Discovering Domain-Specific Composite Kernels

Tom Briggs
Shippensburg University
1871 Old Main Drive
Shippensburg, PA 17257
thb@ship.edu

Tim Oates
University of Maryland Baltimore County
1000 Hilltop Circle
Baltimore, MD 21250
oates@cs.umbc.edu

Abstract

Kernel-based data mining algorithms, such as Support Vector Machines, project data into high-dimensional feature spaces, wherein linear decision surfaces correspond to non-linear decision surfaces in the original feature space. Choosing a kernel amounts to choosing a high-dimensional feature space, and is thus a crucial step in the data mining process. Despite this fact, and as a result of the difficulty of establishing that a function is a positive definite kernel, only a few standard kernels (e.g. polynomial and Gaussian) are typically used. We propose a method for searching over a space of kernels for composite kernels that are guaranteed to be positive definite, and that are tuned to produce a feature space appropriate for a given dataset. Composite kernel functions are easily interpreted by humans, in contrast to the output of other work on kernel tuning. Empirical results demonstrate that our method often finds composite kernels that yield higher classification accuracy than the standard kernels.

Introduction

Kernel-based algorithms are among the most theoretically and empirically powerful algorithms known for such common data mining tasks as classification (Joachims 2001), clustering (Ben-Hur *et al.* 2002), dimensionality reduction (Schölkopf, Smola, & Müller 1999), feature extraction (Bach & Jordan 2003), and many others. A function that maps from pairs of vectors to real numbers is a (positive definite) kernel if the value it produces for any pair of vectors corresponds to the dot product of the projection of the vectors into some feature space. In general, this feature space has very high dimensionality, and linear decision surfaces in the projected feature space correspond to highly non-linear decision surfaces in the original feature space. Kernels turn algorithms for finding linear decision surfaces, such as Support Vector Machines (SVMs), into algorithms for finding non-linear decision surfaces.

The choice of a kernel corresponds to the choice of a feature space, and thus constrains the type of decision surface that can be formed. For example, the polynomial kernel simply computes the dot product of two vectors, adds 1 to the resulting number, and raises this quantity to the power d. Remarkably, this simple computation corresponds to taking the dot product of the vectors after projecting each of them into a feature space with one axis for each product of d or fewer of the values in the original vector (Schölkopf & Smola 2002). For vectors of length n, this feature space contains $O(n^d)$ dimensions, though computing the dot product in this feature space, i.e. evaluating the kernel, requires only $O(n)$ time. Other common kernels, such as the Gaussian and sigmoid, correspond to dot products of vectors projected into other feature spaces.

Different datasets and different data mining tasks require different feature spaces. This suggests that domain-specific and task-specific kernels would be useful. Kernels represent prior knowledge about the type of structure expected in a domain or task. However, to call a function that maps from pairs of feature vectors to real numbers a (positive definite) kernel, this function must have certain properties that can be difficult to establish. Therefore, the handful of functions for which these properties have already been established are invariably used.

There has been some prior work on tuning kernels for specific tasks and domains ((Ayat, Cheriet, & Suen 2002), (Chapelle *et al.* 2002), (Wahba *et al.* 2001)). However, the vast majority of existing methods operate by adjusting the spectrum of the Gram matrix ((Friedrichs & Igel 2004),(Cristianini 2002)). For a dataset containing m data vectors $x_1 \ldots x_m$, the Gram matrix is an $m \times m$ array containing the values of $K(x_i, x_j)$ for kernel K and $1 \leq i, j \leq m$. This matrix lies at the heart of all kernel-based algorithms. From the standpoint of data mining, spectral methods have two distinct disadvantages. First, their output is exceedingly difficult for humans to understand. This is because they directly modify the Gram matrix produced by a known kernel, resulting in a Gram matrix that corresponds to the values of some other kernel that is *implicitly* defined by the new values. No analytical form for the kernel function is available. Second, in a classification setting, these algorithms are typically transductive and therefore require the spectrum to be tuned anew for each data item to be classified, which is expensive ((Friedrichs & Igel 2004),(Cristianini 2002)).

In this paper we explore an alternative approach that involves searching over a space of *explicitly* defined kernel functions. Given a base set of *atomic* functions that are known to be kernels, there are several ways of combin-

ing these functions to produce *composite* functions that are guaranteed to be (positive definite) kernels. This method involves a search over composite kernel space, which is a trade-off of training time to accuracy when compared to selecting a single kernel.

