Technical Report No. TR-132

Fast Binary and Multi-Output Reduced Set Selection

Jason Weston
NEC Labs America
Princeton NJ, USA
`jasonw@nec-labs.com`
& **GökhanH.Bakır**
Max Planck Institute for Biological Cybernetics,
Tübingen, Germany
`gb@tuebingen.mpg.de`

November 2004

Fast Binary and Multi-Output Reduced Set Selection

Jason Weston & Gökhan H. Bakır

Abstract. We propose fast algorithms for reducing the number of kernel evaluations in the testing phase for methods such as Support Vector Machines (SVM) and Ridge Regression (RR). For non-sparse methods such as RR this results in significantly improved prediction time. For binary SVMs, which are already sparse in their expansion, the pay off is mainly in the cases of noisy or large-scale problems. However, we then further develop our method for multi-class problems where, after choosing the expansion to find vectors which describe all the hyperplanes jointly, we again achieve significant gains.

TR-132November 29, 2004

1 Introduction

Kernel algorithms like Support Vector Machines (SVMs) [1] and Ridge Regression (RR) [2] are popular tools for classification and regression. The success of kernel algorithms is based on the representer theorem [3, 4], which states that, under some mild conditions, the solution of an optimization problem of the form[1] :

$$w^* = \arg\min_w \sum_{i=1}^{N} \ell(y_i, x_i, w) + \Omega[f], \tag{1}$$

can be expressed as linear combination of training points, i.e

$$w^{*\top}\phi(x) = w^{*\top}\mathbf{x} = \sum_{i=1}^{N} \alpha_i k(x_i, x). \tag{2}$$

The representer theorem allows one to solve the *functional* minimization in (1) as a standard *function* minimization problem with $\alpha \in \mathbb{R}^N$ as the variable. The representation in (2) is called the *dual* with $\mathbb{R}^N$ being the dual space.

In general, the set of points $\{x_i\}_{i=1}^N$ do not necessarily constitute a linearly independent family. This is always the case for growing N when k is a non-universal kernel [5]. However in practice, universal kernels like the Gaussian kernel lead also to almost linearly dependent feature vectors due to decaying eigenvalues and finite precision effects.

Due to this redundance, the same point w in version space $w \in \mathcal{V}$ can be expressed by multiple linear combinations of x_i. Note that this is in contrast to the convexity property of the involved optimization problem in (1), e.g. in SVMs the solution w is unique, but may have multiple representations in the dual space.

Of great practical interest is the sparsity property of the dual representation (2) of the final predictor w, which determines the number of necessary kernel compuations and thus prediction speed. For this reason, reduced set techniques try to reduce the amount of points used in the dual representation to speed up prediction speed.

[1]In this paper, we denote by Ω an arbitrary positive regularization functional (for example $||w||^2$), by $D_N := \{x_i, y_i\}_{i=1}^N \subset \{\mathcal{X} \times \mathcal{Y}\}^N$ a training set consisting of N points, by ℓ the chosen loss function and by k and ϕ the kernel function and corresponding feature mapping. The variable $\alpha \in \mathbb{R}^N$ denotes expansion coefficients of the original problem formulation. For notation convenience we will use $\mathbf{x}$ where we denote the mapped point $\phi(x)$ in feature space.

In this paper we explore reduced set selection strategies to compress the linear expansion[2] $w = \sum_{i=1}^{N_1} \alpha_i \mathbf{x}_i$ to $\hat{w} = \sum_{j=1}^{N_2} \beta_j \mathbf{x}_j$ where the ultimate goal is

$$N_2 \ll N_1 \quad \text{and} \quad w = \hat{w}(\beta).$$

However the goal can be relaxed by dropping the equality constraint and just demanding

$$N_2 \ll N_1 \quad \text{and} \quad \min_{\beta} ||w - \hat{w}(\beta)||^2.$$

Essentially, such methods become useful in one of three scenarios: (i) if the predictor one wishes to compress is already sparse, but the basis it chooses still has linearly dependent or *close to* linearly dependent examples, this is true for SVMs in the presence of *noise* [5], (ii) if the predictor was not sparse in the first place, e.g. with methods such as Ridge Regression [2] and (iii) when one wishes to construct a multi-output predictor.

In the following sections, we review existing techniques and then propose new methods for the case of classification $\mathcal{Y} = \{+1, -1\}$ and regression $\mathcal{Y} = \mathbb{R}$. The reduced set selection methods we propose generalize to the d-output case $\mathcal{Y} = \{1, \ldots, d\}$ and $\mathcal{Y} = \mathbb{R}^d$ with $d > 1$. Compared to existing methods, the methods proposed are relatively faster to compute, yield similar or higher compression rates and are especially efficient in the multi-output case. Finally we validate the introduced algorithms experimentally on several classification and regression benchmarks, highlighting when reduced set selection proves useful, and then conclude.

