EURASIP Journal on Applied Signal Processing 2004:16, 2515–2521

Modified Kernel Functions by Geodesic Distance

Quan Yong
Institute of Image Processing & Pattern Recognition, Shanghai Jiaotong University, Shanghai 200030, China
Email: quanysjtu@hotmail.com

Yang Jie
Institute of Image Processing & Pattern Recognition, Shanghai Jiaotong University, Shanghai 200030, China
Email: jieyang@sjtu.edu.cn

Received 20 August 2003; Revised 9 March 2004

When dealing with pattern recognition problems one encounters different types of prior knowledge. It is important to incorporate such knowledge into the classification method at hand. A common prior knowledge is that many datasets are on some kinds of manifolds. Distance-based classification methods can make use of this by a modified distance measure called geodesic distance. We introduce a new kind of kernels for a support vector machine (SVM) which incorporates geodesic distance and therefore is applicable in cases where such transformation invariance is known. Experiments results show that the performance of our method is comparable to that of other state-of-the-art methods, such as SVM-based Euclidean distance.

Keywords and phrases: support vector machine, geodesic distance, kernel function.

1. INTRODUCTION

Support vector machine (SVM) is a new promising pattern classification technique proposed recently by Vapnik and coworkers [1, 2]. Unlike traditional methods which minimize the empirical training error, SVM aims at minimizing an upper bound of the generalization error through controlling the margin between the separating hyperplane and the data. This can be regarded as an approximate implementation of the structure risk minimization principle. What makes SVM attractive is the property of condensing information in the training data and providing a sparse representation by using a very small number of data points.

SVM is a linear classifier in the parameter space, but it is easily extended to a nonlinear classifier of the ϕ-machine type by mapping the space $S = \{x\}$ of the input data into a high-dimensional feature space $F = \{\phi(x)\}$. By choosing an appropriate mapping ϕ, the data points become linearly separable or nearly linearly separable in the high-dimensional space so that one can easily apply the structure risk minimization. Instead of computing the mapped patterns $\phi(x)$ explicitly, we only need the dot products between the mapped patterns. They are directly available from the kernel function which integrates $\phi(x)$. By choosing different kinds of kernels, the SVM can realize radial basis function (RBF) and polynomial and multilayer perceptron classifiers. Compared with the traditional way of implementing them, the SVM has an extra advantage of automatic model selection, in the sense that both the optimal number and the locations of the basis functions are automatically obtained during training.

Like neural networks, SVM also uses a distance function to determine how close an input vector x is to each stored data [3]. As mentioned in [4], a variety of distance functions are available for such use, including the Minkowski, Mahalanobis, Camberra, Chebychev, and Chi-square distance metrics. Although there have been many distance functions proposed, by far the most commonly used is the Minkowski distance. Euclidean distance is the most special case of the Minkowski distance. The choice of distance function influences the bias of a learning algorithm. A bias is a rule or method that causes an algorithm to choose one generalized output over another. A learning algorithm must have a bias in order to generalize, and it has been shown that no learning algorithm can generalize more accurately than any other when summed over all possible problems [5]. It follows then that no distance function can be strictly better than any other in terms of generalization ability, when considering all possible problems with equal probability. So, an appropriate distance function should be selected according to datasets. Especially for those data points which lie in a manifold, Euclidean distance cannot reflect the real distance between two points.

In 2000, Tenenbaum proposed the geodesic distance [6]. The basic idea is that for a neighborhood of points on a manifold, the Euclidean distances provide a fair approximation of geodesic distance. For faraway points, the geodesic distance is estimated by the length of the shortest path through neighboring points.

In this paper, we focus on the design of SVM based on the geodesic distance. We first propose an improved geodesic distance to eliminate isolated embeddings, then the geodesic distance is applied to the kernel of SVM. Experiments show that good generalization can be obtained using the revised SVM.

