Joint feature-basis subset selection

Shai Avidan
Mitsubishi Electric Research Labs
201 Broadway, 8th FL
Cambridge, MA 02139

Abstract

We treat feature selection and basis selection in a unified framework by introducing the masking matrix. If one considers feature selection as finding a binary mask vector that determines which features participate in the learning process, and similarly, basis selection as finding a binary mask vector that determines which basis vectors are needed for the learning process, then the masking matrix is, in particular, the outer product of the feature masking vector and the basis masking vector. This representation allows for a joint estimation of both features and basis. In addition, it allows one to select features that appear in only part of the basis functions. This joint selection of feature/basis subset is not possible when using feature selection and basis selection algorithms independently. thus, the masking matrix help extend feature and basis selection methods while blurring the lines between them. The problem of searching for a masking matrix is NP-hard and we offer a sub-optimal probabilistic method to find it. In particular we demonstrate our ideas on the problem of feature and basis selection for SVM classification and show results for the problem of image classification on faces and vehicles.

1. Introduction

Modern machine learning applications deal with large amounts of data. Tasks such as visual recognition, text classification, speech recognition or bio-informatics require the learning algorithm to deal with examples that contain from several hundreds to many thousands of feature per example, as well as thousands or tens of thousands of examples. As a result, methods that reduce the amount of data and focus on the important features and examples are constantly developed and used.

Feature selection and basis selection methods are used in machine learning for various reasons. They can help reduce the run-time complexity of classification tasks, discover the underlying structure of the data or improve the performance of the classifier, by removing irrelevant and redundant features. For example, feature selection was recently shown to be very effective for face detection [20], where the idea was to form a huge bank of possible features and use a feature selection algorithm to greedily select features that reject as much of the negative examples as possible. Feature selection from huge filter sets was also suggested by [1] that use a three-stage algorithm to remove irrelevant and redundant features before picking the informative features for image classification. In text classification, feature selection methods are used to dramatically reduce the vocabulary size while maintaining accurate classification [18].

In a similar vein, basis selection algorithms select a subset of basis vectors from a given set of basis vectors to infer a given learning task [14]. For instance, the run time complexity of SVM classifiers depend linearly on the number of support vectors, so decreasing the number of support vectors will reduce the run time complexity, as well as the memory requirements of SVM. This can be done through reduced set methods [4] that come in two flavors. One variant aim at selecting the "best" support vectors from the given set of support vectors while the other aim at constructing a totally new set of support vectors that approximate the original solution [17]. Viewed this way, one can think of the first variant of reduced set methods as a basis selection problem, in which a subset of the support vectors is to be chosen from a given set. Alternatively, one might reduce the number of features in the support vectors to accelerate the dot-product operation required by the SVM in run-time and improve the overall performance of SVM in the presence of irrelevant features [21].

Much of the work treat these two problems separately, i.e., either do basis selection or feature selection, even though both of them often employ very similar techniques. In this work we only consider binary selection methods, as our goal is to select the "best" basis vectors (i.e. support vectors) *and* the "best" features associated with them. It will therefor be convenient, for our purposes, to consider subset selection as the problem of finding a binary mask vector that determines which elements (either features or basis vectors) are selected and which are not.

Clearly one can do basis and feature selection in sequence. First, find a binary mask of the basis vectors, and given that mask find a binary mask for the features. This will not be optimal as changing the feature subset might al-

ter the selection of the basis vectors.

Therefor we introduce the *masking matrix* that can be formed by the outer product of the two binary masking vectors mentioned above. In this case, the masking matrix is a rank-1 binary matrix whose rows correspond to the features and its columns correspond to the basis vectors. Hence, a feature selection algorithm amounts to choosing a subset of the rows of the masking matrix, while a basis selection method corresponds to choosing a subset of the columns of the masking matrix. Finding both corresponds to choosing a subset of both rows and columns of the masking matrix. So, instead of finding the binary vector masks in sequence we can now opt to find both of them simultaneously.

In fact, we opt for more. Estimating the feature mask and the basis mask simultaneously implicitly implies that the chosen feature is used by *all* the selected basis vectors and vice-versa. We have no freedom to attach particular features to particular basis vectors.