There are a variety of methods for improving the performance of SVMs by optimizing the hyper-parameters of the model ((Friedrichs & Igel 2004),(Chapelle *et al.* 2002)). Each of these methods is certainly more expensive (with respect to training time) than methods, such as some spectral methods, that use only a single kernel and training step. The complexity of this technique is due to the search for the SVM hyper-parameters and is not unique to the evaluation of the composite kernels. For a given dataset, this search over composite kernel space is guided by performance with respect to some generalization estimate, such as radius-margin (R^2M^2), target alignment, or training error; and yields a kernel that is tuned to the structure of the data. Empirical results show that this approach often yields better accuracy than SVMs using the standard atomic kernels.

This paper shows that domain-specific composite kernels and their hyper-parameters can be selected to improve the accuracy of SVM classifiers. This paper also shows that simple composite combinations of simple kernels give nearly the same performance of more complex combinations. We also demonstrate that existing search techniques can be applied to select composite kernel parameters at the same cost as selecting simple kernel parameters, but the resulting models are better than the simple kernels. Finally we show that the terms of the Taylor series for composite and simple Gaussian kernels are distinctly different, and thus induce different feature spaces, and the different feature spaces lead to improved SVM performance.

Background

Given a set of labeled instances of the form (x_i, y_i) where $x_i \in \Re^n$ and $y_i \in \{-1, 1\}$, SVMs attempt to find an n-dimensional hyperplane that separates the positive instances (i.e., those for which $y_i = 1$) from the negative instances (i.e., those for which $y_i = -1$). This is accomplished by solving a quadratic optimization problem to find the *maximum margin* hyperplane, i.e., the hyperplane that perfectly separates the instances and is as far as possible from the convex hulls around the positive and negative instances. SVMs use dot products of instances, denoted $\langle x_i, x_j \rangle$, to find this hyperplane.

If the data are not linearly separable, there are two options available. The optimization problem can be modified by the addition of slack variables that allow some points to be on the wrong side of the hyperplane (i.e. to be misclassified), resulting in a soft margin SVM. Alternatively, SVMs are amenable to the "kernel trick", meaning that they can use kernels to project the instances into high-dimensional feature spaces wherein the data might be linearly separable (Cristianini 2001). Let $\Phi(x_i)$ be the projection of instance x_i into this high-dimensional feature space. For the polynomial kernel of degree d described earlier, $\Phi(x_i)$ contains one dimension for each product of d or fewer of the values in the original vector. Importantly, there is no need to compute $\Phi(x_i)$, $\Phi(x_j)$, or $\langle \Phi(x_i), \Phi(x_j) \rangle$ explicitly. Rather, the kernel produces the value of the dot product in the feature space, but with far less computation than the explicit method.

Not all functions from $\Re^n \times \Re^n$ to $\Re$ are kernels. A function is a kernel if and only if the value it produces for two vectors corresponds to the dot product of those vectors after projecting them into some feature space Φ. This is stated more formally in Mercer's theorem:

Theorem 1 (Mercer's Theorem). *Every positive definite, symmetric function is a kernel. For every kernel K, there is a function $\Phi(x) : K(x_1, x_2) = \langle \Phi(x_1), \Phi(x_2) \rangle$. (Cristianini 2001)*

Composite Kernels

Proving that a function is a kernel is relatively difficult. However, there are several well known kernels, including the polynomial and radial basis function (RBF) kernels, shown in Table 1. We leverage these established kernel functions by applying operators that are closed over the kernel property (Joachims, Cristianini, & Shawe-Taylor 2001). Using these operators, any known kernels can be combined to create a new kernel, avoiding the need to prove that the composite kernel is positive definite. A table of kernel composition operators and the following theorem appear in (Joachims, Cristianini, & Shawe-Taylor 2001). With this information, the four basic kernels shown in Table 1 can be used to create an infinite combination of composite kernels.

Theorem 2 (Composite Kernels). *Let K_1, K_2 be valid kernel functions over $\Re^n \times \Re^n$, $x, z \subseteq \Re^n$, $\alpha \in \Re^+$, $0 < \lambda < 1$, $f(x)$ a real valued function, and $\phi X \rightarrow \Re^m$ with K_3 a kernel over $\Re^m \times \Re^m$. The following are also kernels:*

$$K(x,z) = \lambda K_1(x,z) + (1-\lambda)K_2(x,z) \quad (1)$$
$$K(x,z) = \alpha K_1(x,z) \quad (2)$$
$$K(x,z) = K_1(x,z)K_2(x,z) \quad (3)$$
$$K(x,z) = f(x)f(z) \quad (4)$$
$$K(x,z) = K_3(\phi(x), \phi(z)) \quad (5)$$

For example, let $K_0(x,z), K_1(x,z), K_2(x,z)$ be different kernels. The following is one possible composite kernel:

$$K_3(x,z) = (0.35K_0(x,z) + 0.65(K_1(x,z)K_2(x,z))) \quad (6)$$

By using composite kernels, arbitrarily complex feature spaces can be created, and, so long as each input kernel is a known Mercer kernel, then the composition of those kernels will be as well.

