernels and the distance matrix as illustrated in Fig. 2a–c as colored block images. It should be stressed that the Gaussian kernel's bandwidth is selected manually to be two significantly different values. For the polynomial kernel, we set the degree parameter to 2. Having decided on three base kernels, the optimal kernel is constructed using the proposed approach, and the resulting matrix is illustrated in Fig. 2d. In the matrix visualization depicted in Fig. 2, the points with similar colors have closer value in the kernel matrix. As it can be clearly inferred from Fig. 2, although the object in the images differs in terms of viewpoint, this diversity has not been captured well by the base kernels, while the kernel constructed by the proposed method is able to clearly differentiate these images. This ability is reflected in the colored matrices shown in Fig. 2d where similar images are shown in similar colors. In general, it is expected that a good kernel function will return very close values for similar points and also shows the difference between the data points. Two eigenvectors corresponding to two largest eigenvalues of each of the kernel matrices are plotted in Fig. 3. As expected, highly influential points are grouped closer together by virtue of their similarity and points that are of lesser influence appear scattered apart.

5.2 Dimensionality reduction for face recognition

The formulation of an unsupervised method to learn the kernel paves the way for the use of the proposed approach in any of the kernel-based learning algorithms, which requires no labeled data to be made available. To test our unsupervised approach, we selected kernel-PCA as one of the possible learning algorithms that could be considered. Kernel-PCA is a nonlinear dimensionality reduction technique that has recently been proposed which tries to find a nonlinear projection of data that maximizes its variance.

As the dimensions of the face images are large, and many of the neighboring pixels have similar intensities, dimensionality reduction, in general, and kernel-PCA, in particular, had been used in face recognition problems [40]. In our experiment, we use the Carnegie Mellon PIE face dataset [34] as shown in Fig. 4. A subset of images in the dataset is selected and used to derive the optimal kernel, which is later used within the kernel-PCA. The feature vector of the images is constructed from raw pixel intensity values. Kernel-PCA is used to project the data to lower dimensions. Subsequently, the projected examples in the dataset are used to train the KNN classifier with neighborhood size 5. The result of the classification of the

Fig. 4 Several sample images from two different classes in PIE

Table 1 In this table, the accuracy (%) of classification using the low-dimensional representation of the face images using kernel-PCA is illustrated

Ex.	Dim.	L	P	G_1	G_2	Affine comb.					Convex comb.				
						LP	LG_1	LPG_1	LPG_2	LPG_1G_2	LP	LG_1	LPG_1	LPG_2	LPG_1G_2
50	20	77.33	86	50	57.33	90.67	90	92	**93.33**	**93.33**	86	77.33	86	86	86
	30	90	92	50	58	92	77.33	92	92	**93.33**	92	90	90	90	90
	40	92	**92.67**	50	67.33	92	92	92	92	92	**92.67**	92	91.67	91	91.33
100	20	86	96	50	68	97.33	86	96.67	**97.33**	**97.33**	96	86	96	96	96
	30	95.33	96.67	50	68.67	96.67	95.33	**97.33**	**97.33**	**97.33**	96.67	95.33	96.67	95.33	95.33
	40	96.67	96.67	50	66	**97.33**	96.67	**97.33**	**97.33**	**97.33**	96.67	96.67	96.67	**97.33**	**97.33**
150	20	44	46.67	25	27.67	**47**	44	45.33	44.6	44.67	46.67	44	46.67	45	45
	30	47.33	47.67	25	27.67	**49.33**	47.33	**49.33**	49	49	46.67	47.33	47.67	47	47.67
	40	49	49.33	25	27	49.33	49	49.67	48	48.67	49.33	49	49.33	**50**	**50**

In columns labeled by "Affine comb." and "Convex comb.", the accuracy of the optimal kernel obtained from the proposed approach in two respective settings are presented. The best results in each row are bolded

images is shown in Table 1. In this table, L stands for linear kernel and P denotes polynomial kernel. G_1 and G_2 indicate the Gaussian kernel with bandwidth 10000 and 150000, respectively. The combination of kernels is demonstrated using the concatenation of initial letters. Number of images used to derive the optimal kernel is shown in the first column. In the second column, various target dimensionalities are given. In other columns, the percentage of correct classification obtained from applying KNN to the dataset of reduced dimensions is shown. The results show that the proposed approach achieves better results from KNN, especially with smaller training examples. It can also be observed that the number of base kernels and their initial values influence the optimal kernel.

