Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. V (2010), No. 4, pp. 469-476

Hierarchical and Reweighting Cluster Kernels for Semi-Supervised Learning

Z. Bodó, L. Csató

Zalán Bodó, Lehel Csató
Department of Mathematics and Computer Science
Babeş–Bolyai University
Kogălniceanu 1, 400084 Cluj-Napoca, Romania
E-mail: {zbodo, lehel.csato}@cs.ubbcluj.ro

Abstract: **Abstract:** Recently semi-supervised methods gained increasing attention and many novel semi-supervised learning algorithms have been proposed. These methods exploit the information contained in the usually large unlabeled data set in order to improve classification or generalization performance. Using data-dependent kernels for kernel machines one can build semi-supervised classifiers by building the kernel in such a way that feature space dot products incorporate the structure of the data set. In this paper we propose two such methods: one using specific hierarchical clustering, and another kernel for reweighting an arbitrary base kernel taking into account the cluster structure of the data.

Keywords: Kernel methods, semi-supervised learning, clustering

1 Introduction

Extracting information from large data collections is an important research topic in mathematical modeling: it helps designing automated inference procedures with limited or no user intervention [9]. The resulting algorithms are used in various domains like bioinformatics or natural language processing, both involving the processing of large data sets. Data sets are usually labeled; this manual labeling is done before the automated information extraction procedure takes place. The limitation of the procedure is that the *total number* of items cannot be labeled. In this scenario the semi-supervised learning (SSL) develops methods that handle partially labeled data sets where only a small portion has labels, the rest of it is collected but unlabeled. Since unlabeled data is ubiquitous, in semi-supervised learning we jointly handle both the labeled and the unlabeled parts to improve the performance of the algorithm. In the following we only consider semi-supervised classification, *i.e.* those methods that assign single or multiple labels to a given input.

To use the unlabeled part of the data set, some assumptions have to be made [5]: (i) smoothness assumption, (ii) cluster assumption, (iii) manifold assumption. Most SSL methods are built on the top of the supervised algorithms by using these assumptions together with estimates of the input distribution. We use the input distribution to define a change in the input metric, leading to a modified distance between items. We study SSL methods comprising the following two steps: first we determine the new distance – dot product or kernel function – between the learning examples, and in the second step with a supervised method we obtain the decision function by using the new distance obtained in the first step.

In this paper we focus on methods that exploit the induced changes in distances and characterize this induced distance measure with kernels [3]. Kernel methods constitute a powerful tool to rewrite a linear algorithm into a non-linear one. They are based on a symmetric positive semi-definite (kernel) function, that is a dot product in a high-dimensional space [10]. The kernel in the first step of the generic SSL method mentioned above is *data-dependent*. Data-dependent kernels combine kernel algorithms and semi-supervised learning by providing a new representation for the examples that uses both the labeled and the unlabeled parts of the data set. A formal definition of the data-dependent kernel is the following: if $D_1 \neq D_2$,

$$k(\mathbf{x}_1, \mathbf{x}_2; D_1) \bumpeq k(\mathbf{x}_1, \mathbf{x}_2; D_2)$$

where "$\bumpeq$" reads as "not necessarily equal" and the semicolons denote conditioning. It is important that these kernel functions are conditioned on the data sets, nevertheless we omit it in the following: it will be clear from the context if a kernel is data-dependent.

We propose two data-dependent kernels in this paper: (i) a kernel using the distances induced by hierarchically clustering the labeled and unlabeled data; (ii) a reweighting kernel based on the Hadamard product of some base kernel matrices.

The paper is structured as follows: Section 2 outlines the notations, in Section 3 the hierarchical and graph-based hierarchical cluster kernels for semi-supervised classification are described. In Section 4 we introduce the reweighting kernels based on data clustering and kernel combination. In Section 5 we present the experiments and results and in Section 6 we present the conclusions and discussions on the proposed kernels.

