A Review of Kernel Methods

D Ranjeeth Kumar and C. V. Jawahar
Center for Visual Information Technology
International Institute of Information Technology
Gachibowli, Hyderabad, 500032, INDIA.

Abstract

In this report, kernel methods, the new class of algorithms popular in machine learning, are reviewed. Kernel methods combine the stability and efficiency of linear algorithms with the descriptive power of nonlinear features. The kernel trick results in several advantages : i) detecting complex nonlinear relationships in the input data while ensuring the statistical stability of the detected relationships ii) treating different types of data (such as numerical and categorical, structured and unstructured) under a single framework iii) incorporation of prior information in the kernel function etc. These unique features led to immense popularity kernel-based methods such as the Support Vector Machine and Kernel Principal Component Analysis. This report introduces the fundamental ideas behind the kernel trick along with elementary theory of kernel functions. Some popular kernel methods are reviewed and current challenges in the domain and promising directions of research are discussed.

Keywords Kernel Methods, Linear Algorithms, Kernel PCA, Support Vector Machine

1 Introduction

Kernel Methods received wide attention in the last one and a half decade and are used extensively a number of domains such as computer vision, bio-informatics, data mining, pattern recognition etc [1, 2, 3, 4, 5]. Traditionally, linear algorithms are the first choice for a wide number of tasks such as classification, unsupervised learning etc. This is due the numerical and statistical stability of linear algorithms. Linear relationships are easier to detect from data and are the simplest and most natural estimate of an unknown relationship among several variables. A number of linear methods such as Principal Component Analysis (PCA) [6], Linear Discriminant Analysis(LDA) [7], Linear Predictive Coding [8] found are used widely for tasks involving compression, recognition, detection, modeling and synthesis. However as the complexity of the tasks increases the low descriptive power of linear functions turns out to be a handicap. Nonlinear functions, on the other hand are more descriptive but the estimation of such functions is fraught with problems concerning stability (both numerical and statistical) and convergence. The approaches to detection of nonlinear relationships in data were rather less principled than linear algorithms [9]. For a long time, in both research world and the industry, one of the complementary benefits of these two classes of algorithms was sacrificed in favor of the other (based on the complexity of the task at hand) since there was no way of combining these two advantages.

The introduction of the Support Vector Machine(SVM) [10] for the classification problem first demonstrated the use of *kernel functions* [11, 12] to combine the advantages of linear and nonlinear functions. The essence of the method is to implement a linear classifier in a feature space that is nonlinearly related to the input space *without* explicitly accessing the feature space. The fundamental idea is that a complex relationship in the input data can be simplified by recoding the data in an appropriate manner. Although this idea was in vogue in the domain of nonlinear modeling for a long time, explicit recoding of data is prohibitively expensive in a number of applications. The paradigm was, thus, of little utility for problems involving high-dimensional data. However using the *kernel function* to indirectly access the recoded data via the inner product facilitates makes estimation of linear functions feasible. Ever since the introduction of SVM, a number of successful linear algorithms such as PCA, LDA are *kernelized* i.e. used the kernel trick to incorporate the power of nonlinearity [13, 14, 15]. The resulting algorithms are superior to their linear counterparts in terms of descriptive power and are as stable.

Descriptive power is only one among the many advantages brought by the use of kernel functions. A kernelized algorithm is akin to template definitions in software : Any kernel function can be used with a kernelized algorithm without effecting the statistical properties such as generalization capability of the algorithm (in case of classification algorithms) etc. This allows domain specific knowledge (or *prior*) to be incorporated in the kernel function without changing the algorithm. This modularity makes development of powerful and stable algorithms easier. The theory of kernels does not place any restrictions on the input space : the input data samples can be numerical values, vectors, text strings, sets, graphs etc. Thus the kernel trick allows a

number of algorithms that are designed for numeric data to be applied to non-numerical data thus enhancing the applicability of the these algorithms and increasing the number of tools for analysis of heterogeneous data. Several other advantage of kernel methods will be described in the following sections. The underlying theory of kernel methods is covered in a number of books [9, 16, 17, 18]. In the following section the kernel trick is introduced with the example of *kernel perceptron*. The essentials of theory of kernel methods, popular kernel methods, their advantages and disadvantages, current challenges in the field are reviewed in the later sections.

2 The kernel trick : An example

Pattern analysis refers to the task of finding inherent regularities in the input data from a finite sample. Such regularities characterize the source distribution generating the data and aid in a number of tasks such as i) compression of data, ii) classification of unseen samples, iii) prediction of unseen values, iv) detecting outliers i.e. samples that are not likely from the source distribution etc. Linear relationships are simplest of such regularities. Detecting such relationships is both numerically and statistically stable. In the following, one such method, *the perceptron* algorithm [19] is described. The use of kernel trick to increase the applicability of the perceptron algorithm is demonstrated.

2.1 The perceptron algorithm

Binary pattern classification is one of the popular problems in machine learning. Given an input data set : $\mathcal{X} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)\}$ where $\mathbf{x}_i \in \mathcal{R}^m$ and $y_i \in \pm 1$ for $i = 1, 2, \cdots, N$, the task is to find a function $f : \mathcal{R}^m \rightarrow \pm 1$ which learns the relationship between the $\mathbf{x}_i$ and y_i. Each $\mathbf{x}_i$ is an *input pattern* and y_i is its *label*. The function $f(.)$ can be used for a number of purposes such as i) prediction of label y_t for a test sample $\mathbf{x}_t$ ii) verifying the authenticity of a new pattern-label pair $(\mathbf{x_t}, y_t)$ etc. The perceptron algorithm [19] sets out by assuming that a function $f(.)$ that is linear in $\mathbf{x}$ can satisfy the constraints $f(\mathbf{x}_i) = y_i, \forall\ i \in \{1, 2, \cdots, N\}$. If such a function exists the set $\mathcal{X}$ is *linearly separable*. The function $f(.)$ is of the following form:

$$y = sign(\langle \mathbf{w}, \mathbf{x} \rangle + w_0) \tag{1}$$

where $\mathbf{w} \in \mathcal{R}^d$ and w_0 is a scalar. In the following discussion the scalar w_0 is absorbed in to the weight $\mathbf{w}$ by augmenting the samples : $\mathbf{x}_i^t = [\mathbf{x}_i^t \quad 1]$ and $\mathbf{w}^t = [\mathbf{w}_i^t \quad w_0]$. This allows us to deal with functions of the form $sign(\langle \mathbf{w}, \mathbf{x} \rangle)$. The function is linear in $\mathbf{x}$. The intuition behind this form is that similar objects are mapped to similar classes since

$$|\langle \mathbf{w}, \mathbf{x}_1 \rangle - \langle \mathbf{w}, \mathbf{x}_2 \rangle| = |\langle \mathbf{w}, \mathbf{x}_1 - \mathbf{x}_2 \rangle| \leq ||\mathbf{w}|| \cdot ||\mathbf{x_1} - \mathbf{x}_2|| \tag{2}$$

where the inequality is a result of the Cauchy-Schwartz inequality. The solution does not change if the solution vector $\mathbf{w}$ is rescaled since for $\lambda \neq 0$,

$$sign(\langle \lambda \mathbf{w}, \mathbf{x} \rangle) = sign(\lambda \langle \mathbf{w}, \mathbf{x} \rangle) = sign(\langle \mathbf{w}, \mathbf{x} \rangle) \tag{3}$$

This ambiguity is removed by enforcing an additional constraint that $||\mathbf{w}|| = 1$. An iterative procedure to estimate $\mathbf{w}$ is proposed by Rosenblatt [19] which operates as shown in Algorithm 1.

Algorithm 1 PerceptronPrimal($\{(\mathbf{x}_i, y_i)\}$ for $i = 1, 2, \cdots, N$)

Require: $\exists \mathbf{w} \ni y_i = sign(\langle \mathbf{w}, \mathbf{x}_i \rangle)$ for $i = 1, 2, \cdots, N$

```
w = 0
repeat
   for i = 1 to N do
      if y_i <w, x_i> ≤ 0 then
         w ← w + y_i x_i
      end if
   end for
until no sample is misclassified
```

The intuition behind the perceptron algorithm is simple : if a sample is misclassified i.e. it lies on the *wrong* side of the plane described by $\mathbf{w}$, then the plane is moved *towards* the sample either by adding/subtracting that sample to the vector $\mathbf{w}$. This reduces the distance between the sample and the plane. The plane eventually moves past the misclassified sample so that it is classified correctly. Figure 1 shows how this adjustment of $\mathbf{w}$ operates. The figure also shows how a plane in this space can be viewed as equivalent to point. The vector describing this point is perpendicular to the plane. This duality of points and planes allows many problems

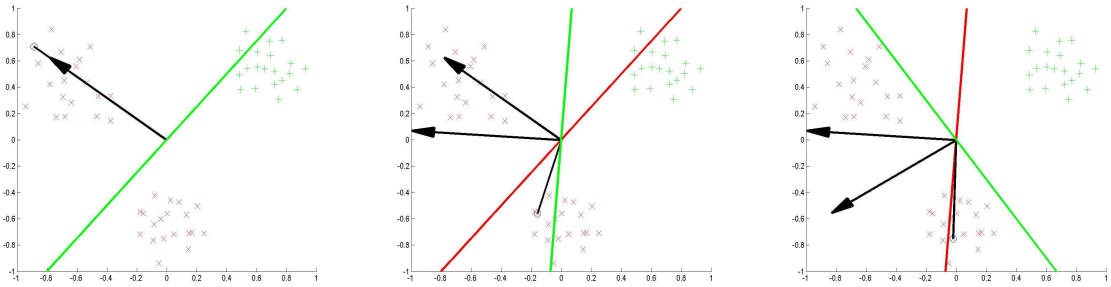


Figure 1: The adjustment of the plane $\mathbf{w}$ in the perceptron algorithm. Each misclassified sample moves the plane (plane in red) to reduce the number of errors in the next iteration (plane in green.

to be recast in an alternate form known as the *dual form*. Algorithm 1 is the *primal form* of the perceptron algorithm. An interesting feature of the perceptron algorithm is that the final solution vector $\mathbf{w}$ is always linear combination of the points in the input data i.e.

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i \tag{4}$$

Estimating the coefficients $\alpha_1, \cdots, \alpha_N$ is equivalent to estimating $\mathbf{w}$. Additionally, the projection of a sample $\mathbf{x}_t$ on to $\mathbf{w}$ can be computed using the coefficients α_i and the input samples $\mathbf{x}_i$ where $i \in \{1, 2, \cdots, N\}$ since

$$\langle \mathbf{w}, \mathbf{x}_t \rangle = \langle \sum_{i=1}^{N} \alpha_i \mathbf{x}_i, \mathbf{x}_t \rangle = \sum_{i=1}^{N} \alpha_i \langle \mathbf{x}_i, \mathbf{x}_t \rangle \tag{5}$$

This computation in the dual form is more expensive than it is in the primal form. However the dual form results in other advantages in terms of expressive power. The perceptron algorithm in its dual form is given in algorithm 2. To speeden the estimation of α_i the inner products between the input data samples can be precomputed and stored in a matrix $\mathbf{G}_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$, know as the Gram Matrix.

Algorithm 2 PerceptronDual($\{(\mathbf{x}_i, y_i)\}$ for $i = 1, 2, \cdots, N$)

Require: $\exists \alpha_j \ni y_i = sign(\langle \sum_{j=1}^{N} \alpha_j \mathbf{x}_j, \mathbf{x}_i \rangle)$ for $i, j \in \{1, 2, \cdots, N\}$
1: $\alpha_i = 0$ for $i = 1, 2, \cdots, N$
2: compute the Gram Matrix $\mathbf{G}_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$
3: **repeat**
4: **for** $i = 1$ to N **do**
5: **if** $y_i \sum_{j=1}^{N} \alpha_j \mathbf{G}_{ij} \leq 0$ **then**
6: $\alpha_i \leftarrow \alpha_i + y_i$
7: **end if**
8: **end for**
9: **until** no sample is misclassified

2.2 Linearization of Nonlinearities

While the simplicity of the perceptron algorithm makes it very attractive, its lack of expressive power is a big disadvantage. Many practical applications involve nonlinear relationships among data and non-linearly separable. An example of linearly non-separable data is shown in figure 2. The perceptron algorithm is guaranteed converge whenever the data is linearly separable but is not guaranteed to converge for data that is not linearly separable. This limits its applicability to a very small number of cases.

A well known technique to handle nonlinear relationships using linear algorithms is to *linearize* the relationships by transforming the data appropriately. For instance the boundary between the positive and negative samples in figure 2 is described by a circle of radius R centered at the origin, $x_1^2 + x_2^2 - R^2 = 0$ in $\mathcal{R}^2$ where each sample is described by $\mathbf{x} = [x_1 \quad x_2]^t$. If the data is mapped to $\mathcal{R}^2$ using the map $\phi : \mathcal{R}^2 \mapsto \mathcal{R}^3$ such that all possible monomials of second degree form the entries of the transformed vector :

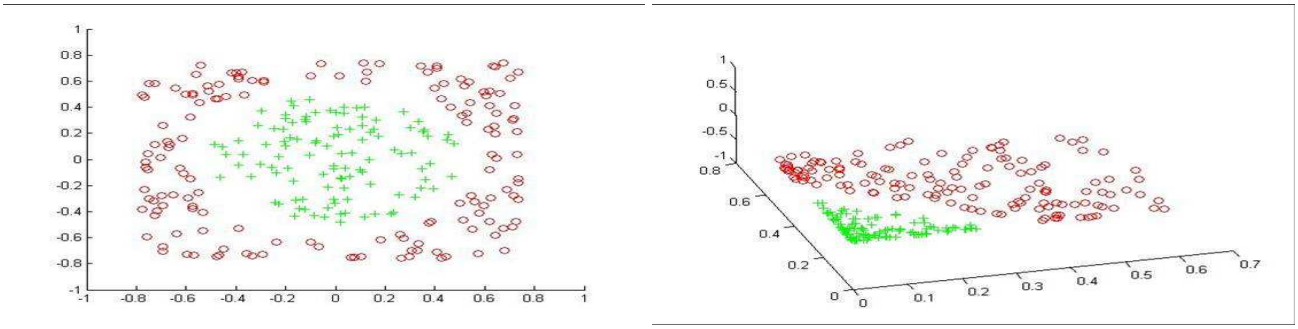


Figure 2: Linearization of nonlinear relationships by recoding the data. The circular boundary in $\mathcal{R}^2$ becomes a plane in $\mathcal{R}^3$.

$$\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) \mapsto \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_1 x_2 \end{bmatrix} \tag{6}$$

then the boundary becomes linear in the transformed space described by $\mathbf{W}^t\phi(\mathbf{x}) - R^2 = 0$ where $\mathbf{W} = [1 \quad 1 \quad 0]^t$. All relationships that can be expressed as second order polynomials in the input feature space can be linearized with the map $\phi()$. Similarly more complex relationships can be linearized by using an appropriate mapping. The perceptron algorithm can be applied to the transformed data thus extending it to handle nonlinear relationships.

