Sparse Max-Margin Multiclass and Multi-label Classifier Design for Fast Inference

Tanuja Ganu[*] Shirish Shevade[†] S. Sundararajan[‡]

Abstract

We address the problems of sparse multiclass and multi-label classifier design and devise new algorithms using margin based ideas. Many online applications such as image classification or text categorization demand fast inference. State-of-the-art classifiers such as Support Vector Machines (SVM) are not preferred in such applications because of slow inference, which is mainly due to the large number of support vectors required to form the SVM classifier. We propose algorithms which solve primal problems directly by greedily adding the required number of basis functions into the classifier model. Experiments on various real-world data sets demonstrate that the proposed algorithms output significantly smaller number of basis functions, while achieving nearly the same generalization performance as that given by SVM and other state-of-the-art sparse classifiers. This enables the classifiers to perform faster inference, thereby making the proposed algorithms powerful alternatives to existing approaches.

1 Introduction

We consider *multiclass* and *multi-label* classification problems. Given a training data set $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, where $\mathbf{x}_i \in X \subset \mathbb{R}^n$ is the i^{th} example and y_i is the corresponding class label ($y_i \in Y = \{1, \ldots, k\}$), the multiclass classification problem is to construct a decision function $h : X \to Y$ which generalizes well. In many real-world applications such as object detection in computer vision or medical diagnosis, more than one class label can be correct for each data point. These problems are formulated as multi-label classification problems. The training data set is of the form, $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{l}$, where $\mathbf{y}_i \in \{\pm 1\}^k$, $y_{i,r} = +1$ if the i^{th} example belongs to the class r and $y_{i,r} = -1$ otherwise. The multi-label classification problem is to construct a decision function $h : X \to \{\pm 1\}^k$ which gives good generalization performance.

Many online applications such as real-time video surveillance, network intrusion detection or credit card fraud detection can be posed as multiclass or multi-label classification problems. Due to their real-time nature, these applications require fast inference (finding class label(s) of a test input). Powerful classification tools such as Support Vector Machines (SVM), though they yield accurate solutions, are not preferred in such applications due to high inference time, when the number of support vectors is large (linear in the training set size [16]). Therefore, for applications which require fast inference, it is important to design classifiers which are not computationally expensive.

For many classification problems, a classifier designed using a smaller subset of *basis functions*[1] of the type $K(\mathbf{x}, \mathbf{x}_i)$ yields generalization performance similar to that of the SVM classifier. In this work, we refer to such classifiers as *sparse* classifiers. Although sparse multiclass and multi-label classifiers can be designed using sparse binary classification algorithms like Sparse SVM (SpSVM) [8] or those presented in [7, 11], the resulting classifiers still use a large number of basis vectors. This is mainly because they do not use any systematic way of finding a common set of basis functions over all the classes. To the best of our knowledge, there have been no attempts to design sparse multi-label classifiers. Therefore, there is a need to develop efficient algorithms to design sparse classifiers for multiclass and multi-label classification problems.

Contributions

- We propose new 'all-together' problem formulations for *sparse* multiclass (Section 3) and multi-label (Section 4) classification and present *novel* and *efficient* primal methods to design sparse classifiers. The proposed algorithms systematically search and output a *common* set of basis vectors for all the classes. The number of basis vectors controls the classifier sparsity, through a user-defined parameter, d_{max}.
- The proposed algorithms are compared with other sparse classifier design algorithms on various real-world data sets. Numerical experiments on differ-

[*]IBM Research, Bangalore, INDIA.
[†]Indian Institute of Science, Bangalore, INDIA.
[‡]Microsoft Research India, Bangalore, INDIA.

[1]In this work, we use the key phrases "basis functions" and "basis vectors" interchangeably.

ent real-world datasets give clear evidence that the proposed algorithms give sparser classifiers without significant degradation in the generalization performance and are, therefore, powerful alternatives when *fast inference* is needed.

Since our focus is on the inference time, we demonstrate the efficacy of the proposed approaches on data sets having moderate number of training set examples, but a large number of test set examples. On a typical real-world multiclass data set with about $25,000$ training set examples and 10^6 test set examples, the classifier designed using the proposed algorithm used only 100 basis functions, and required about 30 seconds for inference on all the test set examples. On the other hand, the classifier designed by extending Sparse SVM [8] to multiclass problems required an order of magnitude larger number of basis functions and inference time. A supplementary file including additional information about related work and experiments is available at `http://drona.csa.iisc.ernet.in/~shirish/SDM2013/supplement.pdf`.