Model Performance Estimators

The performance of SVMs and other kernel-based learning algorithms is highly dependent on the choice of a kernel and kernel parameters. For example, figure 1 shows the CV error rate on the thyroid dataset for an SVM with the RBF kernel as a function of γ, the width of the RBF. Note in this graph, that there is a limited range of values of γ for which CV errors are minimized. In general, "there is no free kernel" (Cristianini 2001). That is, there is no kernel that works best on all datasets or, said differently, there is a need for domain-specific and task-specific kernels.

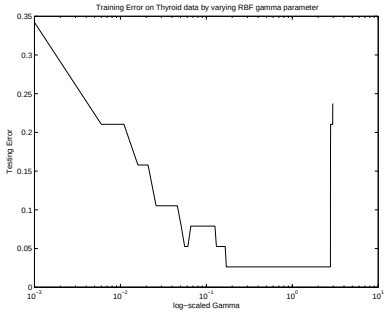

Figure 1: CV Error of RBF as γ varies on Thyroid

As Figure 1 demonstrates, once the functional form of a kernel is chosen, selecting appropriate parameter values is a crucial step. When working with composite kernels, the parameters of multiple atomic kernels may need to be tuned simultaneously. Therefore, we review existing methods for parameter selection to identify those that are most efficient and effective for use in our algorithm.

Most parameter selection algorithms execute a search to find the optimum value of an error estimator (e.g., (Chapelle *et al.* 2002),(Joachims, Cristianini, & Shawe-Taylor 2001)). The quality of the results of the search depends on the quality of the estimator. There are three principle types of error estimators, including *Cross Validation*, *Heuristic*, and *Theoretical Bounds*. The following is a summary of different estimators, and a comparison of how those estimators each performed on the various datasets used in the experiments described in the results section. This information was used to select the estimators used to optimize composite kernel parameters in those experiments.

Cross Validation and Leave One Out Error Estimates Cross Validation (CV) and Leave-One-Out (LOO) are common performance estimators. They are an empirical estimate computed by first selecting a set of parameters (such as γ in an RBF kernel), learning a model using a portion of the training data, and then measuring the error rate of the model using held-out training data. This makes it a good target function for the search to optimize. When the size of the training set is large enough, *n*-way CV produces an error estimate that has little bias and variance (Hastie, Tibshirani, & Friedman 2001). Empirically, this approach tends to create models that most closely approximate the underlying distribution in the data, while still generalizing well to held-out testing data (Duan, Keerthi, & Poo 2003).

Heuristics Heuristics estimate the performance of a model based on empirical evidence. Most of these heurisitic bounds can be derived either directly from the training data or from the resulting model. Heuristic bounds are often not tight, and will tend to over-estimate the true error on the validation set.

One typical example of this type of estimator is the *Support Vector Count* heuristic bound (Chapelle *et al.* 2002):

$$T = \frac{N_{SV}}{\ell} \tag{7}$$

where T is the estimated upper-bound on the errors made by the leave-one-out procedure, and N_{SV} is the number of support vectors for a model with ℓ observations. After the SVM model is created, this heuristic bound is extremely easy to compute. The drawback is that the heuristic bound may overestimate the true error.

Kernel Target Alignment (Cristianini *et al.* 2001) is another heuristic bound which is based on maximizing the spectral alignment between the Gram matrix of the kernel and the labels.

Theorem 3 (Alignment). *Let K_1, K_2 be the $m \times m$ Gram matrices induced by different kernel functions (Cristianini* et al. *2001).*

$$A(K_1, K_2) = \frac{< K_1, K_2 >_F}{\sqrt{< K_1, K_1 >_F < K_2, K_2 >_F}} \tag{8}$$

and $< K_1, K_2 >_F = \sum_{i,j=1}^{m} < K_{1,i}, K_{2,j} >$.

If $K_2(x_i, x_j) = y_i y_j$, then if any kernel K_1 is aligned well with K_2 it should be the case that K_1 will perform well in terms of minimizing error rates.

Theoretical Error Estimates The final type of error estimate is based on structural risk minimization theory. This type of estimate establishes an upper bound on the errors that a model will make on validation data. There are many different upper-error bound estimators in this category, and they differ in accuracy and complexity.

