Data-driven Kernels via Semi-Supervised Clustering on the Manifold

Kernel methods, Semi-supervised learning, Manifolds, Transduction, Clustering, Instance-based learning

Abstract

We present an approach to transductive learning that employs semi-supervised clustering of all available data (both labeled and unlabeled) to produce a data-dependent SVM kernel. In the general case where the domain includes irrelevant or redundant attributes, we constrain the clustering to occur on the manifold prescribed by the data (both labeled and unlabeled). Empirical results show that the approach performs comparably to more traditional kernels while providing significant reduction in the number of support vectors used. Further, the kernel construction technique inherently possesses some of the benefits that would normally be provided by preprocessing with a dimensionality reduction algorithm. However, preprocessing still provides some additional benefit to the data-dependent kernel, including reduction in classification error due to improved cluster quality and robustness with respect to the SVM slack variable.

1. Introduction

In many learning scenarios, it is common for data acquisition to be inexpensive compared to data labeling. In such instances, transductive or semi-supervised learning approaches are often useful. These involve not only the traditional machine learning task of model selection but also the question of how to incorporate unlabeled data into the learning process. We consider the application of SVM classifiers to such situations and address the problem of kernel selection, presenting a method for explicit, data-driven kernel construction that incorporates all available data, both labeled and unlabeled. Because we are working in a transductive setting, it makes sense to utilize the additional information provided by the unlabeled data, something many traditional techniques cannot (naturally) do. Our approach creates a data-dependent distance metric using *all* available data and then uses that metric in classifying the unlabeled portion of the data. We proceed by initially clustering the data in a semi-supervised manner, and then use the resulting pairwise distances to generate an affinity matrix for use as the kernel of an SVM. We also investigate the effect of preprocessing the data with a nonlinear manifold learner for dimensionality reduction, effectively constraining our clustering to the manifold.

In what follows, we formally describe our approach to constructing the data-dependent kernel and present empirical results on several real-world data sets that highlight the characteristic benefits of the technique. Before doing so, however, we briefly mention related work in three significant areas: semi-supervised clustering, kernel selection and manifold learning.

1.1. Semi-supervised Clustering

Semi-supervised clustering, a form of transductive inference, is an easier problem than the standard approach of inductive transfer followed by deductive classification. The presence of unlabeled data allows for better estimation of the data's true distribution which can improve classification accuracy. Vapnik (1998) has proposed a framework to establish upper bounds on the empirical error of on a set of unlabeled data, D_u, given the empirical error on a set of labeled data, D_l.

Given the widespread adoption of support vector machines, also based on Vapnik's statistical learning theory, it is not surprising that significant effort has gone into combining the principles of transductive learning and SVM's. Broadly, these approaches attempt to combine the maximum margin principle of SVM's with the clustering assumption (points that lie close together are likely to belong to the same class).

An early example of this work is the transductive SVM (Joachims, 1999). In this approach the deci-

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

sion boundary is chosen in such a way to maximize the margin for both labeled and unlabeled data. In other words, the transductive SVM attempts to draw a decision boundary in areas of low data density. Related works that followed include (Joachims, 2003) and (Szummer & Jaakkola, 2001) and more recent work includes variations such as semi-supervised regression (Zhou & Li, 2005).

Several approaches to combining the principles of semi-supervised clustering and support vector machines have been proposed. Xu *et. al* (2005) use the maximum margin principle to guide clustering, while Chapelle, Weston and Schölkopf (2003) generate an affinity matrix that leverages available unlabeled data.

We follow the example of Chapelle *et. al* in our efforts to directly construct a SVM affinity matrix using clustering techniques. In contrast, while their method requires the choice of a "transfer function", ours requires choosing only two scalar parameter values; also, while their method is based on spectral clustering, we use a graph-based clustering more reminiscent of LLE or Isomap with their strong non-linear advantages.

Semi-supervised clustering can be seen as a generalization of the transductive inference problem. Labels or pairwise constraints can be used to modify the standard clustering objective function or to learn an appropriate distance metric. Basu, Bilenko and Moody (2004) have proposed several semi-supervised clustering algorithms including Seeded KMeans, HMRF-KMeans, and Pairwise Constraints KMeans.