2 Notation

We denote with $D = \{(\mathbf{x}_i, y_i) \,|\, i = 1, 2, \ldots, \ell\} \cup \{\mathbf{x}_i \,|\, i = \ell+1, \ldots, \ell+u\}$ the training data, with the first set being the labeled and the second the unlabeled data set. We further assume $\mathbf{x}_i \in X$ with X metric space, $y_i \in Y$, and the set of labels is of finite cardinality, *i.e.* $|Y| < \infty$. A key assumption is that the size of labeled data is much smaller than the available unlabeled part, *i.e.* $\ell \ll u$. In the paper N denotes the size of the entire training set, $N = \ell + u$. We use the scalar K to denote the number of clusters, where needed. Boldface lowercase letters denote vectors, boldface capitals are matrices, all other variables are scalars; $\mathbf{A}'$ denotes the transpose. For a matrix $\mathbf{A}$, A_{ij} is its element in the i-th row and j-column, and $\mathbf{A}_{i\cdot}$ and $\mathbf{A}_{\cdot j}$ denote the vectors corresponding to the i-th row and j-th column.

3 Hierarchical cluster kernels

In this section we introduce the proposed hierarchical cluster kernels. We propose the use of distances induced by different clustering algorithms instead of the original distance measure in the input space. If unlabeled data is added to the relatively small labeled data set, we expect that the *new* distance, obtained via clustering and the use of unlabeled data, induces a better representational space for classification. For clustering we use special hierarchical clustering techniques – the ones that result in ultrametric distance matrices – leading to positive semi-definite kernel matrices.

Our method is based on the connectivity kernel [8] and we extend on this kernel construction by involving the unlabeled data and allowing any hierarchical clustering method leading to ultrametric distance matrices. For data sets where the manifold assumption is expected to hold, we construct the hierarchical cluster kernel using distances induced by the k-NN and ε-NN data graphs.

3.1 Hierarchical clustering and ultrametricity

Hierarchical clustering builds a tree in successive steps, where the nodes of the tree represent nested partitions of the data, in contrary to partitional clustering methods, which result in a single partition. For the proposed hierarchical cluster kernel we use special agglomerative clustering methods. To fully specify a hierarchical clustering algorithm, cluster similarities have to be measured; these are called *linkage distances*. Based on the choice of the linkage distance measuring distances between clusters in agglomerative clustering, one can design a large variety of clustering methods.

$$\begin{aligned}
\text{single linkage:} \quad D(C_1,C_2) &= \min\{d(\mathbf{x}_1,\mathbf{x}_2) \mid \mathbf{x}_1 \in C_1, \mathbf{x}_2 \in C_2\} && (1)\\
\text{complete linkage:} \quad D(C_1,C_2) &= \max\{d(\mathbf{x}_1,\mathbf{x}_2) \mid \mathbf{x}_1 \in C_1, \mathbf{x}_2 \in C_2\} && (2)\\
\text{average linkage:} \quad D(C_1,C_2) &= \frac{1}{|C_1||C_2|} \sum_{\mathbf{x}_{1i}\in C_1} \sum_{\mathbf{x}_{2j}\in C_2} d(\mathbf{x}_{1i},\mathbf{x}_{2j}) && (3)
\end{aligned}$$

The linkage distances are based on $d(\mathbf{x}_1,\mathbf{x}_2)$, the *pointwise distance* in the input space that usually is the Euclidean distance $d(\mathbf{x}_1,\mathbf{x}_2) = \|\mathbf{x}_1 - \mathbf{x}_2\|_2$.

In this paper we experiment only with the three linkage distances presented above; for a detailed discussion of these and other distances see [7]. All these three methods lead to *ultrametric* hierarchical clustering. The property is used for constructing positive semi-definite kernels from clusters: suppose that we choose to merge three clusters, C_1, C_2 and C_3 in the following order: we first merge C_1 with C_2 resulting in C_{12}, and then we merge it with C_3. Now if

$$\left.\begin{array}{l} D(C_1,C_2) \le D(C_1,C_3) \\ \text{and} \\ D(C_2,C_1) \le D(C_2,C_3) \end{array}\right\} \quad \text{then} \quad D(C_1,C_2) \le D(C_{12},C_3) \qquad (4)$$

Based on an agglomerative clustering method that uses ultrametric linkage distance, we can define an ultrametric distance matrix, based on that a kernel function that can be used for a better representation.

3.2 The connectivity kernel

Our method of constructing hierarchical cluster kernels is based on [8]. The authors propose a two-step clustering: map the points to a new representational space based on the effective dissimilarities, and cluster them using the new representation.