2. MINKOWSKI METRIC AND ITS LIMITATIONS

The Minkowski metric [7] is widely used for measuring similarity between points. Suppose two points x and y are represented by two p dimensional vectors $(x_1, x_2, \ldots, x_p)$ and $(y_1, y_2, \ldots, y_p)$, respectively. The Minkowski metric $d(x, y)$ is defined as

$$d(x,y) = \left(\sum_{i=1}^{p} |x_i - y_i|^r \right)^{1/r}, \tag{1}$$

where r is the Minkowski factor for the norm. Particularly, when r is set as 2, it is the well-known Euclidean distance; when r is 1, it is the Manhattan distance (or L_1 distance). A point located a smaller distance from a query point is deemed more similar to the query point. Measuring similarity by the Minkowski metric is based on one assumption: the similar point should be close to the query point in all dimensions.

A variant of the Minkowski function, the weighted Minkowski distance function, has also been applied to measure the point similarity. The basic idea is to introduce weighting to identify important features. By assigning each feature a weighting coefficient w_i $(i = 1, \ldots, p)$, the weighted Minkowski distance function is defined as

$$d_w(x,y) = \left(\sum_{i=1}^{p} w_i |x_i - y_i|^r \right)^{1/r}. \tag{2}$$

By applying a static weighting vector for measuring similarity, the weighted Minkowski distance function assumes that similar points resemble the query points in the same features. However, in some cases, Minkowski distance does not reflect the geodesic distance between two points. Figure 1 illustrates how Euclidean distance exploits the distance for two points on a manifold.

As Figure 1 shows, for two arbitrary points (circled) on a nonlinear manifold, their Euclidean distance in the high-dimensional input space (length of dashed line) may not accurately reflect their geodesic distance, as illustrated by the solid curve.

3. REVISED GEODESIC DISTANCE

3.1. Constructing the geodesic distance path

Tenenbaum proposed a new algorithm for estimating geodesic distances. The basic idea is that for a neighborhood of points on a manifold, the Euclidean distances provide a fair approximation of geodesic distance. For faraway points the geodesic distance is estimated by the length of the shortest path through neighboring points.

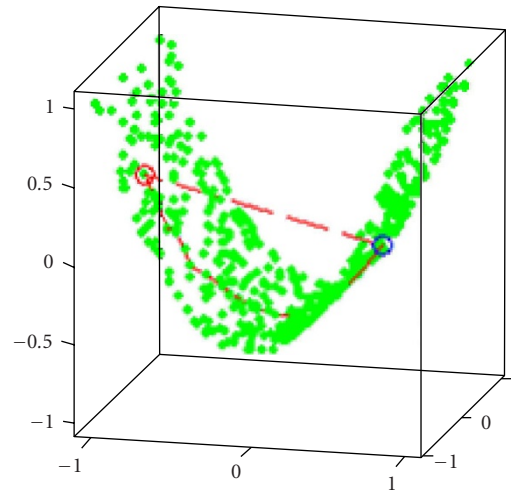


Figure 1: Distance between two points on a manifold.

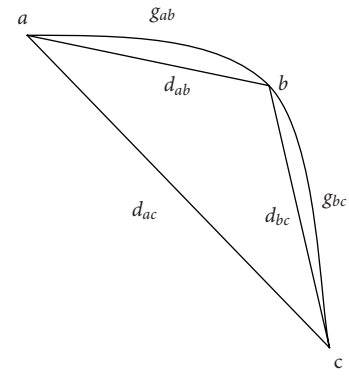


Figure 2: Inaccuracy of Euclidean distance compared with geodesic distance.

Let a, b, and c be given samples from a manifold structure. One does not assume to know the true manifold which would be the curve in this example (see Figure 2). The geodesic distances g_{ab} and g_{bc} would be measured along the manifold, while d_{ab}, d_{bc}, and d_{ac} denote the Euclidean distance. When describing relations between the points, the aim is to take the assumed manifold structure into consideration. Therefore, one tries to estimate the geodesic distance for a given set of points. As one can see for this example, the geodesic distance between neighboring points can be approximated fairly well by their Euclidean distances, so that $\hat{g}_{ab} = d_{ab}$ and $\hat{g}_{bc} = d_{bc}$. The geodesic distance between a and c would be $g_{ac} = g_{ab} + g_{bc}$ for which $\hat{g}_{ac} = d_{ab} + d_{bc}$ is a far better approximation than d_{ac}. So, for neighboring points the geodesic distance is approximated by Euclidean distances and for distant points one considers the length of the shortest path through neighboring points.

