Study of Kernel Selection and Construction Kernel for SVM Classification

1,3 Lou XIONGWE, 2 Huang DECAI, 3 Fang LUMING, 3 Xu AIJUN
1 College of Information Engineering, Zhejiang University of Technology, Hangzhou, Zhejiang, 310032, China
2 School of Computer Science & Technology, Zhejiang University of Technology, Hangzhou, Zhejiang, 310032, China
3 College of Information Engineering, Zhejiang A & F University, Linan, Zhejiang, 311300, China
E-mail: lxwzjfc@163.com

Received: 3 July 2013 /Accepted: 25 September 2013 /Published: 31 October 2013

Abstract: It is the most critical for finding the best kernel to apply the kernel-based algorithms in practice, such as support vector machines (SVMs) for classification. The selection is tightly connected to the encoding of our prior knowledge about the data and the pattern type of the task. In this paper, we discuss the kernel trick and its selection and construction methods for SVM classification. In addition we employed distance learning to construct RBF kernels. Experiments on several dataset proved the advantages of distance metric learning for kernel construction. Copyright © 2013 IFSA.

Keywords: Kernel, SVM, Distance learning.

1. Introduction

The concept “kernel” was an analogy with electrostatics, which was firstly introduced into the field of pattern recognition by Aizerman [1]. After being neglected for many years, the method was re-introduced into machine learning by Boseret et al. [2], which giving rise to the large-margin technique such as support vector machines. Since then, there has been considerable research work in this topic, and the kernel has been extended to handle symbolic objects.

A kernel $k(x,y)$ built with a finite data set $S=\{x_1,x_2,\ldots,x_l\}$ of an input space $X$ should satisfy

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle,$$

where $\phi(x)$ is the feature map which maps the data $x$ and $y$ into a feature space $F$. With the kernel function, the data in the feature space could be represented implicitly. That is, we can implement classification (or regression) tasks with the kernels, no need of the explicit representation of the data in the feature space.

The kernel is formulated as an inner product in a feature space, which allows us to make use of the kernel trick to build interesting extensions of many well-known algorithms. If an algorithm formulated in the form of scalar products, then we can replace that scalar product with some form of kernel functions. For sample, the kernel trick technique can be applied to principal component analysis to develop a nonlinear variant of PCA [3]. Other instances of kernel trick include nearest-neighbor classifiers and the kernel Fisher discriminant [4-5].

Using only the inner product information between all pairs elements of $S$, which is contained in the kernel matrix, could obtain many significant information. Kernel matrix can be regarded as
information bottleneck [6]. The kernel matrix is the key ingredient in the kernel methods, which contain all the available to perform the learning step. It is no doubt that, only through the kernel matrix, the learning algorithms can obtain information about the choice of feature space or model. Through manipulating the kernel matrix before it is passed to the learning machine, some intermediate processing steps can be designed to improve the representation of the samples and hence the overall performance of the system. For example, the addition of a constant to the diagonal of the matrix has the effect of introducing a soft margin or regularization in classification tasks. The data structure is exhibited through the kernel matrix, if the kernel is too general, it will not enough importance to specific type of similarity, which corresponds to giving weight to too many different classification, with the same weight to any pairs of inputs as similar or dissimilar, the off-diagonal entries become very small. The kernel represents the concept of identity. With a kernel we can easily classify a training set correctly but it is hard to generalize to a new data. On the other hand, if the kernel matrix is completely uniform, every input is mapped to the similar feature, which can lead to under fitting of the data, because all the points are mapped to the same class. As an interface between the input data and the learning algorithms, the kernel matrix is critical in the derivation of generalization bounds and their evaluation in practice. Fig. 1 illustrates the stages of kernel methods in the application [7].

As long as a certain kernel function is given, kernel methods can be applied to any kind of data. This gives rise to new questions: Given some structured data, how to define or design a meaningful kernel function?

We begin to introduce the definition of reproducing kernels, which are inner products in the Hilbert space. The reproducing property is equivalent to positive definition of the function.

