Knowl Inf Syst (2012) 31:193–221
DOI 10.1007/s10115-011-0404-6

SURVEY PAPER

A survey of the state of the art in learning the kernels

M. Ehsan Abbasnejad · Dhanesh Ramachandram · Rajeswari Mandava

Received: 10 October 2010 / Revised: 18 March 2011 / Accepted: 19 April 2011 /
Published online: 19 May 2011

Abstract In recent years, the machine learning community has witnessed a tremendous growth in the development of kernel-based learning algorithms. However, the performance of this class of algorithms greatly depends on the choice of the kernel function. Kernel function implicitly represents the inner product between a pair of points of a dataset in a higher dimensional space. This inner product amounts to the similarity between points and provides a solid foundation for nonlinear analysis in kernel-based learning algorithms. The most important challenge in kernel-based learning is the selection of an appropriate kernel for a given dataset. To remedy this problem, algorithms to learn the kernel have recently been proposed. These methods formulate a learning algorithm that finds an optimal kernel for a given dataset. In this paper, we present an overview of these algorithms and provide a comparison of various approaches to find an optimal kernel. Furthermore, a list of pivotal issues that lead to efficient design of such algorithms will be presented.

Keywords Machine learning · Kernel methods · Learning the kernels

1 Introduction

Kernel-based algorithms [72,75,77] (also known as kernel methods or kernel machines) have recently gained a significant attention in machine learning community. Supervised algorithms such as support vector machine (SVM) [81] and kernel discriminant analysis [60] as well as unsupervised algorithms like kernel principle component analysis (kernel-PCA) [73] and

M. E. Abbasnejad (✉) · D. Ramachandram · R. Mandava
Computer Vision Research Group, School of Computer Sciences,
Universiti Sains Malaysia, Penang, Malaysia
e-mail: ehsan.com08@student.usm.my

D. Ramachandram
e-mail: dhaneshr@cs.usm.my

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

support vector clustering [11] have been successfully applied to various real-world problems. Due to lower error rate compared to other learning methods, relatively fast training time and elegant compatibility with high-dimensional data these algorithms are the potential solutions to many problems.

The theoretical origins of the kernel methods may be traced to the work of Aronszajn [8] where the reproducing kernels were developed based on the foundation laid by the Mercer's theorem [57]. In later years, with the paradigm shift in machine learning toward nonlinear techniques, kernel methods have attracted more attention. This is because of their sound mathematical justification and better performance compared to their counterparts such as artificial neural network (ANN) and decision trees particularly in tackling nonlinear problems.

Kernel methods have revolutionized nonlinear learning algorithms by *nonlinear mapping* of data points to a higher, or possibly infinite, dimensional space (known as *feature space*) such that building a model (hypothesis) for a problem is easier. These approaches are called the kernel methods due to their dependency on the concept of *kernel functions*. A kernel function represents the inner product of the implicitly mapped points in a high dimensional space. This implicit mapping eliminates the need for costly feature transformations and leads to an efficient computation.

In kernel methods, the problem is modeled as a pairwise relation between data points which is captured in kernels. Thus, kernel functions (or simply kernels) have a profound impact on the performance of these learning algorithms. However, selection of the appropriate kernel function, and possibly its parameters known as *hyperparameters*, is extremely challenging. It is mostly due to the difficulty of explicitly accessing the high dimensional space. To remedy this problem, a new trend in machine learning known as *learning the kernel* (or *kernel learning* or *kernel selection* [31,51,53,74,87,100]) is becoming popular. Learning the kernel aims to select an optimal kernel, or its hyperparameters, to best define the nature of the underlying data. In simple terms one can describe this area of learning the kernel, as an attempt to find a kernel by either constructing a new kernel or fine-tuning the parameters of a given kernel for a specific dataset to achieve better performance.

In spite of all the recent advances in this area, there has not been any authoritative review of current methods in this relatively new and active area of research. In this paper, we hope to provide an extensive review of the state of the art that helps researchers to further address the problem of learning the kernel. Moreover, to the best of our knowledge, these algorithms have not been classified so that they can be easily compared. The contributions of this paper are twofold: first, to provide a review and comparison of the current approaches and investigate their merits; second, to present the challenging aspects of this problem and the approaches to address them.

This paper is organized as follows: In the subsequent section, the kernel function is introduced in more details. Later in Sect. 3, various aspects of the problem of learning the kernels are presented. Each of these aspects may be considered for further improvements. In Sect. 3.6, the algorithms in the area, categorized based on their optimality conditions, are detailed. The conclusion and the remarks for future developments are presented in Sect. 4.

2 Kernel functions

Kernel functions represent the inner product of the data points which amounts to the angle between their vectors. Consequently, it may be interpreted as the *similarity* between data points. Therefore, any learning algorithm that needs a measure of similarity can use a kernel

function for that purpose. Furthermore, through what is known as *kernel trick*, a large body of well-established linear algorithms may be easily converted to nonlinear methods.

Formally, a kernel function k for a dataset $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ is defined as:

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

where ϕ denotes *feature map* that maps the points nonlinearly to Hilbert space $\mathcal{H}$, i.e.$\phi : \mathbb{R}^d \to \mathcal{H}$. The kernel function k can be further used to build kernel matrix K as:

$$K = (k(x_i, x_j))_{ij} \quad \forall i, j = 1, 2, \ldots, n \tag{2}$$

Examples of popular kernel functions are $k(x, x') = x^\top x'$ (Linear kernel), $k(x, x') = \exp\left(-\|x - x'\|^2 / 2\sigma^2\right)$ (Gaussian kernel with bandwidth σ), and $k(x, x') = \left(x^\top x' + 1\right)^d$ (Polynomial kernel with degree d) where x and x' represent any two points in the data space. The bandwidth σ and degree d are examples of kernel hyperparameters. It can be shown that each of these functions amounts to an inner product of points in $\mathcal{H}$. In fact, as Mercer's theorem states, the only necessary condition for a function to represent inner product of two points in a higher dimensional space is its positive definiteness. It is a remarkable theorem that lays the foundation for constructing kernel functions without any need for direct definition of the mapping functions. Therefore, as long as one can prove that the matrix (function) is positive definite, it surely defines an inner product in a higher dimensional space and may be used in any kernel-based learning algorithm.

3 The challenges of learning the kernel

Learning the kernel is a challenging problem. It is because the implicit mapping of the points to the feature space deters direct analysis. Furthermore, any changes to the kernel have to be performed with respect to the constraint on its positive definiteness which is practically complicated to be applied for the given dataset. Additionally, the mapping function that projects data to the feature space cannot be directly defined or accessed for analysis. By learning the kernel, we hope to tackle these issues to achieve improved performance. The improved performance of a kernel implies the effectiveness of a kernel in defining similarities, capturing distinctions between data points, and representing an optimal inner product that leads to a better generalization to the unseen examples in the kernel methods.

The crucial question to be answered in this area is how well a kernel performs for a given dataset. This question is closely related to other learning algorithms where the learning criteria assess the efficiency of the algorithm (e.g. maximum margin in SVM). In the area of learning the kernels, similar criteria have to be defined to guarantee the performance of the algorithm. These criteria manifest the principles that will lead to an optimal kernel. As these criteria assess the optimality of a kernel and provide the conditions for improvement of the kernel for the given dataset, we shall call them *optimality conditions*. [1] These conditions lay the foundation of learning the kernel and justify the optimization of the kernel.

Optimality conditions in kernel selection, similar to any other learning method, lead to specifying particular objective or formulation of an ideal case whose fulfillment is sought. For example, assessment of the performance of the model based on minimization of the error rates using labeled data is a widely used optimality condition. In definition of the optimality conditions, usually nature of the problem like the availability of labeled data and dimensions

[1] In most of the current approaches, these conditions are imposed on the problem as optimization constraints, as such we may use these terms interchangeably.

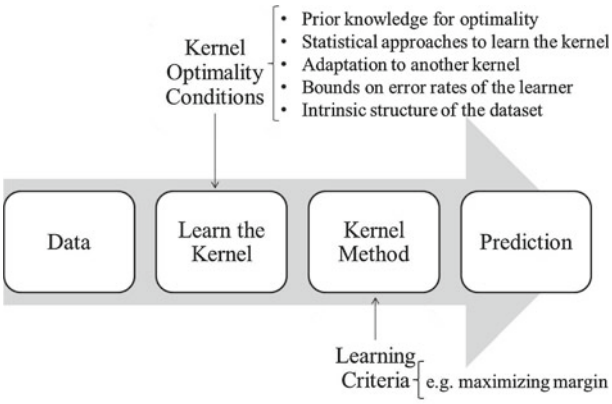


Fig. 1 The common process in learning the kernels

of the data is taken into account. The classification of current optimality conditions is as follows: prior knowledge for optimality, statistical approaches to learn the kernel, adaptation to another kernel, bounds on error rates of the learner and intrinsic structure of the dataset.

Optimality conditions are incorporated into learning the kernels similar to learning criteria in learning algorithms. Any learning algorithm starts by investigating the data and subsequently hypothesizing an appropriate model that best describes the underlying pattern of the problem. In the kernel methods, this model depends on the kernel. As shown in Fig. 1, the kernel may be learned prior to hypothesizing the model in kernel machines. In each stage, the appropriate conditions have to be considered in respective learning problem to ensure the optimality of the solution.

In addition to optimality conditions, other aspects that lead to distinction in learning the kernels may also be considered: optimization, kernel learning model, kernel selection phase, learning type, and the optimal kernel obtained. Further discussion on these aspects is crucial for understanding the nature of these algorithms and their advantages or drawbacks compared to others. On the other hand, such distinctions lead to a systematic classification of algorithms in this area, as shown in Fig. 2, that has not been investigated so far due to the contemporary rise of interest. These aspects shall be detailed below.

3.1 Optimization

Optimization and machine learning are two very closely related fields of research. As typical to most of the learning algorithms, learning the kernels may also be formulated as an optimization problem. There is a wide range of optimization techniques, each of which exhibits different properties. As such, formulation of the objective in a learning problem with superior performance of the optimization is sought. In learning the kernels in particular, variations in convex optimization [12] has been very popular [26,34,53]. A reason for this popularity is the guarantee on the uniqueness of the solution in convex problems. However, the implementation of such techniques and analysis of their performance are not generally straightforward. Nevertheless, there are publicly available toolboxes like `CVX` [29,30] that can solve these problems efficiently. On the other hand, gradient-based approaches [17,78] are easy to implement, efficient and simple to understand. However, gradient-descent requires the objective function to be differentiable which is usually challenging to formulate.