One of the many theoretical estimated upper-bounds is the *Radius Margin* (R^2/M^2) bound. Let T represent the upper-bound on the number of LOO errors, then

$$T = \frac{1}{\ell} R^2 \|\mathbf{w}\|^2 \tag{9}$$

where the margin ($\frac{1}{2}\|\mathbf{w}\|^2$) is defined in (Chapelle *et al.* 2002) as the following, where K_θ is a Mercer kernel with parameters θ:

$$\frac{1}{2}\|\mathbf{w}\|^2 \quad = \quad \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j K_\theta(\mathbf{x_i}, \mathbf{x_j}) \tag{10}$$

subject to $\sum_i^\ell \alpha_i y_i = 0$ and $\forall i : \alpha_i \geq 0$.

Similarly, the radius (R^2) is defined as:

$$R^2 = max_\beta \sum_{i=1}^{\ell} \beta_i K_\theta(\mathbf{x_i}, \mathbf{x_i}) - \sum_{i,j=1}^{\ell} \beta_i \beta_j K_\theta(\mathbf{x_i}, \mathbf{x_j}) \tag{11}$$

subject to $\sum_{i=1}^\ell \beta_i = 1$ and $\forall i : \beta_i \geq 0$.

This theoretical bound is particularly interesting, as it is differentiable with respect to θ, the set of parameters of the kernel (Chapelle *et al.* 2002),(Keerthi 2004). Chapelle demonstrated that using the derivatives of the bound, a gradient descent algorithm is an efficient method to select kernel parameters. This particular heuristic still requires training an SVM model to determine the margin (eq. 10), and it must solve another quadratic problem to determine the β values of the radius R.

Minimization of the error estimator does not guarantee that model error will also be minimized at that point. A

brute-force search using the RBF kernel, and an adjustment factor of 0.001 on each of the RBF parameters in the range 0 to 5 found that the minimum radius-margin point occurred at $\alpha = 0.001$ and $C = 4.510$. The CV error rate observed at that point was 24.68% error. The minimum CV error rate was observed at $\alpha = 4.201$ and $C = 0.401$ with an observed error of 23.38%.

Estimator Selection Each of the previously described estimators have appeared previously in the literature for SVM parameter selection. They each were evaluated to assess their computational overhead and search performance for this paper. Two estimators, *Radius-Margin* R^2M^2 and *Training CV error* gave the best results for both the single RBF kernel and the composite kernel.

Approach

Our goal is to demonstrate that domain-specific composite kernels can be selected from the training data that perform statistically significantly better than the base kernels. Our approach was to find the optimum composite kernel parameters with respect to two performance estimators: *Radius-Margin*, and *Training CV Error*. Our approach required implementing two major steps. A composite kernel language was developed to define and evaluate kernels. A hill climbing search with random restart was used to enumerate composite kernel parameters, and the model with the best performance estimator was selected.

A scanner-parser was developed that recognizes the kernels shown in Table 1 and the composition rules described in Theorem 2. An interpreter was developed to evaluate the value of the parsed kernel function. The interpreter was compiled into SVM*lite*(Joachims 2002) as a custom kernel. An example of a kernel parameter string recognized by the kernel language is: $(K0() + K2(0.35))$.

Table 1: Kernel Notation

Linear	$K0()$	$< X, Y >$
Polynomial	$K1(d, s, r)$	$(s < X, Y > +r)^d$
RBF	$K2(\gamma)$	$\exp^{-\gamma\|X-Y\|^2}$
Sigmoid	$K3(s, c)$	$tanh(s < X, Y > +c)$

Evolutionary Algorithm

An evolutionary algorithm was used to search through a broad range of kernel compositions for a variety of data sets. Composite kernel functions were encoded as expression trees of kernel expressions. This enabled genes to represent arbitrarily complex kernel functions.

Table 2: Kernel Selection Probabilities

Kernel Type	Probability	
Simple Kernel		25%
Linear	10%	
Polynomial	30%	
RBF	60%	
Lambda Kernel		30%
Add/Multiply		25%
Constant		20%