Our solution is to extend the masking matrix from a rank-1 binary matrix to a general rank binary matrix. Such a matrix will let us specifically define which feature should be used by which basis vector. So in the spirit of feature or basis selection we are now interested in *element selection*, an algorithm that will choose particular elements of the masking matrix. This is a refinement of feature and basis selection algorithms that choose an entire row or column in the masking matrix, as figure 1 illustrates.

The problem of finding a binary mask vector that will correspond to the optimal subset (either feature subset or basis subset) requires exhaustive search over all possible subsets and is NP-hard [8]. Hence, many algorithms on subset selection focused on finding sub-optimal approximations to the problem.

We solve the element selection problem using a variant of the sequential forward selection algorithm, a standard technique in the subset selection literature [10], and to accelerate computations we use probabilistic sampling. The result of our algorithm is an extremely sparse masking matrix that combines feature and basis selection seamlessly. However, our definition of sparseness is a little different than the common one. Usually, in sparse coding one is interested in choosing a small number of basis vectors to "explain" the test pattern, but each basis vector is not sparse. We, on the other hand, might find a solution in which many basis vectors are used, but each of them is very sparse.

In this paper we focus on accelerating the run time performance of SVM classifiers, that are among the best performing classifiers available today for various applications [3, 12, 16]. For our purpose, the support vectors are the basis vectors from which we would like to select a subset, as well as choosing a feature subset. We show that the number of support vectors, as well as, the number of features can be reduced considerably with a just slight reduction in classification power.

There are two ways to use our method. In case our solution is good enough it might replace the original SVM classifier altogether. In case our solution is not accurate enough, our method can be used to construct very fast rejectors that will reject most of the patterns and leave only a small fraction to the full scale SVM classifier.

2 Background and notations

We give a brief description of feature selection and SVM methods. The interested reader is referred to [19, 5, 2] for further information.

We denote vectors using bold face fonts $\mathbf{x}$ to distinguish them from scalars x, matrices are denoted by bold capital letters $\mathbf{X}$. In the equations we assume the image data to be vectorized.

2.1 Feature Selection

Feature selection algorithms search for the optimal subset of features for the problem at hand. In classification problems it is assumed that a small number of features is sufficient to solve the problem. Moreover, because of the *curse of dimensionality* it was often observed that many irrelevant or redundant features might actually harm the performance of the classifier. Since the problem is combinatorial in the number of features, greedy sub-optimal solution are often used. The simplest ones involve forward or backward selection algorithms, in which at every step one feature is added (or removed) from the current feature subset [2]. Extensions include the floating search method that combines forward and backward selection [13], branch-and-bound [11] that exhaustively search all possible subsets but relies on the monotonicity assumption to quickly discard large portions of the search space.

Feature selection algorithms are usually categorized as either *filter* or *wrapper* methods [7]. Filter methods are a preprocessing step that is done before the induction process takes place. The feature subset is selected according to some heuristic criteria that is believed to be connected to the true objective function that the induction process tries to optimize. Wrapper methods, on the other hand, use the induction procedure as a subroutine and select features that directly affect it. Filter methods are faster to compute, but wrapper methods are considered more accurate.

2.2 Support Vector Machines

Given a data set $\{\mathbf{x_i}, y_i\}_{i=1}^{l}$ of l examples $\mathbf{x_i} \in \mathcal{R}^d$ with labels $y_i \in \{-1, +1\}$, SVM finds the separating hyperplane

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

(a) feature selection (b) basis selection (c) element selection

Figure 1: The Masking Matrix. The masking matrix is a binary matrix that unifies feature selection and basis selection methods. (a) In feature selection methods a subset of all the rows is chosen to have the value 1, the rest of the rows have the value 0. (b) In basis selection methods a subset of all column is chosen to have the value 1, the rest of the columns have the value 0. (c) In element selection a combination of features and basis elements is selected.

that maximizes the margin between the two classes. Classifying a test pattern $\mathbf{x}$ is given by

$$f(\mathbf{x}) = sgn(\sum_{i=1}^{n} \alpha_i k(\mathbf{x}, \mathbf{s_i}))$$

where $\mathbf{s_i}$ are the support vectors, α_i are their weights, and $k(.,.)$ is the kernel function. The support vectors are a subset of the original training data that are closest to the separating hyperplane.