1.2. Kernel Selection

The selection of a kernel for kernel based learning methods is critical to model performance and is usually the subject of empirical analysis in which parameters for a specific class of kernels are chosen via cross validation. There has been some work on the development of adaptive or data-dependent kernels. Data dependent kernels were first proposed in (Amari & Wu, 1999), where a modified kernel takes the form

$$\tilde{K}(\mathbf{x}, \mathbf{x}') = c(\mathbf{x})c(\mathbf{x}')K(\mathbf{x}, \mathbf{x}')$$

where $K(\mathbf{x}, \mathbf{x}')$ is some standard kernel and $c(\mathbf{x})$ is a data dependent transforming factor. The idea is to increase the resolution of the feature space near the separation boundary. Similar work includes (Xiong et al., 2004), (Heisterkamp et al., 2001) and (Peng et al., 2002).

Another method of developing data dependent kernels, presented in (Cristianini et al., 2001), adapts the kernel matrix based on the notion of "kernel target alignment". Target alignment is defined in terms of the Frobenius inner product and measures the similarity of a kernel matrix and the labels in feature space. The method in (Cristianini et al., 2001) involves finding all the eigenvectors of the kernel matrix generated by a standard kernel function and then finding an optimal linear combination of those eigenvectors.

1.3. Manifold Learning

Manifold learning, typically for the purpose of dimensionality reduction and/or discovering the intrinsic dimensionality of data, is related to both clustering and kernel-based methods as another approach to computing/discovering distance metrics. Seminal work in this area is represented by the well-known manifold learners Isomap (Tenenbaum et al., 2000) and LLE (Roweis & Saul, 2000), while more recent work includes extensions of this technique like Spectral Learning (Kamvar et al., 2003) and Relevant Component Analysis (Bar-Hillel et al., 2005), as well as alternative approaches such as Tensor Voting (Mordohai & Medioni, 2005).

2. Methodology

Given a set D_L of n_l labeled data points and a set D_U of n_u unlabeled data points, we use a graph-based semi-supervised clustering algorithm to learn a data-specific distance metric, represented by a $n \times n$ matrix of point-to-point distances for $D_L \cup D_U$, where $n = n_l + n_u$. This matrix is then used in constructing a kernel matrix for an SVM.

2.1. Distance Matrix Construction

Our goal is to create the matrix $M_D = d_{ij}$, where $1 \leq i, j \leq n$ and d_{ij} is the (data-dependent) distance from point i to point j. We begin by clustering the data. Given data with m classes and n_l labels, we use hierarchical agglomerative clustering with purity thresholding to cluster the data. The purity threshold θ allows us to leverage any labels that are available. The purity ρ of any cluster c is calculated as

$$\rho_c = \begin{cases} \dfrac{n_{k_c}}{n_{l_c}} & \text{if } n_{l_c} > 0 \\ 1 & \text{if } n_{l_c} = 0 \end{cases}$$

where n_{k_c} is the count of the most common label in c and n_{l_c} is the total number of labeled points in c. Initially, the set C of clusters contains only clusters c_i consisting of a single point and having a purity of 1. Clusters are then iteratively agglomerated as follows. For each cluster c_i, in C, we find it's nearest neighbor, c_j (we use complete link clustering and the

Euclidean metric – the distance between any two clusters is defined as the Euclidean distance between their two most distant points). If $\rho_{(c_i \cup c_j)} \geq \theta$, we remove c_i and c_j from C and add the cluster $(c_i \cup c_j)$ to C. This process is repeated until no further clusters can be combined. Finally, any clusters that remain without labeled points are combined with the nearest cluster that has at least one labeled point. C now contains some number of clusters that can each be identified with a dominant label l_c.

For each cluster c, we compute a virtual center of mass point γ^c:

$$\gamma^c = \frac{1}{n_c} \sum_{\mathbf{x} \text{ in } c} \mathbf{x}$$

where n_c is the total number of points in c (both labeled and unlabeled) and the summation is vector addition with the coefficient a scalar applied to each element in the resulting vector. Now, for clusters c_1 and c_2 we define the distance between their centers of mass as

$$\text{dist}(\gamma^{c_1}, \gamma^{c_2}) = \begin{cases} \|\gamma^{c_1} - \gamma^{c_2}\| & \text{if } l_{c_1} \neq l_{c_2} \\ 0 & \text{if } l_{c_1} = l_{c_2} \end{cases}$$

