A SPARSE GREEDY SELF-ADAPTIVE ALGORITHM FOR CLASSIFICATION OF DATA

ANKUR SRIVASTAVA
Mechanical Engineering and Material Science Department, William Marsh Rice University,
6100 Main Street, Houston, Texas 77005-1827, Mail Stop 124
ankur.srivastava@rice.edu

ANDREW J. MEADE
Mechanical Engineering and Material Science Department, William Marsh Rice University,
6100 Main Street, Houston, Texas 77005-1827, Mail Stop 321
meade@rice.edu

Received ()
Revised ()

Kernels have become an integral part of most data classification algorithms. However the kernel parameters are generally not optimized during learning. In this work a novel adaptive technique called Sequential Function Approximation (SFA) has been developed for classification that determines the values of the control and kernel hyper-parameters during learning. This tool constructs sparse radial basis function networks in a greedy fashion. Experiments were carried out on synthetic and real world data sets where SFA had comparable performance to other popular classification schemes with parameters optimized by an exhaustive grid search.

Keywords: Greedy algorithms; radial basis function; method of weighted residuals; control parameters; statistical significance; sparsity.

1. Introduction

Pattern recognition is widely studied in the area of machine learning research. This subject studies tools capable of conducting data analysis and knowledge extraction on real world noisy data. Popular pattern recognition algorithms include back-propagation networks [Cun et al. (1990)], k-nearest neighbor [Dasarathy (1991)], classification trees [Breiman et al. (1984)], support vector machines [Scholkopf and Smola (2002)] and radial basis function networks [Buhmann (2003)]. However, a major drawback of the training and testing approach used by the aforementioned data classification tools result from the dependence on user-determined control parameters. Most classifiers require the use of control parameters and/or kernel hyper-parameters. A widely accepted practice to determine the values of these parameters is to perform a grid search over the training or validation set which is necessary to find optimal values of the control and/or kernel hyper-parameters. Consequently, the optimum performance of the classifier depends heavily on the user-controlled parameters and is difficult to obtain. To address this performance problem, a self-adaptive classifier has been developed that determines the values of the control and kernel parameters during learning. Such a classification tool would assign different parameters to each basis function thereby reducing the dependence of the data classification accuracy on any initial choice of control and kernel hyper-parameters.

In this work we focus on constructing Gaussian radial basis function networks, however, this technique can be applied to a number of basis functions like hyperbolic tangents, polynomials, B-splines, and trigonometric functions. Owing to its greedy nature the algorithm constructs sparse radial basis function networks. Sparsity provides redundancy and robustness to the nonlinear model and is directly related to the generalization capacity of the model [Grapel et al. (2000)].
In this work we propose a self-adaptive data classification technique that estimates the width and center of each Gaussian radial basis function during training. It does not need a validation set to estimate the learning parameters. To evaluate the performance of our algorithm we compare the classification accuracies against popular approaches like support vector machines and nearest neighbor classifiers on simulated and real world data sets with their parameters optimized by an exhaustive grid search.

The developed classifier is called Sequential Function Approximation (SFA) and is based on Method of Weighted Residual interpretation of a greedy sparse approximation technique. The remainder of the paper is organized as follows. In Section 2 a review of other related sequential approximation methods is presented. In Section 3 the algorithm of SFA is presented with its implementation issues. In Section 4 results are shown and experimental comparisons of SFA with other standard classifiers. Conclusions and future avenues of research are discussed in Section 5.

2. Related Algorithms

In this section we present the links between SFA and other sequential approximation algorithms present in the literature. Greedy sequential methods have been an area of active research for many years in the fields of signal processing, neural networks, kernel methods and statistics. Sequential greedy approximation methods construct a series of locally optimum solutions in the spirit of obtaining the global optimum.

2.1. Greedy algorithms

Greedy algorithms are constructive approximation algorithms that decompose the target vector into a linear sum of basis functions. These algorithms are highly nonlinear because they use the target vector to determine the coefficients of the linear sum and to choose the next basis function. We focus on function approximation algorithms that construct models of the form

\[ y_n(x) = \sum_{i=1}^{n} c_i \phi(x, \beta_i) + b_i \]

For classification problems a proxy function, for example

\[ y_n^e(x) = \text{sign}\left( \sum_{i=1}^{n} c_i \phi(x, \beta_i) + b_i \right) \]

denotes the input vector, \( y \in \{-1, 1\} \) denotes the target and \( c_i, b_i \in \mathbb{R} \) are the linear coefficient and the bias respectively. \( y_n^e \) is the approximation of the target vector after the addition of \( n \) basis functions. The basis functions \( \phi(x, \beta_i) \) are Gaussian radial basis functions with kernel parameters \( \beta_i \) that includes the basis center \( x_i^* \) and the width \( \sigma_i \). For the remainder of this work we will assume the width to be uniform in all \( d \) dimensions. To construct the best \( n \)-term approximation of the target vector, we need to solve for four unknowns, namely, \( c_i, b_i, \sigma_i, \) and \( x_i^* \) for each basis function. This would yield a \( 4n \) dimensional optimization problem assuming that the basis centers are chosen from the available training points. A greedy function approximation approach allows us to solve for all of the four unknowns for each basis function at a time.