2 Existing reduced set techniques

Approximations based on rank deficiency It can be shown that linear dependence between $D_N = \{x_i\}_{i=1}^N$ results in a rank deficient gram matrix.

Theorem 2.1. *The maximum linear independent subset $E_{N_2} \subset D_N$ constitutes of rank K points, where K is the matrix of inner products $K_{ij} = x_i^\top x_j$.*

Proof. We give a constructive proof. Write K in its spectral form $K = \sum_{i=1} \lambda_{i=1}^N v_i v_i^\top$. Obviously rank K equals the number of nonzero eigenvalues λ. Furthermore one can show that the eigenvector v_i of K encodes the eigenvector z_i of the empirical correlation matrix C with $C_{ij} = \mathbb{E}[x_i]\mathbb{E}[x_i]^\top$ by $z_i = \sum_{j=1}^N v_{ij} x_j$. Therefore, an eigenvalue larger then zero means that there must be at least one point $x \in D_N$ with $x^\top z_i > 0$. For all nonzero eigenvalues λ_i select one point x with $x^\top z_i > 0 = \sum_{j=1}^N v_{ij} x^\top x_i > 0$. We can only select rank K points and since $z_i \top z_j = 0$ for all $i \neq j$ we have selected r linear independent points. Since the remaining eigenvalues are zero, any further point must be in the span of the selected points since its in the span of the eigenvectors z_i. $\square$

This property was used in Downs et al [7] (see also [8]) to select a subset of size rank K. The subset was selected by computing the row echelon form of K and discarding points which lead to zero rows. Once the subset $E_{N_2} = \{x_{I(i)}\}_{i=1}^{N_2} \in D_N$ is chosen, the expansion coefficients β can be calculated by

$$\beta = K_{E_{N_2}}^{-1} K_{E_{N_2}, D_N} \alpha, \tag{3}$$

with

$$K_{E_{N_2}} = (K_{ij} = k(x_{I(i)}, x_{I(j)})),$$

and

$$K_{E_{N_2}, D_N} = (K_{ij} = k(x_{I(i)}, x_j)).$$

The problem with rank based techniques is their computational cost. The rank is very expensive to calculate for bigger samples since computation complexity is $O(N^3)$ (row echelon form, eigenvector decomposition, etc.). Furthermore, the compression rates may not be as one would like as it is lossless, e.g. for RBF kernels which give kernel matrices of full rank [9] no compression is possible[3].

[2] A related, but more difficult task is that of reduced set *construction*. In that task one searches for weights β_i and vectors $\mathbf{z}_i$, where $\mathbf{z}_i$ are not necessarily in the training set. This results in algorithms which give higher compression rates, but are much slower and more difficult to optimize, see e.g. [6].

[3] Apart from finite precision effects on computers. This may explain why Downs et al. [7] reported a reduction in SVs for RBF kernels.

Using additional penalizers In contrast, the ℓ_1 penalization method suggested in [10, sect. 18.4.2] simply attempts to construct a sufficiently good approximation of w by solving

$$\arg\min_{\beta} \left\| w - \sum_i \beta_i \mathbf{x}_i \right\|^2 + \lambda \sum_i c_i |\beta_i| \tag{4}$$

where parameter λ trades accuracy versus sparsity[4]. Simplifying expression (4) yields a numerically tractable quadratic programming problem. This formulation, although elegant also has the disadvantage to be a quadratic programming problem with cubic complexity in the number of patterns, i.e. $O(N^3)$. The weights c_i specify a prior on the importance of reducing the weight of example i, one could simply choose $c_i = 1$, $i = 1, \ldots, N$. In [8] it is suggested to take $c_i = 1/\alpha_i$ in order put more emphasis on reducing smaller weights. In that case, one can simply ignore the points with $\alpha_i = 0$ which for the SVM method gives a complexity cubic in the number of support vectors, i.e. $O(\#SV^3)$. In practice, however, it is difficult to adjust λ to control the approximation error $\|w - \sum_i \beta_i \mathbf{x}_i\|^2$.

The same concept was used for a multiple output/class scenario which leads to an algorithm with complexity $O((N + 2Nq)^3)$ where N is number of examples and q is number of outputs:

$$\arg\min_{\beta} \sum_j \left\| w^j - \sum_i \beta_i^j \mathbf{x}_i \right\|^2 + \lambda \sum_i c_i \xi_i \tag{5}$$

subject to

$$|\beta_i^j| \le \xi_i, \quad \xi_i \ge 0.$$

However, this is clearly prohibitive when either q or N are large.