The search began with a population of 200 randomly generated genes. Each gene was constructed by recursively populating its expression tree, randomly selecting one of the simple kernels (creating a leaf node) or another composite kernel (an inner tree node). Randomly generated expression trees were capped at a height of 4. Fitness was determined by observing cross-validation accuracy of the gene's kernel function. Two different cross-over algorithms were implemented. When the parents had the same kernel structure (i.e. two simple kernels of the same type), their function parameters were merged by averaging the values of each parameter. When the parents had different structures, a point in the expression tree of each parent was selected, and the child was constructed by splicing the tree of one parent to the tree of the other. The mutation operator would either randomly change a single kernel parameter or randomly replace a node in the expression tree with a random kernel. Successive generations were created by using roulette wheel selection and applying one of the cross-over algorithms. This search proved to be less efficient than other techniques, but was able to explore a much less restricted search space than other types of searches. Empirical evidence from this procedure identified two composite kernel structures that produced good results in a variety of data sets, including those listed in this paper:

$$((\lambda K1(d, s, b)) * ((1 - \lambda)K2(\sigma))) \quad (12)$$

and

$$((\lambda K1(d, s, b)) + ((1 - \lambda)K2(\sigma))) \quad (13)$$

Because the genetic algorithm explored an extremely large number of composite kernels for a variety of datasets and invariably found one of the above kernels to be best, we restrict further attention to just these kernels.

If kernel K_1 projects instances into feature space Φ_1 and kernel K_2 projects instances into feature space Φ_2, into what feature spaces do the composite kernels above, denoted K_+ and K_*, project instances? Let $\Phi^i(x)$ denote the i^{th} feature in feature space Φ. Let $\Phi_1(x) \circ \Phi_2(x)$ denote the concatenation of two feature vectors. It is then easy to see that:

$$\begin{aligned} K_+(x_1, x_2) &= K_1(x_1, x_2) + K_2(x_1, x_2) \\ &= \langle \Phi_1(x_1), \Phi_1(x_2) \rangle + \langle \Phi_2(x_1), \Phi_2(x_2) \rangle \\ &= \sum_i \Phi_1^i(x_1)\Phi_1^i(x_2) + \sum_j \Phi_2^j(x_1)\Phi_2^j(x_2) \\ &= \langle \Phi_1(x_1) \circ \Phi_2(x_1), \Phi_1(x_2) \circ \Phi_2(x_2) \rangle \end{aligned}$$

That is, feature space Φ_+ contains the union of the features in Φ_1 and Φ_2. Similar algebraic manipulation of the expression for $K_*(x_1, x_2)$ leads to the conclusion that $\Phi_*(x) = \Phi_1(x) \times \Phi_2(x)$. That is, feature space Φ_* contains the cross product of the features in Φ_1 and Φ_2, or one dimension for the product of each pair of features drawn from the two feature spaces.

Hill Climbing

Hill Climbing was selected as the composite kernel search algorithm. The properties that made it ideal were that it performed well given the numerous local extrema in the search

space, had little overhead over the cost of model evaluation, and with random restart, yielded the best empirical results over other methods.

The hill climbing search used in this paper used CV accuracy as the target to improve. The search modified each of the parameters of the kernel formula by a fixed amount and moved in the direction of maximum CV accuracy. The search stops when the greatest improvement of any parameter is less than some small ϵ value. The search used 15 random restarts, and selected the models with the greatest accuracy.

The primary drawback of the Hill Climbing approach is the inefficiency from the generation of a large number of model evaluations. The empirical results suggest that this is not a concern in practice. When the hill climber detects a local plateau, it terminates that iteration of the search, and randomly restarts. In practice, there are many local plateaus, and the Hill Climber reaches them after only a few evaluations.

Empirical Results

Description of Data: The experiments were conducted using nine different data sets due to (Ratsch 1999). All of the data sets were 0-1 classification tasks, and composed of continuous, mean-centered, standardized data.

Table 3: BFGS / R^2M^2 Search Timing

	RBF		Composite	
Data	**Evals**	**Time (s)**	**Evals**	**Time (s)**
Banana	166	136	257	245
Breast	47	21	77	49
Diabetis	35	166	47	214
Flare	253	628	257	1041
German	67	1097	55	999
Heart	46	33	55	24
Image	43	1292	55	1587
Splice	35	1343	142	2716
Thyroid	215	25	72	13
Titanic	253	20	257	30

Table 4: Hill Climbing Search Timing

	RBF		Composite	
Data	**Evals**	**Time (s)**	**Evals**	**Time (s)**
Banana	116	11	368	69
Breast	120	9	358	30
Diabetis	116	18	362	106
German	120	46	362	163
Heart	120	342	358	3631
Image	120	323	360	2373
Splice	120	2283	346	16126
Thyroid	120	9	356	268
Titanic	120	156	366	3112

The data sets from Ratsch are distributed with 100 instances of the data split into training and testing data sets. Statistics such as R^2M^2 are generated using the data sets unchanged. Statistics such as Training CV Error are generated by further partitioning the training data. The Ratsch "testing" data is used as held-out data for final cross-validation estimation of the performance of the selected model. This was done to ensure that the error estimation was conservative, and that the models generalized well and were not overfitting the data.