The method is an approximation to pairwise clustering. To compute the effective dissimilarities used in pairwise clustering, the authors build a graph of the data; they assume that on the path between two points belonging to different clusters there will be an edge with large weight, representing the *weakest* link on the path. The effective dissimilarity will be represented by this value. They approximate the effective dissimilarities using a Kruskal-style algorithm [8].

Our method can be viewed as a generalization of the connectivity kernel, since if the ultrametric property is satisfied, we can use an arbitrary linkage distance when performing the agglomerative clustering. Moreover we propose to use the kernel in semi-supervised learning settings, when only a small portion of the data labels is known, and we propose a manifold-based extension of the kernel too.

3.3 Constructing the kernel

The hierarchical clustering results in a dendrogram, whose nodes are labeled with the distance between the clusters that were merged at the respective node. We build a distance matrix by taking the label attached to the lowest common ancestor of the points in the tree. In order to transform distances to dot products we use a method similar to multi-dimensional scaling (MDS) [2]:

$$\mathbf{K} = -\frac{1}{2}\mathbf{JMJ} \qquad \text{with} \qquad \mathbf{J} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}'$$

where $\mathbf{M}$ contains the squared distances based on the dendrogram and $\mathbf{J}$ is the centering matrix built from the identity matrix $\mathbf{I}$ and the tensor product of the vector with all elements 1. The resulting matrix contains the dot products between a set of vectors $\{\mathbf{z}_i\}_{i=1}^N$ with squared Euclidean distances $\|\mathbf{z}_i - \mathbf{z}_j\|_2^2 = M_{ij}$ [8].

In what follows, we construct the cluster kernel using linkage distances from hierarchical clustering. Thus we map the points to a feature space where the pointwise distances are equal to the cluster distances in the input space. The steps are shown in Algorithm 1.

Algorithm 1 Hierarchical cluster kernel

```
1: Perform an agglomerative clustering with ultrametric linkage distances from Section 3.1 on the labeled and unlabeled data.
2: Define matrix $\mathbf{M}$ with $M_{ij}$ = linkage distances of $\mathbf{x}_i$ and $\mathbf{x}_j$; $M_{ii} = 0$.
3: Define the kernel matrix as $\mathbf{K} = -\frac{1}{2}\mathbf{JMJ}$.
```

The resulting kernel **K** is obviously data-dependent. We use the unlabeled data in the clustering step to determine *better* pointwise distances, leading to the kernel; we expect to obtain better similarities then using only the labeled part. It is important that in order to compute the kernel function for the test set we include them into the unlabeled set. This means that if the test point is unavailable at training time, the whole clustering process should be repeated, slowing down the classification. To efficiently compute the kernel for unseen points is left as a future realization and is discussed in Section 6.

3.4 Hierarchical cluster kernel with graph distances

In building the hierarchical cluster kernel, we only used the cluster assumption. Here we extend the above kernel to also exploit the manifold assumption, mentioned in Section 1, using a graph-based hierarchical cluster kernel. We approximate distances by using shortest paths based on k-NN or ε-NN graphs, similar to ISOMAP [11]. In this process we substitute the graph distances for the pointwise distances $d(\cdot,\cdot)$. The result is that the hierarchical clustering algorithm is preceded with the steps shown in Algorithm 2.

Algorithm 2 Graph-based hierarchical cluster kernel

```
-2: Determine the $k$-nearest neighbors or an $\varepsilon$-neighborhood of each point, and let distances to other points equal to $\infty$.
-1: Compute shortest paths for every pair of points – using for example Dijkstra's algorithm.
 0: Use these distances for the pointwise distance in eqs. (1), (2), or (3).
```

We deliberately started the numbering from -2 to emphasize that these steps *precede* the algorithm from the previous subsection. We emphasize that these steps are optional: should only be used if the manifold assumption holds on the data set.

We use here the shortest path distances computed on the k-nearest neighbor or the ε-neighborhood graph of the data, thus – if the data lie on a low-dimensional manifold – approximating pointwise distances on this manifold.

The graph built from the k-nearest or the ε-neighborhoods may contain several disconnected components, for example if k or ε is too small. If this scenario happens we use a method similar to the one described in [13] for building a connected graph.