3 Fast Reduced Set Selection

In this section we detail two novel algorithms for achieving faster reduced set selection: hyperplane matching pursuit, and minimization of the so called zero-norm. In particular, we then focus on generalizations of these methods that are especially efficient in the multiple-output case.

3.1 Hyperplane Matching Pursuit

We propose a greedy forward selection approach to choosing a reduced set approximation of w in the spirit of matching pursuit. The matching pursuit approach has been used before in the machine learning community to greedily train kernel classifiers [11] and to greedily approximate kernel matrices [12]. The general approach works by adding one training example on each iteration, usually by choosing the optimal improvement to an error measure, e.g. squared loss or matrix norm in the previous two examples. In the following we simply adapt the same approach to the pursuit of ”w”, minimizing the norm of $||w - \hat{w}||^2$ where $\hat{w}$ is our approximation.

Algorithm 3.2 (Matching pursuit on w). *Given stopping criteria ϵ, a kernel matrix K, $w = \sum_i \alpha_i \mathbf{x}_i$ and training points $D_N = \{X, Y\}_{i=1}^N$*

```
Initialization
1. i=1, $w^1 = w, \alpha^1 = \alpha$
While $||w^i||^2 = {\alpha^i}^\top K \alpha^i > \epsilon$
2. Let $k^\star = \arg\min_{k, a \in \mathbb{R}} ||w^i - a\mathbf{x}_k||^2 = \arg\max_k |\frac{{w^i}^\top \mathbf{x}_k}{||\mathbf{x}_k||}| = \arg\max_k |\frac{K_{:,k}\alpha^i}{\sqrt{K_{kk}}}|$
3. $a^* = \arg\min_{a \in \mathbb{R}} ||w^i - a\mathbf{x}_{k^*}||^2 = {w^i}^\top \frac{\mathbf{x}_{k^*}}{||\mathbf{x}_{k^*}||^2} = \frac{K_{:,k^*}\alpha^i}{K_{k^*k^*}}$.
```

[4]When the original classifier also has a threshold (constant term), the author's suggest to optimize this by choosing the threshold with minimal training error.

4. $w^{i+1} = w^i - a^* \mathbf{x}_{k^*}, \;\; \alpha_{k^*}^{i+1} = \alpha_{k^*}^i - a^*$

5. $i = i + 1$.

Final approximation of w is given by $\hat{w} = \sum_j (\alpha_j^1 - \alpha_j^i)\mathbf{x}_j$ which is sparse.

We have written each step of the algorithm in both the primal and dual forms. In step (2) the example is added to the reduced set which results in the largest decrease in $w^i = ||w - \hat{w}||^2$ in a greedy forward selection manner. In step (3) the resulting multiplier α is set. This process can be repeated until either one achieves a certain tolerance $||w - \hat{w}||^2 < \epsilon$ or alternatively one could measure error rate (which is what one is really interested in) on a validation set.

We now discuss possible improvements to the basic algorithm.

Backfitting The decrease w_i at each step can be improved considerably by optimally adjusting all α_i in the currently selected examples, rather than just the new incoming example. This approach is called backfitting, which is a standard trick in adaptive filtering [13] and was also used for kernel matching pursuit [11]. To do this one solves

$$\min_\beta || \sum_i \alpha_i \mathbf{x}_i - \sum_i \beta_i \mathbf{x}_{s_i} ||^2 \tag{6}$$

where β are the new expansion coefficients for the so far chosen examples indexed by s. This results in:

$$\beta = K_s^{-1} K_{s,:} \alpha \tag{7}$$

where K_s is the kernel matrix between examples indexed by s only, and $K_{s,:}$ is the kernel matrix between s and all other examples. This clearly has a higher computational cost per iteration than step (3) of the basic algorithm. Backfitting costs $O(|s|^3)$ for a single matrix inversion. However, even in this case it is more efficient than ℓ_1-minimization which has complexity $O(N^3)$ as $|s| < N$.