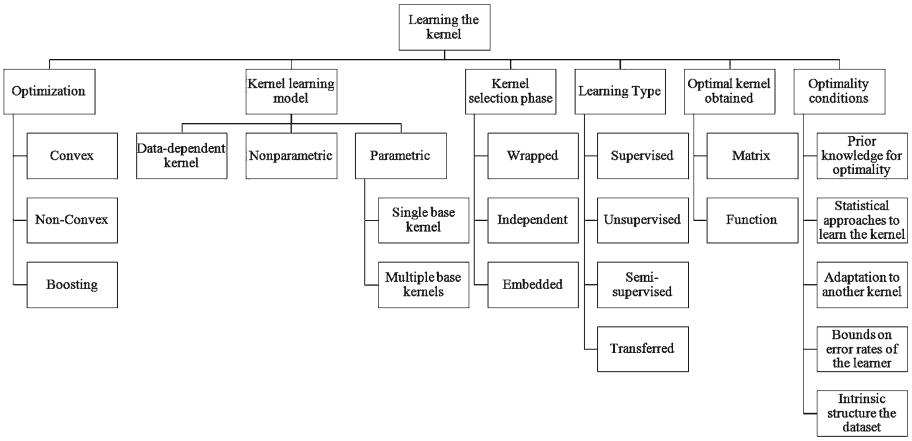


Fig. 2 The most important aspects of the algorithms for learning a kernel is summarized in this classification

3.2 Kernel learning model

In order to learn the kernel, one can develop a procedure to either improve base kernels or construct a new one from scratch based on the information obtained from data. In particular, we consider three families of models: data-dependent, nonparametric, and parametric. First family is the kernels created from available data and referred to as *data-dependent kernels*. These kernels, such as Fisher [38] and marginalization kernel [31], have simple formulation similar to that of a Gaussian kernel. However, the difference is that they have parameters to be determined based on the dataset while the parameter of Gaussian kernel is not dependent on the dataset and may be manually selected. In addition, the positive definiteness of these kernels has to be theoretically justified. However, a remarkable point is that their optimization step is limited to estimating the parameters of the data-dependent model through relatively simple approaches.

Second family of models is *nonparametric* approaches such as [33,35,65]. In these approaches, there is no prior model assumed for the kernel. The objective function of the algorithm is formulated as a set of user-defined criteria whose satisfaction is sought to find an optimal kernel. The drawback of these methods is that during testing, the optimal kernel has to be built from the training and test examples. That is because there is no model to be further used for testing.

The third family of models is *parametric* that consists of most of the available approaches in the literature. The methods in this family generally seek to form an optimization that satisfies user-defined criteria to find the parameters of a predefined model. There are usually two categories of parametric methods:

1. **Single base kernel:** Given a single base kernel, the objective is to find its refinement to an optimal kernel such that it performs better for a given dataset [3,4,15]. In this case, the goal is to either find the appropriate hyperparameters for the base kernel or its transformation to the optimal kernel. In this category of methods, selection of the base kernel is also important and the objective of learning may be interpreted as optimization of an initial kernel for a particular dataset.
2. **Multiple base kernels:** In these approaches, usually a set of base kernels are available and the objective is to find a weighting of them which results in improved performance

on a given dataset [6,10,49]. The combination of the base kernels could vary from linear to nonlinear approaches. However, most of the current methods are formulated as a linear combination of base kernels, i.e.

$$\kappa(x, x') = \sum_{i=1}^{m} \mu_i k_i(x, x') \tag{3}$$

where m is the number of base kernels and k_i denotes each base kernel. Here, μ_i is the value that defines the weight of each kernel and has to be ultimately determined by the learning algorithm. Furthermore, each of these kernels can be assigned to a particular set of features (or source of information) where learning the kernel may be interpreted as a means of fusion.

3.3 Kernel selection phase

The phase in which a kernel is optimized may be classified into three categories. First category consists of the methods that tend to select the kernel prior to its usage in the desired learning algorithm [6,15,49]. In this category, the kernel selection is completely independent of the learning algorithm itself, which makes it generic. In the second category, a wrapper algorithm to the kernel method is designed which iteratively alternates between two procedures: typically, in an outer procedure, the best kernel for that iteration is determined and in the inner one the model is learned [3,4]. The performance of the learning algorithm is evaluated at each step to further improve the kernel. At the end of this procedure, an optimal kernel and its corresponding model are obtained. These methods are more tightly bound to the inner learning algorithm which means in most of the cases the modification of the method is required upon any changes to the inner learning algorithm. The third category of methods is embedded into a particular kernel machine such as SVM [47,78]. A popular example of this category is multi-kernel learning algorithm [10] which led to introduction of a variation in SVM that uses multiple kernels and determines their weighting during learning. As the learning algorithm's objective is achieved jointly with the kernel selection, these embedded methods have better overall performance. However, these methods are solely useful for a particular learning algorithm and ultimately suitable for only a special class of problems.

3.4 Learning type

Selection of the kernel, similar to any learning algorithm, is performed based on the labeled, unlabeled, or mixture of labeled and unlabeled data. This corresponds to supervised [3,4,6,10], unsupervised [31,40], and semi-supervised learning [18,32,33] methods. In addition to this, data from previous examples of same task or similar ones may be *transferred* to be used to learn a more robust kernel [1,69]. In any case, due to the popularity and applicability of supervised learning, most of the existing body of work on the subject has concentrated on learning from labeled datasets.

3.5 Optimal kernel obtained

Generally, learning a kernel leads to either a matrix [18,20,31] or a function [3,4,6]. Although kernel-based algorithms work with both, the drawback of obtaining a kernel matrix rather than a function is that the matrix is learned in an inductive setting. This implies, prior to using the trained model, an extra stage of building the kernel matrix is inevitable. In these methods, the optimization of the kernel matrix is performed on both test and training data.

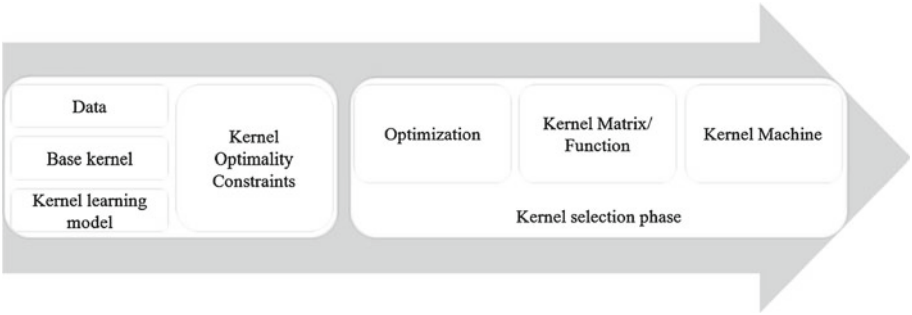


Fig. 3 The various aspects of learning the kernel in a common scenario

As such, it is not possible to learn a kernel from a set of training data alone and use it for testing purposes. However, this may be justified if we consider that one hopes to achieve better accuracy by learning from training and test examples even though it leads to having an extra step of kernel optimization for each test examples and sluggish prediction.

A summary of what has been discussed in this section is demonstrated in Fig. 3. As is illustrated, the data, the base kernel (if there exists any), and the kernel learning model are generally considered in designing the optimality conditions. Consequently, an algorithm is modeled as an optimization which leads to a kernel function or matrix. This kernel is consequently used in kernel machines. The kernel selection phase determines the quality of relation between optimization and the kernel machine. The detailed discussion on examples of these kernel learning algorithms is presented in the subsequent sections.

It is worth mentioning that learning the kernel is very closely related to the problem of distance metric learning [93] where the objective is to find an appropriate metric for a specific dataset. However, learning a kernel for a dataset is a more generic problem because finding a kernel entails a distance metric. It may be noted that several kernel functions amount to same distance metric, although this advantage comes with the price of a relatively complex criteria on the kernel functions to be positive definite [13].

In the subsequent section, state of the art in the area of learning the kernels categorized based on their optimality conditions is detailed.

3.6 Optimality conditions in learning the kernels

In this section, current approaches to learn the kernel are elaborated. In each of these methods, it is assumed that a set of n instances is given as $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$. If the dataset is labeled, a vector of labels y (typically in binary case $y_i = \pm 1$) is also available. The objective is to find the kernel function κ or matrix K. Furthermore, tr is the trace of the matrix and $\|.\|$ is the Euclidean norm.

3.6.1 Prior knowledge for optimality

Prior knowledge of the user is commonly used in learning algorithms. This knowledge may be used in learning the kernels too. For example, the relation between instances in terms of a comparative value that specifies sample a is more similar to sample b rather than c is useful in learning. In this case, [26] showed that a linear programming (LP) approach can be formulated to learn an appropriate kernel that satisfies this prior knowledge as a set of constraints. These constraints are used to find the parameterized kernel as in Eq. 3. The key

to an efficient computation of the optimal kernel is the diagonal dominance theorem that states if

$$M_{ii} \geq \sum_{j \neq i} |M_{ij}| \tag{4}$$

holds for a symmetric matrix M, then it is positive semidefinite.

Examples of other methods developed based on the prior knowledge of the user are those of [35] and [79] which we will discuss later.

3.6.2 Statistical approaches to learn the kernel

The statistical information in terms of probability distribution of data can be used to learn a kernel. The probability distribution in some cases is assumed to be known beforehand or selected based on the user's knowledge of the problem. The probability distribution can also be determined using density estimation techniques like Parzen windows. In any case, a relatively simple phase of either estimating the probability distribution itself or parameters of the user-defined probability distribution is required before using the kernel. It is additionally worth noting that in case the probability distribution is used in a generative approach, i.e. $p(x|\theta)$, the kernel obtained provides a way to combine the generative and discriminative approaches, e.g. SVM, in machine learning. In the following subsections, related approaches that utilized the probabilistic methods to determine the optimal kernel are presented.

Probabilistic kernels

One of the data-dependent kernels that falls in the family of statistical models is *natural kernels* proposed by [38]. If data are generated with probability distribution $p(x|\theta)$ in which θ denotes the parameters of this distribution and $l(x, \theta) = \ln p(x|\theta)$, then the natural kernel is defined as

$$k(x, x') = \nabla_\theta l(x, \theta)^\top M^{-1} \nabla_\theta l(x', \theta) \tag{5}$$

In case $M = I$ (identity matrix) the resultant kernel is known as *Fisher kernel* and if $M = \mathbf{1}\mathbf{1}^\top$ (where $\mathbf{1}$ denotes the column vector with all values set to one) the kernel is called the *plain kernel*. It is obvious that the value of $\nabla_\theta l(x, \theta)$ has to be estimated before kernel's usage. If a known probability distribution is used, the parameters can be estimated from data.