Let \( H \) be a set of functions \( \chi \rightarrow R \) that forms a Hilbert space. A function \( k: \chi \times \chi \rightarrow R \) is called a reproducing kernel if

- For every \( x \in \chi \) the function \( k(x,.) \in H \) and
- For every \( x \in \chi \) and every function \( f(.) \in H \) , the reproducing property holds \( \langle k(x,.), f(.) \rangle = f(x) \)

It follows directly from the definition that every reproducing kernel is symmetric and that for every reproducing kernel \( k: \chi \times \chi \rightarrow R \), there is a function \( \phi: \chi \rightarrow H \) such that \( k(x,y)=\langle \phi(x),\phi(y) \rangle \) for all \( x,y \in \chi \). This function \( \phi \) maps every element of \( \chi \) to a function in the Hilbert space.

A function

\[ k: X \times X \rightarrow R \]

can be decomposed into feature map if and only if it satisfied the finitely positive semidefinite property [8], i.e.

For all \( n \in N \), \( x_1,...,x_n \in X \) and \( c_1,...,c_n \in R \)

\[ \sum_{i,j=1}^{n} c_i c_j k(x_i, x_j) \geq 0 \]

Then

![Fig. 1. The stages in the application of kernel methods.](image1)

The L2 norm of \( \phi(x) \) is

\[ \| \phi(x) \|_2 = \sqrt{\| \phi(x) \|^2 = k(x,x)} \]

The square of the distance between the two points \( \phi(x), \phi(z) \) in the feature space can be calculated as below:

\[ \| \phi(x)-\phi(z) \|^2 = \langle \phi(x)-\phi(z), \phi(x)-\phi(z) \rangle = \langle \phi(x), \phi(x) \rangle - 2\langle \phi(x), \phi(z) \rangle + \langle \phi(z), \phi(z) \rangle = k(x,x) - 2k(x,z) + k(z,z) \]

So, a kernel can be regarded as defining a similarity measure between two samples. Provided the kernel matrices corresponding to any finite training set are positive semi-definite the kernel computes the inner product after projecting pairs of inputs into some feature space. Fig. 2 and Fig. 3 illustrate this point with an embedding showing objects being mapped to feature space.

![Fig. 2. The kernel computes the inner product after projecting pairs of inputs into some feature space.](image2)
Kernel methods can be regarded as a special kind of radial basis functions. The parameters of the estimating function can be determined efficiently and reliably by restricting the class of similarity measures, which are considered to positive definite kernel functions. The hypothesis space of kernel methods consists of linear combinations of a specific class of functions. In the other words, the hypothesis space related to the kernel methods is the set of linear combinations of positive definite kernel functions. Mathematically, this space is a Hilbert space with a well-defined inner product and norm.

In addition, we consider the center of the set $\phi(S)$. The point is inaccessible through the map function.

$$\phi_S = \frac{1}{l} \sum_{i=1}^{l} \phi(x_i)$$

$$\|\phi_S\|^2 = \langle \phi_S, \phi_S \rangle = \left\langle \frac{1}{l} \sum_{i=1}^{l} \phi(x_i), \frac{1}{l} \sum_{i=1}^{l} \phi(x_i) \right\rangle$$

$$= \frac{1}{l^2} \sum_{i,j=1}^{l} \langle \phi(x_i), \phi(x_j) \rangle$$

$$= \frac{1}{l^2} \sum_{i,j=1}^{l} k(x_i, x_j)$$

So the square of the distance of the center is equal to the average of the entries in the kernel matrix. We can calculate the distance of the image of a point $x$ from the center of the set

$$\|\phi(x)-\phi_S\|^2 = \langle \phi(x), \phi(x) \rangle - 2 \langle \phi(x), \phi_S \rangle + \langle \phi_S, \phi_S \rangle$$

$$= k(x,x) - \frac{2}{l} \sum_{i=1}^{l} k(x, x_i) + \frac{1}{l^2} \sum_{i,j=1}^{l} k(x_i, x_j)$$

It is a much more complex transformation to center data in the feature space, but the data-centering can be performed by operations on the kernel matrix, in which the aim is to move the origin of the feature space to the center of mass of the training examples. Furthermore, the center of mass can be characterized as the origin which minimizes the sum of the norms of the samples. This is equal to the sum of its eigenvalues as the sum of the norms is the trace of the kernel matrix.