The geodesic distance algorithm proceeds in two steps based on this idea. Let X be an $(m \times n)$ matrix representing m samples in n dimensions and it can be assumed that the

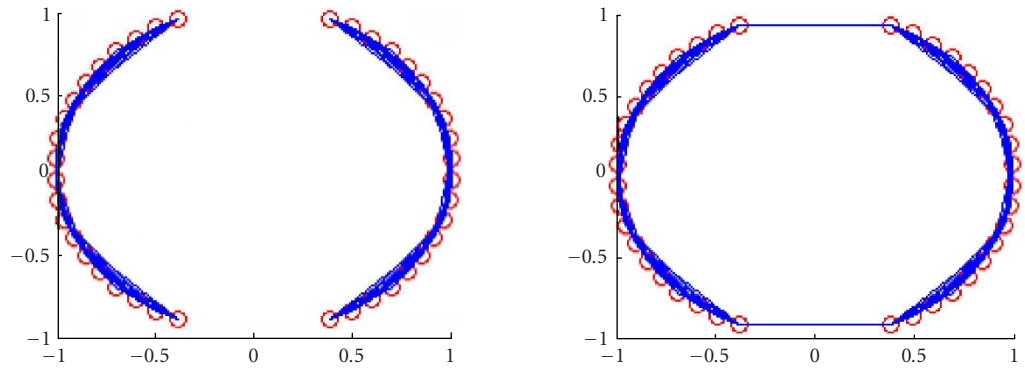


Figure 3: Shortest paths on manifold.

points lie on a submanifold of $\mathbb{R}^n$. In a first step, the Euclidean distance matrix $D_{ij} = d(x_i, x_j)$ is computed and for each $x_i \in X$ a set of neighboring points $Z(x_i)$ is determined by

$$Z(x_i) = \{x_j \mid x_j \in X \text{ is neighbor of } x_j\}. \tag{3}$$

Several variants for defining neighborhood relations can be used, for example, k-nearest neighbors, ε-ball neighborhood. These relations are used to construct the undirected neighborhood path, which will be represented by an $(m \times m)$ adjacency matrix G. The adjacency matrix G is initialized with connections between neighbors, weighted by their Euclidean distance

$$G_{ij} = \begin{cases} d(x_i, x_j) & \text{if } i \in Z(x_j) \text{ or } j \in Z(x_i), \\ \infty & \text{if } x_i \text{ and } x_j \text{ are not neighbored,} \end{cases} \tag{4}$$

where ∞ just denotes that two points are not connected. In this paper, we adopt the k-nearest neighbor algorithm to determine which points are neighbors on the manifold. Since the graph is undirected, the adjacency matrix has to be symmetrized $G_{ij} = \min(G_{ij}, G_{ji})$, which clarifies the symmetry of the neighborhood relation.

In the second step, the geodesic distances for not directly connected points are estimated by the length of the shortest path in G between them. Computationally, this is a standard all-pairs-shortest-path problem for which several algorithms are available. In this paper, a Floyd-Warshall algorithm will be used. If we apply the algorithm to G, the resulting value of G_{ij} is the length of the shortest path and so the approximated geodesic distance between the vertices representing x_i and x_j, if such a path exists.

The accuracy of the estimated intrinsic geometry naturally depends on the density of the samples, errors in the data relative to the manifold, the smoothness of the manifold, and the way the neighborhood graph is constructed.

3.2. Connecting subgraphs

Problems arise when the graph is not connected, which can be a result of noncontinuous sampling, an inadequate construction of neighborhoods or structural breaks in the data. In that case, the construction of neighborhoods has to be modified or additional connections have to be made until the all-pairs-shortest-path procedure gives a connected graph adjacency matrix $(\forall i, j : G_{ij} \neq \infty)$, which then is the matrix of estimated geodesic distance.