Efficient Backfitting Since the expansion is increasing, the main cost of backfitting is calculation of the inverse. One can update the inverse K_s^{-1} since the change in the dictionary is the addition of one new point $\mathbf{x}_l$ to the existing basis set. For this reason we can use the old inverse $K_{s\setminus l}^{-1}$ to calculate K_s. This can be implemented by the Cholesky factorization of terms appearing in the matrix inversion lemma. If $K_s = \begin{pmatrix} K_{s\setminus l} & \mathbf{k}_l \\ \mathbf{k}_l^\top & k_l l \end{pmatrix}$, then

$$\begin{aligned} K_s^{-1} &= \begin{pmatrix} K_{s\setminus l} + & \mathbf{k}_l \\ \mathbf{k}_l^\top & k_{ll} \end{pmatrix}^{-1} & (8) \\ &= \begin{pmatrix} K_{s\setminus l}^{-1} + \lambda K_{s\setminus l}^{-1} \mathbf{k}_l \mathbf{k}_l^\top K_{s\setminus l}^{-1\top} & \lambda K_{s\setminus l}^{-1} \mathbf{k}_l \\ \lambda \mathbf{k}_l^\top K_{s\setminus l}^{-1} & \lambda \end{pmatrix}, & (9) \end{aligned}$$

with $\lambda = \frac{1}{k_l l - \mathbf{k}_l^\top K_{s\setminus l}^{-1} \mathbf{k}_l}$. This can be further optimized by performing a Cholesky factorization of $K_s^{-1} = R_s^\top R_s$, which leads to the final recursion equation

$$R_s = \begin{pmatrix} R_{s\setminus l} & \mathbf{0} \\ -\sqrt{\lambda} K_s^{-1} \mathbf{k}_l & \sqrt{\lambda} \end{pmatrix}. \tag{10}$$

Probabilistic Search A further speed-up is possible using the "59-trick" of [12]. This suggests that by solving around 60 one dimensional minimization problems for randomly chosen examples, examples can be chosen which are probabilistically in the top 95th percentile. We could thus limit step (2) of our basic Hyperplane Matching Pursuit Algorithm to searching only a small random subset of the training examples.

Optimizing the real-valued output We currently minimize $||w - \hat{w}||^2$ however we are not so much interested in this difference as the difference in prediction of the two rules. From the Cauchy-Schwarz equation the latter is only an upper bound to the generalization error, namely:

$$e_x = \mathbb{E}_x ||w^\top \mathbf{x} - \hat{w}^\top \mathbf{x}||^2 \leq \sup_{\mathbf{x}} ||\mathbf{x}||^2 ||w - \hat{w}||^2.$$

While we try to minimize the right hand side, we could try to minimize the left hand side by the empirical estimate of e_x, namely $\hat{e}_x = \frac{1}{N}\sum_{i=1}^{N} ||w^\top \mathbf{x}_i - \hat{w}^\top \mathbf{x}_i||^2$ which is the squared difference of the predictions on the training set. Interestingly, the β which minimize (6) are equivalent to the β which minimize:

$$\frac{1}{|s|}\sum_{i=1}^{|s|} ||w^\top \mathbf{x}_{s_i} - \hat{w}^\top \mathbf{x}_{s_i}||^2 \tag{11}$$

where s are the indices of the currently selected examples, as the minimizer of (11) is also (7). This shows that the $||w - \hat{w}||^2$ minimizer is suboptimal in terms of prediction as it only minimizes the difference in prediction on a *subset* of the examples.

3.2.1 Multiple-Hyperplane Matching Pursuit

In multi-class classification one typically finds a solution which is a combination of binary classifiers, e.g. in the one-vs-the-rest or one-against-one approaches [14]. Such a setting is also applicable to multiple output regression. For example in one-vs-the-rest classification one trains p classifiers for p classes, where the j^{th} hyperplane is learnt with examples labeled positively if they are in class j and negative otherwise. This gives a final classifier:

$$f(x) = \arg\max_j (w_j^\top \mathbf{x}_i)$$

where $w^j = \sum_{j=1}^{p} \alpha_i^j \mathbf{x}_i$. To use reduced set methods, we could compress each hyperplane *independently*. However, the main computational cost we are trying to reduce is in the calculation of kernel functions, and so we would rather minimize the union of expansion vectors $|\{i : \sum_{j=1}^{p} |\alpha_i^j| > 0\}|$ than the sum: $\sum_{j=1}^{p} |\{i : |\alpha_i^j| > 0\}|$.

We thus wish to couple the compression steps to compress the hyperplanes all at once. We can do this in the matching pursuit framework by choosing the next $\mathbf{x}_k$ to minimize

$$\arg\min_k \min_a \sum ||w_j - a_j \mathbf{x}_k||^2$$

For a given $\mathbf{x}$, the optimal a_j for the above equation are given by:

$$a_j = \frac{w_j^\top \mathbf{x}}{||\mathbf{x}||} \tag{12}$$

which gives the optimal $\mathbf{x}$ as

$$\arg\min_{\mathbf{x}_k, k=1,\ldots,m} \sum_j \left(||w_j||^2 - \frac{(w_j^\top \mathbf{x}_k)^2}{||\mathbf{x}_k||^2} \right) \tag{13}$$

To perform Multiple Hyperplane Pursuit, we thus replace step (2) and (3) in Algorithm 3.2 with equations (13) and (12).