Typical sparse kernel modeling techniques simplify this problem by fixing the width of the basis functions thereby reducing the number of optimization problems to be solved. A greedy function approximation algorithm picks a basis function from a given set of basis functions, calculates the residual error of the approximation, and uses the residual to pick the next basis function. Counterparts of greedy algorithms exist as Matching Pursuit [Mallat and Zhang (1993)] in the signal processing community and Boosting [Freund and Schapire (1996)] in the statistics community. The given collection of basis functions is often called a dictionary. The Matching Pursuit algorithm chooses a basis function from the dictionary...
that maximizes the absolute inner product of the basis function \( \phi_n \) with the residual vector \( \tilde{r}_{n-1} \). Geometrically this can be interpreted as choosing a \( \phi_n \) that makes the most acute angle with \( \tilde{r}_{n-1} \) in \( \mathbb{R}^s \) where \( s \) is the number of training points. Here we represent the \( n^{th} \) basis function vector \( \phi_n = [\phi(x_1, \beta_n), \phi(x_2, \beta_n), \ldots, \phi(x_s, \beta_n)] \) compactly as \( \phi_n \) and the residual vector at \( n^{th} \) stage \( \tilde{r}_n = [r_n(x_1), r_n(x_2), \ldots, r_n(x_s)] \) compactly as \( \tilde{r}_n \). In Section 2.2 we stress on the differences between SFA and Matching Pursuit algorithms in order to throw light on some of the key strengths of SFA. Later in Section 4 we compare SFA against these algorithms in order to corroborate our points with experimental results. In Sections 2.3 and 2.4 we present some other greedy approximation techniques that share some similarities with SFA. Comparison of SFA with these algorithms will be a part of the future work.

### 2.2. Kernel matching pursuit

Vincent and Bengio (2002) showed how Matching Pursuit could be used to build a kernel based solution to a machine learning problem and called it Kernel Matching Pursuit (KMP). It is a greedy algorithm for building an approximation of a discriminant function as a linear combination of basis functions chosen from a kernel-induced dictionary. Vincent and Bengio (2002) compared the performance of KMP with SVM and RBF networks on several real world data sets. They concluded that KMP gave comparable results to SVMs with sparser approximations. However, Suykens et al. (2002) showed that sparser solutions can be obtained by SVMs via a pruning strategy.

The proposed algorithm bears some similarities with the basic version of KMP but the primary differences are the following. In KMP the basis function at the \( n^{th} \) stage is chosen such that it maximizes \[ \left| \frac{\tilde{r}_{n-1} \cdot \phi_n}{\tilde{r}_n \cdot \tilde{r}_n} \right|. \] Since a constant width is chosen and the basis functions are centered on the training points, choosing a new basis function amounts to choosing a basis center from the available training data based on the above mentioned criterion. From a geometrical perspective such an approach will choose the basis function \( \phi_n \) that is most aligned with the residual vector \( \tilde{r}_{n-1} \) in \( \mathbb{R}^s \). An improvement is possible if the width of the basis functions is not fixed. Allowing the width to vary would increase the size of the dictionary generated by the training points. A larger dictionary allows us to choose a better basis function that maximizes the above mentioned criterion.

In the proposed approach we optimize for the spread of each basis function and choose the basis center corresponding to the maximum absolute value of the residual at that stage. This can be interpreted as choosing basis functions from a larger dictionary that are aligned more parallel to \( \tilde{r}_{n-1} \) in \( \mathbb{R}^s \). Choosing a new basis function in this manner will diminish more of the residual error and result in sparser approximations as compared to KMP.

The basic version of KMP did not perform to satisfaction in the experiments done by Vincent and Bengio (2002). To address those limitations the authors proposed a back fitting and a pre-fitting version of KMP. Properties of orthogonal projections were used in the back-fitting and the pre-fitting versions of KMP to update all previously chosen basis function and coefficients of linear sum. The aim of the back-fitting and pre-fitting versions of KMP was to select a basis function that is the most aligned to the residual vector and is the most perpendicular to the previously chosen basis function. Keeping the basis centers fixed at the training points and the width of the RBF fixed does not always result in a good choice of basis function from the available dictionary. In this work we show that allowing the width of the basis function to vary will result in a better choice of the basis function because we get one more degree of freedom to choose the basis function. In Section 3 we compare SFA against the basic version of KMP and the pre-
fitting version of KMP on artificial and real world classification problems. The results show that SFA obtained sparser classifiers with similar classification accuracies on almost all data sets. Moreover only one pass through the dictionary was required by SFA to choose the basis function as opposed to the pre-fitting version of KMP that required two passes.

### 2.3. Sparse approximation methods

Sparse approximation methods broadly fall in the following broad categories: a) sequential forward greedy algorithms, b) backward greedy algorithms and c) mathematical programming approaches [Nair et al. (2002)].