The linear combination of support vectors implicitly represent the separating hyper-plane in some high dimensional feature space. If we can find a representation of this hyperplane with a smaller number of support vectors, then we can accelerate the run-time performance of SVM. Reduced set methods aim at reducing the run-time complexity of SVM by using a reduced set of support vectors. There are two methods for computing the reduced set [4]. The first involves *selecting* the most "important" support vectors from the given set of support vectors. The second method involves *constructing* a set of newly synthesized support vectors that will approximate the original set.

In the first case we are given a set of support vectors $\mathbf{s_i} \quad i = 1...N_x$, that we need to approximate with only a subset of it. Put formally, we wish to approximate

$$\Psi(\mathbf{x}) = \sum_{i=1}^{N_x} \alpha_i k(\mathbf{x}, \mathbf{s_i})$$

with

$$\Psi'(\mathbf{x}) = \sum_{j=1}^{N_z} \beta_j k(\mathbf{x}, \mathbf{s_j})$$

where $N_z < N_x$, by minimizing $||\Psi(\mathbf{x} - \Psi'(\mathbf{x})||^2$. Note that the weights in the reduced set are modified to account for the selection of a subset of support vectors.

Instead of choosing from the given set of support vectors, one might construct a new set of support vectors. Now the goal is to find a new set of support vectors $\mathbf{z}_i \in \mathcal{R}^d \quad i = 1...N_z$ and their weights $\beta_i \quad i = 1...N_z$

$$\Psi''(\mathbf{x}) = \sum_{j=1}^{N_z} \beta_j k(\mathbf{x}, \mathbf{z_j})$$

where $N_z < N_x$ and $||\Psi(\mathbf{x}) - \Psi''(\mathbf{x})||^2$ is minimized. This problem is solved by iterative non-linear optimization that adds new support vectors one by one.

The former of the two methods corresponds to the binary basis selection discussed here, but the latter is considered more accurate.

Romdhani *et al* [15] suggested the sequential SVM evaluation, where after evaluating each support vector a decision is made weather to accept, reject or keep on evaluating the SVM expansion.

3. The Masking Matrix

Assume that you are given a set of basis vectors that give the solution to some classification task and your goal is to find a subset of features *and* vectors that will approximate this solution. To put things in context, assume that a full SVM solution was obtained and we are now interested in selecting a subset of support vectors, as well as a subset of features, that will maintain, or even surpass the performance of the original SVM solution. To do this, one resorts to feature selection or basis selection algorithms that search for a binary mask vector that will determine which features or basis vectors should participate in the learning process. Put formally, in feature selection, we look for a binary vector $\mathbf{f} = (f_1, ..., f_d) \quad f_i \in \{0, 1\}$ s.t.

$$\Psi_{\mathbf{f}}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\pi(\mathbf{x}, \mathbf{f}), \pi(\mathbf{s_i}, \mathbf{f})) \qquad (1)$$

will minimize $||\Psi(\mathbf{x}) - \Psi_f(\mathbf{x})||^2$ where the $\pi(\mathbf{x}, \mathbf{f})$ operator projects the vector $\mathbf{x}$ to a low dimensional space according

to the binary mask vector $\mathbf{f}$. In basis selection we look for a binary vector $\mathbf{b} = (b_1, ..., b_n) \quad b_i \in \{0, 1\}$ s.t.

$$\Psi_{\mathbf{b}}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i b_i k(\mathbf{x}, \mathbf{s_i}) \tag{2}$$

will minimize $||\Psi(\mathbf{x}) - \Psi_{\mathbf{b}}(\mathbf{x})||^2$.

Now, let $\mathbf{M} = \mathbf{f}^T\mathbf{b}$ be a masking matrix, then the feature selection equation 1 becomes:

$$\Psi_{\mathbf{f}}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\pi(\mathbf{x}, \mathbf{m_i}), \pi(\mathbf{s_i}, \mathbf{m_i})) \tag{3}$$

where $\mathbf{m_i}$ are the columns of the masking matrix and since the rank of the masking matrix is one, then it follows that all the columns are equal to each other. Similarly, the basis selection equation 2 becomes:

$$\Psi_{\mathbf{b}}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i max(\mathbf{m_i}) k(\mathbf{x}, \mathbf{s_i}) \tag{4}$$

Note that $max(\mathbf{m_i}) \in \{0, 1\}$ because $\mathbf{M}$ is a rank-1 binary matrix. Thus, the masking matrix captures, in a single framework, all the information needed for feature selection and basis selection. Taken together we can give the following definition of the masking matrix.