Experiment Design

Implementation: The hill-climbing algorithm was implemented to optimize training CV error. The hill climber generates a set of parameters and trains an SVM on the first half of the training set and then tests on the second half of the training data. The hill-climber greedily selects the parameters that minimized training CV error, and then starts a new iteration. Ultimately, the model that generates the lowest training CV error was selected. The selected model is then used in 10 way CV on held-out data.

Two different performance estimates were also selected for comparative results, including Chapelle's R^2/M^2 approach, and training error minimization. The BFGS-Line Search found in MATLAB's Optimization Toolbox was used to minimize the R^2/M^2 measurement for the RBF and Composite kernels (Chapelle *et al.* 2002).

Results

The results comparing the performance of RBF and Composite Kernels are shown in Table 5. Using a standard 95% confidence interval, models using a composite kernel performed significantly better on four of the nine data sets (Diabetes, Image, Splice, Thyroid). The composite kernel performed worse than the others on Titanic. There was not a significant difference on the remaining four data sets (Banana, Breast-Cancer, German, Heart) and some other model. In general, models trained with a Composite Kernel performed at least as well as models trained with an RBF kernel, and often performed better.

Kernel formulas selected by the hill climbing search appear in table 6. The C parameter corresponds to the SVM penalization parameter. The formulas shown represent composite kernels consisting of a polynomial kernel, $K_1(d, s, b)$ and an RBF kernel, $K_3(\gamma)$.

Runtime performance for the BFGS-Line Search and Hill Climbing method appear in tables 3 and 4 respectively. These results were collected using a standard PC workstation with a 2.8Ghz Pentium Xeon processor. A comparison of the geometric mean of the run-time shows that for the composite kernels, Hill Climbing approach is 2.7 times slower than the BFGS search. A similar comparison shows that the Hill Climbing is 1.5 times slower for the composite kernel than for the RBF kernel. One reason for this is that each composite kernel requires computing two kernels and combining the results. Each kernel computation must stride through each of the columns in the data vectors. Thus, the data sets (such as Splice and German) that are 'wider' have a higher run-time penalty using the composite kernels. In general, the cost of the more aggressive Hill Climbing search is not unreasonable for the gain in training accuracy. Additionally, evaluation of the results could easily be done in parallel, leading to further reduction in training time.

Decision Boundary: One clue to the improvement in performance of the Composite kernels can be visualized on another dataset. Using Principal Components Analysis

Table 5: **Results** Estimated error rates on held-out data with the RBF and Composite Kernels using 10-way cross validation error and the 95% confidence interval are shown. Results that are not significantly worse than the best model are shown in bold face.

Kernel	RBF Kernel			Polynomial Kernel	Composite Kernel
Target	BFGS/Train$_{err}$	BFGS/R^2/M^2	HC/Train$_{err}$		HC/Train$_{err}$
Banana	$10.3 \pm 0.26\%$	$10.5 \pm 0.37\%$	$\mathbf{10.4 \pm 1.48\%}$	$10.8 \pm 2.00\%$	$\mathbf{9.7 \pm 0.54\%}$
Breast Cancer	$\mathbf{28.9 \pm 0.65\%}$	$\mathbf{29.1 \pm 0.40\%}$	$\mathbf{36.7 \pm 8.40\%}$	$\mathbf{26.0 \pm 16.2\%}$	$\mathbf{29.7 \pm 4.54\%}$
Diabetes	$36.3 \pm 0.39\%$	$36.3 \pm 0.32\%$	$28.6 \pm 5.84\%$	$\mathbf{24.7 \pm 5.2\%}$	$\mathbf{22.9 \pm 3.33\%}$
German	$\mathbf{31.0 \pm 0.42\%}$	$\mathbf{31.0 \pm 0.36\%}$	$\mathbf{31.0 \pm 6.39\%}$	$35.7 \pm 9.2\%$	$\mathbf{29.9 \pm 2.91\%}$
Heart	$\mathbf{19.2 \pm 1.93\%}$	$48.4 \pm 3.55\%$	$44.6 \pm 10.11\%$	$44.6 \pm 16.3\%$	$\mathbf{16.1 \pm 3.97\%}$
Image	$22.8 \pm 4.34\%$	$13.1 \pm 1.29\%$	$7.9 \pm 2.12\%$	$14.0 \pm 3.1\%$	$\mathbf{3.6 \pm 0.60\%}$
Splice	$17.3 \pm 0.79\%$	$49.3 \pm 1.36\%$	$16.4 \pm 1.75\%$	$14.9 \pm 1.7\%$	$\mathbf{10.8 \pm 1.08\%}$
Thyroid	$16.6 \pm 7.93\%$	$7.8 \pm 1.72\%$	$13.1 \pm 5.97\%$	$17.8 \pm 10.2\%$	$\mathbf{5.7 \pm 4.59\%}$
Titanic	$\mathbf{22.3 \pm 0.50\%}$	$\mathbf{22.5 \pm 0.45\%}$	$\mathbf{22.9 \pm 2.91\%}$	$32.5 \pm 2.2\%$	$25.2 \pm 1.80\%$