KMP and SFA are examples of sequential forward greedy algorithms. Other popular examples are Order Recursive Matching Pursuit (ORMP) [Natarajan (1995)] and Orthogonal Least Squares RBF (OLS-RBF) [Chen et al. (1991)]. Vincent and Bengio (2002) described the pre-fitting version of KMP with squared error loss and Gaussian kernel to be identical to the OLS-RBF algorithm. They compared OLS-RBF against Gaussian Support Vector Machines (SVM) to conclude that OLS-RBF performed as well as Gaussian SVM with sparser approximations. One of the contributions of this work is to compare the performance of Least Square Support vector Machines (LS-SVM) against KMP algorithms and SFA. The primary difference between ORMP and SFA is that ORMP normalizes and reorients all unselected basis functions and avoids the recycling problem by choosing from previously unselected basis functions. SFA avoids the recycling problem by assigning a different width to each basis function. So even if the center selection routine makes a poor choice of the basis center from the available training points, because of the width optimization step a different and a relatively better basis function is selected at each step and the recycling problem is avoided. Some other greedy algorithms in the spirit of Natarajan’s algorithm have also been published [Nair et al. (2002)]. But in these works the question of dependence of classification accuracy on the width of the Gaussian radial basis functions has not been addressed.

The backward greedy algorithms [Couvreur and Bresler (1999)] start with a \( s \times s \) Gram matrix, where \( s \) is the number of observations in the training set and iteratively eliminates columns. Even though this helps in achieving guaranteed convergence properties it is more computationally expensive than the forward algorithms. The mathematical programming approaches like Basis Pursuit differs from the sequential forward greedy algorithms in the sense that they use regularization in the cost function to determine a sparse solution. Basis pursuit also employs quadratic programming like support vector machines to minimize the associated cost function. Girosi (1998) proposed a modified version of Basis Pursuit De-noising [Chen (1995)] and showed that SVMs are equivalent to them as they solve the same quadratic programming problem. However sequential forward greedy algorithms like SFA and KMP enforce regularization via sparsity.

### 2.4. Other related methods

In the statistics community the ideas of greedy learning are popular by the name of Boosting [Freund and Schapire (1996)] and Projection Pursuit [Friedman and Stuetzle (1984)]. KMP in its basic form could be seen very similar to Boosting if the weak learners were seen as the kernel functions centered on the training points. The Projection Pursuit method constructs the approximation of the target vector as a linear sum of ridge functions in a forward stagewise manner. The input training points are projected on a unit direction vector which is optimized at each stage. A ridge function which results in the maximum reduction of the residual error is selected. The addition of ridge functions continue till the residual error is smaller than a
Sequential Function Approximation

We start our approximation of the unknown target function \( y \) by noting that a continuous \( d \)-dimensional function can be arbitrarily well-approximated by a linear combination of radial basis functions \( \phi \).

Sequential Function Approximation (SFA) was developed from mesh-free finite element research but shares similarities with the Boosting and Matching Pursuit algorithms. We start our approximation of the target vector \( y \) utilizing the Gaussian radial basis function \( \phi \).

\[
y_n^d = \sum_{i=1}^{s} c_i (\phi(x, \beta_i) + b_i) \quad \text{with} \quad \phi(x, \beta_i) = \exp\left[\log(\alpha_i)(x-x_i^*) \cdot (x-x_i^*)^T\right] \quad \text{for} \quad 0 < \alpha_i < 1. \quad (1)
\]

Traditionally Gaussian radial basis functions are written as \( \phi(x, \beta_i) = \exp\left[-\frac{(x-x_i^*) \cdot (x-x_i^*)}{\sigma_i^2}\right] \). We write the radial basis function as Eq. (1) in order to setup the optimization problem for \( \alpha_i \) as a bounded nonlinear line search instead of an unconstrained minimization problem for \( \sigma_i \). The basic principles of our greedy algorithm are motivated by the similarities between the iterative optimization procedures by Jones (1990; 1992) and Barron (1993) and the Method of Weighted Residuals (MWR), specifically the Galerkin method [Fletcher (1984)]. We can write the function residual vector \( \tilde{r}_n \) at the \( n \)th stage of approximation as in Eq. (2):

\[
\tilde{r}_n = y(x) - y_n^d(x) = y(x) - y_{n-1}^d(x) - c_n (\phi(x, \beta_n) + b_n)
\]

\[
= \tilde{r}_{n-1} - c_n (\phi(x, \beta_n) + b_n) = \tilde{r}_{n-1} - c_n (\tilde{\phi}_n + b_n).
\quad (2)
\]