Figure 3a shows that the resulting graph is unconnected. In this situation, Tenenbaum chooses only the largest component for embedding. This method works only when the largest component covers most of the training samples. Since this would imply that the remaining points would be discarded, this alternative will loose part of information. When the number of points that each subgraph contains is nearly equal, the largest component cannot substitute for the whole sample space. So, errors may occur. In this paper, we choose to link the subgraphs and take all points into consideration.

To link the subgraphs algorithmically, the simplest method to ensure the connectedness is the "single linkage" method. Suppose x_i and x_j are in two unconnected subgraphs, respectively, which show the smallest Euclidean distance D_{ij} among all unconnected points, we link the two points. Their connection can be penalized by enlarging the corresponding length in G. For example, in this paper, we set its length L_{ij} in proportion to D_{ij} and the maximal value g in every subgraph. Then we have to update the distance in G. Note that this does not require a full run of an all-pairs-shortest-path procedure, since if two points are newly connected, the shortest path between them must include the new connection between x_i and x_j. Thus, for all pairs x_a, x_b of previously unconnected points we have to compute

$$G_{ab} = \min\left(G_{ai} + L_{ij} + G_{jb}, G_{aj} + L_{ij} + G_{ib}\right). \tag{5}$$

If still some $G_{ab} = \infty$ remain, the procedure has to be repeated until all points are in one subgraph.

3.3. Supervised geodesic distance

Until now, geodesic distance has been used as an unsupervised technique. However, we can take the classification information into consideration. The geodesic distance problem can easily be rephrased to use class label information, $\omega_i \in \Omega$ ($|\Omega| = c$) with each x_i, during training. The idea is to find a mapping separating within-class structure from between-class structure. The easiest way to do this is to select the neighbors from just the class that x_i itself belongs to.

A slightly more complicated method would be by using the distance matrix formulation as in (5), but adding distance between samples in different classes:

$$G' = G + \mu \max(G)\Delta, \tag{6}$$

where $\Delta_{jm} = 1$ if $\omega_j \neq \omega_m$, and 0 otherwise. In this formulation, $\mu \in [0, 1]$ controls the amount to which the class information should be incorporated. A dataset is created in which there are c "disconnected" classes, each of which should be connected fairly by geodesic distance. These added degrees of freedom are used to separate the classes.

4. TRAINING SVM WITH GEODESIC DISTANCE

SVMs [8] are a general class of learning architecture inspired from the statistical learning theory that performs structural risk minimization on a nested set structure of separating hyperplanes. Given a training data, the SVM training algorithm obtains the optimal separating hyperplane in terms of generalization error.

The support vector algorithm

Suppose we are given a set of examples $(x_1, y_1), \ldots, (x_l, y_l) \in \mathbb{R}^{\mathbb{N}}$. $y_i \in \{-1, +1\}$. We consider functions of the form $\operatorname{sgn}((w \cdot x) + b)$. In addition, we impose the condition

$$\inf_{i=1,\ldots,l} |(w \cdot x_i) + b| = 1. \tag{7}$$

We would like to find a decision function $f_{w,b}$ with the properties $f_{w,b}(x_i) = y_i$; $i = 1, \ldots, l$. If this function exists, condition (7) implies

$$y_i((w \cdot x_i) + b) \geq 1, \quad i = 1, \ldots, l. \tag{8}$$

In many practical situations, a separating hyperplane does not exist. To allow for possibilities violating equation (8), slack variables are introduced:

$$\xi_i \geq 0, \text{ to get } y_i((w \cdot x_i) + b) \geq 1 - \xi_i, \quad i = 1, \ldots, l. \tag{9}$$

The support vector approaches to minimize the generalization error consists of the following. Minimize

$$\Phi(w, \xi) = (w \cdot w) + \gamma \sum_{i=1}^{l} \xi_i \tag{10}$$

subject to the constraints (9).