3.3 ℓ_0-norm Reduced Set Selection

Our central goal is to approximate the vector α with a new sparse vector β, ideally minimizing the number of nonzero coefficients of β, denoted as $||\beta||_0$, i.e. we would like:

$$\min_\beta ||\beta||_0$$

subject to

$$||\sum_i \alpha_i \mathbf{x}_i - \sum_i \beta_i \mathbf{x}_i||^2 = 0 \tag{14}$$

Use of the so-called zero-norm (which isn't really a norm) has been researched before in the field of feature selection [15, 16]. In [15] it was shown how the above problem is related to the following problem:

$$\min_\beta \sum_i \ln(\epsilon + |\beta_i|)$$

subject to (14). This can be solved by gradient descent, resulting in the following iterative procedure:

Algorithm 3.4 (ℓ_0-minimization). *Given a kernel matrix K, $w = \sum_i \alpha_i \mathbf{x}_i$ and training points $D_N = \{X, Y\}_{i=1}^N$*

1. *Set $\beta_i = 1$.*
2. $$\lambda^* = \arg\min_{\lambda} || \sum_i \alpha_i \mathbf{x}_i - \sum_i \lambda_i \beta_i \mathbf{x}_i ||^2$$
   which gives
   $$\lambda^* = (K^1)^{-1} K^2 \alpha$$
   where $K^1_{ij} = \beta_i(\mathbf{x}_i, \mathbf{x}_j)$ and $K^2_{ij} = \beta_i \beta_j (\mathbf{x}_i, \mathbf{x}_j)$.
3. *Rescale data: $\beta_i \leftarrow \lambda_i^* \beta_i$*
4. *Go back to 2, until convergence.*

This procedure typically converges in 10-20 steps. So far this minimization will give lossless compression. To extend to lossy compression, we simply use the same trick as in the ℓ_1-minimization, and replace step (2) with:

$$\arg\min_{\lambda} || \sum_i \alpha_i \mathbf{x}_i - \sum_i \lambda_i \beta_i \mathbf{x}_i ||^2 + \gamma \sum_i \lambda_i \beta_i$$

This gives a fixed deviation that is minimized across iterations. Like in ℓ_1-minimization, one is required to set the parameter γ a priori.

The computational complexity of this optimization is higher than ℓ_1-minimization because one is required to solve multiple iterations of compression, rather than just one. However, on each successive iteration, there are less variables as one can remove the variables which have already obtained $\beta_i = 0$. Moreover, when we come to solve multiple-output compression we will see that compared to ℓ_1-minimization via equation (5) the complexity is greatly reduced.

3.4.1 Multi-output ℓ_0-minimization

To compress multiple hyperplanes w_p we generalize the previous algorithm. We simply learn β^j for each hyperplane j, where we wish to minimize $||(\sum_j |\beta^j|)||_0$. To do this one replaces step (3) with $\beta_i^j \leftarrow (\sum_j |\lambda_i^j|)\beta_i^j$. That is, we perform the following algorithm.

Algorithm 3.5 (Multi-output ℓ_0-minimization). *Given a kernel matrix K, p hyperplanes $w^p = \sum_i \alpha_i^p \mathbf{x}_i$ and training points $D_N = \{X, Y\}_{i=1}^N$*

1. *Set $\beta_i^j = 1, \;\; j = 1, \ldots, p$.*
2. $$\arg\min_{\lambda^j} || \sum_i \alpha_i^j \mathbf{x}_i - \sum_i \lambda_i^j \beta_i^j \mathbf{x}_i ||^2, \;\; j = 1, \ldots, p$$
   which gives
   $$\lambda^j = (K^{j1})^{-1} K^{j2} \alpha^j, \;\; j = 1, \ldots, p$$
   where $K^{j1}_{pq} = \beta^j_p(\mathbf{x}_p, \mathbf{x}_q)$ and $K^{j2}_{pq} = \beta^j_p \beta^j_q (\mathbf{x}_p, \mathbf{x}_q)$.
3. *Rescale data: $\beta_i^j \leftarrow (\sum_j |\lambda_i^j|)\beta_i^j$*
4. *Go back to 2, until convergence.*

Again, one can then easily extend this to the lossy compression case.