### 2. Kernel Functions Selection for SVM

The general processes of C-SVM [9-10] are explained as below.

1) Assuming the training dataset $D$ is given as:

$$D = \{(x_i, y_i) | x_i \in \mathbb{R}^d, y_i \in \{-1, 1\}, i = 1, 2, \ldots m\}$$

where $y_i$ is the label of training sample $x_i$, $m$ is the total number of training samples.

2) In the training phrase, select appropriate kernel function $k(x_i, x_j)$ and penal parameter $C > 0$.

3) Construct and solve the convex programming problem:

$$\min_{\alpha} - \sum_{i=1}^{m} \alpha_i + \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j k(x_i, x_j),$$

s.t. $\sum_{i=1}^{m} \alpha_i y_i = 0$,

$$0 \leq \alpha_i \leq C, \ i = 1, 2, \ldots m;$$

where $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$. The solution $\alpha^* = (\alpha_1^*, \ldots, \alpha_m^*)^T$ is always sparse, i.e. there is small number of non-zero coefficients, the corresponding training samples are defined as the support vector machines.

4) Choose the $\alpha_j^*$ with the value of $(0, C)$, calculate:

$$b^* = y_j - \sum_{i=1}^{m} \alpha_i^* y_i k(x_i, x_j);$$

5) For the test sample $x$, the final classification decision function is

$$f(x) = \text{sgn} \left( \sum_{i=1}^{m} \alpha_i^* y_i k(x_i, x) + b^* \right)$$

Below is some kernel functions in practice.

#### 2.1. Linear Kernel

If $\phi(x) = x$, we get the linear kernel:

$$k(x, z) = x^T z$$

If the original data is already in high dimensional and the original features are relative more informative, the linear kernel is useful. In this case, the decision boundary is represented as a linear combination of the original features. However, not all
high dimensional problems are linearly separable. For instance, the images are in high dimensions, but each individual pixel is not informative, so the linear kernel is not suitable for the problem of image classification. The non-linear kernel functions are as follows.

2.2. Polynomial Kernel

Polynomial kernels are as blows, with three adjustable parameters, i.e. \(\alpha\), constant \(c\), and the degree \(d\).

\[
k(x, z) = (\alpha x^T z + c)^d
\]

2.3. Radial Basis Function Kernel

Radial basis function kernels are as blows, with one adjustable parameter, i.e. \(\sigma\).

\[
k(x, z) = \exp(-\|x - z\|^2) / 2\sigma^2
\]

The adjustable parameter \(\sigma\) plays a major role in the performance of the kernel, and should be carefully tuned to the problem at hand. The parameter \(\sigma\) controls the flexibility just as the similar way to the degree \(d\) in the polynomial kernel. Small value of \(\sigma\) allow kernel classifiers to fit any labels, corresponding to the large value of \(d\) in the polynomial kernel. If overestimated, the exponential will behave almost linearly and the higher-dimensional projection will start to lose its non-linear power. On the other hand, large value of \(\sigma\) gradually reduce the kernel to a constant function, which make it impossible to learn any meaningful classifier. If underestimated, the function will lack regularization and the decision boundary will be highly sensitive to noise in training data. For every value of \(\sigma\), the feature space has infinite dimension, but for the larger one, the weight decays fast on the higher order features. The Gaussian kernel is a polynomial kernel of infinite degree, therefore, its features are all monomials of the input features. The Taylor expansion of the exponential function

\[
\exp(x) = \sum_{j=0}^{\infty} \frac{1}{j!} x^j
\]

The performance of a classifier model is largely dependent on the kernel parameter, different parameter values may cause under or over fitting. The different RBF functions for classification are illustrated as Fig. 3.

Chapelle [11] proposed a clustering of Generalized RBF kernel function:

\[
k(x, z) = \exp\left(-\frac{1}{2\sigma^2} \sum_{i} |x_i^T - z_i^T|^p\right),
\]

if \(p=1\) – the kernel is called Laplacian kernel. The Laplace Kernel is completely equivalent to the exponential kernel, except for being less sensitive for changes in the sigma parameter. And if \(p=0.5\), it becomes sublinear kernel. The Mercer condition can be satisfied on the condition of \(0 \leq b \leq 2\).