The technique presented above, however, does not scale with size and dimensionality of the input data. For instance, for input samples of dimensionality m, the number of all possible degree d monomials is

$$D = \frac{(m+d-1)!}{(m-1)!d!} \tag{7}$$

A typical recognition problem in computer vision involves images of size 256×256 pixels. A common representation of such data is to form a feature vector by stacking all the pixel values resulting in vectors of dimensionality of $m = 65536$. The number of all possible second degree monomials in this case is $(65537 \times 65536)/2 = 214756416$. The numbers increase exponentially with the data size and the degree of the monomials making their explicit evaluation prohibitively expensive. However, the perceptron algorithm in its dual form does not require the data to be explicitly recoded. Instead, only the Gram Matrix in the transformed space (called the *feature space*), $\mathbf{G}_{ij} = \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle$, is required. Similarly classification of a test sample, $\mathbf{x}_t$ requires the inner products $\langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_t)\rangle$. Thus, if the inner product between two samples in the feature space can be computed efficiently, then the dual form can be used to handle complex relationships efficiently.

2.3 The Kernel Trick

The kernel trick enables the efficient computation of inner product in the feature space. Consider the following mapping which is similar to the mapping in Equation 6 except that all ordered products of second degree are taken as the entries of the transformed vector:

$$\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) \mapsto \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_1 x_2 \\ x_2 x_1 \end{bmatrix} \tag{8}$$

This alternate mapping makes it possible to evaluate the inner product easily since,

$$\langle\phi(\mathbf{x}), \phi(\mathbf{y})\rangle = x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 y_1 x_2 y_2 = (x_1 y_1 + x_2 y_2)^2 = \langle\mathbf{x}, \mathbf{y}\rangle^2 \tag{9}$$

The above result can be generalized to any dimension and degree [10] : If $\phi(\mathbf{x})$ maps m-dimensional vector $\mathbf{x}$ to a vector whose entries are all possible degree d ordered products of components of $\mathbf{x}$ then the inner product can be expressed as

$$\langle\phi(\mathbf{x}), \phi(\mathbf{y})\rangle = \sum_{i_1, \cdots, i_d}^{m} x_{i_1} \cdots x_{i_d} y_{i_1} \cdots y_{i_d} = (\langle\mathbf{x}, \mathbf{y}\rangle)^d \tag{10}$$

Alternate forms of $\phi()$ that consider only the unordered products but weight them proportional to the frequency of their occurrence result in the same inner product. For instance, for the $m = 2$ and $d = 2$ the map $\phi(\mathbf{x}) =$

$[x_1 \quad x_2 \quad \sqrt{2}x_1x_2]$ would result in the same inner product as the map in equation 8. The functional form the of the inner product

$$\kappa(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle)^d \tag{11}$$

is called the *kernel function*. The kernel function is simple to evaluate and bypasses the computationally intensive computation of $\phi(.)$. Further, concatenating an additional constant term 1 to $\phi(\mathbf{x})$ is equivalent to modifying the kernel function as follows

$$\kappa(\mathbf{x}, \mathbf{y}) = \langle \begin{bmatrix} \phi(\mathbf{x}) \\ 1 \end{bmatrix}, \begin{bmatrix} \phi(\mathbf{y}) \\ 1 \end{bmatrix} \rangle = 1 + \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle = 1 + (\langle \mathbf{x}, \mathbf{y} \rangle)^d \tag{12}$$

The transformed version of the Gram matrix, $\mathbf{G}$, called the kernel matrix $\mathbf{K}$ can now be computed efficiently, since $\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{y}_j)$. The *kernelized* version of the perceptron algorithm is given in algorithm 3.

Algorithm 3 KernelPerceptron($\{(\mathbf{x}_i, y_i)\}$ for $i = 1, 2, \cdots, N$, $\kappa(.,.)$)

```
1: α_i = 0 for i = 1, 2, ··· , N
2: compute the Kernel Matrix K_ij = κ(x_i, x_j)
3: repeat
4:    for i = 1 to N do
5:       if  y_i Σ_{j=1}^{N} α_j K_ij ≤ 0 then
6:          α_i ← α_i + y_i
7:       end if
8:    end for
9: until no sample is misclassified
```

The kernel perceptron algorithm described above takes the higher order statistics of the data in to account without explicitly computing the higher order terms (the complexity of which increases combinatorially) while the original perceptron algorithm fails to do so. Figure 3 shows how the kernel perceptron captures nonlinear relationships in the data. The traditional perceptron algorithm failed to converge in this case. The images in figure 3 show the input space and the how the relationship learned using a kernel function looks like in the input space and are good of visualizing the advantages of using a kernel function. Such images will be used through this report for demonstrating algorithms.

The kernel functions of the form in equation 11 are called *polynomial* kernels. Even this simple family of inner products in the feature space enable us to detect a wide variety of nonlinearities since the kernels can be combined to give rise to other inner products. For instance, consider kernel function $\kappa_1(.)$ and $\kappa_2(,)$ corresponding to to two mappings $\phi_1()$ and $\phi_2()$. The inner product in the feature space that includes features from both mappings is equivalent to :

$$\begin{aligned} \kappa_3(\mathbf{x}, \mathbf{y}) \quad &= \langle \begin{bmatrix} \phi_1(\mathbf{x}) \\ \phi_2(\mathbf{x}) \end{bmatrix}, \begin{bmatrix} \phi_1(\mathbf{y}) \\ \phi_2(\mathbf{y}) \end{bmatrix} \rangle = \langle \phi_1(\mathbf{x}), \phi_1(\mathbf{y}) \rangle + \langle \phi_2(\mathbf{x}), \phi_2(\mathbf{y}) \rangle \\ \Rightarrow \kappa_3(\mathbf{x}, \mathbf{y}) \quad &= \kappa_1(\mathbf{x}, \mathbf{y}) + \kappa_2(\mathbf{x}, \mathbf{y}) \end{aligned} \tag{13}$$

The above result, in conjunction with the binomial expansion, results in the following kernel which takes in to account, all the monomials up degree d or less :

$$\kappa(\mathbf{x}, \mathbf{y}) = (1 + \langle \mathbf{x}, \mathbf{y} \rangle)^d \tag{14}$$

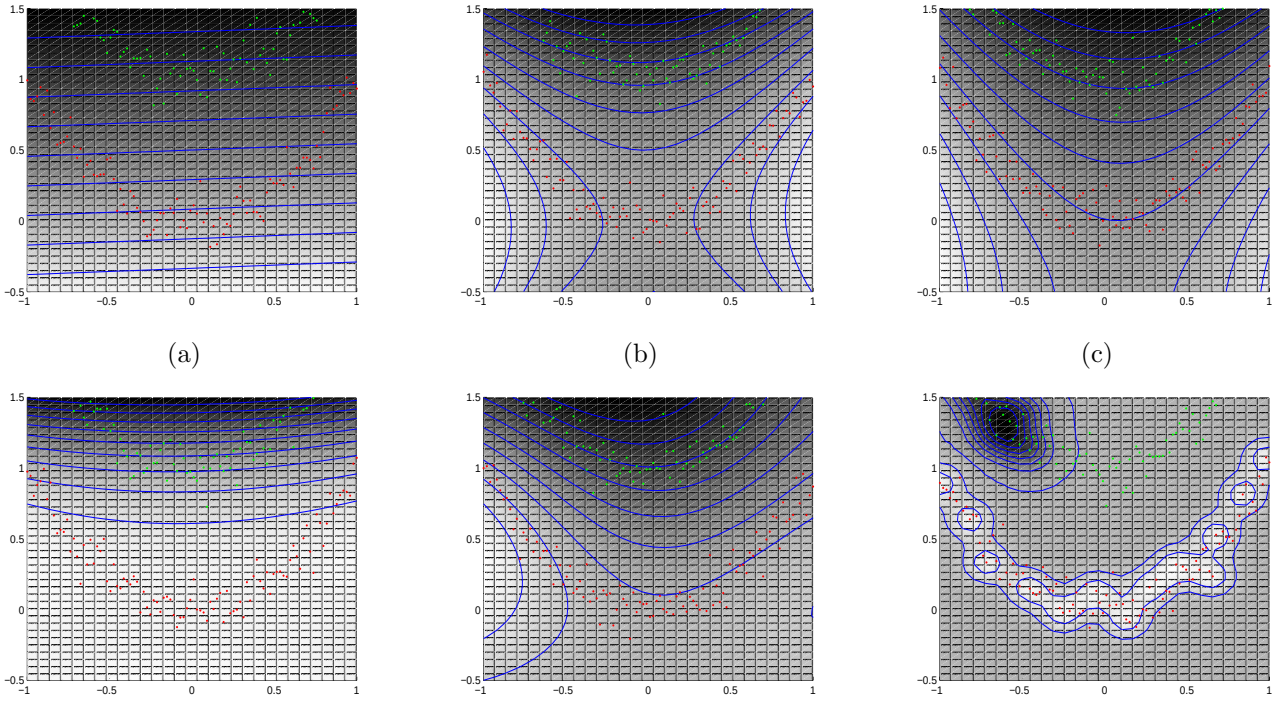
From the discussion above, it can be seen that the functional form $\kappa(.,.)$ is of greater use in practice than the mapping $\phi()$ since the mapping is never used in the computation. Although the mapping $\phi()$ is introduced first in the current discussion, in general, the choice of $\kappa(.,.)$ implicitly defines the mapping $\phi(.,.)$. As seen earlier, there may be several mappings that corresponding to the inner product $\kappa(,)$. There are several function which are valid kernel functions such as the Gaussian radial basis kernel [10]

$$\kappa(\mathbf{x}, \mathbf{y}) = \exp(-\frac{||\mathbf{x} - \mathbf{y}||^2}{2\sigma^2}) \tag{15}$$

and the sigmoid kernel [10]

$$\kappa(\mathbf{x}, \mathbf{y}) = tanh(\kappa(\langle \mathbf{x}, \mathbf{y} \rangle) + \Theta) \tag{16}$$

However, not all functional forms $\kappa(,)$ correspond to inner product in some feature space. The question of what functions $\kappa(,)$ satisfy this criterion is connected to Reproducing Kernel Hilbert Spaces and is discussed widely in literature [10, 20]. The theory of kernel function is briefly reviewed in the following section.



(a) (b) (c)

(d) (e) (f)

Figure 3: The kernel perceptron detects nonlinear relationships. The kernel perceptron algorithm is used to learn a classification function to separate two data sets (one in green and the other in red) each distributed in parabolic shapes. The blue lines in each image are isocontours i.e contours along which the classification function's value is constant. The background color at each point indicates the value of function at each point (higher values are shown brighter). Thus each blue line corresponds a boundary between the class at certain threshold. (a) shows the result using a linear kernel $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$ which fails to find a satisfactory solution. (b) and (c) show results with $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle^2$ and $\kappa(\mathbf{x}, \mathbf{y}) = (1 + \langle \mathbf{x}, \mathbf{y} \rangle)^2$ respectively. The exact parabolic boundary that separates the two distributions are found. (d) shows the result with $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle^3$. Although the boundary is nonlinear it is not useful for classifying unseen examples. (e) and (f) show the results with $\kappa(\mathbf{x}, \mathbf{y}) = \exp(-\frac{||\mathbf{x}-\mathbf{y}||^2}{2\sigma^2})$ with $\sigma = 1, 0.1$ respectively. The solution in the first case is expected to be more stable on unseen examples.

3 The theory of kernel functions

The main theme of the kernel trick is to operate in a feature space via a kernel function $\kappa(,)$. The feature space is accessed indirectly via pairwise inner products. To understand how this indirect access to a different feature space enables the implementation of linear algorithms we first review some properties of linear functions and the inner product.

3.1 Inner product and Hilbert Spaces

Linear Function Given a vector space X over $\mathcal{R}$, a function $f : X \mapsto \mathcal{R}$ is said to be *linear* if i) $f(\alpha\mathbf{x}) = \alpha f(\mathbf{x})$ and ii) $f(\mathbf{x}+\mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in X$ and $\alpha \in \mathcal{R}$.

Inner Product Space A vector space X over $\mathcal{R}$ is know as an *inner product space* over $\mathcal{R}$ if there exists a real symmetric bilinear map $\langle ., . \rangle : X \times X \mapsto \mathcal{R}$ such that $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0$ for all $\mathbf{x} \in X$. The map $\langle ., . \rangle$ is known as the inner product or dot product. The inner product is said to be *strict* if $\langle \mathbf{x}, \mathbf{x} \rangle = 0 \Leftrightarrow \mathbf{x} = \mathbf{0}$. The *norm* on a strict inner product space and the associated distance between two vectors $\mathbf{x}$ and $\mathbf{y}$ are defined as

$$||\mathbf{x}|| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$$

$$d(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||$$

Cauchy-Schwartz Inequality For all $\mathbf{x}, \mathbf{y}$ in an inner product space X over $\mathcal{R}$ the following inequality holds

$$\langle \mathbf{x}, \mathbf{y} \rangle \leq ||\mathbf{x}||||\mathbf{y}|| \tag{17}$$

where, in a strict inner product space, the equality holds if and only if $\mathbf{x}$ and $\mathbf{y}$ are linearly dependent.

Proof Consider the following inequality where $\alpha \in \mathcal{R}$

$$\langle \alpha\mathbf{x} + \mathbf{y}, \alpha\mathbf{x} + \mathbf{y} \rangle = \alpha^2 ||\mathbf{x}||^2 + 2\alpha\langle \mathbf{x}, \mathbf{y} \rangle + ||\mathbf{y}||^2 \geq 0 \tag{18}$$

The above inequality holds for all α. Hence, considering the equality as a quadratic equation the determinant is non-positive since there is at most one root

$$4\langle \mathbf{x}, \mathbf{y} \rangle^2 - 4||\mathbf{x}||^2||\mathbf{y}||^2 \leq 0$$

$$\Rightarrow \langle \mathbf{x}, \mathbf{y} \rangle \leq ||\mathbf{x}||||\mathbf{y}|| \tag{19}$$

The equality, in a strict inner product space, implies $\alpha\mathbf{x} + \mathbf{y} = 0$ which shows $\mathbf{x}$ and $\mathbf{y}$ are rescalings of the same vector.