Name	Inputs	Outputs	Train	Test
German	20	1	700	300
Waveform	21	1	1000	4000
Banana	2	1	1000	4300
Image	18	1	2000	300
USPS	256	10	7329	2000
Letter	17	26	16,000	4000
Abalone	8	1	3133	1044
Kin-32nm	32	1	3000	5192

Table 1: **Datasets used in the experiments.** All the datasets are classification problems, apart from Abalone and Kin-32nm which are treated as regression problems.

3.6 Chunking Method: handling large datasets

In this subsection, we explore a chunk-based reduced set selection strategy. The linear expansion of the solution can be grouped without changing the result, namely:

$$\begin{aligned} w &= \sum_{i=1}^{N} \alpha_i \mathbf{x}_i \\ &= \sum_{i=1}^{N_1} \alpha_i \mathbf{x}_i + \sum_{i=N_1+1}^{N_2} \alpha_i \mathbf{x}_i \cdots \sum_{i=N_{n-1}+1}^{N} \alpha_i \mathbf{x}_i, \\ &= \sum_{j=1}^{n} w_j, \end{aligned}$$

with $w_j = \sum_{i=N_{j-1}}^{N_j} \alpha_i \mathbf{x}_i$. The idea is to apply any of the previously described techniques (ℓ_1, ℓ_0 or Matching Pursuit) to a smaller problem, keeping only a subset of the variables active that we wish to compress, and leaving the rest fixed. We then move the active set to another subset, and iterate through the subsets. This means we never face an optimization problem larger than a fixed size, of our choice. (However, if the chunk size is set too small compression will not be easy to perform as linearly or nearly linearly independent points may not exist in the active set.) Essentially, the basic Hyperplane Matching Pursuit algorithm 3.2, where no backfitting is performed, is an instantiation of this with a chunk size of 1.

4 Experiments

We conducted experiments on four types of data: (i) artificial data, (ii) two-class classification problems, (iii) multi-class classification problems and (iv) regression problems. The summary of the datasets we use, apart from the artificial one of Section 4.1, are given in Table 1. For each dataset we describe the machine learning algorithm that is used for learning, along with its parameters, in Table 2. These are the models we would like to perform reduced set selection on. The parameters selected for German, Image, Waveform and Banana come from [17]. For USPS and Letter we tried to choose parameters which matched the test error quoted elsewhere [1, 18]. For the regression datasets, Abalone and Kin-32nm, we tried to pick the best possible parameters on the test set. Apart from the first toy dataset, in all cases we used RBF kernels. Although attempting reduced set selection on kernels that are not full rank such as polynomial kernels can result in compression with exactly the same decision rule, we decided to stick to the more difficult case of approximating a kernel with full rank.

The compared methods are all implemented in Matlab. The source code and datasets are available at `http://www.kyb.tuebingen.mpg.de/bs/people/weston/rss`. In our experiments with hyperplane matching pursuit, we perform backfitting, and optimize the criterion $||w - \hat{w}||^2$. For all methods we also optimize the threshold b for the classification tasks by minimizing the training error.

Name	Algorithm	SVs	Test Err
German	SVM ($\sigma = 5.24, C = 3.16$)	400	0.2570
Waveform	SVM ($\sigma = 3.16, C = 1$)	331	0.0917
Banana	SVM ($\sigma = 0.7, C = 316$)	220	0.1016
Image	SVM ($\sigma = 3.9, C = 500$)	216	0.0281
USPS	SVM ($\sigma = 128, C = 1000$)	1526	0.0416
Letter	SVM ($\sigma = 4, C = 1000$)	4522	0.0373
Abalone	RR ($\sigma = 15, \gamma = 1e - 5$)	3133	0.410
Kin-32nm	RR ($\sigma = 20, \gamma = 0.005$)	3000	0.6187

Table 2: **Predictors used in the experiments**. Included are the parameters of the initial trained predictors we are trying to compress. The error rates for the first four datasets are averaged over ten splits. The last two datasets, Abalone and Kin-32nm, have error rate measured using mean squared error (after normalizing the outputs).

Figure 1: **Reduced Set Selection on Artificial data.** The number of SVs increases linearly for SVM due to *noise points* becoming SVs, but is constant for RSS methods such as ℓ_0-minimization.