Definition 1 (The masking matrix) The masking matrix $\mathbf{M}$ of n d-dimensional basis vectors is a $d \times n$ binary matrix, whose elements are denoted by m_{ij}. $m_{ij} = 1$ if feature i is used by basis vector j, otherwise $m_{ij} = 0$. The columns of $\mathbf{M}$ are denoted by $\mathbf{m_j}$. The joint feature-basis selection expressed by the masking matrix is given by:

$$\Psi_{\mathbf{M}}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i max(\mathbf{m_i}) k(\pi(\mathbf{x}, \mathbf{m_i}), \pi(\mathbf{s_i}, \mathbf{m_i})) \tag{5}$$

The $max(\mathbf{m_i})$ operation determines if basis vector i participates in the expansion while the elements in $\mathbf{m_i}$ determine which features in the basis vector should be used. It is important to include the $max(\mathbf{m_i})$ term to cancel basis functions since the kernel function $k(\mathbf{x}, 0)$ might not be zero.

We might try and optimize $\Psi_{\mathbf{M}}$ directly, with respect to the generalization error of the classifier, or we might try to minimize its difference from a known optimal solution (i.e. minimize $||\Psi(\mathbf{x}) - \Psi_{\mathbf{M}}(\mathbf{x})||^2$). In any case, the masking matrix captures all there is to know about which features and which basis vectors should be used.

it can readily be seen that feature selection and basis selection are special instances of the masking matrix. In feature selection we have that $\mathbf{M} = \mathbf{f}^T\mathbf{1}$, where $\mathbf{1}$ is a $1 \times n$ vector of ones. In basis selection, on the other hand, we have that $\mathbf{M} = \mathbf{1}^T\mathbf{b}$, where $\mathbf{1}$ is a $d \times 1$ vector of ones. The masking matrix representation reveals the assumptions we make in feature or basis selection algorithms and suggest ways to extend them. For instance, instead of fixing $\mathbf{b}$ or $\mathbf{f}$ to be 1 we might search for a joint feature-basis selection algorithm by estimating directly a rank-1 binary matrix $\mathbf{M}$, instead of searching for $\mathbf{b}$ and $\mathbf{f}$ independently.

But why limit ourselves to a rank-1 masking matrix? A rank-1 masking matrix indicates that the same feature must be used in all the basis vectors. But it might be the case that different features are needed for different basis vectors. Such a situation can not be handled by feature selection or basis selection algorithms as it requires interaction between the two. This interaction is possible with the masking matrix. Trying to find the masking matrix *directly* allows us to recover the interactions between features and basis vectors. We give an algorithm to estimate the masking matrix in the next section.

4 The element selection algorithm

We modify a standard feature selection algorithm for estimating the masking matrix. To avoid confusion we term it *element selection*. This is a forward selection, greedy algorithm that chooses elements that help optimize some objective function. In fact, our algorithm is equivalent to standard feature selection algorithms only that we work in a different space. While feature selection algorithms work in the "row" space of the masking matrix and basis selection algorithms work in the "column" space of the masking matrix, our element selection method work in the "element" space of the masking matrix.

In particular our objective function is to approximate the full SVM expansion, as given in the following equation:

$$\Psi = \mathbf{K}\alpha \approx \tilde{\mathbf{K}}\beta \tag{6}$$

That is, we are given the full design matrix $\mathbf{K}$ and weights α and we need to find an approximated matrix $\tilde{\mathbf{K}}$ and appropriate weights β. The approximated matrix $\tilde{\mathbf{K}}$ is given by:

$$\tilde{k_{ij}} = k(\pi(\mathbf{x_i}, \mathbf{m_j}), \pi(\mathbf{b_j}, \mathbf{m_j})) \tag{7}$$

So, our approach is first to approximate the full design matrix $\mathbf{K}$ with the approximated matrix $\tilde{\mathbf{K}}$ and then calculate the weights β.