It can be shown that minimizing the first term in (10) amounts to minimizing the VC-dimension, and minimizing the second term corresponds to minimizing the misclassification error [8]. The minimization problem can be posed as a constrained quadratic programming (QP) problem. The solution gives rise to a decision function of the form

$$f(x) = \operatorname{sgn}\left[\sum_{i=1}^{l} y_i a_i (x \cdot x_i) + b\right]. \tag{11}$$

Only a small fraction of the a_i coefficients is nonzero. The corresponding pairs of x_i entries are known as support vectors and fully define the decision function. The support vectors are geometrically the points lying near the class boundaries. We use linear kernels for SVM, nonlinear kernels may also be used.

4.1. Geodesic distance for kernel functions

Kernel functions are used in SVM. A possible interpretation of their effects is that they represent dot products in some feature space F, that is,

$$k(x_i, x_j) = \phi(x_i) \cdot \phi(x_j), \tag{12}$$

where ϕ is a map from input (data) space X into F. Another interpretation is to connect ϕ with the regularization properties of the corresponding learning algorithm. These expensive calculations can be reduced significantly by using a suitable function k, leading to decision functions of the form

$$f(x) = \operatorname{sgn}\left[\sum_{i=1}^{l} y_i a_i \cdot k(x, x_i) + b\right]. \tag{13}$$

Most popular kernels are translation invariant kernels

$$k(x_i, x_j) = k(x_i - x_j). \tag{14}$$

Usually, the translation invariant kernels can be expressed as $k(x_i, x_j) = f(d(x_i, x_j))$. Then the distances D can be used to compute a feature space Gram matrix by $K_{ij} = f(D_{ij})$. Compared with Euclidean distance, the geodesic distance reflects the true distance. In this paper, we use the geodesic distance to measure similarity. Thus, kernel function becomes $K_{ij} = f(G_{ij})$. In the experiments, RBF kernel is taken. So,

$$k(x_i, x_j) = \exp\left(-\frac{g(x_i, x_j)}{2\delta^2}\right). \tag{15}$$

4.2. Computing geodesic distance for new observations

When the geodesic distance is used as a data processing function in a classification task, the ability to compute geodesic distance is essential for classifying previously unseen instances. Let X_l be the training set. The task is to find geodesic distance of an additional instance $x_t \in \mathbb{R}^{n_l}$ relative to the points in X_l.

Let G_l be the matrix of estimated geodesic distances computed from X_l. The geodesic distances of the instance x_t towards all points in X_l can be estimated as follows. First, the $(m \times 1)$ vector d_t of Euclidean distances between the rest observation x_t and the points in the training set X_l is calculated. Then, the neighborhood $Z(x_t)$ of x_t is determined using the same neighborhood rule as was used for the initialization of G_l. So, the geodesic distance between x_t and x_i can be estimated by

$$g(x_t, x_i) = \min_{j:x_j \in Z} \left(G_{ij} + d(x_j, x_t)\right). \tag{16}$$

4.3. Computational complexity

What makes geodesic distance algorithm more practical in terms of processing time is that it can be optimized using Dijkstra's algorithm for computation of shortest paths in a graph. Compared to Euclidean distance, the standard geodesic distance algorithm tends to have more computational complexity. One has to calculate the $l \times l$ shortest-path distance matrix G_l. The simplest way is Floyd's algorithm with complexity $O(l^3)$. Dijkstra's algorithm is very significant, because this optimization reduces the processing time from the order of hours to the order of minutes, even to seconds. Such a reduction in time is explainable by observing that the time complexity for Dijkstra's algorithm is $O(l \log l + E)$, where l is the number of vertices and E is the number of edges. For meshes with $O(10^3)$ vertices, exact geodesics can be computed in about 25 seconds on a 633 MHz Celeron II PC.

Moreover, training an SVM requires the solution of a very large QP optimization problem. For datasets of $O(10^3)$ samples, it will take nearly 163 seconds to solve the QP problem. So, the computational time of geodesic distance has a small part in the whole training time of SVM. Once the training process is finished, the SVM can be used in online classification tasks. One will have to compute the geodesic distances of a testing sample to all of the training samples. Instead of recomputing the geodesic distance matrix, Section 4.2 introduces a new method to solve this problem and speeds the computation greatly.