5.3 Text classification

In the last experiment, we evaluate our approach in a text classification task. We select 20-newsgroup dataset collected from 20 different newsgroups represented in 26214 size feature vectors.[1] The data is classified into 20-newsgroups, each corresponding to a different topic. For the purpose of this classification, we use C-SVC setting in Java implementation of LibSVM [6]. To have an equal setting for all our experiments, we set parameter C in C-SVC to its default value of 1.0. To build a training set for binary classification, we select different number of examples from the same class and assign a positive class value $+1$ to them. The negative training examples are also randomly selected from the other classes. Similarly, we create a test set from examples that are not selected in the training set. The accuracy of SVM with various kernels are shown in Table 2. As it is generally expected for text classification, performance of the linear kernel is comparatively better among the base kernels. In Table 2, the columns labeled as "Affine Comb." and "Convex Comb." present the accuracy of running SVM using the optimal kernel obtained from the proposed approach which corresponds to the affine and convex combination of base kernels, respectively. It is clear that in most of the cases, the accuracy of using the affine combination of base kernels outperforms other kernels in case of text classification. As it is shown in Table 2, if the base kernels are selected appropriately the result of the kernel optimization proposed in this paper is noticeably better. It may be noted that in cases where a pair of kernels is used, the overall performance is improved, even if one of the base kernels produces inferior results.

Ultimately, the comparison of accuracy results of running SVM using various kernels (convex or affine combination and the default which refers to the base kernels) is shown in

[1] Available at: http://www.people.csail.mit.edu/jrennie/20Newsgroups/.

Table 2 In this table, the accuracy (%) of running SVM with various kernels in the task of text classification is illustrated

Ex.	L	P	G_1	G_2	Affine comb.					Convex comb.				
					LP	LG_1	LPG_1	LPG_2	LPG_1G_2	LP	LG_1	LPG_1	LPG_2	LPG_1G_2
50	80.33	76.67	80.33	80	**87.67**	82.33	82.33	82	79.33	80.33	46.6	40	79	79.33
100	87.67	83	72	88	87.33	87.67	87.67	88	88.67	88.67	46.6	40	**97**	46.67
150	87.67	84.33	72	72	**89.67**	89.33	**89.67**	**89.67**	89	87.67	46.6	46.67	80	46.67
200	87	84.67	72	87	**89.67**	89	**89.67**	**89.67**	88.67	87.33	87	46.67	80	46.67

In columns labeled by "Affine comb." and "Convex comb.", the accuracy of the optimal kernel obtained from the proposed approach in two settings are presented. The best results in two respective settings are bolded