The set of vectors $X_0 = \{\mathbf{x} \in X \ni ||\mathbf{x}|| = 0\}$ forms a linear subspace of X. This is so since if $\mathbf{x}, \mathbf{y} \in X_0$, then for all $\alpha, \beta \in \mathcal{R}$ we have

$$||\alpha\mathbf{x} + \beta\mathbf{y}||^2 = \langle \alpha\mathbf{x} + \beta\mathbf{y}, \alpha\mathbf{x} + \beta\mathbf{y} \rangle = \alpha^2||\mathbf{x}||^2 + 2\alpha\beta\langle \mathbf{x}, \mathbf{y} \rangle + \beta^2||\mathbf{y}||^2 = 0$$

since $\langle \mathbf{x}, \mathbf{y} \rangle^2 \leq ||\mathbf{x}||^2||\mathbf{y}||^2 = 0$ by the Cauchy-Schwartz inequality. Hence $\alpha\mathbf{x} + \beta\mathbf{y} \in X_0$. This means that any non-strict inner product space can be converted to a strict inner product by taking the quotient space with respect to this subspace.

Separability and Completeness A Cauchy sequence is a sequence of elements $\{h_n\}_{n\geq 1}$ from an inner product space X with the following property

$$\lim_{n\to\infty} \sup_{m>n} ||h_n - h_m|| \to 0 \tag{20}$$

If every Cauchy sequence $\{h_n\}_{n\geq 1}$ of elements of X converges to an element $h \in X$, then X said to be *complete*. If for all $\epsilon > 0$ there exists a finite set of elements $h_1, \cdots, h_n$ such that for all $h \in X$

$$\min_i ||h_i - h|| < \epsilon, \tag{21}$$

then X is said to be *separable* .

Hilbert Space A separable and complete inner product space is know as a Hilbert Space. One particular Hilbert space of interest to us is the space L_2 which is the set of all countable sequences of real numbers $\mathbf{x} = (x_1, \cdots, x_n, \cdots)$ satisfying

$$\sum_i^\infty x_i^2 < \infty$$

where the inner product is defined by

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{\infty} x_i y_i$$

Similarly the space of square integrable functions on a compact subset X of $\mathcal{R}^n$,

$$\{f : \int_X f(x)^2 dx < \infty\}$$

with the inner product defined by

$$\langle f, g \rangle = \int_X f(x)g(x)dx$$

is a Hilbert space.A Hilbert space is isomorphic to either $\mathcal{R}^n$ for some finite n or to the space L_2. This implies that a coordinate system can be given to such spaces. Since kernel functions need to map input data to a space where the mapped data can be viewed as vectors, the feature space needs to a Hilbert space. Although the feature vectors are never explicitly accessed this condition is crucial for the kernel function to be a valid inner product.

Linear algorithms search for a weight vector in the feature space. However this weight vector is also point the feature space and each point $\mathbf{w}$ is a function via the inner product : $f_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle$. Thus estimating the linear function is equivalent to find the element $\mathbf{w}$ of the feature space. The key property that allows the dual form of a linear algorithm to be used is that this element $\mathbf{w}$ is a linear combination of the feature vectors of the training samples.

Dimensionality and Rank Given two vectors $\mathbf{x}$ and $\mathbf{y}$ in a strict inner product space X, the angle between them is defined by

$$\theta = \arccos \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{||\mathbf{x}||||\mathbf{y}||} \tag{22}$$

The vectors are said to be *orthogonal* if $\theta = 0$.A set of vectors $B = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ is called *orthonormal* if

$$\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \delta_{ij} \tag{23}$$

Given an orthonormal set B and a vector $\mathbf{w}$ the following expansion of the vector in terms of the elements of B is called it *Fourier Series* of the vector :

$$\hat{\mathbf{w}} = \sum_{i=1}^{n} \langle \mathbf{x}_i, \mathbf{w} \rangle \mathbf{x}_i \tag{24}$$

If $\hat{\mathbf{w}} = \mathbf{w}$ for all $\mathbf{w}$ the B is a *basis* of X and the number of elements of in B is the *dimensionality* of X. An alternate representation of a set of vectors ,given their coordinates in some basis, is the *data matrix* representation. Given m vectors, $\{\mathbf{x}_1, \cdots, \mathbf{x}_m\}$ from an n-dimensional space the *data matrix* $\mathbf{X}$, of size $n \times m$, is formed by arranging the vectors as the columns of the matrix i.e.

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_m \end{bmatrix}$$

Hereafter, the coordinates of a data sample will always be taken as a column vector and the ith column of the matrix $\mathbf{X}$ will be denoted by $\mathbf{X}^i$. The dimensionality of the space spanned by columns of $\mathbf{X}$ (also known as the *column space*) is called the *rank* of the matrix. Thus if the rank is r the there exist r linearly independent vectors $\mathbf{r}_1, \cdots, \mathbf{r}_r$ which form an $n \times r$ data matrix $\mathbf{R}$. The coordinates describing $\{\mathbf{x}_1, \cdots, \mathbf{x}_m\}$ in this basis form an $r \times m$ matrix $\mathbf{S}$. Thus $\mathbf{X}$ can be factorized as

$$\mathbf{X} = \mathbf{RS} \tag{25}$$

The rank of $\mathbf{X}^t$ is equal to the rank of $\mathbf{X}$. The matrix $\mathbf{X}$ is said to be *full rank* matrix if $rank(\mathbf{X}) = \min(m, n)$.

Eigen Values and Eigen Vectors An $n \times n$ matrix $\mathbf{A}$ is said to be singular if it is not *full rank*. This implies that the columns of $\mathbf{A}$ are linearly dependent. Thus there exists coefficients $x_1, \cdots, x_n$ such that $\sum_{i=1}^{n} x_i \mathbf{A}^i = \mathbf{0}$. Thus with $\mathbf{x} = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}^t$ we have

$$\mathbf{Ax} = \mathbf{0} = 0\mathbf{x} \tag{26}$$

The columns of an $n \times n$ non-singular matrix span a space of dimension n. Thus all the n unit vectors $\mathbf{e}_1, \cdots, \mathbf{e}_n$ can all be expressed as linear combination of the columns. The $n \times n$ matrix $\mathbf{B}$ with the corresponding coefficients $\mathbf{b}_1, \cdots, \mathbf{b}_n$ as columns is the multiplicative inverse of $\mathbf{A}$ since

$$\mathbf{Ab}_i = \mathbf{e}_i \tag{27}$$

$$\mathbf{AB} = \mathbf{I} \tag{28}$$

where $\mathbf{I}$ is the $n \times n$ identity matrix. If there exists a real number λ and a vector $\mathbf{x}$ such that, for a square matrix $\mathbf{A}$

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x} \tag{29}$$

the λ is called an eigen value of the matrix $\mathbf{A}$ and $\mathbf{x}$ the corresponding eigen vector. Thus, for a singular matrix 0 is always an eigenvalue. Conversely, if 0 is an eigenvalue then the matrix is singular. Eigen values and eigen vectors of matrices are important in solving optimization problems that appear in a number of kernelized linear algorithms. The following optimization problem (involving a quadratic term) appears frequently:

$$\max_{\mathbf{x}} \frac{\mathbf{x}^t\mathbf{A}\mathbf{x}}{\mathbf{x}^t\mathbf{x}} \tag{30}$$

Since the solution is invariant to scale an additional constraint on the norm of the $\mathbf{x}$ so that $||\mathbf{x}|| = \mathbf{x}^t\mathbf{x} = 1$ and the equation can be solved using Lagrangian multiplier λ. The can be posed as :

$$\max_{\mathbf{x}} \mathbf{x}^t\mathbf{A}\mathbf{x} - \lambda(\mathbf{x}^t\mathbf{x} - 1) \tag{31}$$

Differentiating with respect to $\mathbf{x}$ and setting to 0 we have

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x} \tag{32}$$

This implies that the solution is an eigen vector of the matrix $\mathbf{A}$. Further, the value of the expression itself is

$$\frac{\mathbf{x}^t\mathbf{A}\mathbf{x}}{\mathbf{x}^t\mathbf{x}} = \frac{\mathbf{x}^t\lambda\mathbf{x}}{\mathbf{x}^t\mathbf{x}} = \lambda \tag{33}$$

which is maximum when the eigen vector corresponds to the maximum eigen value. This result is used in a number of linear algorithms. A symmetric matrix of size $n \times n$ has at most n non-zero eigen values. In this case, The eigen vectors corresponding to distinct eigen values are orthogonal. Usually, the eigen vectors are assumed to be normalized such that their norm is 1. Given eigen vectors $\mathbf{v}_1, \cdots, \mathbf{v}_n$and corresponding eigen values $\lambda_1, \cdots, \lambda_n$ the matrix $\mathbf{V}$ whose columns are the eigen vectors and a diagonal matrix Λ, with $\Lambda_{ii} = \lambda_i$ satisfy :

$$\mathbf{A}\mathbf{V} = \mathbf{V}\Lambda \tag{34}$$

This is known as the eigen decomposition of the matrix $\mathbf{A}$. An interesting consequence of this decomposition is the following factorization for non-singular matrices:

$$\begin{aligned} \mathbf{A} &= \mathbf{V}\Lambda\mathbf{V}^t \\ \mathbf{A}^{-1} &= \mathbf{V}\Lambda^{-1}\mathbf{V}^t \end{aligned} \tag{35}$$

Further for singular matrices, the only non-zero eigen values and eigen vectors need to be collected in to the matrices $\mathbf{V}$ and Λ the first factorization to hold.

Positive semi-definite matrices A symmetric matrix $\mathbf{A}$ is said to be positive semi-definite if all its eigen values are non-negative. It can be shown that this condition holds if and only if, for all vectors $\mathbf{x}$

$$\mathbf{x}^t\mathbf{A}\mathbf{x} \geq 0 \tag{36}$$

The matrix is said to be positive definite if, for $\mathbf{x} \neq 0$,

$$\mathbf{x}^t\mathbf{A}\mathbf{x} > 0$$

An important property of positive semi-definite matrices is that for every such matrix $\mathbf{A}$ there exists a real matrix $\mathbf{B}$ such that :

$$\mathbf{A} = \mathbf{B}^t\mathbf{B} \tag{37}$$

The eigen decomposition of the matrix $\mathbf{A} = \mathbf{V}\Lambda\mathbf{V}^t$ can be used to make a choice of $\mathbf{B}$ as $\mathbf{B} = \sqrt{\Lambda}\mathbf{V}^t$. The choice of $\mathbf{B}$, however, is not unique. It is easy to see that the converse of the above statement is also true since any matrix that can factorized in the above manner satisfies, for all vectors $\mathbf{x}$:

$$\mathbf{x}^t\mathbf{A}\mathbf{x} = \mathbf{x}^t\mathbf{B}^t\mathbf{B}\mathbf{x} = (\mathbf{B}\mathbf{x})^t(\mathbf{B}\mathbf{x}) = ||\mathbf{B}\mathbf{x}||^2 \geq 0 \tag{38}$$

Positive definiteness plays an important role in establishing the properties of a valid kernel function. These properties and other desirable qualities of a kernel function are discussed in section 3.2.

3.2 What functions constitute valid kernel functions?

As seen in section 2 the kernel function can be used to access the feature space only via a finite set of samples. Thus, the kernelized gram matrix i.e. the kernel matrix is central to the kernelization of algorithms. An important of these matrices is that they are positive semi-definite. For a kernel matrix on the set $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ we have $\mathbf{K}_{ij} = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ and hence for any vector $\mathbf{x}$,

$$\begin{aligned} \mathbf{x}^t \mathbf{K} \mathbf{x} &= \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j \mathbf{K}_{\mathbf{ij}} = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \\ &= \langle \sum_{i=1}^{n} x_i \phi(\mathbf{x}_i), \sum_{j=1}^{n} x_j \phi(\mathbf{x}_j) \rangle = || \sum_{j=1}^{n} x_j \phi(\mathbf{x}_j) ||^2 \geq 0 \end{aligned} \tag{39}$$

Thus the kernel matrix (and its special case, the Gram matrix) are positive semi-definite. This property of matrices can be used to characterize the functions that are valid kernels. The following property is essential for a function to be a kernel function.

Finitely positive semi-definite functions Given an input space (not necessarily a vector space) $\mathcal{X}$ a symmetric function

$$\kappa : \mathcal{X} \times \mathcal{X} \mapsto \mathcal{R}$$

is said to be finitely positive semi-definite if the matrix formed by evaluating the function on all pairs of elements of any finite subset of $\mathcal{X}$ is positive semi-definite.