Using the Petrov-Galerkin approach, we select a coefficient \( c_n \) that will force the function residual to be orthogonal to the basis function and \( b_n \) using the discrete inner product \( \langle \cdot, \cdot \rangle_D \) given by Eq. (3)

\[
\tilde{r}_n \cdot (\tilde{\phi}_n + b_n) = \sum_{j=1}^{s} \tilde{r}_n(x_j) (\tilde{\phi}_n(x_j) + b_n) = \tilde{r}_n^D(\tilde{\phi}_n + b_n) = \frac{1}{2} \frac{\partial \tilde{r}_n^D}{\partial c_n} = 0
\quad (3)
\]

which is equivalent to selecting a value of \( c_n \) that will minimize \( \langle \tilde{r}_n^*, \tilde{r}_n^* \rangle_D \), or

\[
c_n = \frac{s \langle \tilde{r}_{n-1}^*, \tilde{r}_{n-1}^* \rangle_D - \langle \tilde{r}_{n-1}^*, \tilde{\phi}_n^* \rangle_D}{s \langle \tilde{r}_{n-1}^*, \tilde{\phi}_n^* \rangle_D - \langle \tilde{\phi}_n^*, \tilde{\phi}_n^* \rangle_D}
\quad (4)
\]
The discrete inner product $\langle \vec{r}_i, \vec{r}_j \rangle_D$, which is equivalent to the square of the discrete L_2 norm, can be re-written, with the substitution of Eq. (4) and (5), as

$$\sum_{j=1}^{s} r_x(x_j) r_x(x_j) = \left\| \vec{r}_i \right\|_{L^2,D}^2 = \langle \vec{r}_i, \vec{r}_i \rangle_D$$

where $\vec{r}_i$ is the dot product of $\vec{r}_i$ and $\vec{r}_j$ since $\langle \vec{r}_i, \vec{r}_j \rangle_D$ and $\langle \vec{r}_j, \vec{r}_j \rangle_D$ are scalars. With Eq. (8) we note that $\| \vec{r}_i \|_{L^2,D}^2 < \| \vec{r}_j \|_{L^2,D}^2$ as long as $\theta_n \neq \pi / 2$, which is a very robust condition for convergence. By inspection, the minimum of Eq. (8) is $\theta_n = 0$, implying Eq. (9)

$$c_n (\phi(x_i, t_i) + b_n) = r_n(x_i) \quad \text{for} \quad i = 1, \ldots, s .$$

Therefore, to force $\| \vec{r}_i \|_{L^2,D}^2 \to 0$ with as few stages $n$ as possible, a low dimensional function approximation problem must be solved at each stage. This involves a bounded nonlinear minimization of Eq. (6) to determine the two variables $\alpha_i$ ($0 < \alpha_i < 1$) and index $j^*(x_i) = x_j \in \mathbb{R}^d$ for the basis function center taken from the training set. The dimensionality of the nonlinear optimization problem is kept low since we are solving only one basis at a time.
3.1. Implementation of the algorithm

Though our SFA scheme allows the basis center \((x^*_n)\) to be located anywhere in \(\mathbb{R}^d\), the practical application to problems with multiple inputs constrains the centers to the set of sample points \(\{x_1, \ldots, x_s\}\). At each stage we determine \(x^*_n\) such that \(|\tilde{r}_{n-1}(x^*_n)| = \max |\tilde{r}_{n-1}|\). The remaining optimization variable \(\alpha_n\) is continuous and constrained to \(0 < \alpha_n < 1\).

A benefit to using radial basis functions is that in practical applications we can ignore the denominator in the discrete inner product formulation of Eq. (6). As a result, the determination of \(\beta_n\) requires only

\[
\min_{\beta_n} \left[ -\left( \tilde{r}_{n-1} - \frac{\langle \tilde{r}_{n-1} \rangle_D}{S}, \tilde{\phi}_n - \frac{\langle \tilde{\phi}_n \rangle_D}{S} \right)^2_D \right].
\]

Optimization of Eq. (6) with respect to \(\sigma\) is a non-convex and an unconstrained nonlinear optimization problem. But formulating the basis function as shown in Eq. (1) allows us to formulate the optimization problem for \(\alpha\) as a bounded line search problem. Traditional bracketing methods such as Golden selection search algorithms coupled with parabolic interpolation methods can be used for such problems. An advantage of such an approach is that it does not require any initial value of the optimization parameter but is based on the approximation that the objective function is unimodal in the prescribed interval \(0 < \alpha < 1\).

However, in this problem, assuming unimodality is safe because the interval is small. The algorithm terminates when \(\max |\tilde{r}_{n}| \leq \tau\), where \(\tau\) is the tolerance desired by the user. For binary classification, where \(y = +1\) or \(-1\), we set \(\tau = 1.0\). Setting an upper bound on the maximum absolute value of the residual balances the empirical risk and the complexity of the model thereby preventing overfitting. In the case of a classification problem setting a tolerance value of 1.0 suffices because unlike regression problems the classifier needs to learn only the sign of the approximation.

In this work multi-class classification problems are tackled by applying several binary classifiers in parallel. Approaches to combining binary classifiers include the one vs. one combination, the one vs. all pairing, and the error correcting output coding. There is no one best approach to combine several binary classifiers. Even though one vs. all has its own shortcomings, in this work this approach was used because of its simplicity and the widely accepted fact that it often produces good results. Rifkin (2004) compared the one vs. all approach to other existing methods and concluded that one vs. all classification will produce results as good as any other approach if the underlying classifiers are well-tuned. Comparison of other methods to combine binary classifiers deserves a separate study.