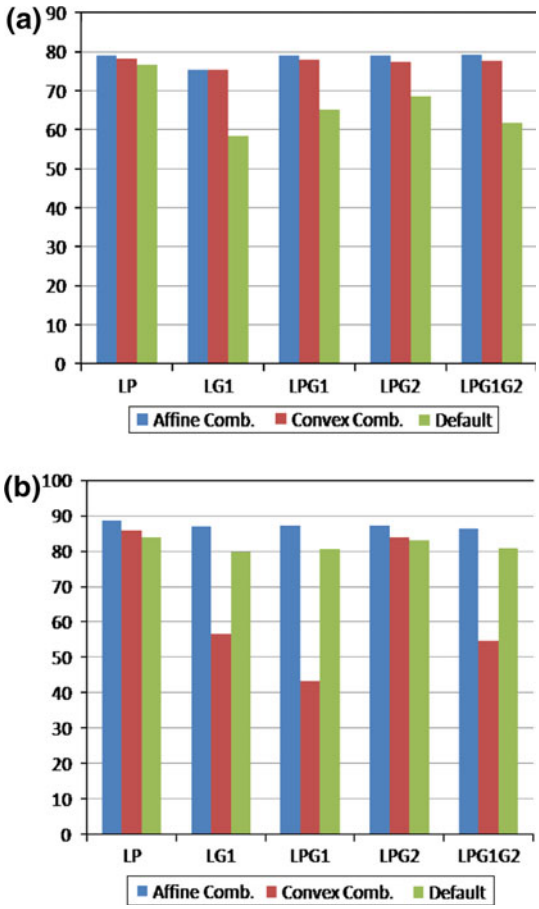

Fig. 5 Average of accuracy values obtained from classification in text and image datasets are illustrated. The comparison is made between the average values of the default kernels and the optimal kernels obtained from their combinations in the proposed approach

Fig. 5. It may be observed that in the task of dimensionality reduction, the optimal kernel obtained from both settings of the proposed approach significantly outperforms the base kernels. However, in case of text classification using SVM, the convex combination of base kernels does not generally produce as accurate results as base kernels. This can be explained if we consider the efficiency of the linear kernels in evaluating the text contents. In spite of this, the affine combination of base kernels provides very good accuracies in text classification task.

Table 3 In this table, the average time (in s) required to find the optimal kernel for given dataset is shown

Ex.	Face recognition			Text classification		
	2 kernels	3 kernels	4 kernels	2 kernels	3 kernels	4 kernels
50						
LP	28	34	42	22	29	35
SDP	27	39	48	23	29	36
100						
LP	121	149	183	131	163	194
SDP	120	163	185	131	162	197
150						
LP	356	432	579	373	446	541
SDP	375	482	640	371	437	538

This average time is computed from 4 times running the optimization problem. The number of base kernels is shown in the header of each column and the number of training examples is also shown in each row. The time is calculated for both linear programming (LP) and semidefinite programming (SDP) cases

The average time required to find the optimal kernel for given dataset is shown in Table 3. This average time is computed over 4 repetitions of the optimization algorithm using 2.66 GHz Core 2 Duo PC with 4 Gb RAM. In each column, the number of base kernels and in each row the number of training examples and corresponding optimization setting is shown. As expected, increasing the number of base kernels or training examples proportionately increases the time required to obtain the optimal kernel. In general, it can be seen that our approach is relatively fast. It is to be noted that the time required for computation can be significantly improved by using compiled languages or faster computers.

6 Conclusion

In this paper, we proposed a parametric approach to learn the kernel function that seeks to find a mapping to the feature space, with an emphasis on the influence of intrinsic characteristics of the dataset in the feature space. The global intrinsic of the dataset is reflected in terms of

stationary probability distribution of each point. It is evident that the global characteristics of a dataset, as used here, may be beneficial for the performance of the kernel, especially in cases where labeled data are not available. In the proposed approach, the intrinsic characteristics captured in the input space are used to find a combination of kernel functions to concatenate feature spaces using the available data. As discussed, the linear combination of kernels could be interpreted as weighting the mapping functions to form the feature space of choice for the dataset at hand. The empirical results in various real-world problems showed effectiveness of the proposed method in finding an optimal kernel without any labeled data. The optimal kernel obtained from the proposed method was used in two different kernel-based algorithms, SVM and kernel-PCA, as two popular supervised and unsupervised algorithms, respectively, to demonstrate the performance of the approach.

Acknowledgments This research has been made possible through the Science Fund Grant "Delineation and 3D Visualization of Tumor and Risk Structures" (DVTRS), No: 1001/PKOMP/817001 by the Ministry of Science, Technology and Innovation of Malaysia.