In other words, if two clusters share a common label, the distance between their centers of mass is defined to be 0; if they have different labels, the distance between their centers of mass is the standard Euclidean distance. Next, a shortest path graph traversal algorithm (optimized Floyd-Warshall) is used to propagate the effect of these "wormholes" between clusters to every pair of data points. In essence, this will create m meta-clusters as the matrix M_D is defined [1]

$$d_{ij} = \|\mathbf{x}_i - \gamma^{c_i}\| + \text{dist}(\gamma^{c_i}, \gamma^{c_j}) + \|\mathbf{x}_j - \gamma^{c_j}\| \qquad (1)$$

where $\mathbf{x}_i$ is the vector representation of point i, c_i is the cluster containing point i, γ^{c_i} is the virtual center of mass of c_i and $\|\cdot\|$ is the Euclidean metric. In other words, we combine all clusters with common labels by decreasing the distance between the points included in such clusters (via the "wormholes").

[1] Actually, the distances calculated by the shortest path algorithm cannot be so simply characterized. Eq. 1 is representative of the effects the "wormholes" can have on the final distance matrix; however, there are actually 5 different scenarios that can represent the shortest path between two points (for the 2-class case). Eq. 1 gives one of the five cases, but in practice we compute all five cases for each pair of points and take the minimum, with the result being the same as running a regular Djikstra's algorithm, just (much) faster.

To improve separability we post-process the matrix M_D to push these meta-clusters apart by increasing the distance between each pair of points in different meta-clusters. The distance is increased by κ times the greatest distance between any two data points sharing the same label (we used $\kappa = 2$ in our experiments). That is,

$$d_{ij} = \begin{cases} d_{ij} + \kappa \max\limits_{k} \max\limits_{i,j \in c_k} d_{ij} & \text{if } l_{c_i} \neq l_{c_j} \\ d_{ij} & \text{if } l_{c_i} = l_{c_j} \end{cases}$$

The resulting distance matrix represents a feature space where each point is grouped with similar points, points that share the same label, and points that are similar to points that share the same label.

2.2. Kernel Construction

The distances learned through the clustering process represent an embedding of data into a vector space, which can be used as the (mapped) feature space for an SVM. If the distance matrix is used to construct a kernel function, a support vector machine can be used to find a decision surface in that space.

Kernel functions work by calculating dot products, and in particular the kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ is defined in general as

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

where $\Phi(\mathbf{x})$ is the transformation of $\mathbf{x}$ into feature space. From the definition of the dot product as $a \cdot b = \|a\|\|b\|cos(\theta)$ and the law of cosines, the kernel function corresponding to any distance matrix can be written as

$$K(\mathbf{x}_i, \mathbf{x}_j) = \frac{-d_{ij}^2 + \|\Phi(\mathbf{x}_i)\|^2 + \|\Phi(\mathbf{x}_j)\|^2}{2} \qquad (2)$$

where d_{ij} is the distance between the points $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{x}_j)$. We chose to define d_{ij} as the distance found by our clustering algorithm between $\mathbf{x}_i$ and $\mathbf{x}_j$, and $\|\Phi(\mathbf{x})\|$ as the distance found during clustering between $\mathbf{x}$ and an arbitrarily chosen origin point. (Because we're just picking an origin so we can convert distances to dot products, any point is as good as any other.) This choice of terms creates a kernel function corresponding to a feature space with the distances induced from the clustering phase. It is interesting to note that this is not an inner-product space, since the "wormholes" allow the distance metric to violate the triangle inequality. As a result, the Mercer conditions required for the performance guarantees of SVMs do not hold for our kernel. In practice some of the most popular kernels (including the Guassian kernel) also violate these conditions, but still perform well on real-world data sets, just as our kernel does.