To implement the SFA algorithm the user takes the following steps:

1. Initiate the algorithm with the labels of the training data \(\tilde{r}_0 = \{y(x_1), \ldots, y(x_s)\}\).
2. **Basis center selection**: Search the components of \(\tilde{r}_{n-1}\) for the maximum magnitude. Record the component index \(j^*\).
3. \(x^*_n = x_{j^*}\).
4. **Basis width optimization**: With \(\tilde{\phi}_n\) centered at \(x^*_n\), minimize Eq. (10) using optimization methods for bounded line search problems.
5. **Coefficient calculation**: Calculate the coefficient \(c_n\) and \(b_n\) from Eq. (4) and (5) respectively.
6. **Residual update**: Update the residual vector $\tilde{r}_n = \tilde{r}_{n+1} - c_n (\phi_n + b_n)$. Repeat steps 3 to 6 until the termination criterion $\max|\tilde{r}_n| \leq 1.0$ has been met.

7. Use the constructed binary classifier to predict on the test set.

8. Repeat steps 1 to 7 for remaining classes.

9. Use one vs. all scheme to obtain the final prediction on the test set.

Our binary method is linear in storage with respect to $s$ since it needs to store only $s + sd$ vectors to compute the residuals: one vector of length $s$ ($\tilde{r}_n$) and $s$ vectors of length $d$ ($x_1, \ldots, x_d$) where $s$ is the number of samples and $d$ is the number of dimensions. To generate the SFA model requires two vectors of length $n$ ($\{c_1, \ldots, c_n\}$ and $\{b_1, \ldots, b_n\}$) and $n$ vectors of length $n+1$ ($\beta_1, \ldots, \beta_n$).

4. Results

In this section we present the comparison of SFA with three other popular data classification algorithms. The comparisons were made over seven real world data sets taken from the UCI Machine Learning Repository [Asuncion and Newman (1998)] and one synthetic data set. We conclude that SFA had performance comparable to other standard classifiers with their control and kernel parameters optimized by an exhaustive grid search.

4.1. Data sets

We first compare prediction abilities of the classifiers on a synthetic data set and then on real world data sets. An artificial binary Gaussian mixture toy data set was generated in the following manner. First ten means were generated from a bivariate Gaussian distribution $N((1,0), I)$ and this class was labeled as “+1” and is shown in red in Figure 1. Similarly ten means from the distribution $N((0,1), I)$ were generated and this class was labeled “-1” and is shown in green in Figure 1. Hundred observations belonging to each class were generated in the following way. For each observation belonging to Class +1 a mean $m1$ was randomly selected from one of the ten means, which generated $N(m1, 0.5*I)$. Similarly for each observation belonging to Class -1 a mean $m2$ was randomly selected from one of the ten means which generated $N(m2, 0.7*I)$ thus producing a mixture of Gaussian clusters for each class.

Seven real world application data sets were also included and downloaded from the UCI Machine Learning Repository. The HOUSING data set was converted to a 3 class problem for this study. The algorithms were compared on a total of 8 data sets shown in Table 1.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Number of Input Dimensions</th>
<th>Number of Classes</th>
<th>Size of the data set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toy</td>
<td>2</td>
<td>2</td>
<td>200</td>
</tr>
<tr>
<td>Iris</td>
<td>4</td>
<td>3</td>
<td>150</td>
</tr>
<tr>
<td>Wine</td>
<td>13</td>
<td>3</td>
<td>178</td>
</tr>
<tr>
<td>Glass</td>
<td>9</td>
<td>6</td>
<td>214</td>
</tr>
<tr>
<td>Housing</td>
<td>13</td>
<td>3</td>
<td>506</td>
</tr>
<tr>
<td>German Credit</td>
<td>24</td>
<td>2</td>
<td>1000</td>
</tr>
<tr>
<td>Balance Scale</td>
<td>4</td>
<td>3</td>
<td>625</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>34</td>
<td>2</td>
<td>351</td>
</tr>
</tbody>
</table>
4.2. Experimental procedure

The SFA classifier is compared to three other algorithms:

1. **LS-SVM**: LS-SVMLab1.5 [Suykens (2002)], a Matlab/C toolbox was used to implement the Least Squares Support Vector Machines with Gaussian radial basis functions. It uses two user-controlled parameters: the regularization parameter ($\gamma$) and the width of the Gaussian radial basis function ($\sigma$).

2. **k-NN**: SPIDER [Weston et al. (2006)], a Matlab-based machine learning toolbox was used to implement the $k$-nearest neighbor algorithm in this work. Gaussian radial basis functions were used to compute the distances between training/test points. It has two user-controlled parameters: the number of nearest neighbors ($k$) and the width of the Gaussian radial basis function ($\sigma$).

3. **KMP**: Kernel Matching Pursuit [Vincent and Bengio (2002)] was also used as one of the algorithms for comparison. The performance of the basic version of KMP (KMP) and the pre-fitting version of KMP (KMP-pre) was investigated with Gaussian radial basis functions. In the implementation of KMP and KMP-pre a tolerance ($\tau = 1.0$) on the maximum absolute value of the residual was used to stop the addition of basis functions. This is the same stopping criterion as used in the implementation of SFA. Other than the tolerance both versions of the algorithm have one user controlled parameter: the width of the Gaussian radial basis function ($\sigma$). Both versions were coded in Matlab version 7.5 to get the results presented in this paper.