Table 6: Selected Kernel Functions

Data	C	Formula
Ban.	1.00	$0.06K_1(3, 1.07, 1.07) + 0.94K_3(4.52)$
BCan.	.20	$0.41K_1(1, 0.99, 0.89) + 0.59K_3(1.20)$
Diab.	.01	$0.05K_1(2, 1.37, 1.37) + 0.95K_3(5.81)$
Ger.	.29	$0.35K_1(1, 0.12, 0.12) + 0.65K_3(4.14)$
Hrt.	.03	$0.31K_1(1, 0.61, 1.23) + 0.69K_3(0.95)$
Img.	.10	$0.62K_1(3, 0.95, 1.01) + 0.38K_3(0.32)$
Spl.	.01	$0.22K_1(2, 0.42, 0.16) + 0.78K_3(5.36)$
Thy.	.23	$0.87K_1(4, 0.38, 1.23) + 0.13K_3(1.11)$
Ttnc	.75	$0.99K_1(2, 1.06, 1.07) + 0.01K_3(3.75)$

(PCA), the *Ionosphere* dataset was projected into two dimensions. Using the parameter selection method described, the parameters for a polynomial, RBF, and composite kernel were selected.

The decision boundary induced by each of the three kernels is shown in Figure 2. The first plot corresponds to the polynomial kernel, and is dominated by a region in the center of the graph that negatively classifies points. Moving outward, the lightest colored ring is the margin, and the zone of positive values is extremely small (in this graph). In the plot for the RBF kernel, there is a nucleus of extremely strong negative classification surrounded by a definite margin, and another nucleus of strong positive classification. The third graph shows the composite kernel's decision boundary. The two nuclei from the RBF plot are still present, although weakened in both shape and magnitude. The margin has been shifted up on the upper-center region. Intuitively, this is the due to the effects of the polynomial kernel's strong central well. It is in this new region that the composite kernel correctly classifies additional points leading to its improved performance.

Discussion

It has often been said of the RBF kernel that it is "universal", or that it has the flexibility via parameter γ to fit, and not overfit, any decision surface. Why, then, would we see improvements in classification accuracy when combining RBF and polynomial kernels? What is the feature space into which the RBF kernel projects the data?

Recall that for RBF kernel K:

$$K(x,y) = e^{-\gamma\|x-y\|^2}$$

$$= e^{-\gamma\sum_i (x_i^2 - 2x_iy_i + y_2^2)}$$

A Taylor series expansion of this expression leads to the following sum:

$$K(x,y) = \sum_{j=0}^{\infty} \frac{\gamma^j}{j!}(-1)^j \left(\sum_i x_i^2 - 2x_iy_i + y_2^2\right)^j$$

Expanding the expression above results in a polynomial with monomials of even degree comprising terms from both x and y. As j grows large, two things happen. The degree of the monomials in the polynomial grows large as well, and the leading term $\gamma^j/j!$ tends to zero. The rate of decline is dependent on γ, which is inversely proportional to the width of the basis function (in this case a Gaussian). When γ is large the basis functions are tight and the multiplier on the monomials goes to zero more slowly, allowing higher order terms to have an impact. Conversely, when γ is small the basis functions are broad and the multiplier on the monomials goes to zero rapidly, causing lower order terms to dominate.

Note that an RBF kernel can never rid itself of the lower order terms. One can only increase γ to increase the weight on higher order terms, at the same time increasing the weight on the lower order terms. Multiplying a polynomial kernel and an RBF kernel increases the degree of all of the terms in the composite kernel, in effect dropping the lower order terms from the RBF kernel. The number of lower order terms dropped depends on the degree of the polynomial kernel. This allows the RBF kernel to fit separators that are best described without the use of lower order terms.