Our solution is to use a greedy sequential forward selection, which is a standard procedure in feature and basis subset selection literature [2, 17]. In each step we choose the next element that minimizes the approximation error and once the desired number of elements is selected we calculate the new weight coefficients.

Here is an overview of the algorithm. It takes as input the matrix of basis vectors $\mathbf{B}$ and their weights α, the matrix of training data $\mathbf{X}$, as well as the desired number of elements t and it outputs the masking matrix $\mathbf{M}$ as well as the weights β, of the selected basis vectors.

Algorithm 1 Element Selection

Input: $\mathbf{B}$ - basis vectors
$\mathbf{X}$ - training set vectors
α - weight vector of original basis vectors
t - number of elements to choose

Output: $\mathbf{M}$ - masking matrix of selected elements
β - weight vector of (sparse) basis vectors

Initialize the masking matrix $\mathbf{M}$
$\mathbf{M} = 0$
Construct the design matrix $\mathbf{K}$
$k_{ij} = k(\mathbf{x_i}, \mathbf{b_j})$
Calculate expected labels $\tilde{\mathbf{y}}$
$\tilde{\mathbf{y}} = \mathbf{K}\alpha$
Add t elements to the masking matrix $\mathbf{M}$
For $i = 1 : t$,
    # Find next element
    [row,col] = chooseNextBestElement($\mathbf{K}, \mathbf{X}, \mathbf{B}, \mathbf{M}$)
    $\mathbf{M}(row, col) = 1$
    # Orthogonalize $\mathbf{K}$
    $\tilde{\mathbf{K}}$ = sparseKernelMatrix($\mathbf{X}, \mathbf{B}, \mathbf{M}$)
    $\gamma = \tilde{\mathbf{K}}^{-1} * \mathbf{K}$
    $\mathbf{K} = \mathbf{K} - \tilde{\mathbf{K}}\gamma$
end
Solve for new weights
$\beta = \tilde{\mathbf{K}}^{-1}\tilde{\mathbf{y}}$

In our case, the matrix $\mathbf{B}$ contains the support vectors, as obtained by solving the SVM problem (each column in $\mathbf{B}$ corresponds to one support vector) and $\mathbf{X}$ is the matrix of the training data (again, each column of $\mathbf{X}$ corresponds to one training example). The function $chooseNextBestElement()$ uses the function $sparseKernelMatrix()$ with different elements and returns the element that best approximates the full design matrix $\mathbf{K}$, where The function $sparseKernelMatrix()$ calculates the approximated kernel matrix using equation 7. We then remove the contribution of the selected element from the matrix $\mathbf{K}$ and repeat the process.

This algorithm is expansive to compute. It costs $O(n \cdot d)$ iterations to add one element, where d is the number of features and n is the number of basis functions. Back of the envelope calculation will show that if we have 1000 support vectors, each of size 400 pixels, we need $O(400,000)$ iterations to find just a single element. Finding the first 100 elements will take $O(40,000,000)$. We therefor propose a probabilistic speedup.

4.1 Probabilistic speedup

Two observations guide us in the probabilistic speedup. First, that it is too expansive to try each and every element as a potential candidate to be added to the element subset. Second, we can use only part of all the training patterns to evaluate each selected element.

It was shown by [17] (pp. 180) that we can find, with high probability, the maximum of m random variables, in a small number of trials. More specifically, in order to obtain an estimate that is with probability 0.95 among the best 0.05 of all estimates, a random subsample of size $\frac{log 0.05}{log 0.95} \approx 59$ will guarantee nearly as good performance as if we considered all the random variables. Assume we already have an element subset of $n-1$ elements and we need to pick the $n-th$ element. If one is willing to assume that element subsets of size n are random variables then randomly sampling 60 elements to be added to our current subset of $n-1$ elements will give us, with high probability, a good element subset of size n. In practice, we sample 60 elements on every step of the algorithm and pick the best one. Secondly, for each element we evaluate it on only a portion of all the training samples, to speed up calculations. That is, in the function $chooseNextBestElement()$ we randomly sample 10% of the data to evaluate each new element. Once an element is selected we need to remove its contribution from the entire design matrix and so we must orthogonalize the design matrix K. Finally, a useful heuristic that proved to be quite effective was to run the element selection algorithm in chunks. Each time we estimated the next, say, 10 elements ahead, but repeated the procedure for several times and picked the best chunk of 10 elements before proceeding.