5. NUMERICAL RESULTS AND COMPARISONS

Experiment 1 (a two-spiral problem). The two-spiral problem [9] is a well-known benchmark problem for testing the quality of neural network classifiers. In this experiment (Figure 4), we illustrate an using modified geodesic distance on a two-spiral problem. The training data are shown on Figure 4 with two classes indicated by "○" and "∗" (100 points with 50 for each class) in a two dimensional input space. Points in between the training data located on the two spirals are often considered as test data for this problem, but are not shown on the figure. The excellent generalization performance is clear from the decision boundaries shown on the figure. In this case, $\delta = 1$ and $\gamma = 10$ were chosen as parameters. This experiment shows that the SVM using modified geodesic distance can work well on complex datasets.

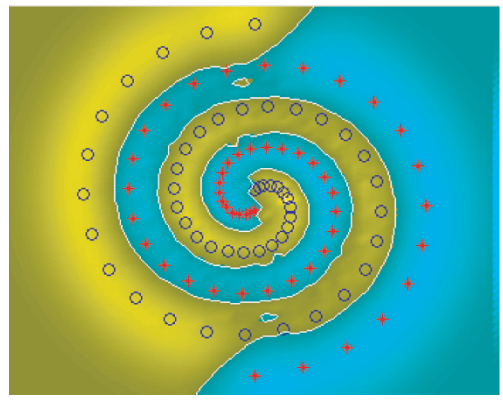


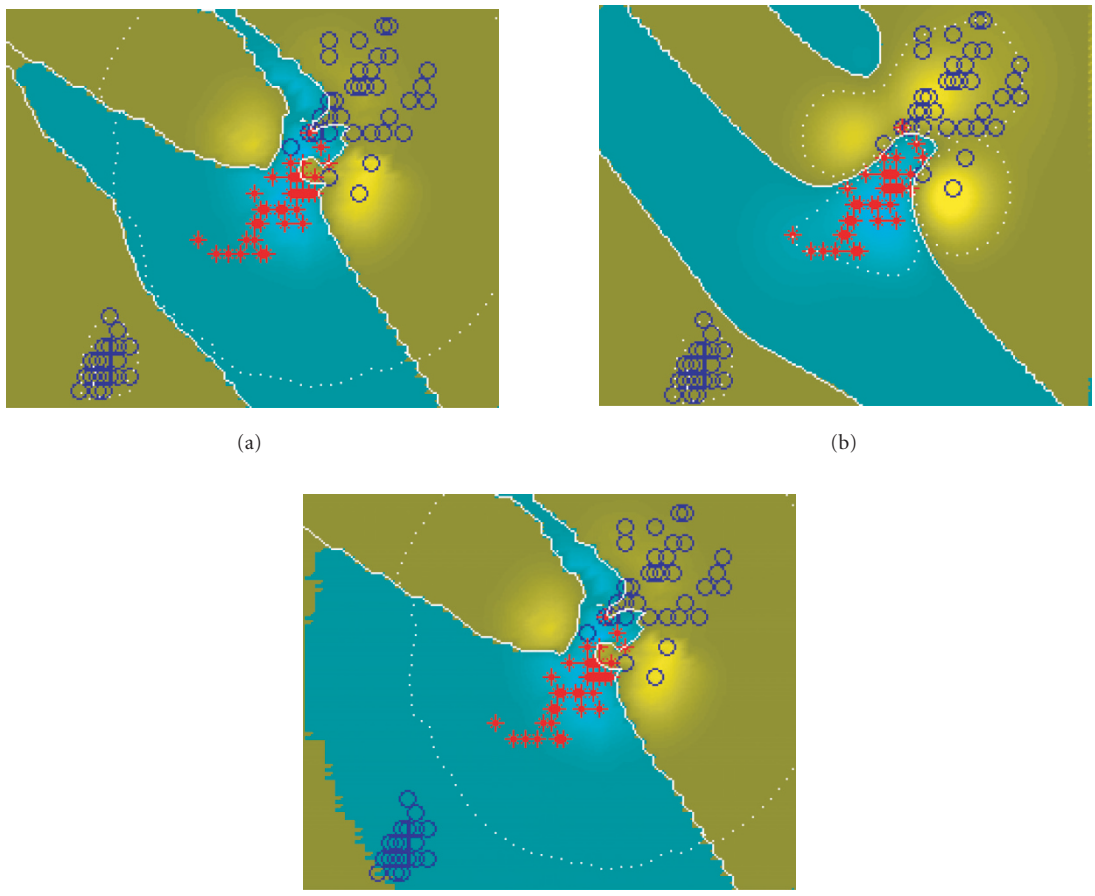
Figure 4: A two-spiral classification problem with the two classes indicated by "○" and "∗."