It can be proved that the Fisher kernel is a special case of *marginalization kernel* [80]. Marginalization kernel is inspired by the hidden Markov model (HMM) and consequently may be specifically useful in cases where data is generated in a sequence, for example a set of strings of characters or more generally in graphs. In marginalized kernels, these data are supposedly generated with a set of hidden parameters $h \in H$. Additionally, it is assumed that data generation is independent, i.e. $p(x, x'|h) = p(x|h)p(x'|h)$. The marginalized kernel is defined as

$$k(x, x') = \sum_{h \in H} p(x|h)p(x'|h)k_z(z, z') \tag{6}$$

where k_z is the joint kernel between two combined variables $z = (x, h)$ and $z' = (x', h')$.

Probability product kernel proposed in [40] defines a kernel between two distributions as:

$$k(x, x') = k_{\mathcal{P}}\left(p^\rho_{\hat{\theta}}(x),\, p^\rho_{\hat{\theta}}(x')\right) \tag{7}$$

In this expression, $\rho > 0$ is a constant and p^ρ is not infinity. Additionally, $k_\mathcal{P}$ provides a generic form of inner product between kernels. In case $\rho = \frac{1}{2}$, the popular *Bhattacharyya kernel* is obtained as $k(x, x') = \int \sqrt{p_{\hat{\theta}}(x)}\sqrt{p_{\hat{\theta}}(x')}\mathrm{d}x$.

In a particular case of probability product kernel, the Kullback-Leibler (KL) divergence is used to measure the similarity of two distributions in a kernel [61]. This kernel is defined as

$$k(x, y) = k_\mathcal{P}(p(x), p(y)) = \exp\left(-\alpha \tilde{D}(p(x)\|p(y)) + \beta\right) \tag{8}$$

In this definition, $\tilde{D}(p(x)\|p(y)) = D(p(x)\|p(y)) + D(p(y)\|p(x))$ is the symmetric version of KL divergence and α, β are constant values representing scale and shift. This kernel function is particularly useful in cases where the input features represent a probability distribution. For example, in computer vision, the image histograms representing the distribution of density values can be compared using this kernel function.

These data-dependent methods are interesting because they bring out the statistical aspects of data into building a kernel function. However, determining the appropriate distribution and their possible parameters may not always be feasible. Additionally, it is not possible to incorporate any preselected kernel into these methods in case user has a prior knowledge about the dataset. In the subsequent section, the probabilistic information is used to further improve the base kernel's performance in a dataset.

Improving kernels with probabilistic information

Apart from kernels obtained directly from distribution of data, it is possible to devise approaches that utilize statistical information to improve the given base kernels. Bayesian inference gives rise to the Gaussian processes [56,68] where the kernel matrix is employed as the covariance matrix in the normal distribution. This Bayesian inference in the Gaussian process may be used to estimate the parameter θ of the kernel K_θ [86]. The likelihood of this distribution is written as:

$$\log p(\mathcal{X}|\theta) = -\frac{1}{2}\log(\det(K_\theta)) - \frac{1}{2}y^\top K_\theta^{-1} y - \frac{n}{2}\log(2\pi). \tag{9}$$

With respect to the base kernel matrix K_θ and the derivative of Eq. 9, it is possible to develop an optimization that finds a local optimum for the problem.

Other than Gaussian process, a semi-supervised Bayesian method to select appropriate hyperparameters is proposed in [45]. Since it is a semi-supervised method, then the cluster assumption is made which means the labeled data is clouded with the unlabeled ones. Thus, the label y are related to the unlabeled data through hidden layer h as $p(y|D) = \int_h p(y|h)p(h|D)$ where D is the set of all labeled and unlabeled data. It is assumed that the kernel and model parameters are captured in parameter Θ. Using the Bayesian rule, the following EM algorithm is proposed to model the problem as $\max_\Theta \log[p(y|\mathcal{X}, \Theta)]$ and obtain Θ:

1. **E-Step** given the current Θ_i, approximate $\mathcal{N}\left(\bar{h}, \Sigma_h\right)$
2. **M-Step** update $\Theta_{i+1} = \arg\max_\Theta \int_h \mathcal{N}(\bar{h}, \Sigma_h) \log \dfrac{p(h|\mathcal{X}, \Theta)\, p(y|h, \Theta)}{\mathcal{N}\left(\bar{h}, \Sigma_h\right)}$

Other examples that construct the kernel matrix using the Bayesian inference include [97–99].

3.6.3 Adaptation to another kernel

One can define learning as the adaptation of the initial hypothesis to an ideal case such that the ultimate kernel obtained generalizes well to unseen examples. The ideal case provides a representation of the ultimate kernel suitable for a dataset. Consequently, adaptation procedure enables the kernel to preserve its initial characteristics while improving its performance by taking the ideal case into account. It is obvious that in the adaptation, the notion of "ideal case" and the way it is compared to the initial kernel play an important role. In the subsequent sections, the concept and examples of ideal case and adaptation will be more thoroughly presented.

Kernel alignment measure

Kernel alignment measure proposed in [18] is one of the most commonly used measures that compare two positive semidefinite kernel matrices. Here, the concept of *ideal kernel* plays an important role that refers to the best possible choice of kernel for the dataset at hand. In case of classification, the ideal kernel is matrix $yy^\top$ built from the labeled data. Then, for a given dataset, one can optimize the given kernel with respect to the ideal kernel. Furthermore, using the eigendecomposition of the kernel matrix given as $K = \sum_i \lambda_i v_i v_i^\top$, the kernel alignment measure is defined as

$$\begin{aligned} A(y) &= \frac{\langle K, yy^\top \rangle}{\sqrt{\langle K, K \rangle \langle yy^\top, yy^\top \rangle}} \\ &= \frac{\langle K, yy^\top \rangle}{n\sqrt{\sum_{ij} \lambda_i \lambda_j \langle v_i v_i^\top, v_j v_j^\top \rangle}} \\ &= \frac{\sum_i \lambda_i \langle v_i, y \rangle^2}{\sqrt{\langle yy^\top, yy^\top \rangle}\sqrt{\sum_i \lambda_i^2}} \end{aligned} \tag{10}$$

With α denoting the Lagrange multiplier, the objective function for the optimization can be formulated as follows:

$$\max \sum_i \lambda_i \langle v_i, y \rangle^2 - \alpha \left(\sum_i \lambda_i^2 - 1 \right) \tag{11}$$

Setting the derivative of the objective function to zero yields $\alpha_i \propto \langle v_i, y \rangle^2$. Replacing value of λ in Eq. 10, we have

$$A(y) = \frac{\sqrt{\sum_i \langle v_i, y \rangle^4}}{n} \tag{12}$$

Finally, the optimal kernel is obtained from an iterative approach that modifies a kernel matrix by combining $v_i v_i^\top$ with coefficient λ. In each iteration, the coefficient λ is updated by measuring the alignment between v_i and y. It is worth noting that the kernel alignment

measure can also be formulated using SDP as [53]:

$$\begin{array}{ll} \underset{A,K}{\text{maximize}} & \langle K, yy^\top \rangle \\ \text{subject to} & \text{tr}(A) \leq 1 \\ & \begin{pmatrix} A & K^\top \\ K & I \end{pmatrix} \succeq 0 \end{array} \tag{13}$$

While the kernel alignment measure has been used by numerous researchers [19,34,43,44,90,101], this technique remains to be ineffective in cases where there is no or limited availability of labeled examples in the dataset. This is because the formulation of the ideal kernel often requires a large set of labeled examples. Furthermore, it has been shown in [62] that it is possible for a kernel function to have a low alignment measure for a particular dataset and still run well for that dataset. Therefore, a new measure of kernel alignment is devised in [62] based on the distribution of data in the feature space. This distribution is assessed by Fisher discriminant analysis.

Divergence measures

The information-theoretic notion of divergence can be employed to measure the similarity between two matrices. In a method proposed by [20], the notation of LogDet divergence is used to measure the similarity of two matrices A and A_0:

$$D_{\ell d}(A, A_0) = \text{tr}\left(AA_0^{-1}\right) - \log\det\left(AA_0^{-1}\right) - n \tag{14}$$

This problem is optimized using the method proposed by [50]. The kernel then may be modeled as $K = \mathcal{X}^\top A\mathcal{X}$. We can alternatively replace x in all computations with $\phi(x)$ to produce a simple kernel-based solution. In this case, the obtained kernel is in the form of

$$\kappa(x, y) = k(x, y) + \sum_{i=1}^{n}\sum_{j=1}^{n} \sigma_{ij} k(x, x_i)k(y, x_j) \tag{15}$$

where the parameter σ should be updated in each iteration to obtain the optimal kernel κ from base kernel k.

Bregman divergence is another measure that may be used in a similar manner. The Bregman divergence for a strictly convex function ψ over a convex set is defined as

$$D_\psi(x, y) = \psi(x) - \psi(y) - (x - y)^\top \nabla\psi(y) \tag{16}$$

This definition is extended to the Bregman matrix divergence as

$$D_\psi(X, Y) = \psi(X) - \psi(Y) - \text{tr}\left((\nabla\psi(Y))^\top (X - Y)\right) \tag{17}$$

The Bregman divergence is used in [50,51] to find a low-rank kernel matrix. A low-rank matrix is sought as it would allow smaller eigenvalues to be omitted and therefore, facilitates dimensionality reduction. The importance of this technique is that if the matrix rank is not constrained, it is possible to derive a matrix with respect to an initial kernel such that they exhibit high similarity. Moreover, the optimization technique used to obtain the kernel allows users to incorporate their prior knowledge as additional constraints into the kernel matrix. The results obtained from this approach are used in a similar method proposed in [100] with more emphasis on preserving the structural representation of the dataset. The structural representation refers to the geometrical aspects of the data captured in the kernel.

3.6.4 Bounds on error rates of the learner

A class of techniques that have been extensively studied to learn the kernels evaluates the bounds of the learner to decide the optimal kernel. These techniques are in line with the general paradigm of learning where the ultimate goal is to have a minimal number of errors in prediction. Consequently, there are methods that seek to either minimize the error bounds directly by assessing the performance of the learner or embed them into the learning algorithm itself to jointly find the optimal values. Those that are embedded into a learning algorithm usually lead to introduction of a new formulation of the kernel-based learning algorithms.

Direct evaluation of error rates

Cross-validation has traditionally been the solution to determine parameters in model selection for most of the well-known machine learning algorithms. Not surprisingly, it found application in kernel methods and specifically SVM [22]. There are variations in cross-validation, e.g. k-fold cross-validation and leave-one-out. In k-fold cross-validation, dataset is split into k subsets (folds) and the SVM decision rule is obtained from the $k-1$ folds leaving one fold out for test. This procedure is repeated k times until all subsets are tested. In the leave-one-out method, the number of k in k-fold is set to be the number of examples. Hence, leave-one-out method requires training and testing equal to the number of training examples.