A function $\kappa(.,.) : \mathcal{X} \times \mathcal{X} \mapsto \mathcal{R}$ that is continuous or has a finite domain can be decomposed in to inner product of the images of its arguments under a feature mapping in to a Hilbert space $\mathcal{F}$,$\phi(.) : \mathcal{X} \mapsto \mathcal{F}$ i.e.

$$\kappa(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle \tag{40}$$

if and only if $\kappa(.,.)$ is finitely positive semi-definite on $\mathcal{X}$. The necessary condition is proved by the fact that the kernel matrix is always positive semi-definite. The sufficient condition is proved by explicit construction of the feature space $\mathcal{F}$and the map $\phi(.)$. The feature space is the following space of functions

$$\mathcal{F} = \{ \sum_{i=1}^{n} a_i \kappa(\mathbf{x}_i, .) | n \in \mathcal{N}, \mathbf{x}_i \in \mathcal{X} \} \tag{41}$$

It can be shown that $\mathcal{F}$ is a Hilbert space over $\mathcal{R}$ with the following definition for the inner product

$$\langle \sum_{i=1}^{n} a_i \kappa(\mathbf{x}_i, .), \sum_{j=1}^{m} b_j \kappa(\mathbf{y}_j, .) \rangle = \sum_{i=1}^{n} \sum_{j=1}^{m} a_i b_j \kappa(\mathbf{x}_i, \mathbf{y}_j) \tag{42}$$

It is easy to see the inner product so defined is real-valued since $\kappa(.,.)$ is real-valued. Let the arguments to the inner product be $f = \sum_{i=1}^{n} a_i \kappa(\mathbf{x}_i, .)$ and $g = \sum_{j=1}^{m} b_j \kappa(\mathbf{y}_j, .)$. The linearity and nonegativity of the inner product in each of its arguments can be seen from the following equations

$$\langle f, g \rangle = \sum_{i=1}^{n} \sum_{j=1}^{m} a_i b_j \kappa(\mathbf{x}_i, \mathbf{y}_j) = \sum_{j=1}^{m} b_j f(\mathbf{y}_j) = \sum_{i=1}^{n} a_i g(\mathbf{x}_i) \tag{43}$$

$$\langle f, f \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j \kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{a}^t \mathbf{K} \mathbf{a} \geq 0 \tag{44}$$

where $\mathbf{a} = [a_1 \quad \cdots \quad a_n]^t$ and $\mathbf{K}$ is the kernel matrix. We have used the fact kernel matrices are positive semi-definite in the second equation. The proofs for completeness and separability of $\mathcal{F}$ are more involved and are omitted for brevity. The specification of the map $\phi : \mathcal{X} \mapsto \mathcal{F}$ completes the decomposition of $\kappa(.,.)$ in to inner product of elements in $\mathcal{F}$. Since the requisite property of this map is $\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle = \kappa(\mathbf{x}, \mathbf{y})$ it can be seen that the map

$$\phi(\mathbf{x}) = \kappa(\mathbf{x}, .) \tag{45}$$

satisfies the requirement. Further it can be shown that this inner product enables us access to an the element $f = \sum_{i=1}^{n} a_i \kappa(\mathbf{x}_i, .)$ of $\mathcal{F}$ since

$$\langle f, \phi(\mathbf{x}) \rangle = \langle \sum_{i=1}^{n} a_i \kappa(\mathbf{x}_i, .), \kappa(\mathbf{x}, .) \rangle = \sum_{i=1}^{n} a_i \kappa(\mathbf{x}_i, \mathbf{x}) = f(\mathbf{x}) \tag{46}$$

Thus a function f in the function space $\mathcal{F}$ can be represented as linear function via the inner product (with itself) defined above. This is the so called *reproducing property* of the kernel function $\kappa(.,.)$. For this reason, the space $\mathcal{F}$ corresponding to the function $\kappa(.,.)$ is called the Reproducing Kernel Hilbert Space (RKHS). This property is a key characteristic of kernel function since any function $\kappa(.,.)$ that satisfies the reproducing property in a Hilbert Space of functions $\mathcal{F}$ satisfies the positive semi-definiteness property. Since $\langle f, \phi(\mathbf{x})\rangle = f(\mathbf{x})$ the function $\kappa(.,.)$ applied to an input pair $\mathbf{x}_i$ and $\mathbf{x}_j$ can be decomposed as follows:

$$\kappa(\mathbf{x}, \mathbf{y}) = \langle \kappa(\mathbf{x}, .), \kappa(\mathbf{y}, .)\rangle \tag{47}$$

Thus for the kernel matrix $\mathbf{K}$ any vector $\mathbf{a} = [a_1 \quad \cdots \quad a_n]^t$ the following result holds

$$\begin{aligned}
\mathbf{a}^t\mathbf{K}\mathbf{a} &= \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \langle \kappa(\mathbf{x}_i, .), \kappa(\mathbf{x}_j, .)\rangle \\
&= \langle \sum_{i=1}^{n} a_i \kappa(\mathbf{x}_i, .), \sum_{j=1}^{n} a_j \kappa(\mathbf{x}_j, .)\rangle \\
&= ||\sum_{i=1}^{n} a_i \kappa(\mathbf{x}_i, .)||^2 \geq 0
\end{aligned}$$

Since the above relation holds for any finite subset the function $\kappa(,)$ is finitely positive semi-definite. When the correspondence between the kernel function and associated RKHS needs emphasis the notation $\mathcal{F}_\kappa$ for the RKHS and $\langle ., . \rangle_{F_\kappa}$ for the inner product in $\mathcal{F}_k$ will be used.

Mercer Theorem Although the positive semi-definiteness characterizes kernel functions, the feature space that is used is a function space which is different from the traditional representation of the data i.e. feature vectors. The Mercer theorem enables the construction of a feature space whose elements are feature vectors and not functions. The theorem is stated below with out proof : Let $\mathcal{X}$ be a compact subset of $\mathcal{R}^n$. Let a continuous symmetric function κ be such that the integral operator

$$I_\kappa : L_2(\mathcal{X}) \mapsto L_2(\mathcal{X})$$

$$I_\kappa f = \int_{\mathcal{X}} \kappa(, \mathbf{x}) f(\mathbf{x}) d\mathbf{x}$$

is always positive for all $f \in L_2(\mathcal{X})$,

$$\int_{\mathcal{X}\times\mathcal{X}} \kappa(\mathbf{x}, \mathbf{y}) f(\mathbf{x}) f(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0 \tag{48}$$

Then $\kappa(\mathbf{x}, \mathbf{y})$ can be expanded in to a uniformly convergent series in terms of orthonormal function ψ_i,

$$\kappa(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{\infty} \lambda_i \psi_i(\mathbf{x}) \psi_i(\mathbf{y}) \tag{49}$$

The above expansion allows the construction of the map ϕ in an explicit feature vector form:

$$\phi(\mathbf{x}) = \left[\sqrt{\lambda_1}\psi_1(\mathbf{x}) \quad \cdots \quad \sqrt{\lambda_i}\psi_i(\mathbf{x}) \quad \cdots\right]^t \tag{50}$$

Other interpretations of the kernel function such as covariance function determined by a probabilistic function can be found in standard literature [9, 16, 18].

3.3 Kernel Design

One of the major challenges related to kernel algorithms is the choice of the kernel function. New kernel function can be constructed from known kernel functions by performing certain operations on them. For instance, given two kernel function $\kappa_1(.,.)$ and $\kappa_2(.,.)$ the following functions are all valid kernel functions

$$\kappa(\mathbf{x}, \mathbf{y}) = a\kappa_1(\mathbf{x}, \mathbf{y}), a \in \mathcal{R}^+$$

$$\kappa(\mathbf{x}, \mathbf{y}) = \kappa_1(\mathbf{x}, \mathbf{y}) + \kappa_2(\mathbf{x}, \mathbf{y})$$

$$\kappa(\mathbf{x},\mathbf{y}) = \kappa_1(\mathbf{x},\mathbf{y})\kappa_2(\mathbf{x},\mathbf{y})$$

$$\kappa(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} a_i \kappa_1(\mathbf{x},\mathbf{y})^i, n \in \mathcal{N}, a_i \geq 0, a_n \neq 0$$

$$\kappa(\mathbf{x},\mathbf{y}) = \exp(\kappa_1(\mathbf{x},\mathbf{y}))$$

However, it is not straight forward to incorporate prior knowledge about a problem in to kernel functions. In case of supervised problems, it is possible to define an *ideal* similarity matrix and try to align a kernel matrix to it. Given a two class problem with the target labels $\{\pm 1\}$, let $\mathbf{y}$ be the vector of labels if the input samples. The matrix $\mathbf{y}\mathbf{y}^t$ defines an ideal similarity matrix since all pairs of examples that have the same label give rise to a similarity score of 1 while those that have different labels given a score of -1. Using the *frobenius inner product*,

$$\langle \mathbf{A}, \mathbf{B} \rangle = trace(\mathbf{A}^t\mathbf{B})$$

the *alignment* or similarity between two matrices $\mathbf{K}_1$ and $\mathbf{K}_2$ can be defined as

$$\frac{\langle \mathbf{K}_1, \mathbf{K}_2 \rangle}{||\mathbf{K}_1||||\mathbf{K}_2||} \tag{51}$$

as a similarity measure between two matrices. For $\mathbf{K}_1 = \mathbf{K}$ and $\mathbf{K}_2 = \mathbf{y}\mathbf{y}^t$ this measure becomes

$$\frac{\mathbf{y}^t\mathbf{K}\mathbf{y}}{n||\mathbf{K}||} \tag{52}$$

where n is the number of samples. The similarity can be used to determine if a kernel matrix suits a task well. Alternate methods for learning the kernel matrix for tasks such as transduction and nonlinear component analysis have been proposed recently [21, 22, 23]. However, choosing the kernel function or learning the kernel matrix that is optimal for a general task is still an open problem.

4 Kernelization of Algorithms

4.1 The Data Matrix Representation

Although properties of kernel function are important, often the kernel matrix plays more important role in practice than the kernel function. Since kernel function allows access to the feature space only via the input samples, the pairwise inner products between elements of a finite input set $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$are the only information available on the geometry of the feature space. This information is embedded in the kernel matrix $\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$. Most kernel algorithms often work with this matrix rather than the kernel function itself. The representation of sets of feature vectors as data matrices greatly simplifies the process of kernelization and to easily *see* kernel matrices or elements of it in equations. Let the images of input samples under the map ϕ form the columns of the matrix $\mathbf{X}$

$$\mathbf{X} = \begin{bmatrix} \phi(\mathbf{x}_1) & \phi(\mathbf{x}_2) & \cdots & \phi(\mathbf{x}_n) \end{bmatrix}$$

The key relationship to note is that $\mathbf{K} = \mathbf{X}^t\mathbf{X}$. Several operations such as summation of the columns, selection of columns etc. can be represented as matrix multiplications. For instance, the solution vector $\mathbf{w}$ in section 2 can be expressed as :

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i) = \begin{bmatrix} \phi(\mathbf{x}_1) & \phi(\mathbf{x}_2) & \cdots & \phi(\mathbf{x}_n) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} = \mathbf{X}\boldsymbol{\alpha} \tag{53}$$

where $\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_n \end{bmatrix}^t$. To see how data matrix representation makes kernelization easy, consider centering of the data in the feature space. The challenge is to remove the sample mean $\boldsymbol{\phi}_C = \frac{1}{n}\sum_{i=1}^{n}\phi(\mathbf{x}_i)$ from each column of the data matrix and then work with the resulting data. The sample mean can be written in terms of the data matrix form by setting $\alpha_i = \frac{1}{n}$ in the above equation. Since there is no way of accessing the sample mean, we must construct an equivalent centered kernel $\hat{\mathbf{K}}$. It turns out that this can be done by operations on the original $\mathbf{K}$ alone. First, the centered data matrix $\hat{\mathbf{X}}$ can be expressed as

$$\begin{aligned} \mathbf{X}_C &= \begin{bmatrix} \phi(\mathbf{x}_1) - \boldsymbol{\phi}_C & \phi(\mathbf{x}_2) - \boldsymbol{\phi}_C & \cdots & \phi(\mathbf{x}_n) - \boldsymbol{\phi}_C \end{bmatrix} \\ &= \begin{bmatrix} \phi(\mathbf{x}_1) & \phi(\mathbf{x}_2) & \cdots & \phi(\mathbf{x}_n) \end{bmatrix} - \begin{bmatrix} \boldsymbol{\phi}_C & \cdots & \boldsymbol{\phi}_C \end{bmatrix} \\ &= \mathbf{X} - \begin{bmatrix} \boldsymbol{\phi}_C \end{bmatrix} \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix} \\ &= \mathbf{X} - \mathbf{X}\begin{bmatrix} \frac{1}{n} & \frac{1}{n} & \cdots & \frac{1}{n} \end{bmatrix}^t \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix} = \mathbf{X} - \mathbf{X}\mathbf{1}_{\frac{1}{n}} \end{aligned} \tag{54}$$

where $\mathbf{1}_a$ is a matrix with all elements equal to a (we do not explicitly show the size in the notation since it is evident from the context. Whenever the size needs emphasis, such matrices will be distinguished explicitly). Thus the centered kernel matrix can be written as

$$\begin{aligned} \mathbf{K}_C &= \mathbf{X}_C^t\mathbf{X}_C = (\mathbf{X} - \mathbf{X}\mathbf{1}_{\frac{1}{n}})^t(\mathbf{X} - \mathbf{X}\mathbf{1}_{\frac{1}{n}}) \\ &= \mathbf{X}^t\mathbf{X} - \mathbf{X}^t\mathbf{X}\mathbf{1}_{\frac{1}{n}} - \mathbf{1}_{\frac{1}{n}}^t\mathbf{X}^t\mathbf{X} + \mathbf{1}_{\frac{1}{n}}^t\mathbf{X}^t\mathbf{X}\mathbf{1}_{\frac{1}{n}} \\ \Rightarrow \mathbf{K}_C &= \mathbf{K} - \mathbf{K}\mathbf{1}_{\frac{1}{n}} - \mathbf{1}_{\frac{1}{n}}\mathbf{K} + \mathbf{1}_{\frac{1}{n}}\mathbf{K}\mathbf{1}_{\frac{1}{n}} \end{aligned} \tag{55}$$

which can be computed from the original kernel matrix $\mathbf{K}$ alone. To use relationships that are inferred by a kernelized algorithm on unseen data, $\mathcal{X}_t = \{\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \cdots, \hat{\mathbf{x}}_m\}$, we need the pairwise inner product information between the unseen samples and the samples from which the algorithm inferred the relationship. This is encoded is an $m \times n$ matrix $\hat{\mathbf{K}}_{ij} = \langle\phi(\hat{\mathbf{x}}_i), \phi(\mathbf{x}_j)\rangle = \kappa(\hat{\mathbf{x}}_i, \mathbf{x}_j)$. Once again, the data matrix notation simplifies equations :

$$\hat{\mathbf{K}} = \begin{bmatrix}\phi(\hat{\mathbf{x}}_1) & \phi(\hat{\mathbf{x}}_2) & \cdots & \phi(\hat{\mathbf{x}}_n)\end{bmatrix}^t \mathbf{X} = \hat{\mathbf{X}}^t\mathbf{X} \tag{56}$$