This new data-dependent kernel is used to train a standard support vector machine. During the training phase of the SVM, all necessary values of the kernel function can be found in the kernel matrix of Eq. 2. During execution, however, it is possible that values of the kernel function may be needed which do not exist in the kernel matrix. In this case, some interpolation is necessary. During testing, all of the calls to $K(\mathbf{x}_t, \mathbf{x}_s)$ are made with a test point and a support vector as parameters, respectively. Since SVMs always choose their support vectors from the set of points available during training, there will be a set of entries in the distance matrix corresponding to each support vector. If there is not an entry in the distance matrix for the test point, its value is interpolated using k-Nearest Neighbor and linear weighted regression. The nearest neighbors ($\mathbf{x}_{i_1 \ldots i_k}$) are chosen as the points which are closest to the test point using the same distance metric that was used to generate the original clusters (here, the Euclidean metric). The value of the distance function is calculated as

$$d_{ts} = \frac{\sum_{i=1}^{k} d_{si_j} \left\|\mathbf{x}_t - \mathbf{x}_{i_j}\right\|}{\sum_{i=1}^{k} \left\|\mathbf{x}_t - \mathbf{x}_{i_j}\right\|} \tag{3}$$

which is then used as the d_{ij} term in Eq. 2. The $\Phi(\mathbf{x}_t)$ term in Eq. 2 is calculated in a similar manner, as a weighted average of $\Phi(\mathbf{x}_{i_j})$.

3. Empirical Results

We performed extensive empirical testing of eight different kernels on ten different data sets measuring error rates, the effect of cluster purity and slack on performance, number of support vectors used, and number of clusters used. The kernels used were a polynomial kernel, a Gaussian kernel, a mixed poly/Gaussian kernel, and a data-dependent kernel matrix. In addition, each of the kernels was applied after preprocessing the data with a non-linear dimensionality reduction algorithm (we used Relevant Component Analysis (Bar-Hillel et al., 2005)). Nine of the data sets (*Chess*, *German*, *Heart*, *Pima*, *Spect*, *Voting*, *WBC1*, *WBC2* and *WBC3*) are taken from the UCI repository (Blake & Merz, 1998) and the tenth (*ADA*) is from a recent performance prediction challenge at *IJCNN 2006*.

For each data set and for each kernel, we obtained support vector count and classification error on the unlabeled portion of the data. We varied slack settings and amounts of unlabeled data, and, in addition, for the data-dependent kernels, the cluster purity was varied as well. We varied the slack variable C over the values $\{0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000\}$, the cluster purity θ over the values $\{0.80, 0.85, 0.90.0.95, 1.00\}$, and the amount of unlabeled data as a percentage of the total data available in 10% increments from 10% to 90%. All results were obtained using 10-fold cross validation, resulting in $40,500$ experiments for each of the two data-dependent kernels and $8,100$ experiments for each of the six traditional kernel variations, for a grand total of $129,600$ experimental runs.

Table 1 summarizes the best classification error for all kernels for all data sets using 90% unlabeled data. Figures 1 and 2 show representative learning curves for the different kernels on two of the data sets. Results reported are the best average error (across the 10-folds) for any value of C and θ. Note that, as expected, kernel performance is data dependent (note in particular the learning curves for the RCA/poly kernel, which had difficulty with many of the data sets.)

Figure 3 shows the number of support vectors each kernel type requires, compared with the performance for each kernel type. The numbers reported in the figure are normalized averages across all ten data sets, with the support vector count for each 10-fold cross validation experiment being reported as a median and classification error over the the 10-folds being reported as an average. Note that while there is some variance in performance between the different kernels, there is a marked difference in numbers of support vectors required for that performance. In particular, the data-dependent kernels require many fewer support vectors on average than do the other kernel types.

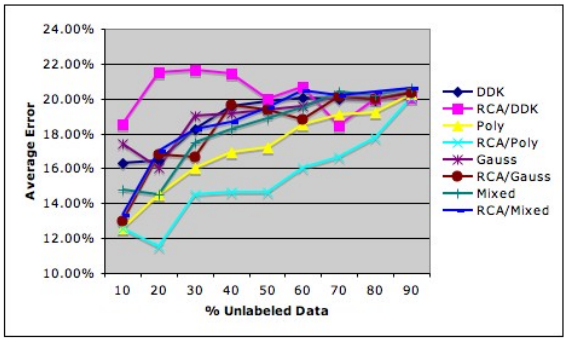

Figure 1. Classification error using 10-fold cross validation on the *Spect* data set for varying amounts of unlabeled data. Data points on the curves represent the lowest error value for any value of C for the traditional kernels and any value of C and any value of θ for the data-dependent kernels.

Table 1. Performance comparison for various kernels, with and without RCA preprocessing. Results reported are the highest average accuracy for any value of C and θ using 90% unlabeled data.