It is obvious that cross-validation only provides a way to select the hyperparameters in a range defined by the user. Additionally, it is not efficient especially when the number of examples is large, thus more efficient methods are needed to be developed. In a method proposed by [3], the concept of empirical error criterion [9] is used to determine the performance of the learner. This method is solely developed for SVM and uses its decision function. Let $t_i = (y_i + 1)/2$, then empirical error is defined as $E_i = |t_i - \hat{p}_i|$ where $\hat{p}_i$ is the posterior probability corresponding to data point x_i. This posterior probability is the sigmoid function used in Platt's probabilistic SVM as [64]:

$$\hat{p}_i = \frac{1}{1 + \exp(A.f_i + B)} \tag{18}$$

where A and B are parameters to be determined during training phase. Assuming the kernel function is parameterized by θ, the objective of the algorithm is formulated in terms of a gradient descent for minimizing the empirical error. Therefore, an iterative approach is employed that takes a subset of the dataset in initial stage and attempts to incrementally increase its size such that the remaining points in the set at the end of each iteration are as close as possible to the margin.

The variation in this method can be formulated in a well-known boosting scenario. Boosting seeks to formulate a strong hypothesis about a problem based on several weak learners. In each iteration, a weighing of the weak learners is decided based on the error rate of the resulting strong hypothesis. AdaBoost [16,71] as a popular type of boosting has been used in [32] to learn a kernel function. This method is semi-supervised with particular focus on the cases where the number of training examples is limited. In this approach, a combination of base kernel functions that minimizes the loss function is used. The weak learner, on the other hand, is based on the probabilistic method of Gaussian mixture model (GMM). The GMM model learned from data is used to create a new kernel which is actually the product of the probability of each pair of points belonging to the same Gaussian component. The advantages of this boosting approach are its relatively simple implementation and the ability

to run with limited training examples. However, the number of parameters that has to be determined by the user such as the number of iterations in the boosting and the number of Gaussian mixtures make it intractable. Another variation in this approach is proposed by [42]. The use of multiple base kernels to learn the kernel using boosting has also been considered in [49].

Implicit bounds on the error rates

Instead of assessing the performance of the learning algorithm directly, one can use the objective function of the learning algorithm as an optimality condition. Specifically in case of supervised learning, variations in SVM are desirable choices. The objective function of SVM exhibits some favorable aspects: sound geometrical interpretation, relatively simple formulation as a convex optimization which results in a globally optimal value and extensive body of literature in various related issues. As such, the optimal kernel is one that performs well in compliance with SVM objective.

One of the most remarkable and influencing methods in this class is proposed by [53] in which the process of learning the kernel is formulated in two settings of convex optimizations. The kernel is formulated as a combination of training and test dataset.

The parametric optimal kernel is formulated using a linear combination of base kernels as in Eq. 3. The parameters of the optimal kernel matrix are obtained from the objective of the dual of the hard and soft margin classifiers while the trace of kernel matrix is limited to a constant value. The algorithm is designed in two settings with respect to the coefficients of the base kernels: If all the coefficients are positive the resulting optimization is quadratically constrained quadratic program (QCQP) and otherwise it can be solved using SDP. In these two formulations, the kernel selection is embedded into the kernel machine (SVM) itself which leads to a new formulation of the learning algorithm.

The Lanckriet's work is one of the groundbreaking methods in the literature that defines a parametric kernel function which can be optimized with respect to a specific dataset. Furthermore, the use of established criteria of maximum margin classifiers to obtain convex optimization is another significant aspect because firstly maximizing the margin is a well-established condition and secondly the objective of learning the kernel and learning algorithm comply. Additionally, unlike many other methods that seek to find an optimal kernel matrix using only labeled examples, this approach used labeled and unlabeled data in a semi-supervised setting. In spite of all these advantages, this approach is suitable for cases where abundant number of training examples is available. The value of the parameter that limits the trace of the kernel matrix is also left to the user to be decided which may not be straightforward and have a great impact on the obtained kernel. One example of using such a method for data fusion is discussed in [52].

The use of combination of kernels considered by [53] has been revisited in [10] where authors have considered solving the problem in a technique similar to sequential minimization optimization (SMO) [63]. The developed method became to be known as *multiple kernel learning* (MKL) in which the use of multiple base kernels for decision-making is considered. In MKL, each kernel may be assigned to a specific data source and the result is a variation in SVM formulation. The remarkable aspect of this model is that the resulting optimization problem is sparse on two levels: data and kernel. While the former is shared with normal SVM, the latter indicates the sparsity in the kernel functions. In other words, there may be a kernel that is not suitable for the dataset or, on the other hand, the set of features assigned to a specific kernel do not play a significant role in decision-making.

MKL is a new formulation similar to SVM that needs to be further implemented in programming languages and tested accordingly. However, a variation in MKL is devised in [66] that is able to use existing implementations of SVM by decomposing the problem into two stages: In the first stage, the appropriate combination of base kernels is updated, and in the subsequent step, the parameters of SVM are estimated accordingly. These two stage alternating approaches have attracted other researchers as well Rakotomamonjy2008, ZenglinXu2010. In particular, in [91], the connection between MKL and group lasso [96] is used to formulate a simple algorithm where the combination coefficients of kernels are easily calculable. This simplicity leads to an efficient and applicable algorithm.

Another variation in MKL is proposed in [78] that improves the learning by introducing the concept of grouping in the kernels. This means, kernels corresponding to similar evaluation of the feature sets can be related. Additionally, the case of large or possibly infinite set of base kernels in MKL is considered in [28].

The use of multiple base kernels has extended in [17] to the case of nonlinear case in which the kernel matrix of the following form is designed

$$\mathcal{K} = \sum_{s_1+\ldots+s_q=q} \mu_1^{s_1} \ldots \mu_m^{s_m} K_1^{s_1} \ldots K_m^{s_m} \tag{19}$$

This formulation of the kernel matrix is in fact a polynomial of degree q that has m base kernels. Consequently, the objective of the algorithm is to find the vector $\mu = (\mu_1, \ldots, \mu_m)^\top$. The algorithm is specially designed for the min–max formulation of the regression in [70]. The solution to the problem is developed as two-stage optimization that consists of solving the regression and subsequently using its solution in a gradient decent to find the kernel. However, the empirical results do not show any consistent performance improvement. Additionally, the setting of the proposed approach is restricted to one particular learning algorithm which cannot be extended to other problems. One can further question whether the nonlinear parameterization of the ultimate kernel functions is necessary or efficient. In any case, the proposed approach set the stage for further development in the field and possible improvements in the empirical evaluations.

In another approach, [82] devised an algorithm to find the optimal hyperparameter for a given dataset by modeling the changes in the hyperparameter as a function that seeks to ultimately minimize the hinge loss in SVM.

Regularization for learning the kernel

Regularization minimizer, as a well-developed condition on the bounds of hypothesis function, provides a robust foundation for kernel construction as considered in [5,47,58]. It is showed that a convex problem can be formulated from the problem with an appropriate continuous loss function which leads to an interior-point method [24,25]. Considering the regularization problem with weight vector w as:

$$\underset{w}{\text{minimize}} \quad \frac{1}{n}\sum_{i=1}^{n} q\left(y_i, w^\top \phi(x_i) + \nu\right) + \lambda\langle w, w\rangle \tag{20}$$

Then, [47] showed that the optimal kernel matrix for the loss function q can be obtained from the following convex optimization

$$\text{minimize} \quad \sum_{i=1}^{n} q(y_i h_i + y_i \nu) + \lambda h^\top K^\dagger h \tag{21}$$

where $h = K\alpha \in \mathbb{R}^n$, $\nu \in \mathbb{R}$ and $K^\dagger$ is the pseudo-invers of the matrix K. This approach toward minimization of the regularization method is further extended by [6] where DC (difference of convex) programming [36] is used for optimization. This supervised method works based on the regularization functional:

$$Q(f) = \sum_j q(y_i, f(x_i)) + \lambda \|f\|_K^2 \tag{22}$$

In this method, a class of continuously parameterized set of base kernels is considered as

$$K = \left\{ \int_\Omega G(\omega) \mathrm{d}p(\omega) : p \in \mathcal{P}(\Omega) \right\} \tag{23}$$

where ω is the set of parameters and $\mathcal{P}(\Omega)$ is a set of probability measures on Ω and $G(.)(x, x')$ is the continuous base kernel with parameter ω for $x, x' \in \mathcal{X}$. As an example Gaussian kernel can be considered, i.e., $G(\omega)(x, x') = \exp\left(-\omega \|x - x'\|^2\right)$. The remarkable point of consideration about this formulation is that it potentially enables usage of infinite number of base kernels. The objective of the algorithm, on the other hand, is formulated as the minimization of the regularization error similar to [58]:

$$E(K) = \min\{Q(f) : f \in \mathcal{H}_K\} \tag{24}$$

The regularization framework has also been studied in an intersection with Bregman divergence in [54] where the ultimate solution is obtained from a Newton method. Further justification of these methods can be found in [59].

Transferred setting

In a different trend of learning, one can use multiple sources of data to strengthen the trained model as in *transfer learning* and *multi-task learning* [14]. These sources of data can be either labeled or unlabeled and exhibit some level of conceptual relation with the test dataset that the algorithm is designed to work on. By using multiple datasets, one may hope to learn a model that eventually generalizes better on the test dataset. As such, [69] considered a case where the problem is to use multiple datasets in a regularization framework to find the optimal kernel. These multiple datasets may refer to the same problem obtained from different time periods or similar problems. This algorithm works by alternating between learning the appropriate model and optimal kernel. Since this algorithm is specifically designed to work with SVM, the model is also learned by training SVM. The minimization of the error bounds of the model will lead to finding the linear combination of base kernels.

In case the additional datasets are unlabeled, the task of learning the kernel is tackled in an unsupervised transfer learning approach as in [65]. The algorithm is designed in a two-stage process: In the first phase, the important features from which final decision may be made are selected using sparse coding as an unsupervised feature selection technique. Once these impactful features are known, the second phase reuses this information about features to find their transformation in the labeled set. Finally, the transformed features are used in a learning algorithm such as SVM to find an appropriate model.

This is an interesting approach as it shows the ability to learn a kernel from labeled and unlabeled data while the unlabeled data are collected from a similar problem. However, a drawback of such method is that the features found in the unlabeled dataset may not necessarily reflect the most important features in the labeled training set. In this case, the

parameters learned in the first phase may even misguide the learning. Similar approaches of using transfer learning to construct a kernel are considered in [1,7,23,39].