5. Experiments

We tested our method on two datasets, one of vehicles and one of faces.

The vehicle image dataset consisted of 10589 training images and 13881 test images, each consisted of roughly 2500 vehicle images and the rest were non-vehicle images. The size of the images is 20×20 pixels. We trained a RBF SVM classifier on the data set and obtained a SVM solution with 2943 support vectors. We then ran our element selection algorithm and compared it to the second variant of the reduced set method of [4], as discussed in subsection 2.2. This method constructs a new set of support vectors and weights that approximates the original set of support vectors. This method is not pure basis selection as each reduced set vector is *synthesized* not selected, nevertheless it represents the state-of-the-art in accelerating run-time SVM classification.

Figure 2 shows the results on the vehicle test set. As can be seen, the 400 element set classifier is comparable to

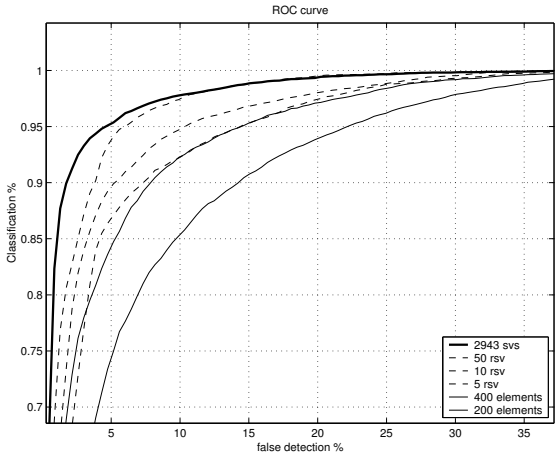

Figure 2: Vehicle Dataset: Comparing element selection Vs. reduced set method. We show the ROC curve for the full set of 2493 support vectors (bold solid line), compared to three reduced sets of 5 10 and 50 support vectors from bottom to top, respectively (all in dashed lines), and two elements sets, one with 200 elements and the other with 400 elements (both in solid line). An element set of 400 elements is equivalent to a *single* support vector. Hence, the 400 element set is equivalent to the 5 reduced set in terms of classification power but uses much less memory.

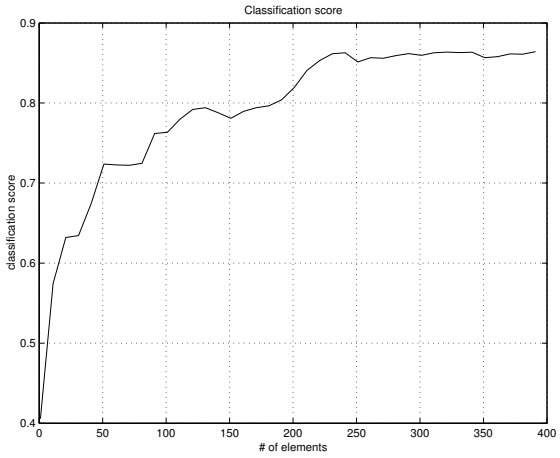

Figure 3: Vehicle Dataset: We show the classification score as a function of the number of elements used. The x-axis measures the number of elements and the y-axis measures the classification score, where we define the classification score as the breakeven point where the percentage of false positive is equal to the percentage of false negative.

the 5 reduced set vector classifier. However, the element-set classifier requires roughly 400 kernel operations, because almost each element belonged to a different support vector, compared to only 5 required by the reduced set method. In terms of memory requirements the element selection method stores only 800 numbers (400 elements and 400 weights) compared to $2005 = 5 * 400 + 5$ numbers to be stored in the reduced set method. The large number of kernel operations might be quite expansive to compute but it can be solved by using a lookup table. In a separate experiment we found that a look-up table of size 16K entries can replace the actual use of the exponent function, without noticeable difference in classification performance. In case of polynomial kernels there is no need for lookup table in the first place.

It will require a very large number of elements to *replace* the reduced set method. However, the element selection method can be quite effective in *rejecting* patterns. For instance, the 200 element set can correctly reject about 70% of the negative patterns while falsely rejecting less than 2% of the positive patterns.