4.1 Artificial Problems

We first constructed artificial data, by generating two classes from two Gaussian clouds in 10 dimensions with means $(1, 1, 1, 1, 1, 0, 0, 0, 0, 0)$ and $(-1, -1, -1, -1, -1, 0, 0, 0, 0, 0)$ and standard deviation 4 following [19]. We trained a linear SVM for differing amounts of training points. This is an unrealistic case, as one can represent the data in primal form by calculating w in this case, nevertheless it serves to show that SVM do not optimally compress in dual space. In this case, any of the reduced set selection methods (with appropriate hyperparameter choice) will choose 10 SVs with no loss in accuracy. The results are given in Figure 1 using the ℓ_0-minimization method with $\lambda = 0$. Note the linear increase in number of SVs for SVM compared to the fixed number of SVs independent of training set size for reduced set methods. This is due to the face that all mislabeled points become SVs in SVM, and for large datasets this has a signficant effect on computation time. Indeed the fraction of SVs in SVM is lower bounded by the number of training errors, and hence assymptotically by the Bayes error, as pointed out in [5].

4.2 Two-class Classification

We took four different datasets and trained SVMs with the parameter choices quoted in [17]. We then compared the error rate to compression ratio of the competing methods: ℓ_1- minimization, ℓ_0- minimization and hyperplane matching pursuit. For ℓ_1- and ℓ_0- minimization we chose an array of parameter choices for λ, and then computed the error rate for hyperplane matching pursuit with the same number of obtained SVs. We averaged the results over ten splits, linearly interpolating between points as the parameter λ does not give the same number of SVs each time. The results, reported in Table 2 indicate similar performance between the methods, although hyperplane matching pursuit yields slightly lower test error for higher compression rates.

The time given for ten parameter choices for each of the methods is given in Table 3. We show this for ten parameter choices because to find a good loss to compression ratio a hyperparameter search is necessary. For example, one can evaluate accuracy using a validation set. As hyperplane matchng pursuit essentially gives all parameter choices at no extra cost through a single run of greedy minimization, it yields much faster execution times. Moreover, the hyperparameter λ in the other two methods is difficult to control - it is difficult to know which value yields which number of SVs. We also analyzed the training error and value of the objective function for all three algorithms, this can be seen for the Waveform dataset in Table 3. We found the objective function and training error were lower for hyperplane matching pursuit particularly in the case of high compression rates, mirroring its test error performance.

german image

waveform banana

Figure 2: **Comparison of Reduced Set Selection Methods for Two-Class Classification** The three reduced set selection methods perform similarly, but the Hyperplane Matching Pursuit method yields lower test error for high compression rates on Waveform and Image.

Figure 3: **Training Error and Objective Function Value on the Waveform Dataset** The training error (left) and value of the objective function $||w - \hat{w}||^2$ (right) are both lower for the Hyperplane Matching Pursuit method relative to the other two methods.

Note that it is easier to compress the expansions on the problems with higher error rates (Image, which has the smallest test error, is the hardest to compress.) This corroborates the results found with artificial data in Section 4.1.

Table 3: **Calculation Time of Reduced Set Selection Methods for Two-Class Classification for ten parameter choices.**

Dataset	German	Image	Waveform	Banana
ℓ_1-min	613 secs	152 secs	290 secs	136 secs
ℓ_0-min	645 secs	260 secs	250 secs	131 secs
M-Pursuit	21 secs	16 secs	10 secs	2 secs

4.3 Multi-Class Classification

We then performed reduced set selection on multi-class problems. In this setting, one might expect to achieve a greater gain if the reduced set selection is coupled across the different classifiers. We tested the compression of SVM solutions using the one-against-the-rest method [14] on two datasets: USPS Digit Recognition and the Letter database, comprising of letter recognition - there are 10 and 26 classes respectively. We performed ℓ_1-minimization on SVM solutions for a range of different values of λ (the compression parameter.) The results are given in Figure 4. We performed Matching Pursuit using the same number of SVs for each hyperplane as the ℓ_1 method to enable direct comparison. Again, although SVs produce sparse solutions, these can be compressed considerably by methods such as ℓ_1-minimization or the Matching Pursuit method. The latter again outperforms the former for higher compression rates. Finally, we then evaluated the "coupled" Multi-hyperplane Matching Pursuit method which finds SVs which are relevant to all hyperplanes (for each class) at once. The compression rate is significantly improved. We did not compare with the algorithm given in (4) because it was too slow to compute, however in [8] the authors do report a single run of this system with an error rate of 5.5% for 570 SVs which they compare to 10.8% using the ℓ_1-minimization of equation (5) approach which does not take into account the multi-class problem.

We can expect these results to carry over to other similar settings, such as regression with multiple outputs. Moreover, we can expect larger gains on other multi-class methods such as one-vs-one or error correcting codes [14] which use more support vectors.