3.6.5 Intrinsic structure of the dataset

There is recently an increasing interest in the use of intrinsic structural aspect of the dataset for finding the optimal kernels. This is mostly because the kernel function of the choice should be able to preserve and reflect the relational information of a pair of points. Moreover, by using the characteristics of the dataset, the need for prior knowledge of the algorithm designer or the label of each example may be exempted. In this section, several of these methods will be further discussed.

Input space conditions

Optimality of the selected kernel can be based on some aspect of the dataset inferred from investigating the representation of the dataset in its input space. The inferred information will be used in the construction of the feature space. That is, the kernel or the mapping function is enforced to have strongest compliance with these inferred constraints. As an example, *heat* or *diffusion kernel* [48] is a well-known data-dependent kernel function defined over the statistical manifold of data with a strong physical interpretation. The name along with the justification of the formulation derived from the physical interpretation of the diffusion of heat through continuous media which is obtained from equation: $\frac{\partial}{\partial t}\psi = \mu\Delta\psi$. The kernel is also obtained from a similar equation:

$$\frac{\partial}{\partial \beta} K_\beta = H K_\beta \tag{25}$$

In Eq. 25, the matrix H indicates the geometrical representation of the dataset. For example, H may be obtained from a graph Laplacian that $H_{ij} = 1$ if vertices i and j are connected, $H_{ii} = -d_i$ where d_i counts the number of edges emanating from point i in a graphical representation of data and $H_{ij} = 0$ otherwise. The remarkable point about the heat kernel is that the intrinsic structure of the dataset in terms of a graph captured in H is used to create a new kernel function.

The solution to Eq. 25 in continuous case may be obtained from

$$k_x(x') = \frac{1}{\sqrt{4\pi\beta}} \exp\left(\frac{-|x - x'|}{4\beta}\right) \tag{26}$$

which reduces to Gaussian kernel with selection of $\beta = \sigma^2/2$. One drawback of the heat kernel is that the parameter β itself should be decided beforehand. Additionally, in process of generating the matrix H, the proximity between points should be examined and decided by the user which may not be easy.

Another data-dependent kernel function is *graph Laplacian kernel* [31,101] which has been used extensively to find the structural representation of a dataset. In addition, the kernel function construction is centered on the notion of graph Laplacian which is built from the adjacency matrix W where each entry w_{ij} denotes the distance between points i and j. The graph Laplacian L is defined as $L = D - W$, where D is a diagonal matrix such that $D_{ii} = \sum_{i=1}^{n} w_{ij}$ and the normalized graph Laplacian is defined as $\tilde{L} = I - D^{-1/2} L D^{-1/2}$. It can be shown that the pseudo-inverse of the graph Laplacian is always positive semidefinite

and can be used as the kernel matrix, i.e.

$$K = L^{\dagger} = \sum_{i=r+1}^{n} \lambda^{-1} u_i u_i^{\top} \tag{27}$$

where λ, u denote the eigenvalues sorted in ascending order and eigenvectors, respectively. Additionally, r indicate the index of the first positive eigenvalue. Finally, it is worth noting that the diffusion kernel can be related to the graph Laplacian by

$$K_{\text{heat}} = \exp(-\beta L) \tag{28}$$

Graph Laplacian has been one of the basic methods to extract the information about the structure of the dataset. It has a simple formula that is easily implementable in various programming languages. In the rest of this section, several examples of the algorithms that used graph Laplacian for kernel construction will be presented.

The prior knowledge of the user about the problem may be jointly used with the notion of graph Laplacian to formulate a new kernel. Such a problem is tackled by [35] (similar to [50] discussed earlier) where the objective is to use graph Laplacian as the kernel with structural information in addition to a set of predefined constraints given by the user as relative similarity between pairs of examples to formulate the optimal kernel. This algorithm aims at minimizing the inconsistency between optimal kernel and the graph Laplacian kernel matrix while satisfying the predefined constraints. The ultimate optimization is SDP; however, authors argued that it can be further simplified to a conjugate gradient method.

This approach [35] has been further extended using active learning in [33]. Active learning is a class of learning algorithms that aims at improving the model by taking the most informative examples into account. It has been very popular in semi-supervised approaches where labeled data are more informative. In [33], authors considered using active learning to find the most informative pair of examples to the optimal kernel. The most important justification of using active learning is the appropriate utilization of the limited labeled examples. The proposed active learning is a nonparametric approach and the main improvement made compared to its ancestor is that the assumption of the existence of the relative similarity between points is voided. Additionally, the optimization obtained in this method is slightly complicated. However, an improvement of [35] proposed in [102] attempts to find a simpler optimization. The use of graph Laplacian inherited from [35] brings the ability to capture the structure of the dataset. The loss function and the graph Laplacian are jointly used as the criteria to evaluate the performance of the kernel function obtained. One major drawback is that, this is a nonparametric method in which there is always the possibility of over-fitting. It makes the kernel matrix over-trained for a special dataset while it does not truly represent the correct level of similarity between points.

In another interesting semi-supervised approach proposed in [76], a small number of labeled examples are used as the initial points for kernel construction. The feature space construction is started with the labeled examples and led by the unlabeled ones. In this method, the cluster assumption plays a crucial role because it is assumed that the labeled examples are surrounded by a "cloud" of unlabeled examples. The kernel function is constructed based on the notion of reproducing kernel Hilbert space (RKHS) and the assumption that the evaluation functional is bounded at each point. RKHS provides the necessary framework to build a function in the feature space based on a given dataset and kernel function. In order to obtain the new kernel, the linear combination of predefined kernels in $\mathcal{H}$ is considered as

$$\kappa^*(x, .) = k(x, .) + \sum_j \beta_j(x) k(x_j, .) \tag{29}$$

In this equation, β is a data-dependent function of x. By defining the inner product over $k(x_i, .)$ at x the ultimate kernel is obtained as:

$$\kappa^*(x, x') = k(x, x') - k_x^\top (I + MK)^{-1} M K_{x'} \tag{30}$$

where k_x is the column vector of the kernel matrix. The value of the matrix $M = L^p$, where p is an integer value and L is the Laplacian graph defining the geometry of the points. This work, in addition to defining a family of kernel functions that favors the geometrical structure of the dataset, sets the stage for future development of techniques that uses RKHS to utilize the geometry of a dataset.

The base kernel obtained in Eq. 30 represents a kernel defined on the structure of the dataset. Hence, it can be used as a basis for other methods to be developed. Specifically, in combination with the divergence measures introduced earlier, [100] proposed an approach that seeks to find a kernel matrix with highest similarity to a structural aspect of the dataset represented in another kernel matrix (such as graph Laplacian or the kernel matrix in Eq.30 as a more generic case). To serve this purpose, the LogDet matrix divergence, as defined in Eq.14, is used and its main advantage is its invariance to rescaling of the feature space. Considering K as the initial kernel, the ultimate kernel $\mathcal{K}$ may be modeled as:

$$\begin{aligned} \mathcal{K} &= \left\{K - K(I + TK)^{-1} TK\right\}, \\ T &= \sum_{i=1}^{r}, \lambda_i v_i v_i^\top \lambda_1 \geq \cdots \geq \lambda_r \geq 0 \end{aligned} \tag{31}$$

The significance of these kernels is their ability to be easily extended to unseen examples. On the other hand, the convexity of this set is certain as long as $\{v_1, \ldots, v_r\}$ are orthogonal. To obtain a solution, it should be firstly noted that in general $D_{\ell d}(K, \mathcal{K})$ is not jointly convex in K and $\mathcal{K}$. Therefore, the goal of this algorithm is formulated as finding T and consequently λ. Hence, a cyclic projection algorithm is used where at each step the solution is projected on the constraints. The optimization to solve the divergence of $D_{\ell d}(K, \mathcal{K})$ is similar to the one used in [20].

Feature space conditions

In the previous subsection, methods that define the kernel based on the information extracted from the input space are considered. As it is shown, most of these methods consider variations in the graph Laplacian to capture the structural representation of the dataset. However, in this subsection, focus is on the methods that typically define optimality as the condition applied directly to the mapped points in the feature space. Consequently, the optimal kernel represents the desired characteristics. As an example, [95] proposed to devise a kernel matrix such that the squared Euclidean distance between pairs in the same class in the feature space is reduced.

One important approach in this family that investigates the geometric representation of the mapped points in the feature space is considered in [4]. This method is concentrated on the supervised learning specifically in case of SVM. The mapped points are investigated using Riemannian geometry. The Riemannian geometry provides the foundation to analyze data in a highly nonlinear structure smoothly. The Riemannian metric used in the feature space is obtained from

$$g_{ij} = \frac{\partial}{\partial x_i} \frac{\partial}{\partial x'_j} k(x, x') \mid_{x'=x} \tag{32}$$

Furthermore, the volume of the Riemannian space is defined as

$$\mathrm{d}V = \sqrt{\det |g_{ij}(x)|}\ \mathrm{d}x_1 \ldots \mathrm{d}x_n \tag{33}$$

The factor $\sqrt{\det |g_{ij}(x)|}$ represents how a local area is magnified in the feature space under mapping ϕ. Subsequently, the Riemannian distance is defined as

$$\mathrm{d}s^2 = \sum_{i,j=1}^{n} g_{ij}\, \mathrm{d}x_i\, \mathrm{d}x_j \tag{34}$$

By increasing the value of metric g_{ij}, the distance between points i and j around the decision boundary is increased. Thus, the nonlinear mapping ϕ is modified such that $\sqrt{\det |g_{ij}(x)|}$ is enlarged around the boundaries. These boundaries may then be selected to be the support vectors in SVM. A conformal transformation $\tilde{g}_{ij}(x) = \Omega(x) g_{ij}(x)$ is proposed to solve the problem where the conformal map $\Omega(x)$ has a large value around the boundaries. The conformal transformation will not change the angle between points and therefore the spatial characteristics remain unchanged. This conformal transformation is defined as:

$$\kappa(x, x') = q(x) q(x') k(x, x') \tag{35}$$

with factor $q(x)$. The factor of this transformation ensures the modification on the mapping function. It may be defined as weighted sum of Gaussian kernels:

$$q(x) = \sum_{\ell=1}^{|\mathcal{S}|} \alpha_\ell \exp\left(-\frac{||\ \mathbf{x} - s_\ell\ ||^2}{2\sigma}\right) \tag{36}$$

where s_ℓ denotes each support vector in the set of support vectors $\mathcal{S}$ and parameter α left to be determined during learning process. Finally, the kernel selection algorithm is proposed in three steps: First, SVM is trained using the kernel function k and the support vectors are obtained; Second, the conformal transformation of kernels according to Eqs. (35) and (36) is used to obtain κ; third, train SVM using the modified kernel κ. Although this simple iterative approach may not be an efficient way of obtaining the optimal kernel, but the notion of kernel conformal transformation as a mathematically justified method to modify a kernel function has been used jointly with other conditions to further ensure the optimality of the solution. Examples of such approaches can be found in [27,87–89].