A word about our notations. All vectors will be column vectors and the row vectors will be denoted by a superscript, ^{T}. $\|\mathbf{v}\|$ denotes the 2-norm of the vector $\mathbf{v}$. The cardinality of the set J will be denoted by $|J|$. For a n-dimensional vector $\mathbf{v}$, let $\mathbf{v}_J$ denote the $|J|$ dimensional vector containing $\{\mathbf{v}_j : j \in J\}$. For a matrix $A \in \mathbb{R}^{m\times n}$, $A_{i,j}$ denotes the $(i,j)^{th}$ element of A. $A_{i,\cdot}$ denotes the i^{th} row of A and $A_{\cdot,j}$ denotes the j^{th} column of A. $A_{I,J}$ refers to the submatrix of A made of the rows indexed by I and the columns indexed by J. If A is a square matrix $Tr(A)$ denotes the trace of A. δ is the Kronecker delta, $\delta_{i,j} = 1$ if $i = j$ and 0 otherwise.

2 Related Work

2.1 Multiclass classifiers A number of different methods have been proposed in the literature to solve the multiclass classification problem. Some methods such as *one-vs-all* (OVA) or *all-vs-all* (AVA), decompose a multiclass classification problem into multiple independent binary classification problems. Other approaches, called *all-together* approaches [6], design multiclass classifiers by formulating an optimization problem which considers all the classes together. Some prominent all-together multiclass classification approaches include [2, 19, 1, 12]. Particularly relevant to the proposed work is the formulation proposed in [19] which we describe briefly.

Most of the *all-together* multiclass classifiers design real-valued functions $f_r(\cdot)$, $r = 1, \dots, k$, which assign a set of *similarity scores* to every data sample. The classifier function[2] h uses the following form:

$$h(\mathbf{x}) = \operatorname*{argmax}_{1\le r\le k} f_r(\mathbf{x}). \tag{2.1}$$

Weston and Watkins [19] proposed to use $f_r(\mathbf{x}) = \mathbf{w}_r \cdot \phi(\mathbf{x})$ and formulated the multiclass classification problem as,

$$\min_{\mathbf{w},\xi} \quad \frac{\lambda}{2}\sum_{r=1}^{k}\|\mathbf{w}_r\|^2 + \sum_{i=1}^{l}\sum_{r\neq y_i}\xi_{i,r} \tag{2.2}$$

where $\xi_{i,r} = \max(0, \mathbf{w}_r\cdot\phi(\mathbf{x}_i) + 2 - \mathbf{w}_{y_i}\cdot\phi(\mathbf{x}_i))$ denotes the misclassification error for every example $\mathbf{x}_i$ and class $r,\ \ r \neq y_i$. Computations involving ϕ are handled using the kernel function, $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j)$. The bias term can be included in this formulation in a straightforward way. For notational convenience, we have not included it throughout this article. The solution to the optimization problem (2.2) can be found by solving the equivalent dual problem, as in the case of the binary SVM optimization problem. Note that the dual problem of (2.2) is a large dimensional optimization problem, which may be difficult to solve using standard quadratic programming techniques. This difficulty is alleviated by the use of a single slack variable for every example[2].

$$\begin{aligned}\min_{\mathbf{w},\xi} \quad & \frac{\lambda}{2}\sum_{r=1}^{k}\|\mathbf{w}_r\|^2 + \sum_{i=1}^{l}\xi_i \\ \text{s.t. } & (\mathbf{w}_{y_i}\cdot\mathbf{x}_i) + \delta_{y_i,r} - (\mathbf{w}_r\cdot\mathbf{x}_i) \ge 1 - \xi_i \ \forall i, r.\end{aligned} \tag{2.3}$$

In this formulation, ξ_i denotes the misclassification error for every example $\mathbf{x}_i$. A simple and efficient algorithm was proposed to solve the dual problem of (2.3) [2]. In either case, the resulting classifiers are often not sparse, though their generalization performance is good.

2.2 Multi-label classifiers incorporating label correlations In the multi-label classification problem, given a set Y containing k class labels, an example can be tagged with any of the 2^k possible subsets of Y. The main challenge is to search over the exponentially large label space. This problem can also be treated as a simpler version of multi-task learning problem, where the same input data are used for all the tasks, but the output labels differ for each task and depend upon the task correlation information [4] Some approaches solve the multi-label classification problem by converting to other canonical forms like binary classification, multiclass classification, regression or ranking (see [18] and the references therein). Often, these algorithms do not consider the inherent label correlation information in

[2]The one-vs-all (OVA) approach to solve a multiclass classification problem also uses the same decision function in (2.1).

solving a multi-label classification problem. The multi-label classifier design problem becomes more difficult if the label correlations are also taken into account. The max-margin multi-label classifier design approach proposed by Hariharan et al. [5], uses the prior knowledge about the co-occurrences of labels in the test categories. This approach, relevant to our work, is discussed below.