2.3. Chi-Square Kernel [12]

The Chi-Square kernel comes from the Chi-Square distribution.

\[
k(x, z) = \exp(-\|x - z\|^2) / 2\sigma^2,
\]

where:

\[
\chi^2(x, z) = \sum_{i} \frac{(x_i - z_i)^2}{x_i + z_i}
\]

2.4. Histogram Intersection Kernel

The Histogram Intersection Kernel is also known as the Min Kernel and has been proven useful in image classification.

\[
k(x, z) = \min_{i} (x_i, z_i)
\]

The Generalized Histogram Intersection kernel [13] is built based on the Histogram Intersection Kernel for image classification but applies in a much larger variety of contexts. It is given by:

\[
k(x, z) = \sum_{i} \min(x_i^p, z_i^p)
\]

The Mercer condition is satisfied as \(\beta \geq 0\).
3. Kernel Construction

Kernel can be regarded as an oracle to guess the similarity of two inputs. For the so-called normalized kernels, they can be regarded as the a priori probability of the inputs being in the same class minus the a priori probability of their being in different classes. In the case of a covariance kernel over a class of classification functions this is precisely the meaning of the kernel function under the prior distribution since

\[ k(x,z) = \int f(x)f(z)p(f)d\mu = P(f(x) = f(z)) - P(f(x) \neq f(z)) \]

It is the most critical for finding the best kernel among the extensive range of possibilities to apply the kernel-based algorithms in practice. The selection is tightly connected to the encoding of our prior knowledge about the data and the pattern type of the task. One possible way is to treat the kernel as a modular fashion, constructing it from simple components and modifying by some well-defined operations.

Kernels can be constructed with the case of infinite-dimension feature space at the cost of a few kernel computations. We can design kernel functions according to the task at hand. However, there exist some general principles.

Creating a new kernel: the normalized kernel \( k(x,z) \) corresponds to the map:

\[ x \mapsto \phi(x) \mapsto \left[ \frac{\phi(x)}{\|\phi(x)\|}, \frac{\phi(z)}{\|\phi(z)\|} \right] = \frac{k(x,z)}{\sqrt{k(x,x)k(z,z)}} \]

It is helpful to make clear of the effect of kernel combination on the structure of feature space.

\[ \phi(x) = [\phi_1(x), \phi_2(x)] \]

since

\[ k(x,z) = \langle \phi(x), \phi(z) \rangle = \langle \phi_1(x), \phi_1(z) \rangle + \langle \phi_2(x), \phi_2(z) \rangle = k_1(x,z) + k_2(x,z) \]

We can create more complicated kernels from simple kernels:

Let \( k_1 \) and \( k_2 \) is kernel designed over \( X \times X, X \in \mathbb{R}^+ \), \( f(.) \) is a real-value function on \( X \), \( p \) is a polynomial with positive coefficients. Then the following functions are kernels:

1) \( k(x,z) = k_1(x,z) + k_2(x,z) \)
2) \( k(x,z) = ak_1(x,z) \)
3) \( k(x,z) = k_1(x,z)k_2(x,z) \)
4) \( k(x,z) = f(x)f(z) \)
5) \( k(x,z) = p(k_1(x,z)) \)
6) \( k(x,z) = \exp(k_1(x,z)) \)

The kernels created with simple operations are finitely positive semidefinite.

4. Distance Learning for Kernel Construction

A norm \( ||.|| \) defines a metric on a space, i.e., a function that measures the “distance” between two elements \( x \) and \( y \), which holds the following four properties:

1) \( d(x, y) = d(y, x) \)
2) \( d(x, y) > 0 \) if \( x \neq y \);
3) \( d(x, x) = 0 \);
4) \( d(x, z) \leq d(x, y) + d(y, z) \)

Below are some commonly used distance metrics:

1) Minkowski-Form Distances:

\[ L_p(x,y) = \left( \sum_i |x_i - y_i|^p \right)^{1/p} \]

If \( p=1 \), \( L_p(x,y) \) defines L1 distance, i.e. streetdistance:

\[ L_1(x,y) = \sum_i |x_i - y_i| \]