Another way of providing constraints on the feature space while ensuring the global optimality of the solution is proposed in [46]. In this method, discriminant analysis is used to model a convex problem for a given dataset. The objective function of this method is formulated as a solution to the following problem:

$$F_1(w, K) = \frac{\left(w^\top \left(\bar{\phi}^+ - \bar{\phi}^-\right)\right)^2}{w^\top \left(n_+/n\Sigma^+ + n_-/n\Sigma^- + \lambda I\right) w} \tag{37}$$

In this equation, $\bar{\phi}$ is the mean, Σ is the covariance matrix for the positive or negative classes obtained from the mapped points in the feature space, and $\lambda > 0$ is a regularization parameter. The optimal weight vector is determined as

$$w^* = \arg\max_{w} \quad F_1(w, K) \tag{38}$$

For a fixed kernel matrix K and regularization parameter λ, Eq. 38 is equal to

$$w^* = \left(n_+/n\Sigma^+ + n_-/n\Sigma^- + \lambda I\right)^{-1} \left(\bar{\phi}^+ - \bar{\phi}^-\right) \tag{39}$$

Fixing $w = w^*$, the optimal value for K is obtain as

$$F_1^*(K) = \left(\bar{\phi}^+ - \bar{\phi}^-\right)^\top \left(n_+/n\Sigma^+ + n_-/n\Sigma^- + \lambda I\right)^{-1} \left(\bar{\phi}^+ - \bar{\phi}^-\right) \tag{40}$$

In order to maximize $F_1^*(K)$, a semidefinite program is proposed that can be solved using the available optimization toolboxes. This algorithm is further extended and simplified by [41] with the advantage of being formulated as a QCQP problem. The advantage of the QCQP is that it is faster than SDP and computationally affordable for moderate sized problems. Discriminant analysis provides an effective criterion to assess the feature space and consequently has been studied in various approaches to learn the kernel [15,55,46,89,94]. In [83], authors proposed an optimization approach that maximizes the linear discriminant analaysis's objective in the feature space which leads to finding the parameter of Gaussian kernel.

While discriminant analysis as well as conformal transformation provides explicit constraints on the feature space, a technique that amounts to an implicit constraint is proposed in [79]. This implicit constraint in addition to the user-defined linear conditions as the prior knowledge is incorporated into the optimization objective to find an optimal kernel. Intuitively, this method aims to distribute the mapped points in the feature space as evenly as possible. In other words, all the points are given equal chance to be mapped in the feature space while the linear constraints decide the relation between them. This implicit constraint is defined as the von Neumann's entropy for positive definite matrices, i.e.

$$E(K) = -\operatorname{tr}(K \log(K)), \quad K \succ 0, \operatorname{tr}(K) = 1 \tag{41}$$

Since $\operatorname{tr}(K \log(K))$ is convex, it is shown that maximization of the von Neumann entropy with respect to a set of linear constraints can be used to find the optimal kernel. This amounts to taking the global aspect of the dataset using the entropy and use it jointly with local constraints defined by the user. Ultimately, it is shown that the kernel obtained from this approach is equivalent to the diffusion kernel.

Another unsupervised method has been proposed in [2] where the random walk on the graph is used to learn a linear combination of the kernels. Random walk is performed to assess the influence of each point in the input or feature space. Then, the feature space is probabilistically constructed such that the points with higher influence in the input space remain in the dense areas in the feature space. This algorithm is presented in two settings, and the solution to both of them can be obtained from two convex optimizations, namely linear and semidefinite programming.

Intrinsic structure for dimensionality reduction

Kernel methods and the mapping to a higher dimensional space as in kernel-PCA have been used for dimensionality reduction. It is therefore, rational to try to select an appropriate kernel optimized for the task of dimensionality reduction. In [84] authors proposed a method that utilizes the nonlinearly mapped points to reveal the lower dimensional representation of the dataset, in accordance with kernel-PCA. The algorithm's goal is to construct a kernel matrix that maximizes the variance of the dataset in the feature space while preserving the distance between neighboring data points. This goal is formulated as a semidefinite program. The pairwise distance between data points can be written as:

$$\frac{1}{2n} \sum_{i=1}^{n} \sum_{j=1}^{n} \|\phi(x_i) - \phi(x_j)\|^2 = \operatorname{tr}(K) \tag{42}$$

On the other hand, the distance between data points should be changed symmetrically. This is in line with most of the nonlinear dimensionality reduction techniques that seek to preserve some aspects of the dataset, such as distance between points in this case. This amounts to the local constraint on the distances between two neighboring points, i.e.

$$\|\phi(x_i) - \phi(x_j)\|^2 = \|x_i - x_j\|^2 \tag{43}$$

This unsupervised method is useful for cases where the kernel function is intended to be used for dimensionality reduction, e.g. with kernel-PCA. A similar method has also been developed in [74] where the topological structure of the dataset is considered. The goal of this algorithm is to find an appropriate embedding of a constructed graph from the dataset in a Euclidean space. This approach is based on the fact that the distance between two points is a linear function of the kernel: $d(x, x') = k(x, x) + k(x', x') - 2k(x, x')$. Additionally, the points that are not connected in the graph (or not close enough in a K-Nearest Neighbor selection) and correspond to the zero entry in connectivity matrix A should be placed in distant i.e. $D_{ij} > (1 - A_{ij}) \max_s (A_{is} D_{is})$. Consequently, the objective of the algorithm is formulated as a SDP optimization that seeks to maximize $\mathrm{tr}(KA)$ while satisfies the constraint $\mathrm{tr}(A) < 1$.

Finally, it should be noted that two methods mentioned earlier in Sect. 3.6.3 [50,51] in which divergence is used to find a low-rank kernel matrix may be utilized to obtain kernel matrices for dimensionality reduction. However, because those techniques are more strongly related to adaptation and divergence, they are not included here. Furthermore, dimensionality reduction techniques discussed in this subsection are more task-specific and cannot be easily extended to other problems like the classification cases.

4 Conclusions

In this paper, the state of the art techniques related to the area of learning the kernels have been discussed (in Table 1, a list of most important methods in this area and their significance is listed). We find that the current methods to learning the kernel vary, and improvements in various aspects of these approaches can be done. Most importantly, the current approaches classified based on their optimality conditions were detailed (Table 2).

As it is presented, there are various aspects that the merits of the algorithms in the area can be discussed. In optimization, it is important if any guarantee on the optimal solution can be given. Also, it is more desirable to define the model of the learning beforehand so that the resultant optimal kernel does not need to be refined during testing phase. Additionally, it is useful to have an algorithm that learns the kernel independent of the kernel machine of choice so that the algorithm can be applied to various problems.

Furthermore, the following is the list of possible directions for further research and development in this area:

– The current methods are mostly concentrated on the cases where the number of training examples is abundant. However, there is not always enough labeled examples available to evaluate the model's performance or even incorporate into optimality conditions. In addition to what is presented, further investigation on new unsupervised approaches is required so that the resultant optimal kernel can be used in unsupervised kernel-based algorithms, such as clustering, too. An unsupervised algorithm, either independent of the learning problem or embedded into it, may be very useful in providing more efficient clustering algorithms since the most important aspect of clustering is the measure of similarity.

Table 1 List of most important methods to learn the kernel and their highlights

Method	Comments
Fisher kernel [38]	Data-dependent kernel A generative approach Statistical manifold of data is investigated Supervised
Conformal transformation of the kernels [4,87,88]	The structural representation of the dataset is modified The mapping function is modified through conformal map Iterative optimization and examination of kernel's performance The conformal map is required to be cautiously selected Multiple training and testing of SVM is required Supervised
Boosting for learning the kernels [32,42,49]	Use of celebrated boosting algorithm Easy to implement Iterative approach without any need for sophisticated optimization methods or specific toolboxes Supervised, semi-supervised
Divergence and alignment measures [18,20]	Challenging formulation of an appropriate divergence/alignment measure Selection of appropriate ideal kernel Supervised, Semi-supervised
Maximizing the entropy [79]	Desirable interpretation in the feature space Incorporation of prior knowledge in terms of linear constraints Independent of the kernel machine An unsupervised approach Unsupervised
Multi-kernel learning [10,53]	Maximizes the margin Embedded into the kernel machine Entails a new formulation of a kernel method similar to SVM Computationally efficient Supervised
Dimensionality reduction [74,84]	Captures the structure of the data for nonlinear dimensionality reduction Seeks to preserve some aspects of the data in the feature space Unsupervised
Graph Laplacian [31]	Data-dependent kernel Frequently used to capture the dataset's characteristics Mostly constructed using Gaussian kernel which is a parametric kernel Unsupervised
Using RKHS to find a function space [76]	Solid mathematical foundation and strong connection to the learning problem Formulation of a data-dependent kernel with interpretation of the structure of the dataset A semi-supervised approach Semi-supervised
Minimizing the regularized objective function [5,34,58]	Solid foundation with strong relation to the well-established learning criteria Lays the foundation for continuous and smooth formulation of base kernel Supervised
Maximizing the discriminant analysis [15,46,89]	Strong interpretation Good measure to be used in feature space Independent of kernel machine Supervised

Table 1 Continued

Method	Comments
Minimizing the empirical error [3,9]	Iterative optimization of the kernel with multiple training of SVM Requires Platt's sigmoid function May be used in incremental manner Supervised
Transferred settings [1,65,69]	Determine the important features from the auxiliary dataset Leads to a robust learned kernel Suitable for cases where the number of labeled training examples is insufficient
Non-parametric kernel learning [35,102]	Produces a kernel matrix rather than a function Ability to incorporation of prior knowledge Semi-supervised
Alternating MKL [66,91]	Easy extension of the current SVM implementations Iterative improvement with multiple training of SVM Strong connection to the learning criteria Applicable to a wider range of kernel machines Supervised
Non-linear combination of kernels [17]	Innovative idea in combination of base kernels Gradient descent optimization in regression formulation Further investigation may exhibit better performance, although the formulation of a convex problem may be challenging Supervised
Random walk for learning the kernel [2]	Random walk to derive the structural information of the dataset Independent of the kernel machine An unsupervised approach Unsupervised

Table 2 List of the available source codes for learning the kernels

Method	Link
Multiple kernel learning (support kernel machine) [10]	http://www.stat.berkeley.edu/~gobo/SKMsmo.tar
Probability product kernels [40]	http://www.cs.columbia.edu/~jebara/code/elkernel.m
Multi-task feature and kernel selection for SVMs [39]	http://www.cs.columbia.edu/~jebara/code/multisparse
Convex combinations of basic kernels [5]	http://ttic.uchicago.edu/~argyriou/code/dc/dc.tar
Using RKHS to find a function space [76]	http://people.cs.uchicago.edu/~vikass/manifoldregularization.html
Semidefinite embedding (maximum variance unfolding) [85]	http://www.cse.wustl.edu/~kilian/code/files/lmvu.zip
Simple multiple kernel learning [67]	http://asi.insa-rouen.fr/enseignants/~arakotom/code/mklindex.html

– The current trend is to use a kernel or its linear combination to model an initial hypothesis for further optimization. However, other possible models may be opted for this purpose. One possible direction is to use the nonlinear combination of the base kernels. The nonlinear case may produce better performance and flexibility. There has not been enough research done in optimizing the kernels with respect to the RKHS which provide a solid foundation and may lead to promising algorithms. Additionally, use of infinite number of base kernels has been investigated which has not yet produced outstanding performance. It is possible that further studies lead to better results.
– Geometry of the mapped points in the feature space provides sound justification on the optimality of the kernel function. However, in these methods, the optimization of the objective function or forming the dual is the most important aspect that has to be closely scrutinized.