4 Reweighting kernels

Combining kernels to improve classification performance was thoroughly studied [10]. In the following – based on kernel combination – we propose three techniques to reweight a base kernel using the cluster assumption of semi-supervised learning.

We make use of the following properties: for any $\mathbf{K}_1$ and $\mathbf{K}_2$ positive semi-definite matrix and any positive scalar value $a > 0$, the following combinations are positive semi-definite matrices:

$$\mathbf{K}_1 + \mathbf{K}_2, \quad a\mathbf{K}_1, \quad \mathbf{K}_1 \odot \mathbf{K}_2,$$

where $\odot$ denotes the Hadamard, or direct product. We develop techniques that reweight the kernel matrix by exploiting the cluster structure of the training data. Thus, if two points are in the same cluster, their similarity obtains a high weight, while lying in different clusters induces a lower weight; the resulting kernel is called the *reweighting kernel*, or $k_{\mathrm{rw}}(\mathbf{x}_1, \mathbf{x}_2)$. The similarity weights are combined with the values of the base kernel $k_{\mathrm{b}}(\mathbf{x}_1, \mathbf{x}_2)$, thus forming the final kernel matrix. To sum up, the new cluster kernel is

$$k(\mathbf{x}_1, \mathbf{x}_2) = k_{\mathrm{rw}}(\mathbf{x}_1, \mathbf{x}_2)\, k_{\mathrm{b}}(\mathbf{x}_1, \mathbf{x}_2)$$

where $k_{\mathrm{rw}}(\cdot,\cdot)$ is the reweighting and $k_b(\cdot,\cdot)$ is the base kernel. In matrix form it can be written as

$$\mathbf{K} = \mathbf{K}_{\mathrm{rw}} \odot \mathbf{K}_{\mathrm{b}}$$

We are faced with two problems in the construction of the above cluster kernel: (i) the reweighting kernel must be positive semi-definite, (ii) the base kernel matrix has to be positive semi-definite and *positive*. The first requirement is obvious: it is needed to guarantee the positive semi-definiteness of the resulting kernel. The second condition is crucial, since for negative values in the base kernel matrix a quite different reweighting should be performed. To avoid complications due to negativity, we require a positive base kernel, $k_b(\mathbf{x}_1, \mathbf{x}_2) \geq 0$.

The bagged cluster kernel, proposed in [12], reweights the base kernel values by the probability that the points belong to the same cluster. For computing this probability the bagged cluster kernel uses k-means clustering, together with its property that the choice of the initial cluster centers highly affects the output of the algorithm. Assuming we have N data items and K clusters, the kernel is constructed by running k-means T times, each time with different initialization resulting in different cluserings. The resulting kernel is called the bagged kernel. The final cluster kernel is the Hadamard product of the base kernel and the bagged kernel.

Borrowing the underlying idea of the bagged cluster kernel in the following we develop reweigthing kernels based on various clustering algorithms.

4.1 Gaussian reweighting kernel

Suppose that we are given the output of a clustering algorithm, the cluster membership matrix $\mathbf{U}$ of size $K \times N$. We assume that two points belong to the same cluster(s) if their cluster membership vectors are similar or close to each other. We define similarity via the Gaussian kernel in the following way:

$$k_{\mathrm{rw}}(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\frac{\|\mathbf{U}_{\cdot \mathbf{x}_1} - \mathbf{U}_{\cdot \mathbf{x}_2}\|^2}{2\sigma^2}\right) \tag{5}$$

where $\mathbf{U}_{\cdot \mathbf{x}}$ denotes the cluster membership vector of point $\mathbf{x}$, *i.e.* the column of $\mathbf{x}$ in $\mathbf{U}$. We know that the resulting matrix is positive semi-definite [10] with each element between 0 and 1. In this case the parameter σ defines the amount of separation between similar and dissimilar points: if σ is large, the gap between the values expressing similarity and dissimilarity becomes smaller, while for smaller σ these values get farther from each other.