If \( p=2 \), \( L_p(x,y) \) defines L2 distance, i.e. Euclidean distance:

\[ L_2(x,y) = \sqrt{\sum_i (x_i - y_i)^2} \]

If \( p \to \infty \), \( L_p(x,y) \) defines maximum distance

\[ L_\infty(x,y) = \max_i |x_i - y_i| \]

2) Kullback-Leibler Divergence

\[ KL(x,y) = \sum_i x_i \log \frac{x_i}{y_i} \]

Since Kullback-Leibler Divergence is non symmetric, i.e. \( KL(x,y) \neq KL(y,x) \), it is not a real metric. Jensen-Shannon Divergence is defined as a revised symmetric KL metric as below:

\[ JS(x,y) = \frac{1}{2} KL(x,y) + \frac{1}{2} KL(y,x) = \frac{1}{2} \sum_i \log \frac{x_i}{y_i} + \frac{1}{2} \sum_i \log \frac{y_i}{x_i} \]
3) $\chi^2(x,y) = \frac{1}{2} \sum_{i} \frac{(x_i - y_i)^2}{x_i + y_i}$

$\chi^2(x,y)$ is common used for histogram distance, whose advantage is to reduce the effect of large bins. However, $\sqrt{\chi^2(x,y)}$ (instead of $\chi^2(x,y)$) is a metric.

4) Quadratic-form distance

Bin-to-bin distance such as L1 L2 $\chi^2$ is sensitive to quantization. The quadratic-form distance is defined to avoid the effects of quantization:

$$d^2_i = \|x - y\|_d = (x - y)^T A(x - y) = \sum_{y}(x_i - y_i)(x_j - y_j)A_{ij}$$

$A_{ij}$ is the similarity of bin i and bin j. If A is inverse of covariance – it is called mahalanobis distance.

It is critical to learn a good distance metric in feature space for many tasks. For example, distance metrics are important in image classification applications. For the nearest-neighbor (NN) image classifier, it is critical to obtain the labeled image that is closest to a given query image in the visual feature space, which will involve the choice or designation of a good distance metric. Many considerable researches on distance metric learning over the past few years [14-23] has made the conclusion that, compared to the standard Euclidean distance, appropriately-designed distance metrics can significantly boost the accuracies of NN classifiers.

Distance metric learning can be divided into two categories: supervised distance metric learning and unsupervised distance metric learning. The supervised distance metric learning can get the availability of the training examples which are given class labels, while the unsupervised distance metric learning has no labeled samples to employ. There are two categories for the supervised distance metric learning: the global distance metric learning, and the local distance metric learning.

The former learns the distance metric in a global way, while the second learn a distance metric in a local setting. The global distance metric learning attempt to learn metrics to make all the samples within the same classes close and separate all the data from different classes far apart.

In distance metric learning, the label information is usually specified in the form of pairwise constraints on the samples, which is different from the typical supervised learning. (In the supervised learning, each training sample is annotated with their class label). The first is the so-called equivalence constraints, which indicate that, in the learned metric, the given pair are semantically-similar and should be close together. The second is “inequivalence constraints”, which state that the given samples are semantically-dissimilar and should not be near in the learned metric. Most learning algorithms try to find a distance metric that keep all the data pairs in the equivalence constraints close while separating those in the inequivalence constraints. [24] discuss the problem of learning linear representation functions, or equivalently an optimal Mahalanobis distance between data points, using equivalence relations. Specifically, they focus here on the Relevant Component Analysis (RCA) algorithm, which was first introduced in [24]; they present a new analysis, based on a novel information theoretic optimality criterion. RCA is shown to be an optimal learning procedure in this sense. In G. Lebanon [25], new types of K-nearest-neighbor procedures are described that estimate the local relevance of each input variable, or their linear combinations, for each individual point to be classified. This information is then used to separately customize the metric used to define distance from that object in finding its nearest neighbors. In [26], the distance metric is explicitly learned to minimize the distance between data points within the equivalence constraints and maximize the distance between data points in the inequivalence constraints.

The metric learning can make the kernel more generalization [27-31]. In this paper, the purpose of metric learning is to learn the Mahalanobis distance.