It is easy to show that to center the $\hat{\mathbf{X}}$ with respect to the sample mean ϕ_C one needs to modify the kernel matrix $\hat{\mathbf{K}}$ as follows

$$\hat{\mathbf{K}}_C = \hat{\mathbf{K}} - \hat{\mathbf{K}}\mathbf{1}_{\frac{1}{n}} - \mathbf{1}'_{\frac{1}{n}}\mathbf{K} + \mathbf{1}'_{\frac{1}{n}}\mathbf{K}\mathbf{1}_{\frac{1}{n}} \tag{57}$$

where $\mathbf{1}'_{\frac{1}{n}}$ is an $m \times n$ matrix with all entries equal to $\frac{1}{n}$. Another important advantage with the data matrix notation is the connection to the sample covariance matrix which encodes the second order statistics of the input set of samples. The covariance matrix $\mathbf{C}$ in the feature space is $N \times N$ matrix with $\mathbf{C}_{ij} = \sum_{k=1}^{n}(\phi(\mathbf{x}_k)_i - \boldsymbol{\phi}_{Ci})(\phi(\mathbf{x}_k)_j - \boldsymbol{\phi}_{Cj})$, where N is dimension of the feature space. Using the data matrix notation the sample covariance matrix can be represented as :

$$\mathbf{C} = \mathbf{X}_C\mathbf{X}_C^t \tag{58}$$

The eigen vectors of the sample covariance matrix are of great practical importance for many tasks involving compression, dimensionality reduction etc. However, it is not possible to explicitly compute the matrix $\mathbf{C}$ since $\phi()$ typically maps the input samples to a very high dimensional space. Even if $\phi()$ were the identity map the size of $\mathbf{C}$ is $N \times N$,where N is the dimensionality of the data, making it impossible to work with such matrices. The following property gives a way to workaround this problem :

$$\begin{aligned} \mathbf{A}\mathbf{A}^t\mathbf{x} &= \lambda\mathbf{x} \\ \Rightarrow \mathbf{A}^t\mathbf{A}\mathbf{A}^t\mathbf{x} &= \mathbf{A}^t\lambda\mathbf{x} \\ \Rightarrow (\mathbf{A}^t\mathbf{A})(\mathbf{A}^t\mathbf{x}) &= \lambda(\mathbf{A}^t\mathbf{x}) \end{aligned} \tag{59}$$

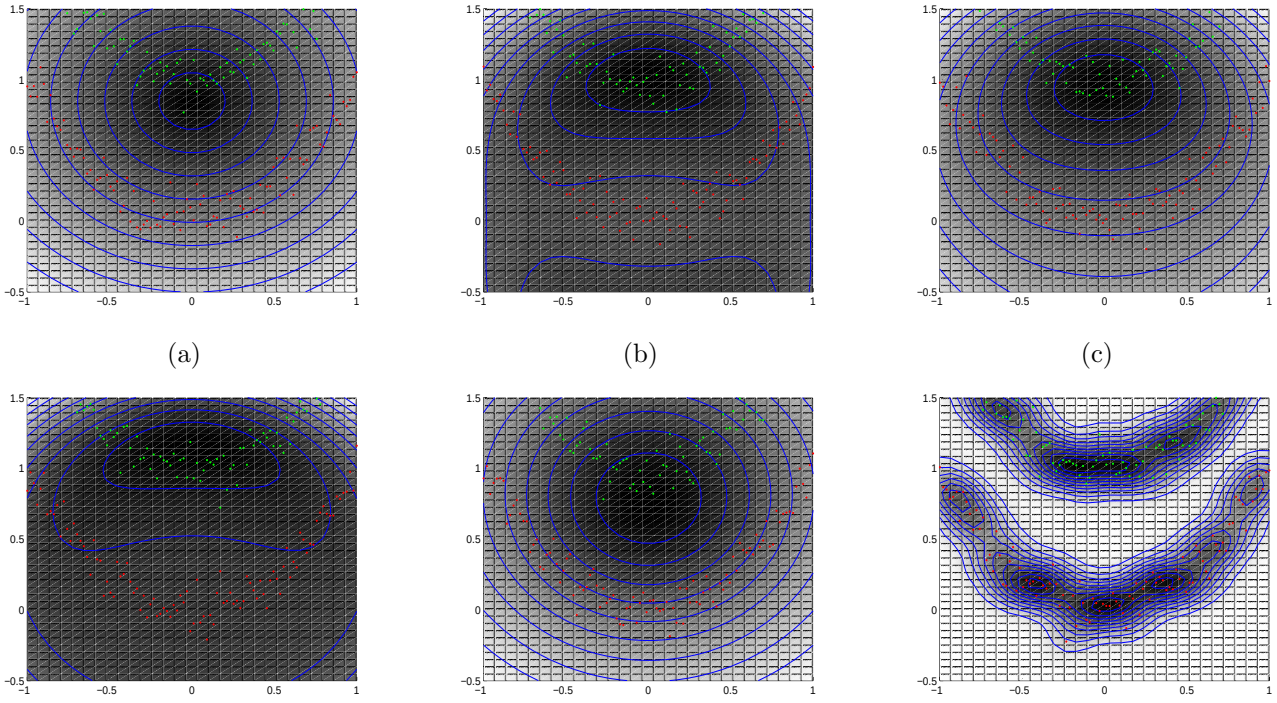
The above equations show that the eigen values of $\mathbf{A}\mathbf{A}^t$ and $\mathbf{A}^t\mathbf{A}$ are same. Moreover $\mathbf{v}$ is an eigen vector of $\mathbf{A}\mathbf{A}^t$ only if $\mathbf{A}^t\mathbf{v}$ is an eigen vector $\mathbf{A}^t\mathbf{A}$. Thus, the eigen values of the matrix $\mathbf{K}_C$ are same as that of the matrix $\mathbf{C}$. However, the eigen vectors are still not accessible since $\mathbf{X}_C^t$ is not directly accessible. The dual representation helps in overcoming this problem. If each eigen vector $\mathbf{v}$ of the matrix $\mathbf{C}$ lies in span of columns of $\mathbf{X}_C$ i.e $\mathbf{v} = \mathbf{X}_C\boldsymbol{\alpha}$, then $\mathbf{X}_C^t\mathbf{v} = \mathbf{X}_C^t\mathbf{X}_C\boldsymbol{\alpha} = \mathbf{K}_C\boldsymbol{\alpha}$ is an eigen vector of $\mathbf{K}_C$. This fact is the foundation for kernel principal component analysis(KPCA) [13] which is popular for nonlinear component analysis. In the remainder of this section examples kernelized versions of several linear algorithms are presented. In many cases the data matrix representation simplifies the task and is useful when kernelizing new algorithms.

4.2 Basic Algorithms : Distance, Variance, Normalization...

Distances and Normalization Distance between two points in the feature space is one of the simplest measures used for a variety of tasks. Distance is defined in terms of inner product and hence the kernelized version of distance finding is straightforward :

$$\begin{aligned} d(\phi(\mathbf{x}), \phi(\mathbf{y})) &= \sqrt{||\phi(\mathbf{x}) - \phi(\mathbf{y})||^2} \\ &= [\langle\phi(\mathbf{x}) - \phi(\mathbf{y}), \phi(\mathbf{x}) - \phi(\mathbf{y})\rangle]^{\frac{1}{2}} \\ &= [\langle\phi(\mathbf{x}), \phi(\mathbf{x})\rangle + \langle\phi(\mathbf{y}), \phi(\mathbf{y})\rangle - 2\langle\phi(\mathbf{x}), \phi(\mathbf{y})\rangle]^{\frac{1}{2}} \\ &= \kappa(\mathbf{x}, \mathbf{x}) + \kappa(\mathbf{y}, \mathbf{y}) - 2\kappa(\mathbf{x}, \mathbf{y}) \end{aligned} \tag{60}$$

The ability to compute distance enables us to kernelize one of the simplest algorithms for anomaly detection i.e. finding if a new sample $\mathbf{x}_t$ belongs to the same distribution as the training set $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$. A simple (albeit



(a) (b) (c)

(d) (e) (f)

Figure 4: Visualizing distance in the feature space gives an idea of the the geometry of the features. The distance finding algorithm is used to calculate the distance of the points in the input space and the sample mean of two data sets (one in green and the other in red) each distributed in parabolic shapes. The contours are curves along which the distance to the mean is constant Results : (a) linear kernel, (b) and (c) with $\kappa(\mathbf{x},\mathbf{y}) = \langle \mathbf{x},\mathbf{y}\rangle^2$ and $\kappa(\mathbf{x},\mathbf{y}) = (1+\langle \mathbf{x},\mathbf{y}\rangle)^2$ respectively, (d) with $\kappa(\mathbf{x},\mathbf{y}) = \langle \mathbf{x},\mathbf{y}\rangle^3$. (e) and (f) with $\kappa(\mathbf{x},\mathbf{y}) = \exp(-\frac{||\mathbf{x}-\mathbf{y}||^2}{2\sigma^2})$ with $\sigma = 1, 0.1$ respectively. It can be seen that the isosurfaces in each feature space define complex curves in the input space. Also, in (f), note the complexity of the feature space corresponding to Gaussian kernel function where points from both the distributions are close to the mean.

naive) way of doing this is to approximate the distribution with a normal distribution with unit covariance (i.e spherical) $\mathcal{N}(\mu, \mathbf{I})$ where μ is the sample mean $\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$. The distance from the μ then becomes the criterion to decide if $\mathbf{x}_t$ is an anomaly or not. The equations are all again simplified using the data matrix notation. The sample mean is $\mathbf{X}\mathbf{1}_{\frac{1}{n}}$ and the distance d from $\phi(\mathbf{x}_t)$ to the sample mean is given by

$$\begin{aligned} d^2 &= (\phi(\mathbf{x}_t) - \mathbf{X}\mathbf{1}_{\frac{1}{n}})^t(\phi(\mathbf{x}_t) - \mathbf{X}\mathbf{1}_{\frac{1}{n}}) \\ &= \phi(\mathbf{x}_t)^t\phi(\mathbf{x}_t) - 2\phi(\mathbf{x}_t)^t\mathbf{X}\mathbf{1}_{\frac{1}{n}} + \mathbf{1}_{\frac{1}{n}}^t\mathbf{X}^t\mathbf{X}\mathbf{1}_{\frac{1}{n}} \\ &= \kappa(\mathbf{x}_t,\mathbf{x}_t) - 2\mathbf{k}_t^t\mathbf{1}_{\frac{1}{n}} + \mathbf{1}_{\frac{1}{n}}^t\mathbf{K}\mathbf{1}_{\frac{1}{n}} \end{aligned} \tag{61}$$

where $\mathbf{k}_t = \begin{bmatrix}\kappa(\mathbf{x}_1,\mathbf{x}_t) & \kappa(\mathbf{x}_2,\mathbf{x}_t) & \cdots & \kappa(\mathbf{x}_n,\mathbf{x}_t)\end{bmatrix}^t$ is the vector containing the inner products of $\phi(\mathbf{x}_t)$ with the images of training samples under $\phi(.)$. It should be noted matrices of the form $\mathbf{A}^t\mathbf{B}$, where $\mathbf{A}$ and $\mathbf{B}$ are data matrices, are accessible in the feature space. Figure 4 show how the distance in the feature space with various kernels looks like in the input space.

Like centering of the data, normalization of data can also be done by operations on the kernel matrix alone. Given a point $\phi(\mathbf{x})$ in the feature space, the normalized vector is

$$\hat{\phi}(\mathbf{x}) = \frac{\phi(\mathbf{x})}{||\phi(\mathbf{x})||}$$

Thus the new inner product in the feature space becomes

$$\begin{aligned}\hat{\kappa}(\mathbf{x},\mathbf{y}) &= \langle\hat{\phi}(\mathbf{x}),\hat{\phi}(\mathbf{y})\rangle \\ &= \langle\frac{\phi(\mathbf{x})}{||\phi(\mathbf{x})||},\frac{\phi(\mathbf{y})}{||\phi(\mathbf{y})||}\rangle \\ &= \frac{\langle\phi(\mathbf{x}),\phi(\mathbf{y})\rangle}{||\phi(\mathbf{x})||||\phi(\mathbf{y})||} \\ &= \frac{\kappa(\mathbf{x},\mathbf{y})}{\sqrt{\kappa(\mathbf{x},\mathbf{x})\kappa(\mathbf{y},\mathbf{y})}}\end{aligned} \tag{62}$$

The original kernel matrix $\mathbf{K}$ can modified to obtain a new kernel matrix $\hat{\mathbf{K}}$ with the above inner product. If $\mathbf{D}$ is a diagonal matrix such that $\mathbf{D}_{ii} = \frac{1}{\sqrt{\mathbf{K}_{ii}}}$ then $\hat{\mathbf{K}}$ can be expressed as a simple matrix multiplication :

$$\hat{\mathbf{K}} = \mathbf{DKD} \tag{63}$$

Variance Apart from distance between two points and norms of vectors, the projection of a vector $\mathbf{x}$ on to a line $\mathbf{w}$ and variance of a collection of points $\{\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_n\}$ along the line $\mathbf{w}$ are some of the important quantities in analysis of data. These quantities can be computed in the feature space provided the line in the feature space lies in the span of images of a known set of points under the mapping ϕ. Thus using the data matrix notation the line is $\mathbf{w} = \mathbf{X}\alpha$ and the sample mean is $\mu = \mathbf{X}\mathbf{1}_{\frac{1}{n}}$. The projection of a vector $\mathbf{x}$ on to $\mathbf{w}$ is given by :

$$P_{\mathbf{w}}(\mathbf{x}) = \frac{\langle\mathbf{x},\mathbf{w}\rangle}{||\mathbf{w}||}\mathbf{w} \tag{64}$$

Kernelizing this using the data matrix notation, the projection of $\phi(\mathbf{x}_t)$ on to $\mathbf{w} = \sum_{i=1}^{n}\phi(\mathbf{x}_i)$ is:

$$\frac{\langle\mathbf{x}_t,\mathbf{w}\rangle}{||\mathbf{w}||}\mathbf{w} = \frac{\phi(\mathbf{x}_t)^t\mathbf{X}\boldsymbol{\alpha}}{\boldsymbol{\alpha}^t\mathbf{X}^t\mathbf{X}\boldsymbol{\alpha}} = \frac{\mathbf{k}_t^t\boldsymbol{\alpha}}{\boldsymbol{\alpha}^t\mathbf{K}\boldsymbol{\alpha}} \tag{65}$$

Many linear algorithms search for a direction along which the variance of input samples meets certain criterion. The variance of the input set along the line $\mathbf{w}$ can be computed by combining the results above. For simplicity, we assume $\mathbf{w}$ is normalized in feature space such that $||\mathbf{w}|| = \sqrt{\boldsymbol{\alpha}^t\mathbf{K}\boldsymbol{\alpha}} = 1$. The variance can be expressed as below :

$$\begin{aligned}\sigma_{\mathbf{w}}^2 &= \frac{1}{n}\sum_{i=1}^{n}(\phi(\mathbf{x}_i)^t\mathbf{w} - \boldsymbol{\phi}_C^t\mathbf{w})^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}(\phi(\mathbf{x}_i)^t\mathbf{X}\boldsymbol{\alpha} - \mathbf{1}_{\frac{1}{n}}^t\mathbf{X}^t\mathbf{X}\boldsymbol{\alpha})^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}(\mathbf{k}_i^t\boldsymbol{\alpha} - \mathbf{1}_{\frac{1}{n}}^t\mathbf{K}\boldsymbol{\alpha})^2\end{aligned} \tag{66}$$