4.2 Dot product-based reweighting kernels

Another possibility of using the cluster membership vectors is to define the following reweighting kernel:

$$\mathbf{K}_{\mathrm{rw}} = \mathbf{U}'\mathbf{U} + \frac{\alpha}{N}\mathbf{1}\mathbf{1}' \tag{6}$$

where $\mathbf{U}$ denotes the cluster membership matrix and $\alpha \in [0,1)$. The first term scores point similarity according to the cluster memberships, and the second term is used to avoid zero similarities: if two membership vectors are orthogonal, leading to a zero in the dot product matrix. We may assume that the obtained clustering is not too *confident*, *i.e.* we should use a small value α, the crisp cluster membership is thus alleviated with the term $(\alpha/N)\mathbf{11}'$.

Shoving that the reweighting kernel from equation (6) is positive semi-definite is straightforward: both the first and the second term is an external product, thus positive semi-definite, and owing to the properties defined in Section 4, results that the kernel is indeed positive semi-definite.

Another version of the kernel in (6) is

$$\mathbf{K}_{\text{rw}} = \beta\,\mathbf{U}'\mathbf{U} + \frac{1}{N}\mathbf{11}' \tag{7}$$

where $\beta \in (0,\infty)$. Here the kernel values for which the dot product matrix of cluster membership vectors correspond to zero, by $(1/N)\mathbf{11}'$, remain the same, however if the points lie in the same cluster $\beta\mathbf{U}'\mathbf{U}$ gives a weight greater than zero, thus this kernel value will be increased.

The above equations could clearly be merged, but we left them as separate reweighting kernels to differentiate between the underlying ideas.

5 Experiments and results

In this section we present the results obtained using our cluster kernels, and we compare it to other data-dependent kernels. For learning we used support vector machines (SVMs) [3], namely the LIBSVM (version 2.85) implementation [4]. The data sets used for evaluating the kernels were the following: USPS, Digit1, COIL2, Text. The detailed descriptions of these sets can be found in [5]. Each data set has two variations: one with 10 and one with 100 labeled data; furthermore each data set contains 12 labeled/unlabeled splits of the data. We used only the first split from each set. The columns having labels 10 and 100 in the tables showing the obtained result indicate which version of the data set was used, *i.e.* they show the number of labeled and unlabeled examples used.

Table 1 shows the accuracy results obtained using different kernels. We used accuracy as the evaluation measure, and the results are given in percentage. For each data set we indicated the best, second best and third best results obtained.

The first two rows contain the *baseline* results obtained with linear and Gaussian kernels. Principally we wanted to improve on these results. The hyperparameter for the Gaussian kernel was set using a cross-validation procedure.

The following 7 rows show the results obtained using the ISOMAP kernel [11], the neighborhood kernel [12], the bagged cluster kernel [12], the multi-type cluster kernel with different transfer functions [6] and Laplacian SVM [1], respectively.

The next 15 rows – below the second horizontal line – show the results obtained using our kernels. HCK and gHCK denote the hierarchical cluster and graph-based hierarchical cluster kernels from Section 3, respectively. RCK1, RCK2 and RCK3 denote the reweighting cluster kernels defined in equations (5), (6) and (7), respectively. Here we experimented with three clustering techniques: *k*-means, hierarchical and spectral clustering.

Because of lack of space we omitted the description of setting the parameters.

6 Discussion

As the results show we obtained good results with the proposed hierarchical and reweighting cluster kernels, and in many cases the results provided by our kernels are very close to the best results, as shown