	Chess	German	Heart	Pima	Spect	Vote	WBC1	WBC2	WBC3	ADA
DDK	21.7%	30.0%	24.7%	31.0%	20.3%	7.0%	5.7%	23.3%	8.0%	20.3%
RCA/DDK	15.2%	33.1%	28.9%	29.8%	20.1%	3.7%	6.1%	23.4%	4.8%	20.2%
Poly	5.0%	27.8%	18.0%	28.1%	20.3%	5.5%	4.4%	23.9%	5.9%	24.7%[a]
RCA/Poly	10.8%	29.1%	39.3%	33.9%	20.1%	17.8%	8.6%	23.5%	16.6%	40.2%
Gauss	10.2%	29.2%	25.6%	29.2%	20.4%	8.1%	6.0%	23.3%	5.9%	24.8%
RCA/Gauss	6.1%	27.1%	21.0%	26.5%	20.3%	3.6%	4.5%	23.4%	4.5%	16.4%
Mixed	4.6%	28.6%	21.0%	25.3%	20.6%	5.2%	5.4%	23.3%	6.0%	26.9%
RCA/Mixed	4.2%	28.8%	22.9%	27.8%	20.6%	3.7%	6.0%	23.9%	5.4%	16.7%

[a] the poly kernel learned to invert its output, so the reported value is actually $(1 - \text{error})$

Also, note that while RCA preprocessing results in a reduction in support vector count for the traditional kernels, there is an apparent lack of correlation between RCA preprocessing and number of support vectors for the data-dependent kernel. This seems to suggest that the semi-supervised clustering technique used to construct the kernel is discovering the same information that RCA is providing. This, in turn, suggests an intimate link between (semi-supervised) clustering and non-linear manifold learning techniques.

As further evidence of this, we examined the correlation between RCA preprocessing and number of support vectors. Figure 4 shows a scatter plot of support vector counts with and without preprocessing. Note the slopes of the trend lines indicate that RCA preprocessing tends to reduce the number of support vectors for polynomial, Gaussian and mixed kernels while tending to slightly increase the number of support vectors for the data-dependent kernel.

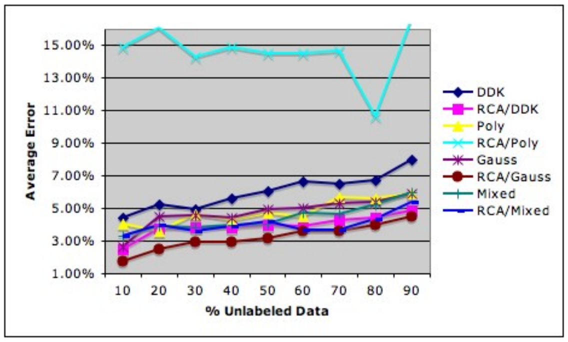

Figure 2. Classification error using 10-fold cross validation on the *WBC3* data set for varying amounts of unlabeled data. Data points on the curves represent the lowest error value for any value of C for the traditional kernels and any value of C and any value of θ for the data-dependent kernels.

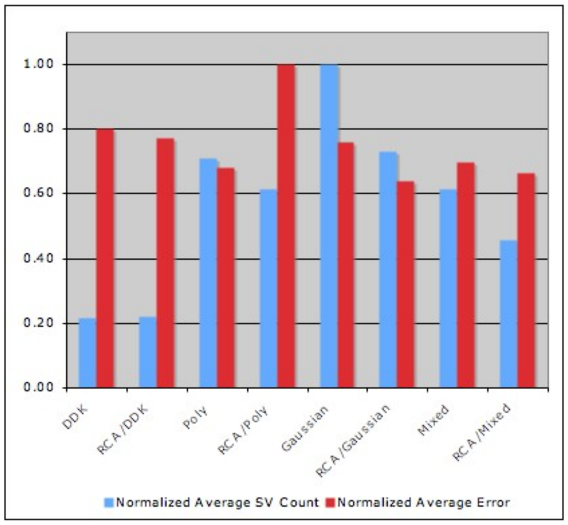

Figure 3. Support vector count vs. error rate – normalized average median number of support vectors compared with normalized average classification error. Averages are over ten data sets. For support vector count, the median value of the 10 folds was averaged over the ten data sets and then normalized, while for classification error the average value for the 10 folds was averaged over the ten data sets and then normalized. Error values reported for each algorithm are the best average for any value of C and θ with 90% of the training data unlabeled.