The one vs. all scheme is used to combine different binary classifiers and perform multi-class classification for all the above mentioned algorithms. All algorithms were implemented using the MATLAB programming environment on a Windows-configured PC with a Pentium 4 2.66 GHz processor and 1.0 GB of RAM.

4.2.1. Grid search

The parameters of LS-SVM, k-NN, KMP and KMP-pre were determined by an exhaustive grid search. Both LS-SVM and k-NN have two control parameters so a two-dimensional grid was constructed between the following grid limits. For LS-SVM $\gamma$ is chosen from the range $2^{-5}$ to $2^{12}$ and $\sigma$ is chosen from the range $2^{10}$ to $2^{5}$. These grid intervals are taken from the technical report [Hsu et al. (2003)] where authors give recommendations about parameter grid search for support vector machines. For k-NN the grid search was performed over the two-dimensional parameter space ($k, \sigma$), $k$ is chosen from the range 1 to 100 and $\sigma$ is picked from the range same as chosen for LS-SVM. Each of the parameter ranges were discretized into ten equally spaced points to obtain a grid of hundred ($\gamma, \sigma$) and ($k, \sigma$) combinations. Once the training and test sets are obtained, they are loaded in the respective programs of the classifiers. Then training and testing were repeated on each grid point and classification accuracy was stored. Once this loose grid search is over a refined grid search is performed in the neighborhood of the obtained maximum classification accuracy. The above mentioned two dimensional grid is divided in 4 equal quadrants, and the quadrant in which the obtained maximum classification accuracy lies is determined. Then that quadrant is divided into 400 equally spaced grid points and training and testing are repeated on each grid point in that quadrant. For KMP and KMP-pre a line search was performed on 100 grid points equally spaced in the range $2^{10}$ to $2^{5}$. Consequently, optimal classification accuracy is obtained on the chosen test set for the above mentioned grid intervals.
4.3. Results

First we compare the classifiers on the binary artificial data set described in Section 4.1. A data set with 200 points was generated and then was randomly partitioned into 100 training and 100 test points. Performance of the classifiers is measured by calculating the percentage accuracy as shown below:

\[
\text{Percentage accuracy} = \left( \frac{N_{\text{test}} - m_{\text{test}}}{N_{\text{test}}} \right) \times 100 \%
\]

where \(N_{\text{test}}\) = number of points in the test set and \(m_{\text{test}}\) = number of misclassifications. Percentage error of each classifier on the Toy data set is shown below in Table 2.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>% accuracy</th>
<th>Sparsity</th>
<th>(\sigma)</th>
<th>Other parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>LS-SVM</td>
<td>89.00</td>
<td>58</td>
<td>2.24</td>
<td>(\gamma = 1401.30)</td>
</tr>
<tr>
<td>KMP</td>
<td>88.00</td>
<td>100</td>
<td>1.23</td>
<td>-</td>
</tr>
<tr>
<td>KMPpre</td>
<td>91.00</td>
<td>95</td>
<td>1.14</td>
<td>-</td>
</tr>
<tr>
<td>(k)-NN</td>
<td>81.00</td>
<td>-</td>
<td>28.58</td>
<td>(k = 1)</td>
</tr>
<tr>
<td>SFA</td>
<td>83.00</td>
<td>27</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

The first column of the table shows the percentage classification accuracies of all classifiers on the Toy test set obtained by the grid search method mentioned in Section 4.2.1. The resulting parameters are given in the last two columns. The width parameter \(\sigma\) for SFA is not mentioned because SFA being adaptive in nature uses a different width for each basis function. The classification accuracy of SFA is similar to other standard algorithms with their parameters optimized by an exhaustive grid search.

The second column of the table shows the number of training points that were used as basis functions. KMP used 100 basis functions but it used only 28 training points as basis centers, the rest of 72 basis functions were repetitions of the 28 already chosen basis functions. Since the width of all basis functions is the same choosing the same basis center means picking the same \(s\)-dimensional vector from the dictionary of all basis functions. KMPpre on the other hand is an improvement over KMP since it does not pick an already chosen basis function. But it does not result in a sparse classifier choosing 95 of the 100 training points. Now it should be noted here that the sparsity of a greedy classifier is dependent on the stopping criterion. It is possible that choosing a different stopping criterion such as an upper bound on the support vectors, minimum description length criterion, Akaike information criterion or a combination of different criterions can result in a sparser classifier. But in this work we chose a tolerance level on the maximum absolute value of the residual as the stopping criterion as is usually done by signal processing algorithms. A comparison of the different stopping criterion for the proposed greedy algorithm deserves to be a part of a separate study. SFA used 27 basis functions to train on the Toy training set. Out of the 27 basis functions, 7 of them were repetitions of already chosen basis centers. But since each basis function has a different width, repetition of a basis center does not mean that the same \(s\)-dimensional vector was picked from the dictionary of basis functions. LS-SVM used 58 training points as basis functions or support vectors. Figure 1 shows the classification boundaries obtained by the different classifiers on the Toy data set. Blue crosses represent support vectors used by each classifier. Since \(k\)-NN does not involve the use of support vectors, blue crosses are not shown in Figure 1c.
Fig. 1. Class decision boundaries produced by (a) SFA, (b) LS-SVM (c) $k$-NN, (d) KMP (e) KMPpre. Class +1 is shown in red and -1 in green. The hollow circles represent the test points. The blue crosses mark the support vectors used by the classifier. Blue crosses are not shown in Figure 1(c) because $k$-NN does not use support vectors to construct its approximation.