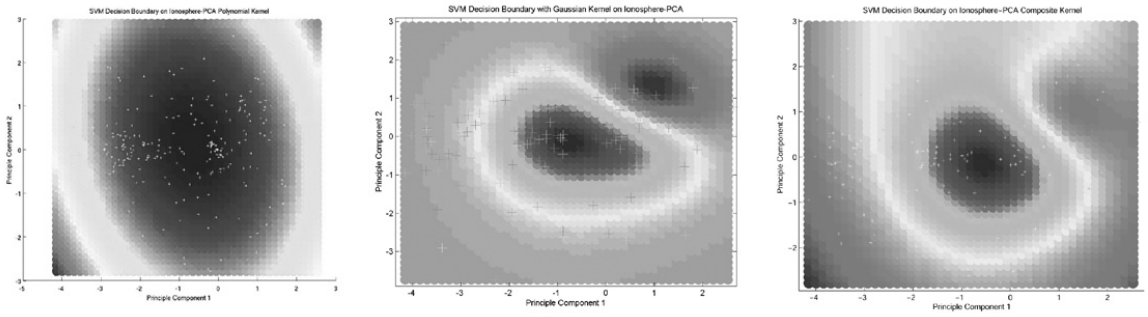

Figure 2: Decision Boundary of polynomial, RBF, and Composite Kernels

Conclusion

We have demonstrated that composite kernels can improve classification accuracy over the standard atomic kernels. We demonstrated that composite kernels can be selected by standard minimization techniques using the same target functions employed in selecting parameters for the RBF kernel.

The performance of the search methods used here yielded surprisingly good general performance. In general, parameter selection requires only a few minutes to complete. We demonstrated that our approach can improve training accuracy, at a cost of increased time to select kernel parameters.

One area of future work is to compute the gradients of the composite kernel parameters. The gradients are used to by the BFGS-Line Search algorithm to adjust the parameters of the model. Finite-difference analysis is less efficient, as it must generate models to estimate the gradients of the parameters. This would likely only improve the time cost and not the classification accuracy of the resulting models.

References

Ayat, N.; Cheriet, M.; and Suen, C. 2002. Kmod-a two parameter svm kernel for pattern recognition.

Bach, F., and Jordan, M. 2003. Kernel independent component analysis. *J. Mach. Learn. Res.* 3:1–48.

Ben-Hur, A.; Horn, D.; Siegelmann, H.; and Vapnik, V. 2002. Support vector clustering. *J. Mach. Learn. Res.* 2:125–137.

Chapelle, O.; Vapnik, V.; Bousquet, O.; and Mukherjee, S. 2002. Choosing multiple parameters for support vector machines. *Machine Learning* 46(1-3):131–159.

Cristianini, N.; Shawe-Taylor, J.; Elisseeff, A.; and Kandola, J. 2001. On kernel-target alignment. In *NIPS*, 367–373.

Cristianini, N. 2001. Support vector and kernel machines. website. Presented at ICML 2001.

Cristianini, N. 2002. Kernel methods for pattern analysis. website. Presented at NATO Advanced Study Institute.

Duan, K.; Keerthi, S.; and Poo, A. 2003. Evaluation of simple performance measures for tuning svm hyperparameters. *Neurocomputing* 51:41–59.

Friedrichs, F., and Igel, C. 2004. Evolutionary tuning of multiple svm parameters.

Hastie, T.; Tibshirani, R.; and Friedman, J. 2001. *The Elements of Statistical Learning; Data Mining, Inference, and Prediction*. Springer Series in Statistics. New York, NY: Spring-Verlag.

Joachims, T.; Cristianini, N.; and Shawe-Taylor, J. 2001. Composite kernels for hypertext categorisation. In *Proceedings of ICML-01, 18th ICML.*, 250–257.

Joachims, T. 2001. A statistical learning learning model of text classification for support vector machines. In *In. Proc. 24th ACM SIGIR conf. on Research and development in information retrieval*, 128–136. ACM Press.

Joachims, T. 2002. Svm^{lite}. software.

Keerthi, S. 2004. Efficient tuning of svm hyperparameters using radius/margin bound and iterative algorithms. URL. http://guppy.mpe.nus.edu.sg/ mpessk/nparm.shtml.

Ratsch, G. 1999. Benchmark repository.

Schölkopf, B., and Smola, A. J. 2002. *Learning with Kernels*. MIT Press.

Schölkopf, B.; Smola, A.; and Müller, K. 1999. *Kernel principal component analysis*. MIT Press.

Wahba, G.; Lin, Y.; Lee, Y.; ; and Zhang, H. 2001. On the relation between the gacv and joachims' xi-alpha method for tuning support vector machines, with extensions to the non-standard case. Technical Report 1039, Dept. of Statistics, Univ. of Wisc.