Experiment 2 (iris dataset problem). In the second experiment, we compare the feature of the SVM using modified geodesic distance proposed in this paper with the SVM using unmodified geodesic distance. Here, the unmodified geodesic distance means the geodesic distance implemented by Tenenbaum. The iris dataset problem [10] is also a well-known benchmark problem for testing the quality of neural network classifiers. In this experiment (Figure 5), we illustrate the results of SVMs using modified geodesic distance (Figure 5a), unmodified geodesic distance (Figure 5b) and Euclidean distance (Figure 5c) on iris dataset problem. The training data are showing on Figure 5 with two classes indicated by "○" and "∗" in a two-dimensional input space. Like Experiment 1, we take points in between the training data located on the plane as test data for this problem. In this case, $\delta = 0.5$ and $\gamma = 20$ were chosen as parameters.

From Figure 5b we can see that many points are misclassified by the SVM using unmodified geodesic distance. For the iris dataset, there are two subgraphs when constructing geodesic distance path. The unmodified geodesic distance, implemented by Tenenbaum, is to choose only the largest subgraph for learning. So, only part of training examples is used, that is the reason why some important information contained in training examples may be lost. That leads to a disappointing result.

In our modified geodesic distance algorithm, we connect the subgraphs ex-post. As a result, we take all training examples into consideration when using SVM. From Figure 5a we can see that the SVM using modified geodesic distance can correctly classify the examples that is misclassified in Figure 5b. This experiment shows that the SVM based on modified geodesic distance can take full advantage of all training examples and gain better classification results than that of the SVM based on unmodified geodesic distance.

From Figures 5a and 5c, we can see that when the structure of the training samples is simple, the SVM using modified geodesic distance can gain comparable results as that of the SVM using Euclidean distance. But when training samples are in high-dimensional space, the structure is usually very complicated and the results are very different. Experiment 3 shows this situation.



(a) (b)

(c)

Figure 5: Iris dataset classification problem with the two classes indicated by "○" and "∗."

Experiment 3 (real datasets). In order to compare the classification accuracy of the SVM using modified geodesic distance proposed in this paper, the SVM using Euclidean distance on massive datasets, and the SVM using unmodified geodesic distance, we test this algorithm on three real-world datasets. Here, the proposed SVM means the SVM using modified geodesic distance proposed in this paper, the classical SVM means the SVM using Euclidean distance, and the geodesic SVM means the SVM using unmodified geodesic distance.

In this experiment, we adopt the same datasets used in [11], that is, we choose the Boston housing and the Abalone dataset from the UCI repository [Blake et al., 1998] and the USPS database of handwritten digits. The first data are of size 506 (350 training, 156 testing), the Abalone dataset of size 4177 (3000 training and 1177 testing). In the first two cases, the data were rescaled to zero mean and unit variance, coordinate-wise, while the USPS dataset remained unchanged. Its data size is 40337 (29463 training and 10874 testing). Finally, the gender encoding in Abalone (male/female/infant) was mapped into $\{(1,0,0),(0,1,0),(0,0,1)\}$.

Table 1 illustrates testing error rate, training set size, and the number of support vectors for classical SVM and the proposed SVM algorithms. Here, we can see that in almost every dataset, the new proposed SVM can gain lower testing error rate than the classical SVM and the geodesic SVM. When a dataset contains nonlinear structure, the geodesic distance reflects intrinsic relations in the data and contains properties of the curvature of the manifold. Therefore, more prior knowledge about the training data is considered. Especially in large amount of datasets as shown in Table 1, the new proposed SVM can gain higher classification accuracy than that of the other two methods.