In a second experiment we tested how fast does the element selection algorithm improves. Figure 3 show that the first few elements give a very large improvement, but after about 250 elements the improvement is negligible.

Next, we tested our method on the face classification problem. The training set consisted of 9666 images of size 24×24 pixels each. 4666 of them were faces, the rest were non-faces. The test set consisted of a different set of 9666 images of size 24×24 pixels and it, as well, consisted of 4666 face images and 5000 non-face images. We then used SVM^{light} [6] to train an RBF SVM on the data and achieved a classifier with 1434 support vector and 95% success rate in classifying both positive and negative examples on the test set.

Figure 4 show the results on the face test set. As can be seen, for comparable classification score the element selection method uses only 40% of the memory used by the reduced set method. Again, almost for every element we have a weight coefficient to store. As before, the method is probably more useful for *rejection* than *replacing* the reduce-set method. In the case of the face database it can be seen that 80% of the negative patterns can be correctly rejected with false negative rate of less than 2%.

A second experiment tested how fast does the element selection algorithm improves for the face dataset. Figure 5 show that the first few elements give a very large improvement, but after about 250 elements the improvement is negligible.

6. Acknowledgment

I would like to thank Mike Jones, from MERL, for supplying me with the face database and the MobilEye research team for supplying me with the vehicle database.

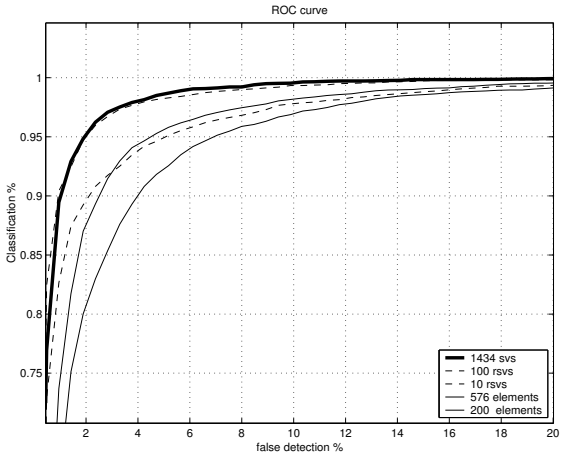

Figure 4: Face Dataset: We show the ROC curve for the full set SVM of 1434 support vectors (bold solid line), two reduced set methods of 10 and 100 reduced sets (dashed line). The dashed line of the 100 reduced set coincide almost entirely with the full set of support vectors. In addition, we show two element sets of 200 and 576 elements (solid line). Note that an element set of 576 elements is equivalent to a *single* support vector. Hence, the 576 element set is equivalent to the 10 reduced set in terms of classification power but uses much less memory.

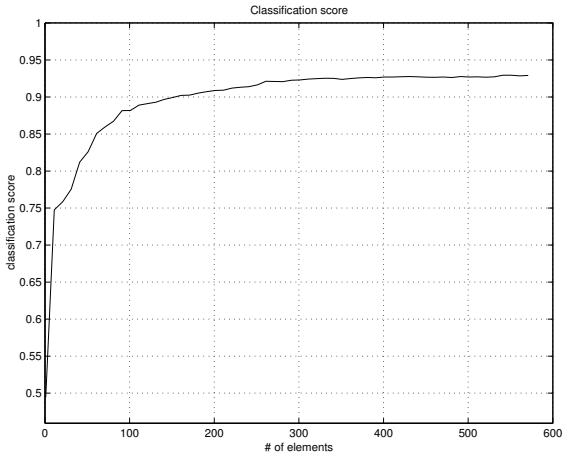

Figure 5: Face Dataset: We show the classification score as a function of the number of elements used. The x-axis measures the number of elements and the y-axis measures the classification score, where we define the classification score as the breakeven point where the percentage of false positive is equal to the percentage of false negative.

7. Summary and Conclusions

We unified feature selection and basis selection algorithms using the masking matrix. The masking matrix is a natural representation that treats features and basis vectors alike. Moreover, we showed that the masking matrix representation leads to more general subset selection algorithms in which different features can be assigned to only part of the basis vectors. The problem is NP-hard and we gave a probabilistic greedy algorithm for solving it.

