Geodesic Distance for Support Vector Machines1)

QUAN Yong YANG Jie
(*Institute of Image Processing and Pattern Recognition, Shanghai Jiaotong University, Shanghai* 200030)
(E-mail: quanysjtu@sjtu.edu.cn)

Abstract When dealing with pattern recognition problems one encounters different types of prior knowledge. It is important to incorporate such knowledge into classification method at hand. A very common type of prior knowledge is many data sets 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 support vector machines which incorporate geodesic distance and therefore are applicable in cases such transformation invariance is known. Experiments results show that the performance of our method is comparable to that of other state-of-the-art method.

Key words 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 co-workers[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.

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

In 2000, Tenenbaum proposed geodesic distance[5]. 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 geodesic distance. We first propose an improved geodesic distance to eliminate isolated embeddings. And then the geodesic distance is applied in 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[6] 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 $d(\boldsymbol{x}, \boldsymbol{y})$ is defined as

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

1) Supported by National Natural Science Foundation of P. R. China (50174038, 30170274)
Received June 20, 2003; in revised form March 23, 2004

where r is the Minkowski factor for the norm. Particularly, when r is set as 2, it is the well known Euclidean distance. A variant of the Minkowski function, the weighted Minkowski distance function, has also been used to measure point similarity. By assigning each feature a weighting coefficient, the weighted Minkowski distance function is defined as

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

However, in some cases, Minkowski distance does not reflect the intrinsic similarity.

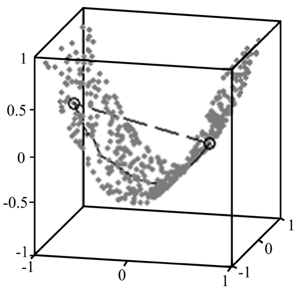

Fig. 1 Distance between two points on a manifold

As Fig. 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 intrinsic similarity which is illustrated by 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 and for faraway points, the geodesic distance is estimated by the length of the shortest path through neighboring points.

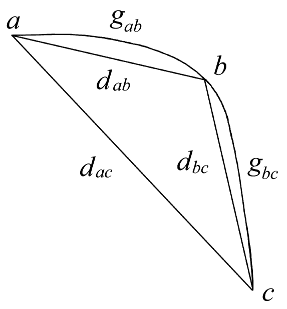

Fig. 2 Inaccuracy of Euclidean distance compared with geodesic distance

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. 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 distances. The aim of describing relations between the points 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 from this example, the geodesic distance between neighboring points can be approximated fairly well by their Euclidean distance, so that $\hat{g}_{ab} = \mathrm{d}_{\mathrm{ab}}$ 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 distance 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 a $(m \times n)$ matrix representing m samples in n dimensions and it can be assumed that the points lie on a submanifold of R^n. In the 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

$$Z(x_i) = \{x_j | x_j \in X \text{ is neighbor of } x_j\}$$

Several variants for defining neighborhood relations can be used. 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

$$G_{ij} = \begin{cases} d(x_i, x_j) & \text{if } i \in Z(x_j) \text{ or } j \in Z(x_i) \\ \infty & \text{if } x_i \text{ and } x_j \text{ are not neighbored} \end{cases} \tag{3}$$

where ∞ just denotes that two points are not connected. 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 the points that are not connected directly 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.

3.2 Connecting subgraphs

Problems arise when the graph is not connected as a result of uncontinuous 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 adjacent matrix ($\forall i, j : G_{ij} \neq \infty$), which then is the matrix of estimated geodesic distance.

Fig. 3 (a) shows the resulting graph that is unconnected. In this situation, Tenenbaum chooses only the largest component for embedding. Since this method works only when the largest component covers most of the training samples with the implication that the remaining points would be discarded, this alternative will lose part of information. When each subgraph contains nearly equal number of points, the largest component can not 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.

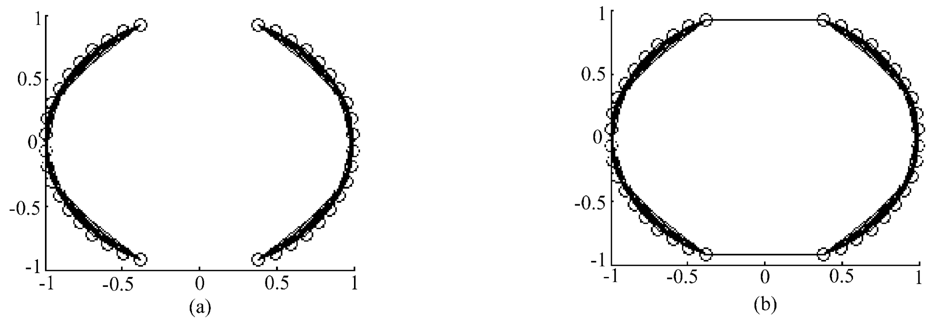

Fig. 3 Shortest paths on manifold

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 and 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 connected yet, 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

$$G_{ab} = \min(G_{ai} + L_{ij} + G_{jb}, G_{aj} + L_{ij} + G_{ib}) \tag{4}$$

If some $G_{ab} = \infty$ still remains, 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 just from the class that x_i itself belongs to.