	USPS		**Digit1**		**COIL2**		**Text**	
	10	100	10	100	10	100	10	100
linear	72.82	86.43	81.07	90.86	60.74	80.43	58.26	67.86
Gaussian	80.07	89.71	56.11	93.86	57.38	82.50	59.06	56.43
ISOMAP	85.10	86.71	**94.43**	**97.43**	62.62	80.64	59.80	72.43
neighborhood	76.31	94.14	87.11	94.21	64.43	84.43	51.68	62.79
bagged	**87.38**	92.79	**93.29**	**96.93**	**71.28**	85.57	63.29	66.14
multi-type, step	80.07	92.86	91.01	91.29	55.77	84.86	53.56	**74.79**
multi-type, linear step	80.07	92.86	91.01	91.36	55.77	84.86	53.02	**75.29**
multi-type, polynomial	80.07	80.29	48.86	65.07	54.23	82.29	50.60	56.71
LapSVM, Gaussian	81.95	**95.93**	84.50	**97.64**	**76.64**	**97.71**	**63.42**	62.50
HCK, single	80.07	81.79	48.86	70.21	67.85	**96.00**	**66.78**	**73.14**
HCK, complete	82.01	89.50	60.67	89.71	55.64	86.36	50.27	49.57
HCK, average	81.48	92.86	71.75	93.79	68.05	91.71	**64.63**	50.14
gHCK, single	80.07	81.79	48.86	70.21	60.60	**93.86**	**66.78**	**73.14**
gHCK, complete	**88.26**	**95.64**	75.50	93.71	**68.52**	88.79	56.17	67.71
gHCK, average	**89.26**	**95.64**	**94.70**	95.21	60.54	90.64	47.32	66.86
RCK1, k-means	84.45	92.98	83.14	94.28	58.55	83.76	–	–
RCK1, hierarchical	86.17	95.29	89.06	94.94	62.08	85.93	62.35	68.07
RCK1, spectral	81.43	90.87	88.32	95.20	58.22	83.83	63.26	66.93
RCK2, k-means	83.86	92.45	84.58	94.08	58.95	83.76	–	–
RCK2, hierarchical	86.11	**95.50**	89.06	95.29	62.08	85.64	61.28	71.14
RCK2, spectral	81.63	91.39	88.32	94.64	58.03	83.37	61.50	70.07
RCK3, k-means	83.66	92.59	84.13	92.96	58.60	83.28	–	–
RCK3, hierarchical	84.97	95.29	89.06	94.57	62.95	86.07	59.13	71.21
RCK3, spectral	81.16	91.56	88.32	94.73	55.83	83.20	59.26	71.00

Table 1: Accuracy results using different kernels. The results are given in percentage. For each data set the best three results were formatted in boldface.

in Table 1. Individually LapSVM outperformed every other method, possibly because of the careful selection of its parameters, but also because it is a very powerful technique.

Thus the results show that the proposed kernels for semi-supervised classification can be used for different types of data sets, and they provide better performances compared to simple, data-independent kernels, *e.g.* the Gaussian kernel. Moreover with data-dependent kernels any supervised kernel method can be easily turned into a semi-supervised method, without changing the underlying learning algorithm.

In order to compute the kernel for the test points one needs to include these points in the unlabeled data set. That is one can say that the methods resemble transductive learning, where the decision function is computed only on the points in question. Thus if a new point arrives the whole process must be repeated. To overcome this costly process approximation methods could be implied, but this is left as a future work. We also plan to develop methods or heuristics for automatically choosing the parameters of the proposed kernels.

Acknowledgments The authors acknowledge the partial support of the Romanian Ministry of Education and Research via grant PNII 11-039/2007.

Bibliography

[1] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[2] Ingwer Borg and Patrick J. F. Groenen. *Modern Multidimensional Scaling, 2nd edition.* Springer-Verlag, New York, 2005.

[3] B. E. Boser, I. Guyon, and V. N. Vapnik. A Training Algorithm for Optimal Margin Classifiers. *Computational Learning Theory*, 5:144–152, 1992.

[4] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001.

[5] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. MIT Press, September 2006.

[6] Olivier Chapelle, Jason Weston, and Bernhard Schölkopf. Cluster Kernels for Semi-Supervised Learning. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *NIPS*, pages 585–592. MIT Press, 2002.

[7] Richard Duda, Peter Hart, and David Stork. *Pattern Classification*. John Wiley and Sons, 2001. 0-471-05669-3.

[8] Bernd Fischer, Volker Roth, and Joachim M. Buhmann. Clustering with the Connectivity Kernel. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *NIPS*. MIT Press, 2003.

[9] Imre J. Rudas and János Fodor. Intelligent systems. *Int. J. of Computers, Communication & Control*, III(Suppl. issue: Proceedings of ICCCC 2008):132–138, 2008.

[10] B. Schölkopf and A. J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, 2002.

[11] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, December 2000.

[12] Jason Weston, Christina Leslie, Eugene Ie, and William Stafford Noble. Semi-Supervised Protein Classification Using Cluster Kernels. In Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors, *Semi-Supervised Learning*, chapter 19, pages 343–360. MIT Press, 2006.

[13] Quan Yong and Yang Jie. Geodesic Distance for Support Vector Machines. *Acta Automatica Sinica*, 31(2):202–208, 2005.