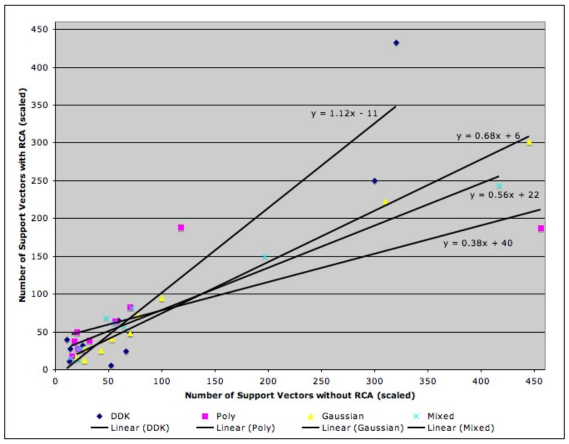

Figure 4. Correlation of number of support vectors with and without RCA preprocessing. Data points represent scaled (SV, SV) count pairs for the best average (10-fold cross validation) accuracy for the ten data sets.

Figure 5 gives further evidence that the data-dependent kernel is independently making gains associated with RCA preprocessing for the other kernels. It shows two scatter plots correlating the number of clusters discovered with number of support vectors: one using RCA preprocessing and one without. Trend lines for both indicate a nice correlation – the more clusters (and thus the higher their purity), the fewer support vectors are required. Further, this negative correlation is independent of RCA preprocessing.

However, it is not the case that RCA provides no benefit as a preprocessing step to the data-dependent kernel. As can be seen in Figure 3, there is a performance

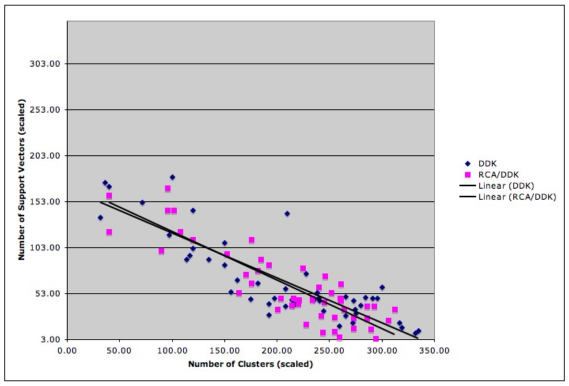

Figure 5. Correlation of number of support vectors with number of clusters. Data points represent scaled (cluster, SV) count pairs for different values of θ for the ten data set. Since the number of clusters is independent of the value of C we arbitrarily chose $C = 1$.

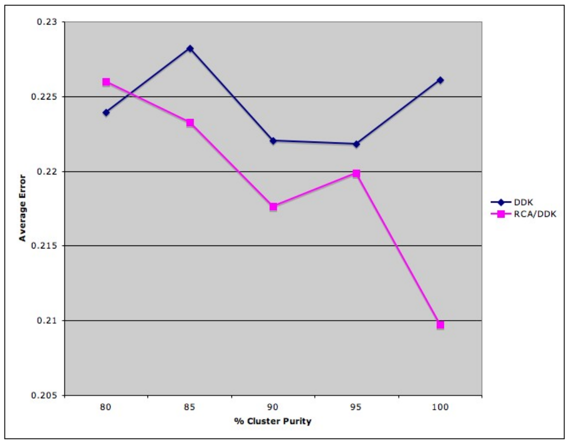

Figure 6. Effect of cluster purity on accuracy for DDK – classification error using 10-fold cross validation for ten data sets for different values of θ in the data-dependent kernel, both with and without RCA preprocessing. Reported error values are averages over all ten data sets. The SVM slack variable C was held constant at 1.0.

improvement when RCA is used. Figure 6 sheds further light on this phenomenon by examining the effect of cluster purity on classification error for the data-dependent kernel. In the absence of RCA preprocessing, classification performance appears robust to the cluster purity threshold – "better" clusters do not improve accuracy. However, after preprocessing, performance improves as the purity threshold is increased; therefore, RCA is providing some additional guidance to the clustering algorithm that results in more effective clustering.