A slightly complicated method is to use the distance matrix formulation as in (4), but it adds distance between samples in different classes

$$G' = G + \mu \max(G) \triangle$$

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

4 Training SVM with geodesic distance

Support vector machines[7] are a general class of learning architecture inspired from statistical learning theory that perform 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 $(x_1, y_1), \ldots, (x_1, y_1) \in R^N$. $y_i \in \{-1, +1\}$. We consider functions of the form $\mathrm{sgn}((w \times x) + b)$, in addition we impose the condition

$$\inf_{i=1,\cdots,l} |(w \times x_i) + b| = 1 \tag{5}$$

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 (5) implies

$$y_i((w \times x_i) + b) \geqslant 1, \quad i = 1, \ldots, l \tag{6}$$

In many practical situations, a separating hyperplane does not exist. To allow for possibilities violating Equation (6), slack variables are introduced

$$\xi_i \geqslant 0, \text{ to get } y_i((w \times x_i) + b) \geqslant 1 - \xi, \quad i = 1, \ldots, l \tag{7}$$

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

$$\text{Minimize} \quad \Phi(w, \xi) = (w \times w) + \gamma \sum_{i=1}^{l} \xi_i \tag{8}$$

subject to the constraints (7).

It can be shown that minimizing the first term in (8) amounts to minimizing the VC-dimension, and minimizing the second term corresponds to minimizing the misclassification error[7]. The minimization problem can be posed as a constrained quadratic programming (QP) problem. The solution gives rise to a decision function of the form $f(x) = \mathrm{sgn}\Big[\sum_{i=1}^{l} y_i a_i (x \times x_i) + b\Big]$.

Only a small fraction of the a_i coefficients are non-zero. 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. Nonlinear kernels, as well as linear kernels, can be used for SVM.

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, *i.e.*,

$$k(x_i, x_j) = \phi(x_i) \times \phi(x_j) \tag{9}$$

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. Most popular kernels are translation invariant kernels

$$k(x_i, x_j) = k(x_i - x_j) \tag{10}$$

Usually, this kind of 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 $K_{ij} = f(D_{ij})$. Compare with Euclidean distance, geodesic distance reflects the intrinsic similarity. In this paper, we use geodesic distance to measure similarity. Thus kernel function becomes $K_{ij} = f(G_{ij})$. In the experiments, RBF kernel is taken. So $k(x_i, x_j) = \exp(-\frac{g(x_i, x_j)}{2\delta^2})$.

4.2 Computing geodesic distance for new observations

When geodesic distance is used as data processing 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 $x_t \in R^{n_l}$ 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 test 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 is used for the initialization of G_l. So the geodesic distance between x_t and x_i can be estimated by

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

4.3 Computational complexity

What makes geodesic distance algorithm more practical in terms of processing time is that it can be optimized by using Dijkstra′s algorithm for the computation of shortest paths in a graph. Compared with 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 a complexity of $O(l^3)$. 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 quadratic programming (QP) optimization problem. For datasets of $O(10^3)$ samples, it will cost 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 re-computing the geodesic distance matrix, section 4.2 introduces a new method to solve this problem and speeds up the computation greatly.

5 Numerical results and comparisons

The revised SVM algorithm is tested against a standard SVM learning algorithm on a series of benchmarks. Both algorithms are written in C++, using microsoft Visual C++ 6.0 compiler. Both algorithms are run on a 633 MHz Celeron II processor running Windows 2000. The CPU time for both algorithms is measured. The CPU time covers the execution of the entire algorithm, including kernel evaluation time, but excluding file I/O time.

Experiment 1. A two-spiral problem The two-spiral problem[8] is a well-known benchmark problem for testing the quality of neural network classifiers. In this experiment (Fig. 4) we illustrate a support vector machine using geodesic distance on a two-spiral problem. The training data are shown in Fig. 4 with two classes indicated by 'o' 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 in the figure.

The excellent generalization performance is clear from the decision boundaries shown in the figure. In this case $\delta = 1$ and $\gamma = 10$ are chosen as parameters.