Since SFA, KMP and KMPpre are greedy methods the rate of convergence of $\{\tilde{r}_s, \tilde{r}_s\} \cap D$ can be studied as the basis functions are added.
It can be seen from Figure 2 that convergence of the residual error when SFA is used is faster than convergence rate of KMP or KMPpre. All three algorithms start with the initial residual error of $\langle \tilde{r}_{s}^1, \tilde{r}_{s}^1 \rangle_D = 100$. After the addition of the first basis function $\langle \tilde{r}_{s}^1, \tilde{r}_{s}^1 \rangle_D = 84.70$ for KMP, $\langle \tilde{r}_{s}^1, \tilde{r}_{s}^1 \rangle_D = 70.57$ for KMPpre and $\langle \tilde{r}_{s}^1, \tilde{r}_{s}^1 \rangle_D = 49.78$ for SFA. Also it should be noted that convergence rate of SFA on this problem is almost exponential. In Figure 2 residual error convergence is shown up to 25 basis functions because all three algorithms used at least 25 basis functions to converge. SFA used only 27 basis functions before stopping, residual error by KMPpre further decreases while the residual error almost remains constant using KMP.

4.4. Classification data sets

In this section compare the classifiers on seven real world data sets taken from the UCI Machine Learning Repository. The training set for each classification data set constitutes of fifty percent of randomly chosen points and the test set contains the rest of the points. Kernel hyper-parameters for each algorithm, except SFA, are selected by an exhaustive grid search method described in Section 4.2.1. Table 3 shows the hyper-parameter values for each data set obtained after the grid search.

After obtaining the hyper-parameter values mean and standard deviation of the classification accuracies were estimated on twenty random training and test sets containing fifty percentage of all the points in the data set. The percentage accuracies shown in Table 4 are unweighted in the sense that misclassification of one class is equally important as misclassification of any other class present in the data set.

Table 3. Kernel hyper-parameter values of the classifiers obtained after an exhaustive grid search.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>LS-SVM</th>
<th>k-NN</th>
<th>KMP</th>
<th>KMP-pre</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\gamma$</td>
<td>$\sigma$</td>
<td>$k$</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>Iris</td>
<td>107.82</td>
<td>8.00</td>
<td>9</td>
<td>22.63</td>
</tr>
<tr>
<td>Wine</td>
<td>431.19</td>
<td>10.46</td>
<td>15</td>
<td>27.39</td>
</tr>
<tr>
<td>Glass</td>
<td>323.40</td>
<td>9.93</td>
<td>6</td>
<td>25.01</td>
</tr>
<tr>
<td>Housing</td>
<td>107.82</td>
<td>12.38</td>
<td>1</td>
<td>5.96</td>
</tr>
<tr>
<td>German Credit</td>
<td>323.40</td>
<td>11.90</td>
<td>1</td>
<td>26.20</td>
</tr>
<tr>
<td>Balance Scale</td>
<td>215.61</td>
<td>10.86</td>
<td>14</td>
<td>23.82</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>2048.00</td>
<td>19.25</td>
<td>1</td>
<td>32.00</td>
</tr>
</tbody>
</table>
Table 4. Percentage accuracy obtained by the algorithms using half of the data set to train and the other half to test the classifier using Gaussian basis functions. For each data set the highest mean accuracies are shown in bold.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>LS-SVM</th>
<th>k-NN</th>
<th>KMP</th>
<th>KMP-pre</th>
<th>SFA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris</td>
<td>95.46 ± 1.88</td>
<td>96.25 ± 1.43</td>
<td>94.21 ± 3.98</td>
<td>91.58 ± 5.94</td>
<td>94.93 ± 2.87</td>
</tr>
<tr>
<td>Wine</td>
<td>95.22 ± 2.11</td>
<td>70.78 ± 3.46</td>
<td>71.61 ± 3.17</td>
<td>68.61 ± 2.99</td>
<td>70.22 ± 3.63</td>
</tr>
<tr>
<td>Glass</td>
<td>57.59 ± 3.80</td>
<td>62.13 ± 4.44</td>
<td>67.82 ± 3.39</td>
<td>66.99 ± 4.12</td>
<td>68.80 ± 4.04</td>
</tr>
<tr>
<td>Housing</td>
<td>72.95 ± 1.99</td>
<td>66.16 ± 2.05</td>
<td>66.24 ± 2.88</td>
<td>67.33 ± 2.84</td>
<td>68.44 ± 2.52</td>
</tr>
<tr>
<td>German Credit</td>
<td>72.52 ± 1.46</td>
<td>65.70 ± 1.10</td>
<td>70.79 ± 1.32</td>
<td>70.01 ± 1.99</td>
<td>69.87 ± 1.39</td>
</tr>
<tr>
<td>Balance Scale</td>
<td>84.30 ± 1.41</td>
<td>88.85 ± 1.96</td>
<td>88.43 ± 1.51</td>
<td>88.50 ± 1.14</td>
<td>86.92 ± 1.23</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>94.09 ± 1.55</td>
<td>85.57 ± 3.35</td>
<td>89.60 ± 1.62</td>
<td>82.74 ± 3.06</td>
<td>83.60 ± 1.66</td>
</tr>
</tbody>
</table>