An alternate form that is more suitable for solving optimization problems can be obtained by noting the following identities

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}(\phi(\mathbf{x}_i)^t\mathbf{w} - \boldsymbol{\phi}_C^t\mathbf{w})^2 &= \frac{1}{n}\sum_{i=1}^{n}(\phi(\mathbf{x}_i)^t\mathbf{w})^2 - (\boldsymbol{\phi}_C^t\mathbf{w})^2 \\ \sum_{i=1}^{n}(\mathbf{k}_i^t\boldsymbol{\alpha})^2 &= \boldsymbol{\alpha}^t\mathbf{K}^2\boldsymbol{\alpha}\end{aligned} \tag{67}$$

Using these identities the variance can be expressed more compactly as

$$\sigma_{\mathbf{w}}^2 = \frac{1}{n}\boldsymbol{\alpha}^t\mathbf{K}^2\boldsymbol{\alpha} - (\mathbf{1}_{\frac{1}{n}}^t\mathbf{K}\boldsymbol{\alpha})^2 \tag{68}$$

Orthonormalization Many pattern analysis algorithms, given an input set $\{\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_n\}$, construct an orthonormal basis $\mathcal{Q} = \{\mathbf{q}_1,\cdots,\mathbf{q}_d\}$ that spans the input set. For simplicity we assume that the given input samples are linearly independent so that $d = n$. The popular Gram-Schmidt orthonormalization procedure builds the basis inductively as follows:

- The first basis vector is obtained by normalizing $\mathbf{x}_1$ i.e.,

$$\mathbf{q}_1 = \frac{\mathbf{x}_1}{||\mathbf{x}_1||}$$

- The ith basis vector $\mathbf{q}_i$ from $\mathbf{x}_i$ by first subtracting the projections of $\mathbf{x}_i$ on to $\mathbf{q}_1, \cdots, \mathbf{q}_{i-1}$ and normalizing the residual i.e.,

$$\mathbf{q}_i = \frac{\mathbf{x}_i - \sum_{j=1}^{i-1} \langle \mathbf{q}_j, \mathbf{x}_j \rangle \mathbf{q}_j}{||\mathbf{x}_i - \sum_{j=1}^{i-1} \langle \mathbf{q}_j, \mathbf{x}_j \rangle \mathbf{q}_j||}$$

Building an orthonormal basis in the feature space using the above algorithm is not possible since the images $\phi(\mathbf{x}_i)$ are not accessible directly. However, since the basis set is in the span of the input set of samples each basis vector $\mathbf{q}_i$ in the feature space can be expressed as $\mathbf{q}_i = \mathbf{X}\boldsymbol{\alpha}^i$. The problem of basis construction now can reposed as estimation of the vectors $\boldsymbol{\alpha}^1, \cdots, \boldsymbol{\alpha}^n$. Since $\phi(\mathbf{x}_i)$ can be represented as $\mathbf{X}\mathbf{e}_i$ where $\mathbf{e}_i$ is the ith canonical vector we have the following procedure for orthonormal basis construction:

- The first basis vector can be written as

$$\mathbf{q}_1 = \frac{\mathbf{X}\mathbf{e}_1}{\sqrt{\mathbf{e}_1^t \mathbf{X}^t \mathbf{X} \mathbf{e}_1}} = \mathbf{X} \frac{\mathbf{e}_1}{\sqrt{\mathbf{e}_1^t \mathbf{K} \mathbf{e}_1}}$$

  and hence the dual coefficients of the first basis vector are given by

$$\boldsymbol{\alpha}^1 = \frac{\mathbf{e}_1}{\sqrt{\mathbf{e}_1^t \mathbf{K} \mathbf{e}_1}} \tag{69}$$

- The unnormalized ith basis vector is given by

$$\begin{aligned} \hat{\mathbf{q}}_i &= \mathbf{X}\mathbf{e}_i - \sum_{j=1}^{i-1} (\mathbf{X}\mathbf{e}_i)^t (\mathbf{X}\boldsymbol{\alpha}^j) \mathbf{X}\boldsymbol{\alpha}^j \\ &= \mathbf{X}\mathbf{e}_i - \sum_{j=1}^{i-1} (\mathbf{e}_i^t \mathbf{X}^t \mathbf{X}\boldsymbol{\alpha}^j) \mathbf{X}\boldsymbol{\alpha}^j \\ &= \mathbf{X}\mathbf{e}_i - \mathbf{X} \sum_{j=1}^{i-1} (\mathbf{e}_i^t \mathbf{K}\boldsymbol{\alpha}^j) \boldsymbol{\alpha}^j \end{aligned} \tag{70}$$

  Thus the dual coefficients of the unnormalized ith basis vector are given by

$$\hat{\boldsymbol{\alpha}}^i = \mathbf{e}_i - \sum_{j=1}^{i-1} (\mathbf{e}_i^t \mathbf{K}\boldsymbol{\alpha}^j) \boldsymbol{\alpha}^j \tag{71}$$

  The normalized coefficients $\boldsymbol{\alpha}^i$ can be done by setting $||\hat{\mathbf{q}}_i|| = \sqrt{\hat{\boldsymbol{\alpha}}^{it} \mathbf{X}^t \mathbf{X} \hat{\boldsymbol{\alpha}}^i} = 1$ resulting in

$$\boldsymbol{\alpha}^i = \frac{1}{\sqrt{\hat{\boldsymbol{\alpha}}^{it} \mathbf{X}^t \mathbf{X} \hat{\boldsymbol{\alpha}}^i}} \hat{\boldsymbol{\alpha}}^i \tag{72}$$

The above procedure gives the coefficients of the basis vectors in terms of the original. To use the the basis for computation we should be able to project a new sample $\phi(\mathbf{x}_t)$ on to the basis vectors. This is straightforward since $\langle \phi(\mathbf{x}_t), \mathbf{q}_i \rangle = \mathbf{k}_t^t \boldsymbol{\alpha}^i$.

The methods describe above help us in analysis of the data in the feature space using measures directly related to inner product such as distances and projections. However deeper analysis of data is required for tasks such as compression data without losing much of the inherent structure (compression and dimensionality reduction) and selecting features that best suit a task e.g classification (feature selection). Examples of kernelization of algorithms which perform such tasks are presented in the following sections.

5 Feature Extraction, Feature Selection and Modeling

5.1 Principal Component Analysis

The goal principal component analysis (PCA) is the to find the directions along which the variance is maximum. The reason for seeking these direction is that the directions with low variance do not provide any information about the data and such directions may be discarded by projecting the data on the more informative directions (called principal components). It can be shown that these directions are the eigen vectors corresponding to larger eigen values of the covariance matrix. The kernelization of this method was proposed by Scholkopf *et.al* [13] As shown in section 4, the dual coefficients $\boldsymbol{\alpha}$ of these eigen vector(in the feature space) satisfy the property that $\mathbf{K}_C\boldsymbol{\alpha}$ is an eigen vector of $\boldsymbol{\alpha}$. Thus we have

$$\begin{aligned} \mathbf{K}_C(\mathbf{K}_C\boldsymbol{\alpha}) &= \lambda\mathbf{K}_C\boldsymbol{\alpha} \\ \mathbf{K}_C\boldsymbol{\alpha} &= \lambda\boldsymbol{\alpha} \end{aligned} \tag{73}$$

The above equation indicates that the eigen vectors corresponding to the maximum eigen values of the centered kernel matrix give the dual coefficients of the principal components in the feature space. This result can be obtained by directly optimizing the variance along a line $\mathbf{w} = \mathbf{X}_C\boldsymbol{\alpha}$. Assuming that the data is centered the variance ,as shown in section 4, can be expressed as

$$\sigma^2_{\boldsymbol{\alpha}} = \frac{1}{n}\boldsymbol{\alpha}^t\mathbf{K}_C^2\boldsymbol{\alpha} \tag{74}$$

Maximizing the above equation subject to the constraint $||\mathbf{w}||^2 = \boldsymbol{\alpha}^t\mathbf{K}_C\boldsymbol{\alpha} = 1$ using lagrangian multiplier λ_1 results in :

$$\begin{aligned} \frac{\partial(\sigma^2_{\boldsymbol{\alpha}} - \lambda_1(\boldsymbol{\alpha}^t\mathbf{K}_C\boldsymbol{\alpha} - 1))}{\partial\boldsymbol{\alpha}} &= \frac{2}{n}\mathbf{K}_C^2\boldsymbol{\alpha} - 2\lambda_1\mathbf{K}_C\boldsymbol{\alpha} = 0 \\ \Rightarrow \mathbf{K}_C\boldsymbol{\alpha} &= n\lambda_1\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha} \end{aligned} \tag{75}$$

Since the eigen vectors need to be normalized in the feature space, the eigen vectors corresponding to the top k eigen values $\lambda_1, \cdots, \lambda_k$ of $\mathbf{K}_C$, $\{\boldsymbol{\alpha}^1, \cdots, \boldsymbol{\alpha}^k\}$,need to be normalized such that $||\mathbf{X}\boldsymbol{\alpha}^i||^2 = 1$. Since

$$||\mathbf{X}\boldsymbol{\alpha}^i||^2 = \boldsymbol{\alpha}^{it}\mathbf{K}_C\boldsymbol{\alpha}^i = \boldsymbol{\alpha}^{it}\lambda_i\boldsymbol{\alpha}^i = \lambda_i||\boldsymbol{\alpha}||^2 = \lambda_i$$

the normalized vectors $\{\lambda_1^{-1/2}\boldsymbol{\alpha}^1, \cdots, \lambda_k^{-1/2}\boldsymbol{\alpha}^k\}$ give the final dual coefficient of the principal components in the feature space (called kernel principal components). Kernel principal component analysis has been extensively used for extraction of nonlinear feature descriptors [24, 25]. Figure 5 demonstrates how kernel principal components aid in description of nonlinear relationships among data.

5.2 Linear Discriminant Analysis

Principal Component Analysis aids in extracting description that described the data well. However, not all such features are useful for every task. For instance, in the case of face recognition, given three persons $A, B and C$, the features that aid distinguishing between A and B are in general different from those that discriminate between B and C. Hence, the selection of features that are optimal for a given task is an important problem in pattern analysis. Linear Discriminant Analysis is a method to find best direction that aid in distinguishing between different classes. The Fisher Discriminant Analysis is one such method the kernel variant of which was proposed by Mika et.al [14]. The current discussion addresses the case of two classes. The key idea is that sample mean and variance of each class describe the distribution of that class. Let $\boldsymbol{\mu}_+$ and $\boldsymbol{\mu}_-$ be the sample means of the two classes. Then the $\langle\boldsymbol{\mu}_+, \mathbf{w}\rangle - \langle\boldsymbol{\mu}_-, \mathbf{w}\rangle = \mu^+_{\mathbf{w}} - \mu^-_{\mathbf{w}}$ is a measure of how far much distributions differ along the direction $\mathbf{w}$. Discriminating between the classes also would require each the samples of each class to be tightly clustered around the mean. The variance along $\mathbf{w}$ for each class can be used as measure for this fit. There are many objective function involving these terms that can be optimized to obtain the optimal direction $\mathbf{w}$. A popular objective function used frequently is

$$J(\mathbf{w}) = \frac{(\mu^+_{\mathbf{w}} - \mu^-_{\mathbf{w}})^2}{{\sigma^+_{\mathbf{w}}}^2 + {\sigma^-_{\mathbf{w}}}^2} \tag{76}$$

To express the above equation in the dual form using the data matrix notation the following additional symbols are defined. Let n be the number of samples and $\mathbf{X}$ be the data matrix. The number of samples belonging to

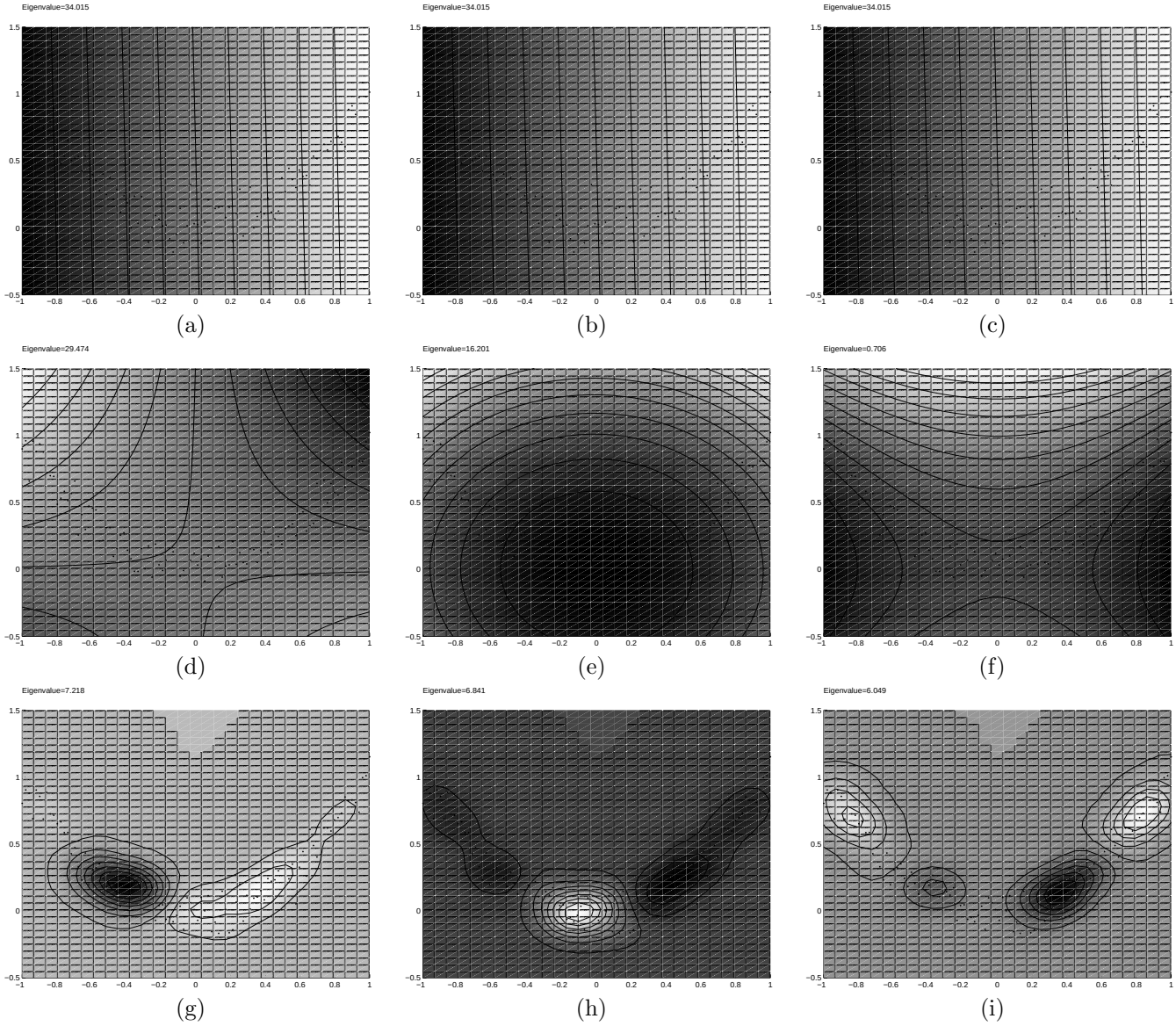


Figure 5: Nonlinear component analysis using kernel function. The top three eigen vectors of the kernel matrix were used to derive the components shown with various kernels. (a),(b),(c) : linear kernel, (d),(e),(f) : quadratic kernel, (g),(h),(i) : Gaussian radial basis kernel with $\sigma = 0.1$. In all cases the third component has very low significance compared to the other two. This is expected since the intrinsic dimension of the input set it 2. However the nonlinear components describe the data better.