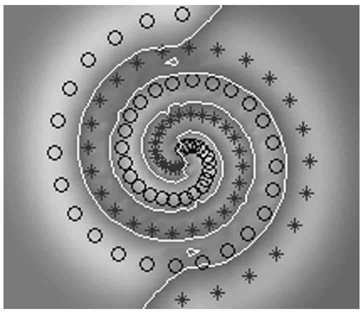

Fig. 4 A two-spiral classification problem with the two classes indicated by 'o' and '$*$'

Experiment 2. Real data sets In order to compare the classification accuracy of the new methods on massive data sets, we test this algorithm on three real-world data sets.

In this experiment, we adopt the same data sets used in [9]. 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 is 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 was 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 Comparison on various data sets

Data set	Classification accuracy %	Training set size	Number of SVs	Training time (s)	SVM parameters δ	γ
	Classical SVM Revised SVM	Both SVMs	Classical SVM Revised SVM	Classical SVM Revised SVM	Both SVMs	
Boston Housing	92.74 ± 0.86 94.42 ± 0.08	350	167 ± 6 143 ± 10	5.8 ± 0.3 6.7 ± 0.4	150	0.5
Abalone	87.44 ± 1.01 93.37 ± 1.24	3000	1317 ± 15 1277 ± 11	378.8 ± 13.1 462.3 ± 24.3	500	20.0
USPS	91.1 ± 0.64 96.5 ± 0.31	29463	11533 ± 5 9874 ± 5	8221.9 ± 147.8 9186.3 ± 276.4	700	5.0

Table 1 illustrates testing error rate, training set size and number of support vectors for classical SVM and the revised SVM algorithms. Here we can see that in almost every data set, the new revised SVM can gain lower testing error rate than classical SVM.

Using the new method introduced in Section 4.2, little time is needed to compute geodesic distance from testing sample to training samples and its influence can be neglected. Here, we concentrate on the difference of training time between the revised SVM and classical SVM. It can also be seen that geodesic distance leads to a slightly increase of about 16.43 percent in training time. One should determine the trade-off between reducing the training time and increasing classification accuracy. When high accuracy is required, one should consider the revised SVM firstly. Otherwise classical SVM or even some linear classifiers should be employed when less training time is more important. Fortunately, in most cases, the revised SVM can be used in the same situation as classical SVM, where a relative long training time is accepted. Once the training process is completed, the proposed one can be used in online classification problem to gain high accuracy.

6 Conclusions

High dimensional data sets, for example human face expression data set, are usually on some sort of manifolds. Application of the geodesic distance to these data sets produces much better results. This algorithm is most efficient on these data sets because it accurately reflects their intrinsic similarity. When datasets distribute irregularly in low dimensional space, the classification accuracy of revised SVM algorithm is still comparable to that of classical SVM algorithm.

We successfully demonstrate the generation of new SVM-kernel by substituting distance-measures in distance-based kernels. We define modifications of geodesic distance, which reflects the intrinsic 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 and real data sets problem show that excellent generalization performance can be obtained using geometric distance based support vector machine.

References

1 Vapnik V. Statistical learning theory. New York: John Wiley, 1998

2 Zhang Xue-Gong. Introduction to statistical learning theory and support vector machines. *Acta Automatica Sinica*, 2000, **1**(1): 32~42

3 Vapnik V. The nature of statistical learning theory. New York: Springer-Verlag, 1995, 422~426

4 Wilson D Randall, Martinz Tony R. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 1997, **6**: 1~34

5 Tenenbaum J B, Silva V d, Langford J C. A global geometric framework for nonlinear dimensionality reduction. *Science*, 2000, **290**(5500): 2319~2323

6 Batchelor, Bruce G. Pattern recognition: ideas in practice. New York: Plenum Press, 1978: 71~72

7 Burges C J C. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 1998, **2**(2): 121~167

8 Ridella S, Rovetta S, Zunino R. Circular back-propagation networks for classification. *IEEE Transactions on Neural Networks*, 1997, **8**(1): 84~97

9 Smola A J, Schölkopf B. Sparse greedy matrix approximation for machine learning. In: Langley, P. In: Proceeding of 17th International Conference on Machine Learning. San Francisco, California: Morgan Kauffman. 2000, 911~918, http://mlg.anu.edu.au/ smola/publications.html

QUAN Yong Received his bachelor and master degrees from Harbin Institute of Technology, P. R. China. Now he is a Ph. D. candidate at the Institute of Image Processing and Pattern Recognition, Shanghai Jiaotong University, P. R. China. His research interests include machine learning and data mining.

YANG Jie Professor at the Institute of Image Processing and Pattern Recognition, Shanghai Jiaotong University, P. R. China. His research interests include image processing, pattern recognition, data mining and application.