As a final remark, it should be noted that most of the current approaches are showing good results in small datasets and fail to scale to large ones. As such, the immediate future work will be developing approaches with faster optimization algorithms like stochastic gradient descent that perform better with larger datasets. The online approaches to learn the kernel with the capability to update with new examples can also be another possible path to improve this area.

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. Abbasnejad ME, Ramachandram D, Mandava R (2009) Optimizing kernel functions using transfer learning from unlabeled data. In: Machine Vision, 2009. ICMV '09. Second international conference on, pp 111–117. doi:10.1109/ICMV.2009.10
2. Abbasnejad ME, Ramachandram D, Mandava R (2010) An unsupervised approach to learn the kernel functions: from global influence to local similarity. Neural Comput Appl. doi:10.1007/s00521-010-0411-7
3. Adankon MM, Cheriet M (2007) Optimizing resources in model selection for support vector machine. Pattern Recognit 40(3):953–963. doi:10.1016/j.patcog.2006.06.012
4. Amari S, Wu S (1999) Improving support vector machine classifiers by modifying kernal functions. Neural Netw 12:783–789. doi:10.1016/S0893-6080(99)00032-5
5. Argyriou A, Micchelli CA, Pontil M (2005) Learning convex combinations of continuously parameterized basic kernels. In: Auer P, Meir R (eds) Learning theory, Lecture Notes in Computer Science, vol 3559. Springer, Heidelberg, pp 338–352. doi:10.1007/11503415_23
6. Argyriou A, Hauser R, Micchelli CA, Pontil M (2006) A dc-programming algorithm for kernel selection. In: ICML '06: Proceedings of the 23rd international conference on machine learning, ACM, New York, NY, USA, pp 41–48. doi:10.1145/1143844.1143850
7. Argyriou A, Evgeniou T, Pontil M, Argyriou A, Evgeniou T, Pontil M (2007) Multi-task feature learning. In: Schölkopf B, Platt J, Hoffman T (eds) Advances in neural information processing systems, vol 19, MIT Press, Cambridge, MA, pp 41–48
8. Aronszajn N (1950) Theory of reproducing kernels. Trans Ame Math Soc 68:337–404
9. Ayat N, Cheriet M, Suen C (2005) Automatic model selection for the optimization of svm kernels. Pattern Recognit 38(10):1733–1745. doi:10.1016/j.patcog.2005.03.011
10. 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. doi:10.1145/1015330.1015424
11. Ben-Hur A, Horn D, Siegelmann HT, Vapnik V (2002) Support vector clustering. J Mach Learn Res 2:125–137
12. Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press, New York
13. Burges CJC (1999) Geometry and invariance in kernel based methods. MIT Press, Cambridge 89–116
14. Caruana R (1997) Multitask learning. Mach Learn 28(1):41–75
15. Chen B, Liu H, Bao Z (2008) A kernel optimization method based on the localized kernel fisher criterion. Pattern Recognit 41(3):1098–1109. doi:10.1016/j.patcog.2007.08.009
16. Collins M, Schapire RE, Singer Y (2002) Logistic regression, adaboost and bregman distances. Mach Learn 48(1–3):253–285
17. Cortes C, Mohri M, Rostamizadeh A (2009) Learning non-linear combinations of kernels. In: Bengio Y, Schuurmans D, Lafferty J, Williams CKI, Culotta A (eds) NIPS: advances in neural information processing systems, vol 22, pp 396–404
18. Cristianini N, Shawe-taylor J, Elissee A, Kandola J (2002) On kernel-target alignment. In: Advances in neural information processing systems, vol 14, MIT Press, pp 367–373
19. Cristianini N, Kandola J, Elisseeff A, Shawe-Taylor J (2003) On optimizing kernel alignment. Tech. rep., UC Davis Department of Statistics
20. Davis JV, Kulis B, Jain P, Sra S, Dhillon IS (2007) Information-theoretic metric learning. In: ICML '07: Proceedings of the 24th international conference on machine learning, ACM, New York, NY, USA, pp 209–216. doi:10.1145/1273496.1273523

21. Domeniconi C, Peng J, Yan B (2010) Composite kernels for semi-supervised clustering. Knowl Inf Syst, pp 1–18
22. 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
23. Evgeniou T, Micchelli CA, Pontil M (2005) Learning multiple tasks with kernel methods. JMLR Org 6:615–637
24. Florian A, Potra SJW (2000) Interior-point methods. J Comput Appl Math 124(1–2): 281–302. doi:10.1016/S0377-0427(00)00433-7
25. Freund RM, Mizuno S (1996) Interior point methods: current status and future directions. Working papers 3924-96., Massachusetts Institute of Technology (MIT), Sloan School of Management. http://ideas.repec.org/p/mit/sloanp/2634.html
26. Fung G, Rosales R, Rao RB (2008) Feature selection and kernel design via linear programming. In: IJCAI'07: Proceedings of the 20th international joint conference on artifical intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 786–791
27. Gang Wu EYC (2003) Adaptive feature-space conformal transformation for imbalanced-data learning. In: Proceedings of IEEE international conference on machine learning, pp 816–823
28. Gehler P, Nowozin S (2008) Infinite kernel learning. Tech. rep., Max Planck Institute For Biological Cybernetics, Tuebingen, Germany
29. Grant M, Boyd S (2008) Graph implementations for nonsmooth convex programs. Recent Adv Learn Control 371:95–110. doi:10.1007/978-1-84800-155-8_7
30. Grant M, Boyd S (2009) Cvx: Matlab software for disciplined convex programming. web page and software. http://stanford.edu/~boyd/cvx
31. 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. doi:10.1145/1102351.1102390
32. Hertz T, Hillel AB, Weinshall D (2006) Learning a kernel function for classification with small training samples. In: ICML '06: Proceedings of the 23rd international conference on machine learning, ACM, New York, NY, USA, pp 401–408. doi:10.1145/1143844.1143895
33. Hoi SCH, Jin R (2008) Active kernel learning. In: ICML '08: Proceedings of the 25th international conference on machine learning, ACM, New York, NY, USA, pp 400–407. doi:10.1145/1390156.1390207
34. Hoi SCH, Lyu MR, Chang EY (2006) Learning the unified kernel machines for classification. In: KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, New York, NY, USA, pp 187–196. doi:10.1145/1150402.1150426
35. Hoi SCH, Jin R, Lyu MR (2007) Learning nonparametric kernel matrices from pairwise constraints. In: ICML '07: Proceedings of the 24th international conference on machine learning, ACM, New York, NY, USA, pp 361–368. doi:10.1145/1273496.1273542
36. Horst R, Thoai NV (1999) Dc programming: overview. J Optim Theory Appl 103(1): 1–43. doi:10.1023/A:1021765131316
37. Huang K, Ying Y, Campbell C (2010) Generalized sparse metric learning with relative comparisons. Knowl Inf Syst
38. Jaakkola TS, Haussler D (1999) Exploiting generative models in discriminative classifiers. In: Proceedings of the 1998 conference on advances in neural information processing systems II, MIT Press, Cambridge, MA, USA, pp 487–493
39. Jebara T (2004) Multi-task feature and kernel selection for svms. In: ICML '04: Proceedings of the twenty-first international conference on machine learning, ACM, New York, NY, USA. doi:10.1145/1015330.1015426
40. Jebara T, Kondor R, Howard A (2004) Probability product kernels. J Mach Learn Res 5:819–844
41. Jieping Ye JC, Shuiwang Ji (2007) Multi-class discriminant kernel learning via convex programming. J Mach Learn Res 9:719–758
42. Joseph KC, Keshet J, Singer Y (2002) Kernel design using boosting. In: Advances in Neural Information Processing Systems, vol 15, MIT Press, pp 537–544
43. Kandola J, Shawe-Taylor J (2002a) On the extensions of kernel alignment. Tech. Rep. NC-TR-2002-120, University of Southampton (United Kingdom). http://eprints.ecs.soton.ac.uk/9745/
44. Kandola J , Shawe-Taylor J (2002b) Optimizing kernel alignment over combinations of kernel. Tech. Rep. NC-TR-2002-121. http://eprints.ecs.soton.ac.uk/9746/
45. Kapoor A, Alan Qi Y, Ahn H, Picard RW (2005) Hyperparameter and kernel learning for graph based semi-supervised classification
46. Kim SJ, Magnani A, Boyd S (2006) Optimal kernel selection in kernel fisher discriminant analysis. In: ICML '06: Proceedings of the 23rd international conference on machine learning, ACM, New York, NY, USA, pp 465–472. doi:10.1145/1143844.1143903