References

1. Amari S, Wu S (1999) Improving support vector machine classifiers by modifying kernal functions. Neural Netw 12(6):783–789. doi:10.1016/S0893-6080(99)00032-5
2. Bach FR, Lanckriet GRG, Jordan MI (2004) Multiple kernel learning, conic duality, and the smo algorithm. In: ICML '04: proceedings of the twenty-first international conference on Machine learning. ACM, New York, NY, USA, p 6. doi:10.1145/1015330.1015424
3. Ben-Hur A, Horn D, Siegelmann HT, Vapnik V (2002) Support vector clustering. J Mach Learn Res 2:125–137
4. Burges CJ (1998) A tutorial on support vector machines for pattern recognition. Data Min Knowl Discov 2:121–167
5. Burges CJC (1999) Geometry and invariance in kernel based methods. pp 89–116
6. Chang CC, Lin CJ (2001) LIBSVM: a library for support vector machines. Software available at: http://www.csie.ntu.edu.tw/cjlin/libsvm
7. Coifman RR, Lafon S (2006) Diffusion maps. Appl Comput Harmon Anal 21(1):5–30. doi:10.1016/j.acha.2006.04.006. http://www.dx.doi.org/10.1016/j.acha.2006.04.006
8. Cristianini N, Shawe-taylor J, Elissee A, Kandola J (2002) On kernel-target alignment. In: Advances in neural information processing systems 14. MIT Press, pp 367–373
9. Duan K, Keerthi SS, Poo AN (2003) Evaluation of simple performance measures for tuning svm hyperparameters. Neurocomputing 51:41–59. doi:10.1016/S0925-2312(02)00601-X. http://www.sciencedirect.com/science/article/B6V10-4625PVR-1/2/cc81e4581eca9413c3784c75c1ba0ee1
10. Graf A, Smola A, Borer S (2003) Classification in a normalized feature space using support vector machines. Neural Netw, IEEE Trans 14(3):597–605. doi:10.1109/TNN.2003.811708
11. Grant M, Boyd S (2008) Graph implementations for nonsmooth convex programs. http://www.dx.doi.org/10.1007/978-1-84800-155-8_7
12. Grant M, Boyd S (2009) Cvx: Matlab software for disciplined convex programming web page and software. http://www.stanford.edu/boyd/cvx
13. Guo Y, Gao J, Kwan P (2008) Twin kernel embedding. Pattern Anal Mach Intell, IEEE Trans 30(8):1490–1495. doi:10.1109/TPAMI.2008.74
14. Herbster M, Pontil M, Wainer L (2005) Online learning over graphs. In: ICML '05: proceedings of the 22nd international conference on Machine learning. ACM, New York, NY, USA, pp 305–312. http://www.doi.acm.org/10.1145/1102351.1102390
15. Kandola J, Shawe-Taylor J, Cristianini N (2002) On the extensions of kernel alignment. NC-TR-2002-120. http://www.eprints.ecs.soton.ac.uk/9745
16. Kandola J, Shawe-Taylor J, Cristianini N (2002) Optimizing kernel alignment over combinations of kernel. http://www.eprints.ecs.soton.ac.uk/9746/
17. Kim SJ, Zymnis A, Magnani A, Koh K, Boyd S (2008) Learning the kernel via convex optimization. pp 1997–2000. doi:10.1109/ICASSP.2008.4518030
18. Kondor RI, Lafferty J (2002) Diffusion kernels on graphs and other discrete structures. In: Proceedings of the ICML, pp 315–322
19. Kulis B, Sustik M, Dhillon I (2006) Learning low-rank kernel matrices. In: ICML '06: Proceedings of the 23rd international conference on Machine learning. ACM, New York, NY, USA, pp 505–512. http://www.doi.acm.org/10.1145/1143844.1143908
20. Vandenberghe L, Vandenberghe SB, Boyd S (1996) Semidefinite programming. SIAM Rev 38(1):49–95. http://www.jstor.org/stable/2132974
21. Lafon S, Lee A (2006) Diffusion maps and coarse-graining: a unified framework for dimensionality reduction, graph partitioning, and data set parameterization. Pattern Anal Mach Intell, IEEE Trans 28(9):1393–1403. doi:10.1109/TPAMI.2006.184
22. Lanckriet GRG, Cristianini N, Bartlett P, Ghaoui LE, Jordan MI (2004) Learning the kernel matrix with semidefinite programming. J Mach Learn Res 5:27–72
23. Marie Szafranski YG, Rakotomamonjy A (2008) Composite kernel learning. In: ICML
24. Meila M, Shi J (2001) A random walks view of spectral segmentation
25. Micchelli CA, Pontil M (2005) Learning the kernel function via regularization. J Mach Learn Res 6:1099–1125
26. Mika S, Ratsch G, Weston J, Scholkopf B, Mullers K (1999) Fisher discriminant analysis with kernels, pp 41–48. doi:10.1109/NNSP.1999.788121
27. Muller KR, Mika S, Ratsch G, Tsuda K, Scholkopf B (2001) An introduction to kernel-based learning algorithms. Neural Netw, IEEE Trans 12(2):181–201. doi:10.1109/72.914517
28. Nadler B, Lafon S, Coifman R, Kevrekidis I (2007) Diffusion maps-a probabilistic interpretation for spectral embedding and clustering algorithms. http://www.dx.doi.org/10.1007/978-3-540-73750-6_10
29. Rakotomamonjy A, Bach F, Canu S, Grandvalet Y (2007) More efficiency in multiple kernel learning. In: ICML '07: Proceedings of the 24th international conference on Machine learning. ACM, New York, NY, USA, pp 775–782. http://www.doi.acm.org/10.1145/1273496.1273594
30. Scholkopf B, Smola A, Muller KR (1998) Nonlinear component analysis as a kernel eigenvalue problem. Neural Comp 10(5):1299–1319. http://neco.mitpress.org/cgi/content/abstract/10/5/1299
31. Shaw B, Jebara T (2009) Structure preserving embedding. In: ICML '09: Proceedings of the 26th annual international conference on machine learning. ACM, New York, NY, USA, pp 937–944.doi:10.1145/1553374.1553494
32. Smola A, Hofmann T, Scholkopf B (2007) Kernel methods in machine learning. Ann Stat 36:1171–1220. http://eprints.pascal-network.org/archive/00003984