Assuming $C = \{x_1, x_2, ..., x_n\}$, n is the total number of samples, $x_j \in \mathbb{R}^m$

$S = \{(x_i, x_j) | x_i, x_j \text{ belongs to different classes}\}$

$D = \{(x_i, x_j) | x_i, x_j \text{ belongs to different classes}\}$

the matrix metric $A \in \mathbb{R}^{m \times m}$

The distance between the two points $x, z$ is:

$$d^2_i = \|x - z\|_d = (x - z)^T A(x - z)$$

$A$ is a positive semi-definite (PSD) matrix. If not, the distance between the two points $x, z$ $d^2_i$ could be negative.

Suppose $\nu = x - z$ is an eigenvector corresponding to a negative eigenvalue $\lambda$ of $A$, then

$$d^2_i(x,z) = (x - z)^T A(x - z) = \nu^T A \nu = \lambda \nu^T \nu < 0$$

We employ the [32] convex programming to optimize the global metric learning:

$$\min_{\lambda \geq 0} \sum_{(x_i, x_j) \in S} \|x_i - x_j\|_d$$

s.t. $A \geq 0$, $\sum_{(x_i, x_j) \in D} \|x_i - x_j\|_d \geq 1$

Probability method is used to implement the distance metric:
$$y_{ij} = \begin{cases} 1, & (x_i, x_j) \in S \\ -1, & (x_i, x_j) \in D \end{cases}$$

$$\Pr(y_{ij} \mid x_i, x_j) = \frac{1}{1 + \exp(-y_{ij} \|x_i - x_j\|_d^2 - u))}$$

\[ u \] is threshold. If \( x_i, x_j \) are from the same class, the distance between the two points is less than \( u \). The maximum probability estimation

$$L_g(A, u) = \log \Pr(S) + \log \Pr(D)$$

$$= -\sum_{(x_i, x_j) \in S} \log(1 + \exp(-y_{ij} \|x_i - x_j\|_d^2 - u))$$

$$-\sum_{(x_i, x_j) \in D} \log(1 + \exp(-y_{ij} \|x_i - x_j\|_d^2 - u))$$

$$\min_{A \in R^{m \times m}, u \in R} L_g(A, u)$$

s.t. \( A \geq 0, \ u \geq 0 \)

Fig. 4 illustrates the comparison of before and after distance metric optimization.

**5. Experiment**

We perform experiments on some real-world datasets. The set \( S \) and \( D \) are constructed by selecting randomly 40 patterns from the training set. The remaining samples are for testing. The wine, irises are small datasets, taken from UCI learning Repository. Each is with less than 500 training samples and just 3 classes.

The experiments were performed on an Intel Pentium CPU 2.66 GHz computer running Windows XP and MATLAB 2011 with 4 GB RAM. The LibSVM [9] implementation Toolbox with one-against-one strategy is used to train the multi-class classifier. We randomly selected 300 training samples per class and test on the remaining images, calculating the average accuracy for all the classes. The kernel functions were constructed through global distance learning.

Table 1 reports the classification errors, averaged over 10 repeated experiments, which compare the performance of the linear and RBF kernels with Euclidean distance and learned Mahanobis distance.

From the result data in Table 1, we can see that the RBF kernel performed better than the linear kernel, and the kernel constructed with learning distance lead to higher accuracies. For linear kernels, the error rates dropped 0.07 % ~ 5.64 %, while for the RBF kernel, the error rates went down from 0.48 % ~ 4.56 %.

**6. Conclusion**

In this paper, we discuss the kernel trick and its selection and construction methods for SVM classification; In addition, we employed distance learning to construct RBF kernels. Experiments proved the advantages of distance metric learning for kernel construction. Future work should focus on the learning methods of kernel construction, local distance learning and its combination with global distance learning.

**7. Acknowledgement**

The research is partly supported by Key Laboratory of Visual Media Intelligent Process Technology of Zhejiang Province (No: 2011E10003), Project of Zhejiang Province Department of Education (No: Y201018063) and The National Natural Science Foundation of China (No: 30972361).

**References**


[2]. B. E. Boser, I. M. Guyon, V. N. Vapnik, A training algorithm for optimal margin classifiers, in