6. CONCLUSIONS

We successfully demonstrate the generation of a new SVM-kernel by substituting distance measures in distance-based kernels. We define modifications of geodesic distance, which reflects the true distance on a manifold relative to Euclidean distance. We present a new algorithm for support vector machines based on geodesic distance. The applications to a difficult two-spiral classification problem, iris dataset classification problem, and real datasets problem show that excellent generalization performance can be obtained using geometric distance-based SVM.

Table 1: Comparison on various datasets.

Dataset	SVM	Classification accuracy (%)	Training set size	Number of SVs	Training time (s)	SVM parameters	
						δ	γ
Boston housing	Classical SVM	92.74 ± 0.86	350	167 ± 6	5.8 ± 0.3	150	0.5
	Geodesic SVM	93.26 ± 0.47	350	153 ± 4	6.3 ± 0.2	150	0.5
	Revised SVM	94.42 ± 0.08	350	143 ± 10	6.7 ± 0.4	150	0.5
Abalone	Classical SVM	87.44 ± 1.01	3000	1317 ± 15	378.8 ± 13.1	500	20.0
	Geodesic SVM	82.17 ± 2.13	3000	932 ± 26	307 ± 21.6	500	20.0
	Revised SVM	93.37 ± 1.24	3000	1277 ± 11	462.3 ± 24.3	500	20.0
USPS	Classical SVM	91.1 ± 0.64	29463	11533 ± 5	8221.9 ± 147.8	700	5.0
	Geodesic SVM	83.6 ± 0.81	29463	9677 ± 19	7994 ± 152.4	700	5.0
	Revised SVM	96.5 ± 0.31	29463	9874 ± 5	9186.3 ± 276.4	700	5.0

ACKNOWLEDGMENTS

This work was Supported by the National Natural Science Foundation of China under Grant no. 50174038. The second author is now supported by the National Natural Science Foundation of China under Grant no. 30170274.

REFERENCES

[1] V. N. Vapnik, *Statistical Learning Theory*, John Wiley, New York, NY, USA, 1998.

[2] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*, Cambridge University Press, Cambridge, UK, 2000.

[3] B. Schölkopf, *Support Vector Learning*, R. Oldenbourg Verlag, Munich, Germany, 1997.

[4] D. R. Wilson and T. R. Martinez, "Improved heterogeneous distance functions," *Journal of Artificial Intelligence Research*, vol. 6, pp. 1–34, 1997.

[5] C. Schaffer, "A conservation law for generalization performance," in *Proc. 11th International Conference on Machine Learning*, pp. 259–265, Morgan Kaufmann, San Mateo, Calif, USA, 1994.

[6] J. B. Tenenbaum, V. D. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, 2000.

[7] B. G. Batchelor, *Pattern Recognition: Ideas in Practice*, Plenum Press, New York, USA, 1978.

[8] C. J. C. Burges, "A tutorial on support vector machines," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[9] S. Ridella, S. Rovetta, and R. Zunino, "Circular backpropagation networks for classification," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 84–97, 1997.

[10] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[11] A. J. Smola and B. Schölkopf, "Sparse greedy matrix approximation for machine learning," in *Proc. 17th International Conference on Machine Learning*, pp. 911–918, Morgan Kaufman, San Francisco, Calif, USA, 2000.

Quan Yong was born in Hei Longjiang province, China, in 1976. He received his B.S. and M.S. degrees in mechanical & electrical engineering from Harbin Institute of Technology, Harbin, China. Since 2000, he has been a Ph.D. candidate at the Institute of Image Processing and Pattern Recognition, Shanghai Jiaotong University, China. His current research areas include machine learning and data mining.

Yang Jie was born in 1964. He is a Professor and Doctoral Supervisor at the Institute of Image Processing and Pattern Recognition, Shanghai Jiaotong University, China. His research interests are in the areas of image processing, pattern recognition, and data mining and application.