Finally, we mention an additional characteristic of note for the data-dependent kernels – they are robust to the value of the slack variable C. The only other kernel that is comparable in this respect is the RCA/poly kernel and its average error was much worse (see Figure 7). Note, that the effect is especially apparent when RCA is combined with the data-dependent kernel, providing another benefit to preprocessing.

4. Discussion

We have presented an approach to building an explicit data-dependent kernel based on the results of a semi-supervised clustering of data. Since the kernel construction is completely data driven, this technique is intended primarily for use in data-rich settings such as those involving transductive learning tasks.

When used in conjunction with a standard SVM, the data-dependent kernel exhibits classification performance competitive with several traditional kernels

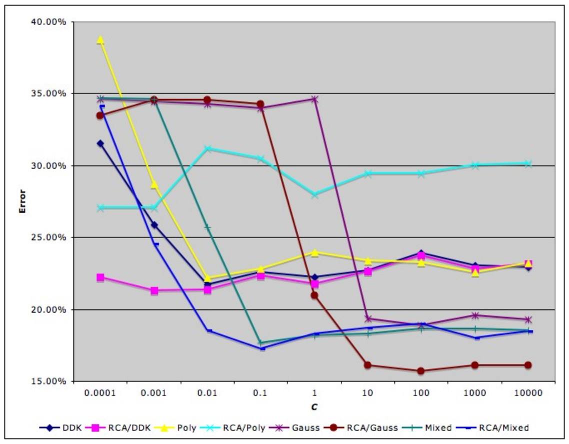

Figure 7. Robustness of various kernels to SVM slack – classification error using 10-fold cross validation for the ten data sets for different values of C for the different kernels. The cluster purity threshold θ for the data-dependent kernel was held constant at 90%.

Table 2. Performance comparison for data-dependent [distance weighted] k-NN ($k = 9$) and data-dependent kernel SVM, with and without RCA preprocessing. Results reported are the average over nine data sets of the lowest average error for any value of C and θ using 90% unlabeled data.

	k-NN	DDK
no RCA	20.84%	19.79%
RCA	19.07%	18.35%

while using significantly fewer support vectors. Further, the data-dependent kernel's production of support vectors exhibits characteristics often associated with dimensionality reduction techniques (and which the other kernels exhibit only after preprocessing for dimensionality reduction). Even the data-dependent kernel, however, gains some benefit from dimensionality reduction prepocessing, specifically, a reduction in classification error due to increased cluster quality. Also, preprocessing results in the data-dependent kernel exhibiting a unique robustness to the value of the SVM slack variable C.

Because we are deriving a data-dependent distance metric, it is reasonable to question whether the distance metric itself contributes all of the classification accuracy or if using the metric to kernelize an SVM provides some benefit. Perhaps the most obvious way to answer this question is to compare our data-dependent kernel SVM with a k-NN model that uses the data-dependent distance metric rather than the standard Euclidean metric. Table 2 shows average of averages for classification error over nine different data sets for 90% unlabeled data and makes two things clear: RCA is beneficial for both k-NN and the SVM, and the data-dependent SVM is more accurate than data-dependent k-NN. Figure 8 shows average error rates (over nine different data sets) for varying amounts of unlabeled data for the data-dependent kernel SVM and for data-dependent (distance weighted) k-NN for several different values of k. Again, the results indicate that the SVM makes a significant contribution to the performance we report here (in other words, the distance metric is not doing all the work).

The current approach to building metaclusters is *ad hoc* and can result in extremely nonlinear distance metrics. While the stated goal of this research has been to create a kernel based solely on the data, there may be more principled approaches, in particular with respect to enforcing both negative and positive cluster constraints, that result in better generalization. Also, it would be desirable to automatically choose values for the parameters θ and κ that best fit the data. This might be done for θ by measuring some global purity value, such as an entropic measure, and for κ by measuring some global variance property of the data. Of course, the nature of the kernel construction may mean