4.5. Sparsity

In this work sparsity of a classifier is measured as the percentage of training data used by the classifier to predict the class of a test point. The sparsity of a classifier is calculated as the average number of support vectors used by all the binary classifiers. The average is expressed as a percentage of the number of training points and the values are recorded in Table 5. It should be noted here that for all algorithms except SFA, the parameters of the classifier and the kernel hyper-parameters used are optimal in the sense that they give the maximum performance on the test set. Also classification of a test point by $k$-nearest neighbors uses the entire training set due to its inherent structure.

Table 5. Percentage of training points used for prediction of a test point. Values corresponding to the $k$-nearest neighbor classifier are not shown as it uses 100% of the training set to predict the label of a test point. Sparsest results are shown in bold.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>LS-SVM</th>
<th>KMP</th>
<th>KMP-pre</th>
<th>SFA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris</td>
<td>54.71</td>
<td>37.76</td>
<td>17.59</td>
<td><strong>8.90</strong></td>
</tr>
<tr>
<td>Wine</td>
<td>69.61</td>
<td>88.50</td>
<td>34.69</td>
<td><strong>20.27</strong></td>
</tr>
<tr>
<td>Glass</td>
<td>60.42</td>
<td>69.68</td>
<td>47.46</td>
<td><strong>16.60</strong></td>
</tr>
<tr>
<td>Housing</td>
<td>63.60</td>
<td>95.12</td>
<td>83.78</td>
<td><strong>23.77</strong></td>
</tr>
<tr>
<td>German Credit</td>
<td>93.60</td>
<td>100.00</td>
<td>75.59</td>
<td><strong>30.40</strong></td>
</tr>
<tr>
<td>Balance Scale</td>
<td>56.67</td>
<td>99.68</td>
<td>36.63</td>
<td><strong>11.50</strong></td>
</tr>
<tr>
<td>Ionosphere</td>
<td>79.49</td>
<td>100.00</td>
<td>43.44</td>
<td><strong>20.20</strong></td>
</tr>
</tbody>
</table>

5. Conclusions and Future Avenues of Research

A sparse approximation algorithm is developed which constructs the approximation in a greedy fashion. The main advantage of this algorithm over any other routine used for classification purposes is that it determines all parameters during learning and does not involve any user interaction. It assures the user that it is the optimum performance given by the algorithm, while any other algorithm cannot give this assurance to the user. Experimental tests were conducted over a number of real world application data sets and its performance was compared to other standard classification routines. Experiments on both artificial and real data sets show that SFA gave sparser approximations with comparable prediction accuracy.

Best possible approximation to a problem can be constructed by solving a three dimensional optimization problem for the coefficients $(c_n, \sigma_n)$, the width parameter $(\sigma_n)$ and the center of the basis function $(x_n^*)$ at each stage. Incorporating the method of weighted residuals we calculated the optimum value of the coefficients which leaves us with a two-dimensional optimization problem for the center and the widths at each stage. In this algorithm we used a heuristic given in Section 3.1 to locate the center of the radial basis function at each stage instead of optimizing it to reduce computational time and expense. Then
we are left with a one-dimensional optimization problem for the width of the basis function which we solve at each stage in the algorithm. This algorithm gives a suboptimal rate of convergence for the approximation error. Research is in progress to mathematically formulate the dependence of the residual as a function of the number of basis functions. We believe that this formulation would result in the optimal value of the width parameter at each stage which could then be used to determine the optimum center of the radial basis function at each stage. An algorithm like this would reduce the number of optimizations to three to four thus accelerating the computational time taken.

Research is in progress to formulate the estimation error of the algorithm. Having obtained convergence rates for approximation and estimation error we can bound the generalization error of the algorithm using radial basis function. This research is being done for radial basis functions but in general SFA could also be used with other standard basis functions like the sigmoid and the hyperbolic tangent.

After obtaining generalization error bounds of SFA, it is envisioned as a tool to work with the experimentalists to better understand the design and working of a system. SFA can not only be used as a pattern recognition tool given a data set but also help the experimentalists accelerate through the test matrix by estimating how many more number of experiments need to be conducted for a user defined accuracy level. Being greedy in nature, ideas of active learning [Niyogi (1995)] can be incorporated with filter or wrapper feature selection methods to better design of the test matrix.

Acknowledgements

Support for this work was provided by the NASA Ames Research grant NCC-2-8077 and NASA Cooperative Agreement No. NCC-1-02038.

References


Sequential Function Approximation