In the current paper we focused on accelerating the runtime performance of SVM classification. In our experiments we found that the sparse basis/feature selection can serve as an excellent rejector that can reject up to 80% of the negative patterns with less than the memory requirements used by a single basis function.

Future research directions will focus on recovering a rank-constrained masking matrix, which implies that there are several prototypes of feature subsets to choose from and each basis vector can choose only one of them. Another direction will be to integrate the element selection strategy into the reduced set method. This way, when a new vector will be added to the reduced set we will use feature selection to find a subset of the features that can approximate the dot-product when using the full vectors to the dot-product using just a subset. Finally, we would like to include the recovery of the masking matrix as part of the SVM process, not as a post-processing step.

References

[1] J. Bins and B. Draper. Feature Selection from Huge Feature Sets. In Int. Conf. on Computer Vision, volume 2, pages 159 165, Vancouver, CA, 2001.

[2] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. Artificial Intelligence, 97:245 271, 1997.

[3] L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. D. Jackel, Y. LeCun, U. A. Muller, E. Sackinger, P. Simard and V. Vapnik. Comparison of classifier methods: a case study in handwritten digit recognition. In *Proceedings of the 12th International Conference on Pattern Recognition and Neural Networks*, Jerusalem, pages 77-87, IEEE Computer Society Press, 1994.

[4] C.J.C. Burges. Simplified support vector decision rules. In L. Saitta, editor, *Proceedings, 13th International Conference on Machine Learning*, pages 71-77, San Mateo, Ca, 1996.

[5] C.J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121-167, 1998.

[6] T. Joachims. Making Large-Scale SVM Learning Practical. In *Advances in Kernel Methods - Support Vec-*

tor Learning, B. Schlkopf and C. Burges and A. Smola (ed.), MIT Press, 1999.

[7] John, G. Kohavi R. and Pfleger, K. Irrelevant features and the subset selection problem. In *Machine Learning: Proceedings of the Eleventh International Conference*, Morgan Kaufmann, pp. 121-129, 1994.

[8] R. Kohavi and G.H. John, Wrappers for Feature Subset Selection , Artificial Intelligence Journal, 97(1-2):273-324, 1997.

[9] S.Z. Li, L. Zhu, Z.Q. Zhang, A. Blake, H.J. Zhang and H. Shum. Statistical Learning of Multi-View Face Detection. In *Proceedings of the 7th European Conference on Computer Vision*, Copenhagen, Denmark, May 2002.

[10] T. Marill and D.M. Green, On the Effectiveness of Receptors in Recognition Systems , IEEE transactions on Information Theory, 9:11-17, 1963.

[11] P. Narendra and K. Fukunaga. A Branch and Bound Algorithm for Feature Subset Selection. IEEE Transactions on Computer, C 26(9):917 922, 1977.

[12] E. Osuna, R. Freund and F. Girosi. Training Support Vector Machines: An Application to Face Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, Puerto Rico, pages 130-136, 1997.

[13] P. Pudil, J. Novovicov, and J. Kittler. Floating Search Methods in Feature Selection. Pattern Recognition Letters, 15(11):1119 1125, 1994.

[14] J.O. Rawlings. Applied Regression Analysis. Wadsworth & Brooks/Cole, Pacific Grove, CA, 1988.

[15] S. Romdhani, P. Torr, B. Scholkopf, and A. Blake. Computationally Efficient Face Detection. In *International Conference on Computer Vision*, Vancouver, 2001.

[16] B. Scholkopf. Support Vector Learning. R. Oldenbourg Verlag, Munich, 1997.

[17] B. Scholkopf and A. Smola. Learning with Kernels Support Vector Machines, Regularization, Optimization and Beyond. MIT Press, Cambridge, MA, 2002.

[18] Yang, Y., Pedersen J.P. A Comparative Study on Feature Selection in Text Categorization Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97), 1997, pp412-420.

[19] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer, N.Y., 1995.

[20] P. Viola and M. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. In *IEEE Conference on Computer Vision and Pattern Recognition*, Hawaii, 2001.

[21] J. Weston, S. Mukherjee, O. Chapelle, M. Pon- til, T. Poggio, and V. Vapnik. Feature selection for SVMs. In Proc. Advances in Neural Information Processing Systems (NIPS'00), pages 668- 674, 2000.