the two classes be n^+ and n^-. The vector $\mathbf{j}_+$ is n-vector with $j_{+i} = \frac{1}{n^+}$ if ith column of $\mathbf{X}$ is in the positive class and $j_{+i} = 0$ otherwise. The vector $\mathbf{j}_-$ is defined similarly. Thus the sample means of the positive and negative classes are :

$$\boldsymbol{\mu}_C^+ = \mathbf{X}\mathbf{j}_+$$

$$\boldsymbol{\mu}_C^- = \mathbf{X}\mathbf{j}_-$$

The positive and negative data matrices $\mathbf{X}_+$ and $\mathbf{X}_-$ are formed by taking the input samples belonging to the respective class. The matrix $\mathbf{I}_+$ is a selector matrix whose ith column is the ith canonical vector $\mathbf{e}_i$ if the ith sample belongs to the positive class and $\mathbf{0}$ otherwise. Thus the data matrices are related by :

$$\mathbf{X}_+ = \mathbf{X}\mathbf{I}_+$$

$$\mathbf{X}_- = \mathbf{X}\mathbf{I}_-$$

Similar the class specific kernel matrices also can be defined as $\mathbf{K}_+ = \mathbf{X}_+^t\mathbf{X}_+$ and $\mathbf{K}_- = \mathbf{X}_-^t\mathbf{X}_-$. Thus the dual objective function can be written as

$$\begin{aligned} J(\boldsymbol{\alpha}) &= \frac{(\boldsymbol{\alpha}^t\mathbf{X}^t\mathbf{X}\mathbf{j}_+ - \boldsymbol{\alpha}^t\mathbf{X}^t\mathbf{X}\mathbf{j}_-)^2}{\frac{1}{n^+}\boldsymbol{\alpha}^t\mathbf{K}_+^2\boldsymbol{\alpha} - (\mathbf{1}^t_{\frac{1}{n^+}}\mathbf{K}_+\boldsymbol{\alpha})^2 + \frac{1}{n^-}\boldsymbol{\alpha}^t\mathbf{K}_-^2\boldsymbol{\alpha} - (\mathbf{1}^t_{\frac{1}{n^-}}\mathbf{K}_-\boldsymbol{\alpha})^2} \\ &= \frac{\boldsymbol{\alpha}^t(\mathbf{K}\mathbf{j}_+\mathbf{j}_+^t\mathbf{K} + \mathbf{K}\mathbf{j}_-\mathbf{j}_-^t\mathbf{K} - 2\mathbf{K}\mathbf{j}_+\mathbf{j}_-^t\mathbf{K})\boldsymbol{\alpha}}{\boldsymbol{\alpha}^t(\frac{1}{n^+}\mathbf{K}_+^2 + \frac{1}{n^-}\mathbf{K}_-^2 - \mathbf{K}_+\mathbf{1}_{\frac{1}{n^+}}\mathbf{1}^t_{\frac{1}{n^+}}\mathbf{K}_+ - \mathbf{K}_-\mathbf{1}_{\frac{1}{n^-}}\mathbf{1}^t_{\frac{1}{n^-}}\mathbf{K}_-)\boldsymbol{\alpha}} \\ &= \frac{\boldsymbol{\alpha}^t\mathbf{A}\boldsymbol{\alpha}}{\boldsymbol{\alpha}^t\mathbf{B}\boldsymbol{\alpha}} \end{aligned} \tag{77}$$

where $\mathbf{A}$ and $\mathbf{B}$ can be computed from the kernel matrix alone. Optimizing this function subject to the constraint $||\mathbf{w}||^2 = \boldsymbol{\alpha}^t\mathbf{K}\boldsymbol{\alpha} = 1$ leads to the following generalized eigen vector problem

$$\mathbf{A}\boldsymbol{\alpha} = \lambda(\mathbf{B} + \mathbf{K})\boldsymbol{\alpha} \tag{78}$$

Thus the most discriminative directions can be found by finding the generalized eigen vectors corresponding the maximum generalized eigen values of the matrices above. The projection of test sample and other computations are similar to the case of kernel PCA and kernel perceptron. Figure 6 shows how kernel fisher discriminants extract nonlinear relationships in the input space.

6 Classification and Support Vector Machine

6.1 Optimal Separating Hyperplanes

While feature extraction and feature selection help us in extracting meaningful and relevant descriptions of data, classification is an altogether different problem where the goal is to learn boundaries that separate different classes of data. This is a more challenging problem since such classification rules which are inferred from a finite set of data are expected to classify unseen examples with high accuracy. The input set from which the classification function is learned is called the *training set*. Since this set is finite, it is not guaranteed that a given classification function would commit errors with the same frequency on unseen examples as it did on this set. However statistical learning theory [20] allows us to relate the error rate on the training set and the true error rate i.e. the expected error on unseen examples. This relation is the foundation for support vector algorithm. A complete review of learning theory is beyond the scope of this report and the additional details can be found in Vapnik [20]. The central result that aids in understanding the development of the support vector algorithm is reviewed below.

Consider a input set $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$, where $y_i \in \{\pm 1\}$, with each example drawn independently and distributed identically (*iid*) according to a distribution $P(\mathbf{x}, y)$. Let f be a function parametrized by α, the task of which is to learn the relationship between $\mathbf{x}_i$ and y_i. Thus expected error rate of f is :

$$R(\alpha) = \int \frac{1}{2}|f_\alpha(\mathbf{x}) - y| dP(\mathbf{x}, y) \tag{79}$$

However $R(\alpha)$ is not computable we have only finite amount of data. What can be computed is the empirical error i.e the error rate of f on the training set :

$$R_{emp}(\alpha) = \sum_{i=1}^{n} \frac{1}{2n}|f_\alpha(\mathbf{x}_i) - y_i| \tag{80}$$

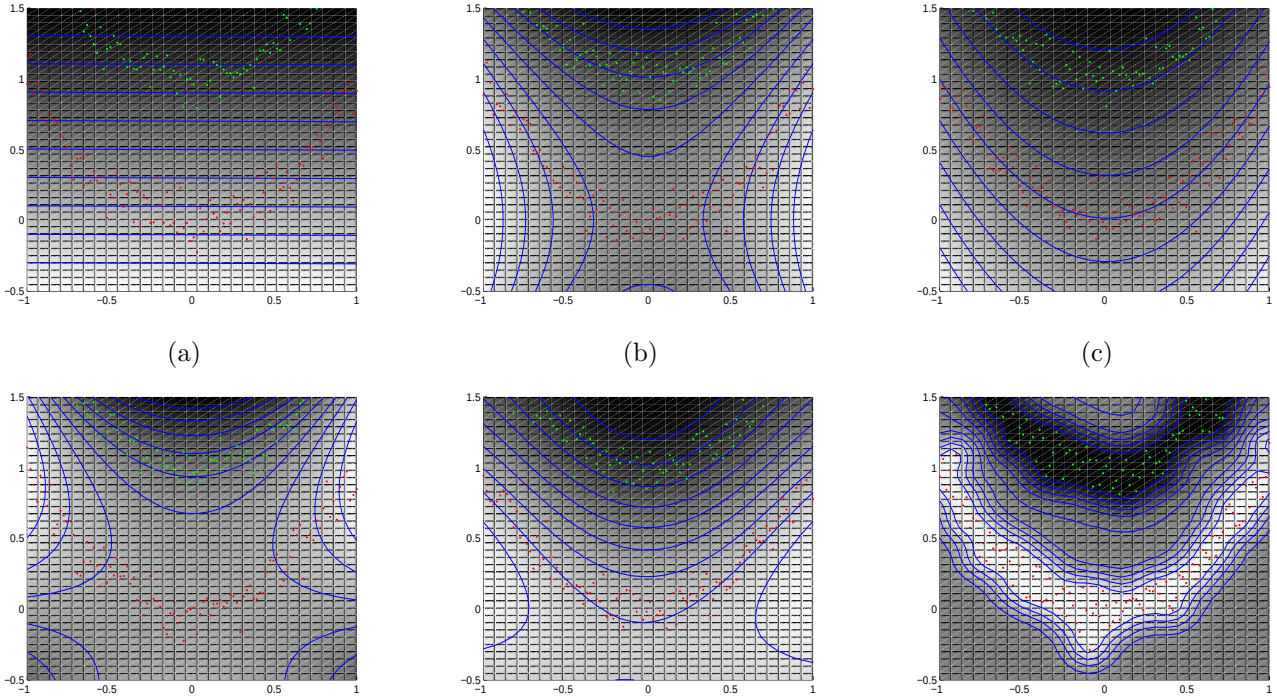


Figure 6: Kernel fisher discriminant analysis finds the most discriminative directions in the feature space. Here the algorithm is used to learn a classification function to separate two data sets (one in green and the other in red) each distributed in parabolic shapes. (a) shows the result using a linear kernel $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$ which fails to find a satisfactory solution. (b) and (c) show results with $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle^2$ and $\kappa(\mathbf{x}, \mathbf{y}) = (1 + \langle \mathbf{x}, \mathbf{y} \rangle)^2$ respectively. (d) shows the result with $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle^3$. (e) and (f) show the results with $\kappa(\mathbf{x}, \mathbf{y}) = \exp(-\frac{||\mathbf{x}-\mathbf{y}||^2}{2\sigma^2})$ with $\sigma = 1, 0.1$ respectively. Note the difference between these boundaries and the once obtained with the kernel perceptron in figure 3

.

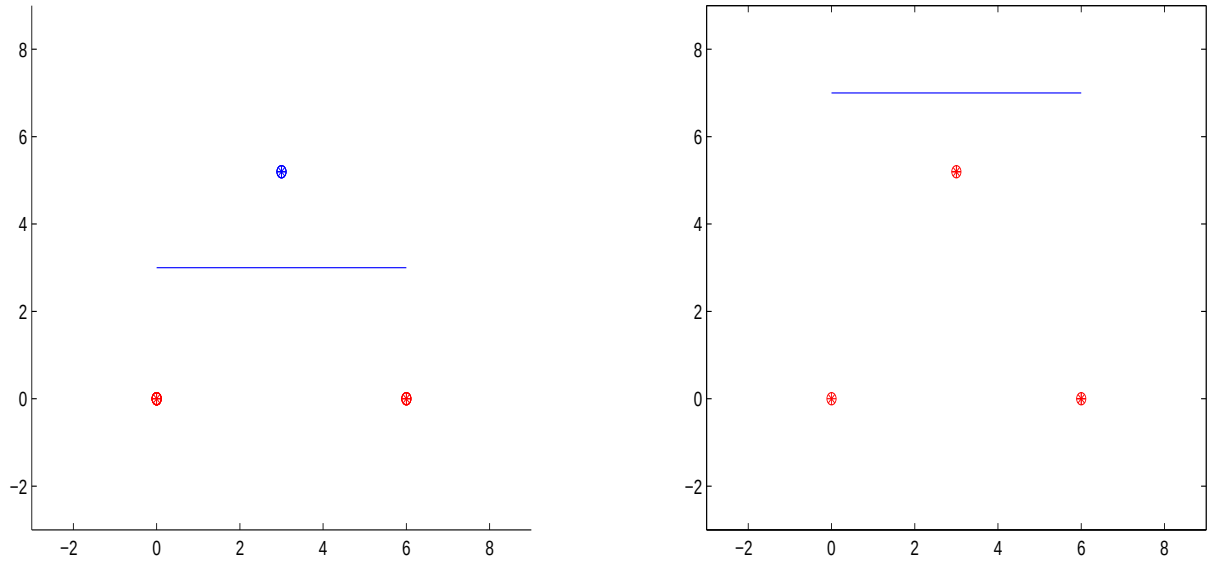


Figure 7: VC dimensions of lines in $\mathcal{R}^2$ is 3. Lines separating two labellings of vertices of an equilateral triangle. By symmetry, the remaining labellings also can be separated.

As the number of input samples $n \to \infty$, the empirical risk $R_{emp}(\alpha)$ approaches $R(\alpha)$. However the rate at which the two quantities two converge and the exact relationship between them is not apparent. The central result of learning theory is that with a probability of $1 - \eta$ where $0 \leq \eta \leq 1$, the following inequality holds

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h \log(2n/h) + h - \log(\eta/4)}{n}} \tag{81}$$

where h is a measure of descriptive power of f called the Vapnik Chervonenkis(VC) dimension. The VC dimension is a property of a class of functions $\{f_\alpha\}$. For a two class classification problem there exists a straightforward characterization of VC dimension. Give a set n input samples there exist 2^n different labellings possible since each sample can be assigned a label $+1$ or -1. If for each such labeling a member $f_\alpha^* \in \{f_\alpha\}$ can found such $R_{emp}(\alpha^*) = 0$, then the input set is said to be *shattered* by the the function class $\{f_\alpha\}$. The VC dimension of a function class $\{f_\alpha\}$ is the maximum number of points that can be shattered by $\{f_\alpha\}$. For instance, for lines in $\mathcal{R}^2$ the VC dimension is 3 as proven by figure 7.