33. Society FRKCAM (1997) Spectral graph theory. Regional conference series in mathmatics. American Mathematical Society, Providence, RI
34. Terence Sim SB, Bsat M (2001) The cmu pose, illumination, and expression (pie) database of human faces. Technical Reports CMU-RI-TR-01-02, Robotics Institute, Pittsburgh, PA
35. Vapnik V (1999) An overview of statistical learning theory. Neural Netw, IEEE Trans 10(5):988–999. doi:10.1109/72.788640
36. Weinberger KQ, Packer BD, Saul LK (2005) Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In: Proceedings of the tenth international workshop on artificial intelligence and statistics, pp 381–388
37. Weinberger KQ, Saul LK (2006) An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In: Unfolding, Proceedings of the 21st national conference on artificial intelligence. AAAI
38. Weinberger KQ, Sha F, Saul LK (2004) Learning a kernel matrix for nonlinear dimensionality reduction. In: ICML '04: Proceedings of the twenty-first international conference on Machine learning. ACM, New York, NY, USA, p 106. doi:10.1145/1015330.1015345
39. Xiong H, Swamy M, Ahmad M (2005) Optimizing the kernel in the empirical feature space. Neural Netw, IEEE Trans 16(2):460–474. doi:10.1109/TNN.2004.841784
40. Yang MH, Ahuj N, Kriegman D (2000) Face recognition using kernel eigenfaces. In: Image Processing, 2000. Proceedings. 2000 International Conference on, vol 1, pp 37–40. doi:10.1109/ICIP.2000.900886
41. Zhang X, Lee WS (2006) Hyperparameter learning for graph based semi-supervised learning algorithms. In: NIPS. http://books.nips.cc/papers/files/nips19/NIPS2006_0148.pdf
42. Zhengdong Lu, Dhillon PJI (2009) Geometry-aware metric learning. In: ICML '09: Proceedings of the 26nd international conference on Machine learning. ACM