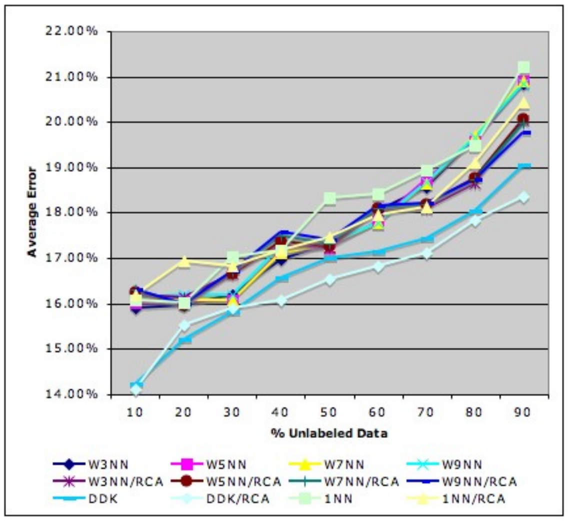

Figure 8. Data-dependent Nearest Neighbor vs. Data-dependent kernel SVM – best average classification error (over data sets) using 10-fold cross validation for different amounts of unlabeled data. The k-NN models use [data-dependent] distance weighted voting. Averages reported are the best average for any value of θ and any value of C (for the SVM results).

there is no principled way to select the two parameters; however, most learning methods, including SVMs of course, involve some amount of parameter tuning. In fact, for SVMs the approach presented here greatly simplifies the need to choose parameters by providing the kernel function automatically – choosing two appropriate scalar values is likely much simpler than picking an appropriate kernel function. (We note that in practice using purity thresholds near the expected classification accuracy often produced the best results, though this is still a preliminary observation and not yet well-supported experimentally).

Finally, the relationship between this clustering-based kernel construction and (nonlinear) dimensionality reduction algorithms should be further explored. Techniques used in manifold learning for preserving local and global point-to-point distances on the manifold may prove useful in aiding cluster formation, with the result being data-driven automatic kernel construction that natively discovers and makes use of the underlying manifold.

References

Amari, S., & Wu, S. (1999). Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, *12*, 783–789.

Bar-Hillel, A., Hertz, T., Shental, N., & Weinshall, D. (2005). Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, *6*, 937–965.

Basu, S., Bilenko, M., & Mooney, R. J. (2004). A probabilistic framework for semi-supervised clustering. *KDD04* (pp. 59–68). Seattle, WA.

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases.

Chapelle, O., Weston, J., & Schölkopf, B. (2003). Cluster kernels for semi-supervised learning. *Advances in Neural Information Processing Systems 15* (pp. 585–592).

Cristianini, N., Kandola, J., Elisseeff, A., & Shawe-Taylor, J. (2001). On kernel target alignment. *Advances in Neural Information Processing Systems 14* (pp. 367–373).

Heisterkamp, D., Peng, J., & Dai, H. (2001). An adaptive quasiconformal kernel metric for image retrieval. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 236–243).

Joachims, T. (1999). Transductive inference for text classification using support vector machines. *Proceedings of the International Conference on Machine Learning* (pp. 200–209).

Joachims, T. (2003). Transductive learning via spectral graph partitioning. *Proceedings of the International Conference on Machine Learning* (pp. 290–297).

Kamvar, S., Klein, D., & Manning, C. (2003). Spectral learning. *Proceedings of the International Joint Conferences on Artificial Intelligence* (pp. 561 – 566).

Mordohai, P., & Medioni, G. (2005). Unsupervised dimensionality estimation and manifold learning in high-dimensional spaces by tensor voting. *Proceedings of the Joint Conferences on Artificial Intelligence* (pp. 908–913).

Peng, J., Heisterkamp, D., & Dai, H. (2002). Adaptive kernel metric nearest neighbor classification. *Proceedings of IEEE International Conference on Pattern Recognition* (pp. 33–36).

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*, 2323–2326.

Szummer, M., & Jaakkola, T. (2001). Partially labeled classification with markov random walks. *Advances in Neural Information Processing Systems 14* (pp. 945–952).

Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*, 2319–2323.

Vapnik, V. (1998). *Statistical learning theory.* Wiley.

Xiong, H., Swamy, M., & Ahmad, M. (2004). Learning with the optimized data-dependent kernel. *Proceedings of the IEEE Workshop on Learning in Computer Vision and Pattern Recognition* (pp. 95–98).

Xu, L., Neufeld, J., Larson, B., & Schuurmans, D. (2005). Maximum margin clustering. *Advances in Neural Information Processing Systems 17* (pp. 1537–1544).

Zhou, Z.-H., & Li, M. (2005). Semi-supervised regression with co-training. *Proceedings of the Joint Conferences on Artificial Intelligence* (pp. 908–913).