47. Kim SJ, Zymnis A, Magnani A, Koh K, Boyd S (2008) Learning the kernel via convex optimization. In: Acoustics, speech and signal processing, 2008. ICASSP 2008. IEEE international conference on, pp 1997–2000. doi:10.1109/ICASSP.2008.4518030
48. Kondor RI, Lafferty J (2002) Diffusion kernels on graphs and other discrete structures. In: Proceedings of the ICML '02: Proceedings of the 23rd international conference on machine learning, pp 315–322
49. Kristin P, Bennett MJE Michinari M (2002) Mark: a boosting algorithm for heterogeneous kernel models. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining, ACM, New York, NY, USA, pp 24–31. doi:10.1145/775047.775051
50. 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. doi:10.1145/1143844.1143908
51. Kulis B, Sustik MA, Dhillon IS (2009) Low-rank kernel learning with bregman matrix divergences. J Mach Learn Res 10:341–376
52. Lanckriet G, Deng M, Cristianini N, Jordan MI, Noble WS (2004a) Kernel-based data fusion and its application to protein function prediction in yeast. Pac Symp Biocomput, vol 11
53. 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
54. Li F, Fu Y, Dai YH, Sminchisescu C, Jue W (2009) Kernel learning by unconstrained optimization. J Mach Learn Res 5:328–335
55. Lu J, Plataniotis KN, Venetsanopoulos AN (2005) Kernel discriminant learning with application to face recognition. In: Wang L (ed) Support vector machines: theory and applications, Springer, Heidelberg, Studies in Fuzziness and Soft Computing, pp 275–296. doi:10.1007/10984697_13
56. MacKay JCD (1997) Introduction to gaussian processes. NATO ASI Ser F: Comput Syst Sci 168:33–165
57. Mercer J (1909) Functions of positive and negative type and their connection with the theory of integral equations. Philos Trans R Soc 209:415–446. doi:10.1098%2Frsta.1909.0016
58. Micchelli CA, Pontil M (2005) Learning the kernel function via regularization. J Mach Learn Res 6:1099–1125
59. Micchelli CA, Pontil M (2007) Feature space perspectives for learning the kernel. Mach Learn 66(2–3): 297–319. doi:10.1007/s10994-006-0679-0
60. 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
61. Moreno PJ, Ho P, Vasconcelos N (2003) A kullback-leibler divergence based kernel for svm classification in multimedia applications. In: Thrun S, Saul LK, Schölkopf B, Thrun S, Saul LK, Schölkopf B (eds) NIPS, MIT Press. http://dblp.uni-trier.de/rec/bibtex/conf/nips/MorenoHV03
62. Nguyen CH, Ho TB (2008) An efficient kernel matrix evaluation measure. Pattern Recognit 41(11):3366–3372. doi:10.1016/j.patcog.2008.04.005
63. Platt JC (1999a) Fast training of support vector machines using sequential minimal optimization. Adv Kernel Methods Support Vector Learn, pp 185–208
64. Platt JC (1999b) Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. Adv Large Margin Classif, pp 61–74
65. Raina R, Battle A, Lee H (2007) Self-taught learning: transfer learning from unlabeled data. 24th International conference on machine learning
66. 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. doi:10.1145/1273496.1273594
67. Rakotomamonjy A, Bach FR, Canu S, Grandvalet Y (2008) Simplemkl. J Mach Learn Res 9:2491–2521
68. Rasmussen CE (2003) Gaussian processes in machine learning 3176:63–71. doi:10.1007/978-3-540-28650-9_4
69. Rückert U, Kramer S (2008) Kernel-based inductive transfer. In: Machine learning and knowledge discovery in databases, Springer, Berlin, Heidelberg, pp 220–233. doi:10.1007/978-3-540-87481-2_15
70. Saunders C, Gammerman A, Vovk V (1998) Ridge regression learning algorithm in dual variables. In: ICML '98: Proceedings of the fifteenth international conference on machine learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 515–521
71. Schapire RE, Singer Y (1999) Improved boosting algorithms using confidence-rated predictions. J Mach Learn 37(3):297–336. doi:10.1023/A:1007614523901
72. Scholkopf B, Smola AJ (2001) Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT Press, Cambridge
73. 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

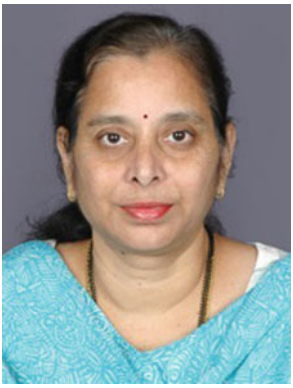
74. Shaw B, Jebara T (2009) Structure preserving embedding. In: ICML '09: Proceedings of the 26th annual international conference on machine learning, ACM, New York, NY, USA, pp 937–944. doi:10.1145/1553374.1553494
75. Shawe-Taylor J, Cristianini N (2004) Kernel methods for pattern analysis. Cambridge University Press, New York
76. Sindhwani V, Niyogi P, Belkin M (2005) Beyond the point cloud: from transductive to semi-supervised learning. In: ICML '05: Proceedings of the 22nd international conference on machine learning, ACM, New York, NY, USA, pp 824–831. doi:10.1145/1102351.1102455
77. Smola A, Hofmann T, Scholkopf B (2007) Kernel methods in machine learning. Annal Stat 36:1171–1220. http://eprints.pascal-network.org/archive/00003984/
78. Szafranski M, Grandvalet Y, Rakotomamonjy A (2008) Composite kernel learning. In: ICML '08: Proceedings of the 25th international conference on machine learning, ACM, New York, NY, USA, pp 1040–1047. doi:10.1145/1390156.1390287
79. Tsuda K, Noble WS (2004) Learning kernels from biological networks by maximizing entropy. Bioinformatics 20(1):326–333. doi:10.1093/bioinformatics/bth906
80. Tsuda K, Kin T, Asai K (2002) Marginalized kernels for biological sequences. Bioinformatics (Oxford, England) 19:2149–2156. doi:10.1093/bioinformatics/18.suppl_1.S268
81. Vapnik V (1999) An overview of statistical learning theory. IEEE Trans Neural Netw 10(5):988–999. doi:10.1109/72.788640
82. Wang G, Yeung DY, Lochovsky FH (2007) A kernel path algorithm for support vector machines. In: ICML '07: Proceedings of the 24th international conference on machine learning, ACM, New York, NY, USA, pp 951–958. doi:10.1145/1273496.1273616
83. Wang J, Lu H, Plataniotis K, Lu J (2009) Gaussian kernel optimization for pattern classification. Pattern Recognit 42(7):1237–1247. doi:10.1016/j.patcog.2008.11.024
84. 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
85. 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
86. Williams CKI, Barber D (1998) Bayesian classification with gaussian processes. IEEE Trans Pattern Anal Mach Intell 20(12):1342–1351. doi:10.1109/34.735807
87. Williams P, Li S, Feng J, Wu S (2007) A geometrical method to improve performance of the support vector machine. IEEE Trans Neural Netw 18(3):942–947. doi:10.1109/TNN.2007.891625
88. Wu S, Amari SI (2002) Conformal transformation of kernel functions: a data-dependent way to improve support vector machine classifiers. Neural Process Lett 15(1):59–67. doi:10.1023/A:1013848912046
89. Xiong H, Swamy M, Ahmad M (2005) Optimizing the kernel in the empirical feature space. IEEE Trans Neural Netw 16(2):460–474. doi:10.1109/TNN.2004.841784
90. Xu Z, Zhu J, Lyu MR, King I (2007) Maximum margin based semi-supervised spectral kernel learning. In: Proceedings of international joint conference on neural networks, Orlando, Florida, USA
91. Xu Z, Jin R, Yang H, King I, Lyu MR (2010) Simple and efficient multiple kernel learning by group lasso. In: ICML '10: Proceedings of the 27th international conference on machine learning, ACM
92. Yang J, Cheung WK, Chen X (2008) Learning element similarity matrix for semi-structured document analysis. Knowl Inf Syst
93. Yang L, Jin R (May, 2006) Distance metric learning: a comprehensive survey. Department of Computer Science and Engineering, Michigan State University
94. Yeung DY, Chang H, Dai G (2007) Learning the kernel matrix by maximizing a kfd-based class separability criterion. Pattern Recognit 40(7):2021–2028. doi:10.1016/j.patcog.2006.12.031
95. Yeung DY, Chang H, Dai G (2008) A scalable kernel-based semisupervised metric learning algorithm with out-of-sample generalization ability. Neural Comput 20(11):2839–2861. doi:10.1162/neco.2008.05-07-528
96. Yuan M, Yuan M, Lin Y, Lin Y (2006) Model selection and estimation in regression with grouped variables. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.79.2062
97. Zhang Z, Yeung DY, Kwok JT (2004) Bayesian inference for transductive learning of kernel matrix using the tanner-wong data augmentation algorithm. In: Brodley CE (ed) ICML, machine learning, proceedings of the twenty-first international conference (ICML 2004), Banff, Alberta, Canada, July 4–8, 2004, ACM, ACM International Conference Proceeding Series, vol 69
98. Zhang Z, Kwok JT, Yeung DY (2006) Model-based transductive learning of the kernel matrix. Mach Learn 63(1):69–101

99. Zhang Z, Yeung DY, , Kwok JT, Kwok JT (2007) Probabilistic kernel matrix learning with a mixture model of kernels
100. Zhengdong Lu PJ, Dhillon I (2009) Geometry-aware metric learning. In: ICML '09: Proceedings of the 26nd international conference on machine learning, ACM
101. Zhu X, Kandola J, Ghahramani Z, Lafferty J (2005) Nonparametric transforms of graph kernels for semi-supervised learning. In: Nonparametric transforms of graph kernels for semi-supervised learning, no. 17 in advances in neural information processing systems
102. Zhuang J, Tsang IW, Hoi SCH (2009) Simplenpkl: simple non-parametric kernel learning. In: ICML '09: Proceedings of the 26th annual international conference on machine learning, ACM, New York, NY, USA, pp 1273–1280. doi:10.1145/1553374.1553537

Author Biographies

M. Ehsan Abbasnejad received his Bachelors in Software Engineering from the Islamic Azad University, Shiraz, Iran in 2007. He consequently earned his Master's degree by research from University of Science, Malaysia. He is currently pursuing his PhD in Department of Computer Sciences at the Australian National University. His research interests are in the areas of machine learning and pattern recognition.

Dhanesh Ramachandram received his B.Tech. (Hons.) in Instrumentation and Quality Control (1997) and PhD in Industrial Technology Robot Vision (2003) from University Science of Malaysia. He is a lecturer in the School of Computer Sciences and his research interests include medical image analysis, machine learning and pattern recognition.



Rajeswari Mandava is an Associate Professor at School of Computer Science, Universiti Sains Malaysia (USM), Penang. She is the Head of Computer Vision Research Group and also the Head of Data-to-Knowledge Research Cluster at USM. Prior to joining the School of Computer Science, she has served USM in the School of Electrical Engineering and the School of Industrial Technology. Mandava's primary research is in Image analysis and her research includes Machine Intelligence, Pattern Recognition and Soft Computing areas as enablers for Image analysis. Her current research is in Semantic Image Knowledge Extraction and Medical Image Analysis. She currently leads a large research group with several researchers in Computer Science and Medical Specialists.