The second result that is used in the development of SVM is the upper bound on the VC dimension of class of hyperplanes in $\mathcal{R}^n$ described by parameters $\alpha = (\mathbf{w}, b)$:

$$f_{\mathbf{w},b} = sign(\langle \mathbf{w}, \mathbf{x} \rangle + b) \tag{82}$$

To make this function class more precise we need to obtain a canonical form to describe each member of $(\mathbf{w}, b)$ in the class $\{f_{\mathbf{w},b}$ since for any $k \neq 0$, $(k\mathbf{w}, kb)$ also describes the same member. This canonical form can be obtained by enforcing the constraint that the closest points to the hyperplane should be at unit distance from the place. Figure 8 shows how this is achieved. Although this constraint does not distinguish between the pairs $(\mathbf{w}, b)$ and $(-\mathbf{w}, -b)$ the class labels allows us to do this. Formally, this constraint can be expressed as

$$\min_{i=1\cdots n} |\langle \mathbf{w}, \mathbf{x}_i \rangle + b| = 1 \tag{83}$$

In Vapnik [20] it is shown that the following upper bound for VC dimension of the above function class holds :

Given an input set $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$, if R be the radius of the smallest hypersphere enclosing these points then the VC dimension h of the function class $\{f_{\mathbf{w},b} : ||\mathbf{w}|| \leq A\}$, where

$$f_{\mathbf{w},b} = sign(\langle \mathbf{w}, \mathbf{x} \rangle + b)$$

satisfies the following inequality

$$h < R^2 A^2 + 1 \tag{84}$$

The above two results aid choosing the plane, among the planes that classify the data correctly, that had the least expected risk. Since $R(\alpha)$ increases with h and h is bounded by A the optimal canonical hyperplane is obtained by minimizing $||\mathbf{w}||$.

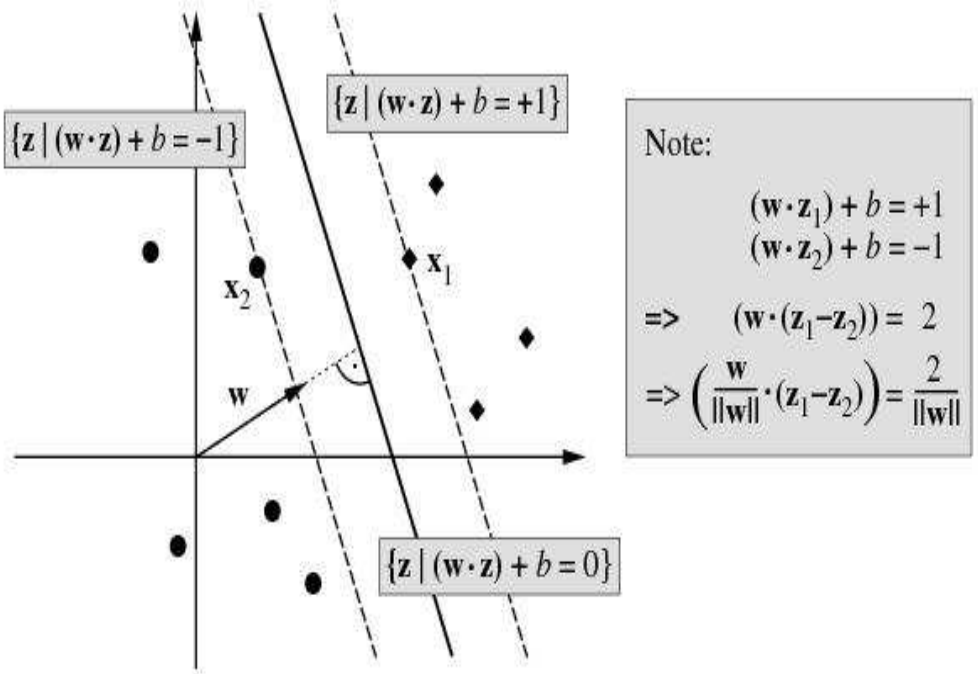


Figure 8: Obtaining a canonical representation of a hyperplane described by $(\mathbf{w}, b)$

6.2 Finding the optimal separating hyperplane

Assuming that there exist hyperplanes that classify the data correctly, the optimization problem can be posed formally as follows. The function to be minimized is

$$J(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2 \tag{85}$$

subject to the following n constraints

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \tag{86}$$

Using the lagrangian multipliers $\alpha_1, \cdots, \alpha_n$ the Lagrangian becomes

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i (y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1) \tag{87}$$

Setting the partial derivatives $\frac{\partial L(\mathbf{w},b,\boldsymbol{\alpha})}{b}$ and $\frac{\partial L(\mathbf{w},b,\boldsymbol{\alpha})}{\mathbf{w}}$ to 0 we obtain

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{88}$$

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{89}$$

A key property of the above optimization (a result of Kuhn-Tucker theorem) is that only those α_i are non-zero for which the input samples lies exactly at unit distance from the canonical plane i.e $(\mathbf{x_i}, y_i)$ satisfies the following constraint

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 = 0 \tag{90}$$

Such vectors are called *support vectors* and are the most important subset of the input samples. In fact, removing the other samples does not affect the final solution. The dual optimization function W($\boldsymbol{\alpha}$) by using this fact and by substituting the expression for $\mathbf{w}$ in $L(.,.,.)$. The function to maximize is :

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \tag{91}$$

subject to the constraints

$$\begin{aligned} \alpha_i \geq 0, i = 1, \cdots, n \\ \sum_{i=1}^{n} \alpha_i y_i = 0, i = 1, \cdots, n \end{aligned} \tag{92}$$

Solving the above optimization problem gives the dual coefficients of the vector $\mathbf{w}$. Note that $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$ and depends on the sign the of the label assigned to $\mathbf{x}_i$ unlike the previous cases where $\mathbf{w}$ was a linear combination of the input samples. A big drawback of the above formulation is the assumption that the data is separable by a hyperplane. However, even when this is not the case the plane that separates a good fraction of examples is still useful. To allows for examples that violate the constraint in equation(86) slack variables $\xi_i \geq 0$ for $i = 1, \cdots n$ are introduced with the following relaxed separation constraints

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \tag{93}$$

Minimizing the bound on the expected error leads to the following optimization : minimize

$$j(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2 + \gamma \sum_{i=1}^{n} \xi_i \tag{94}$$

where γ is a constant that controls the empirical risk i.e the number of mis-classifications on the training set. Setting up the Lagrangian as in the separable case the dual optimization problem can be found to be : maximize

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \tag{95}$$

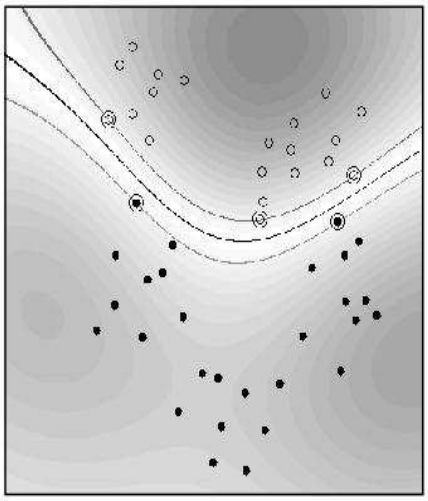


Figure 9: Boundary learned by an SVM with an Gaussian radial basis kernel

subject to the constraints

$$\begin{aligned} \gamma \geq \alpha_i \geq 0, i = 1, \cdots, n \\ \sum_{i=1}^{n} \alpha_i y_i = 0, i = 1, \cdots, n \end{aligned} \tag{96}$$

The final step is to introduce nonlinearity by kernelizing the SV algorithm. This is rather straightforward as equation(95) accesses the input samples only via the inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ which in the feature space is $\kappa(\mathbf{x}_i, \mathbf{x}_j)$. Since the constraints do not involve the input samples the optimization problem in the feature space can be posed as : maximize

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \tag{97}$$

subject to the constraints

$$\begin{aligned} \gamma \geq \alpha_i \geq 0, i = 1, \cdots, n \\ \sum_{i=1}^{n} \alpha_i y_i = 0, i = 1, \cdots, n \end{aligned} \tag{98}$$

Projecting a test sample on to the hyperplane is done in a manner similar to the previous cases. The support vector machine and its many variants are widely studied in literature. Ever since their discovery SVMs have been widely used for a number of applications involving classification and recognition. Many approaches to problems such as parameter selection, parallelization and other complexity issues have been proposed earlier. The reader is referred to [26, 17] for more information on these techniques and applications. Figure 9 shows how SVM learns optimal separating boundaries with various kernels.

7 Further Challenges in Kernel Methods

The above examples demonstrate various ways in which the kernel paradigm can be exploited to make algorithms more descriptive while retaining their statistical properties. Kernel methods possess many advantages other than nonlinearity such modularity, ability to work with heterogeneous descriptions of data, incorporation of prior knowledge etc. However a major issue in all the kernel methods is the choice of kernel function. The kernel function defines the geometry of the space in which an algorithm operates and this is crucial for the performance

of the algorithm in that space. Although many methods have been developed for selection of kernel function optimal for a specific task [22, 21], in general there is no way of choosing or constructing a kernel that is optimal for a given problem.

Another important limitation is the complexity of kernel algorithms. Kernel methods access the feature space via the input samples and hence kernel algorithms need to store all the relevant input samples. For instance, in the case of SVMs all the support vectors need to stored so that they can be used to project a test sample on the the separating hyperplane. Similarly, while inferring relationships from large number of samples, the size of the kernel matrices grows quadratically with the number of samples. Methods such as the reduced set SVM [27], approximate factorization of kernel matrix [28] attempt to reduce the space and time complexity in the case of SVMs. Despite these recent developments scalability of kernel methods still is a challenging problem with scope for further research.

Apart from kernel selection and scalability, the design of kernels, incorporating priors knowledge and invariances in to kernel functions are some of the challenges in kernel methods. Despite these shortcomings the numerous advantages of kernel method over traditional linear algorithms make them the preferred choice for practical applications. With a number of problems still open, kernel methods is an active area of research with promising future prospects.

References

[1] M. Pontil, A. Verri, Support vector machines for 3d object recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (6) (1998) 637–646.

[2] S. Avidan, Support vector tracking., IEEE Transcations on Pattern Analysis and Machine Intelligence 26 (8) (2004) 1064–1072.

[3] G. Guo, S. Li, K. Chan, Face recognition by support vector machines, in: Proc. of the International Conferences on Automatic Face and Gesture Recognition, 2000, pp. 196–201.

[4] T. Joachims, Text categorization with support vector machines: learning with many relevant features, in: Proceedings of ECML-98, 10th European Conference on Machine Learning, no. 1398, Springer Verlag, Heidelberg, DE, 1998, pp. 137–142.

[5] A. Sun, E. Lim, W. Ng, Web classification using support vector machine, in: Proceedings of the fourth international workshop on Web information and data management, 2002, pp. 96–99.

[6] I. T. Jolliffe, Principal Component Analysis, Springer, 1986.

[7] R. Fisher, The Use of Multiple Measurements in Taxonomic Problems, Annals of Eugenics 7 (1936) 179–188.

[8] J. Makhoul, Linear Prediction : A Tutorial Review, in: Proc. IEEE, 63(4), 1975, pp. 561–580.

[9] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, New York, NY, USA, 2004.

[10] B. E. Boser, I. M. Guyon, V. N. Vapnik, A training algorithm for optimal margin classifiers, in: COLT '92: Proceedings of the fifth annual workshop on Computational learning theory, ACM Press, New York, NY, USA, 1992, pp. 144–152.

[11] A. Aizerman, E. M. Braverman, L. I. Rozoner, Theoretical foundations of the potential function method in pattern recognition learning, Automation and Remote Control 25 (1964) 821–837.

[12] N. Aronszajn, Theory of reproducing kernels, Transactions of the American Mathematical Society 68.

[13] B. Scholkopf, A. Smola, K. uller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Computation 10 (1998) 1299–1319.

[14] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, K. Muller, Fisher discriminant analysis with kernels, in: Proceedings of IEEE Neural Networks for Signal Processing, pp. 41–48.

[15] J. Wang, J. Lee, C. Zhang, Kernel trick embedded gaussian mixture model., in: Algorithmic Learning Theory, 14th International Conference, ALT 2003, Sapporo, Japan, October 17-19, 2003, Proceedings, Vol. 2842, 2003, pp. 159–174.

[16] R. Herbrich, Learning Kernel Classifiers: Theory and Algorithms, MIT Press, Cambridge, MA, USA, 2001.

[17] N. Cristianini, J. Shawe-Taylor, An introduction to support Vector Machines: and other kernel-based learning methods, Cambridge University Press, New York, NY, USA, 2000.

[18] B. Scholkopf, A. J. Smola, Learning with Kernels : Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, Cambridge, MA, USA, 2001.

[19] F. Rosenblatt, The perceptron: A probabilistic model for information storage and organization in the brain, Psychological Review 65 (1958) 386–408.

[20] V. N. Vapnik, The nature of statistical learning theory, Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[21] G. R. G. Lanckriet, N. Cristianini, P. L. Bartlett, L. E. Ghaoui, M. I. Jordan, Learning the kernel matrix with semidefinite programming., Journal of Machine Learning Research 5 (2004) 27–72.

[22] T. Graepel, Kernel matrix completion by semidefinite programming., in: Artificial Neural Networks - ICANN 2002, International Conference, Madrid, Spain, August 28-30, 2002, Proceedings, Vol. 2415, 2002, pp. 694–699.

[23] K. Q. Weinberger, F. Sha, L. K. Saul, Learning a kernel matrix for nonlinear dimensionality reduction, in: ICML '04: Proceedings of the twenty-first international conference on Machine learning, ACM Press, New York, NY, USA, 2004, p. 106.

[24] M. Yang, N. Ahuja, D. Kriegman, Face recognition using kernel eigenfaces, in: Proceedings of the International Conference on Image Processing, 2000, pp. 37–40.

[25] C. J. Twining, C. J. Taylor, Kernel principal component analysis and the construction of non-linear active shape models., in: Proceedings of the British Machine Vision Conference 2001, BMVC 2001, Manchester, UK, 10-13 September 2001, 2001.

[26] C. J. C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery 2 (2) (1998) 121–167.

[27] C. J. C. Burges, B. Schölkopf, Improving the accuracy and speed of support vector machines, in: M. C. Mozer, M. I. Jordan, T. Petsche (Eds.), Advances in Neural Information Processing Systems, Vol. 9, The MIT Press, 1997, p. 375.

[28] G. Wu, E. Chang, Y. K. Chen, C. Hughes, Incremental approximate matrix factorization for speeding up support vector machines, in: KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM Press, New York, NY, USA, 2006, pp. 760–766.