4.4 Regression

We performed experiments on two datasets in a regression setting. We attempted to compress the solution of Ridge Regression in dual variables [2] which does not give sparse solutions, hence the potential gain here should be much greater. We found that was indeed the case. We normalized both input and outputs to have mean zero and standard deviation one, and quote the mean squared error on the normalized outputs. Figure 5 shows the error rate for differing compression rates using Hyperplane Matching Pursuit as the RSS method, using the probabilistic search method to further speed up computation. The results show that compressing RR solutions can give significant postal letter

Figure 4: **Comparison of Reduced Set Selection for Multi-Class Classification.** The Hyperplane Matching Pursuit Algorithm designed for Multi-Class problems reduced the number of kernel computations required on the USPS Postal Database (left) and UCI Letter Database (right) compared to binary classification-based compression.

Abalone Kin-32nm

Figure 5: **Comparison of Reduced Set Selection for Regression.** The Hyperplane Matching Pursuit Algorithm reduced the number of kernel computations required on the Abalone (left) and Kin-32nm (right) problems compared to standard Ridge Regression, which is not sparse.

efficiency gains, and we expect this result to hold for other non-sparse predictors as well, such as the Nadaraya-Watson Estimator or Parzen's Windows [14], to name two.

5 Discussion

Reduced set techniques fall into two categories: reduced set selection and reduced set construction. Reduced set construction tries to reduce the amount of kernel points, by constructing a new set of points which are not necessarily subset of the training data. In contrast, reduced set selection methods try to reduce the amount of points by *selecting* a linear independent subset of the training data. Often the compression factor in reduced set selection is worse than in reduced set construction but in general reduced set selection methods are more efficient and numerically stable than reduced set selection techniques.

We have reviewed and proposed new optimization strategies for reduced set selection. Our initial goal was to find more efficient techniques of achieving compression, and while both Hyperplane Matching Pursuit and ℓ_0-minimization present viable novel alternatives we prefer the former for (1) its basic computational properties, (2) the ability to compute every compression level in one training run rather than relying on retraining with new parameter choices, and (3) its good performance with high compression rates. However, the main finding of this work is that in the multi-class case, significant gains can be made in terms of compression by coupling the compression of all trained hyperplanes at once. This led to high compression levels in the USPS and Letter datasets. We expect these results to also carry over to other predictors with multiple hyperplanes, such as the multi-output regression case (some preliminary experiments on artificial data, not shown, confirmed this). This result suggests one could gain significant speedups in the training phase also by performing such coupling, if one could design an algorithm which does that. The most straight-forward way of achieving such a goal would be to adapt the kernel matching pursuit algorithm for direct multi-class classification. This is the subject of our future research.

Acknowledgements

We would like to thank Leon Bottou and Vladimir Vapnik for useful discussions. Part of this work was funded by NSF grant CCR-0325463.

References

[1] V. N. Vapnik. *Statistical Learning Theory*. Springer, 1998.

[2] Craig Saunders, Alexander Gammerman, and Volodya Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 515–521. Morgan Kaufmann Publishers Inc., 1998.

[3] G. S. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33:82–95, 1971.

[4] B. Schölkopf, R. Herbrich, A. J. Smola, and R. C. Williamson. A generalized representer theorem. Technical Report 2000-81, NeuroCOLT, 2000. Published in *Proceedings COLT'2001*, Springer Lecture Notes in Artificial Intelligence, 2001.

[5] Ingo Steinwart. On the generalization ability of support vector machines. *Journal of Machine Learning*, 2:67–93, December 2001.

[6] C. J. C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector learning machines. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 375–381, Cambridge, MA, 1997. MIT Press.

[7] T. Downs, K. E. Gates, and A. Masters. Exact simplification of support vector solutions. *Journal of Machine Learning Research*, 2:293–297, December 2001.

[8] B. Schölkopf, S. Mika, C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. J. Smola. Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, 1999.

[9] C. A. Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2:11–22, 1986.

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

[11] Pascal Vincent and Yoshua Bengio. Kernel matching pursuit. *Mach. Learn.*, 48(1-3):165–187, 2002.

[12] Alex J. Smola and Bernhard Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 911–918. Morgan Kaufmann Publishers Inc., 2000.

[13] Ali Sayed. *Fundamentals of Adaptive Filtering*. Wiley, 2003.

[14] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, New York, second edition, 2001.

[15] J. Weston, A. Elisseeff, and B. Schölkopf. Use of the ℓ_0-norm with linear models and kernel methods. Technical report, Biowulf Technologies, New York, 2001.

[16] Paul S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 82–90. Morgan Kaufmann Publishers Inc., 1998.

[17] G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001. Also: NeuroCOLT Technical Report 1998-021.

[18] C. Gentile. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242, 2001.

[19] G. Bakir, L. Bottou, and J. Weston. Breaking svm complexity with cross-training. In *Advances in Neural Information Processing Systems*, 2004.