Assuming that the matrix P encodes the label correlations and the matrix W contains the weight vectors associated with all the labels, by defining a real-valued score function $f(\mathbf{x},\mathbf{y}) = \mathbf{y}^T P^T W^T \phi(x)$, the multi-label learning problem was formulated [5] as:

(2.4)
$$\begin{aligned} \min_f \quad & \frac{1}{2}\|f\|^2 + C\sum_{i=1}^{l}\xi_i \\ \text{s.t.} \quad & f(\mathbf{x}_i,\mathbf{y}_i) \geq f(\mathbf{x}_i,\mathbf{y}) + \Delta(\mathbf{y}_i,\mathbf{y}) - \xi_i \ \forall \mathbf{y} \in \{\pm 1\}^k \setminus \{\mathbf{y}_i\} \\ & \xi_i \geq 0 \ \forall i \end{aligned}$$

where $\Delta(\cdot)$ is the multi-label loss function. The decision function h for a new point $\mathbf{x}$ is given by

$$h(\mathbf{x}) = \underset{\mathbf{y}\in\{\pm 1\}^k}{\operatorname{argmax}} f(\mathbf{x},\mathbf{y}). \tag{2.5}$$

Note that there are $l2^k$ constraints in the formulation in (2.4). This exponentially large number of constraints makes the optimization very slow. This problem was alleviated by reformulating (2.4) as the minimization of k correlated sub-problems, each having only l constraints [5]. The resulting multi-label classifier is, however, not sparse. Our goal is to provide an efficient algorithm to design a sparse multi-label classifier in this setting.

2.3 Sparse Classifier Design Several methods have been proposed in the literature to design sparse multiclass classifiers. These include k-class linear programming machines [19], Import Vector Machine (ImVM) [20], Multiclass Relevance Vector Machines [15] and Informative Vector Machine [10]. The design of a k-class linear programming machine requires sophisticated linear program solvers which are computationally expensive. Our work uses basis function selection ideas similar to those proposed in Sparse SVM (SpSVM) [8] and ImVM [20] for sparse model design and differs from probabilistic approaches [15, 10] by the way sparse model is constructed.

A sparse classifier design algorithm involves efficient selection of the set J of basis functions, as well as the the corresponding model parameters. SpSVM minimizes regularized quadratic hinge loss function while ImVM is built on optimizing regularized kernel logistic regression loss function to get the basis functions and the model parameters. Starting with an empty model, both the algorithms use a greedy approach to add a desired number (d_{max}) of basis vectors to the model, by making use of the respective objective function decrease, for basis vector selection. Our experiments using ImVM for sparse multiclass classifier design on large data sets confirm that ImVM is not scalable. Also, compared to the proposed approach, an ImVM or a sparse multiclass classifier designed using SpSVM in one-vs-all (OVA) mode typically require an order of magnitude larger number of basis vectors, thereby resulting in slower inference.

3 Multiclass Classification - Problem Formulation and Algorithm

Our proposed problem formulation for multiclass classification problem uses L_2 penalization of the training errors and the proposed algorithm directly minimizes the primal problem over the set of vectors $\mathbf{w}_r$ of the form

$$\mathbf{w}_r = \sum_{j\in J}\alpha_{j,r}\phi(\mathbf{x}_j) \tag{3.6}$$

where J is an index set of basis functions that form a subset of the training set input. Finding an appropriate set J and the corresponding parameters α is done by optimizing the primal objective function. We consider k classifier functions, $f_r(\mathbf{x}),\ r = 1,\ldots,k$, of the form

$$f_r(\mathbf{x}) = \mathbf{w}_r \cdot \phi(\mathbf{x}) = \sum_{j\in J}\alpha_{j,r}K(\mathbf{x}_j,\mathbf{x}). \tag{3.7}$$

Note that the set J is *common* for all the classes. The learning algorithm finds this set J as well as the matrix of model parameters, $\boldsymbol{\alpha}_{J,\cdot} \in \mathbb{R}^{|J|\times k}$. The multiclass decision function in (2.1) can then be used, where $f_r(\mathbf{x}_i)$ is represented in a compact form as,

$$f_r(\mathbf{x}_i) = {\alpha_{J,r}}^T K_{J,i}. \tag{3.8}$$

3.1 Method Our formulation uses quadratic penalization of errors in the formulation (2.2) proposed in [19]. We consider the following problem formulation:

$$\min_{\mathbf{w},\mu} \quad \frac{\lambda}{2}\sum_{r=1}^{k}\|\mathbf{w}_r\|^2 + \frac{1}{2}\sum_{i=1}^{l}\sum_{r=1}^{k}(\max(0\ ,\ \mu_{i,r}))^2 \tag{3.9}$$

where $\mu_{i,r} = \mathbf{w}_r\cdot\phi(\mathbf{x}_i) + 1 - \mathbf{w}_{y_i}\cdot\phi(\mathbf{x}_i) - \delta_{y_i,r}\ \forall\ i,r$ and the misclassification error, $\xi_{i,r} = \max(0\ ,\ \mu_{i,r})$. The quadratic penalization of the errors in (3.9) makes the objective function differentiable. Using $\mathbf{w}_r$ in (3.6), the problem in (3.9) can be written as:

(3.10)
$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & e_J \equiv \frac{\lambda}{2}\sum_{r=1}^{k}{\alpha_{J,r}}^T K_{J,J}\alpha_{J,r} + \frac{1}{2}\sum_{i=1}^{l}\sum_{r=1}^{k}\xi_{i,r}^2 \\ \text{s.t.} \quad & \xi_{i,r} = \max\left(0, {\alpha_{J,r}}^T K_{J,i} + 1 - {\alpha_{J,y_i}}^T K_{J,i} - \delta_{y_i,r}\right) \forall i,r \end{aligned}$$

Note that the objective function in the above problem is piecewise quadratic. Although it is not twice differentiable, one can define the generalized Hessian [13] and use Newton method with line-search to solve this problem. Our aim is to obtain an approximate solution that uses a common set of basis functions for all the classes and to find $\alpha_{J,\cdot}$ such that the resulting classifier has the desired level of complexity. For this purpose, the choice of the basis vector set J becomes crucial. Given that $|J| \le d_{max}$, searching through all possible combinations of d_{max} basis vectors to get an optimal set J is computationally expensive. Therefore, one has to resort to approaches like those suggested in [20, 8]. A typical sparse classifier design algorithm is given below. This

Algorithm 1 : Sparse Classifier Design Algorithm

```
Input: $D = \{\mathbf{x}_i, y_i\}_{i=1}^{l}$, $d_{max}$
Output: $J$, $\boldsymbol{\alpha}_{J,\cdot}$
1: $J = \phi$
2: while $|J| < d_{max}$ do
3:     Select a new basis function $j^*$ which gives a maximum
       decrease in the objective function
4:     $J := J \cup \{j^*\}$
5:     Optimize the objective function w.r.t $\boldsymbol{\alpha}_{J,\cdot}$
6: end while
```

approach of sparse classifier design is not fundamentally new. For multiclass classification problems, the algorithm involves efficient selection of the basis vector set J, *common for all the classes*. As we describe later, this procedure is crucial as the selection of J affects $\boldsymbol{\alpha}_{J,\cdot}$. We now give details related to step 3 (basis function selection) and step 5 (optimization w.r.t. $\alpha_{J,\cdot}$).

3.2 Basis Function Selection We now discuss a systematic and efficient way of selecting basis functions iteratively. Given the basis function set J and the coefficients $\boldsymbol{\alpha}_{J,r}$, our aim is to select a training set example $j^* \in \{1, ..., l\} \setminus J$, as a new basis function, such that its inclusion in the set J would give a maximum decrease in the objective function in (3.10). The training set examples which incur the misclassification error ($\xi_{i,r} > 0$) are distributed over the two sets of training examples I^r and I^r_m,

$$I^r = \Big\{ i : i \in \{1, ..., l\},\ \xi_{i,r} > 0,\ r \ne y_i \Big\}$$
$$I^r_m = \Big\{ i : y_i = r,\ \xi_{i,p} > 0 \text{ for some } p \ne y_i \Big\}. \tag{3.11}$$

Clearly, $I^r \cap I^r_m = \phi$. Note that I^r and I^r_m are functions of J and $\boldsymbol{\alpha}_{J,\cdot}$. For notational convenience, we do not indicate this dependence explicitly. Suppose we add a new basis vector j to the existing set J. Let $\hat{J} = (J; j)$. To find the effect of this basis function addition on the objective function, it becomes necessary to solve the problem (3.10) with respect to $\alpha_{\hat{J},r}$. Note that this problem needs to be solved for all the candidate basis vectors from the set $\{1, \ldots, l\} \setminus J$ and the basis vector j^* is selected which gives a maximum decrease in the objective function in (3.10). This procedure is computationally expensive, especially when the training set is large. So in our implementation, we solved k one-dimensional optimization problems (with respect to $\alpha_{j,r}$, $r = 1, \ldots, k$, keeping $\boldsymbol{\alpha}_{J,\cdot}$, fixed).

$$\min_{\alpha_{j,r}} \frac{\lambda}{2}\alpha^T_{\hat{J},r} K_{\hat{J},\hat{J}} \alpha_{\hat{J},r} + \frac{1}{2}\sum_{i \in I^r} (K_{i,\hat{J}}\alpha_{\hat{J},r} + 1 - K_{i,\hat{J}}\alpha_{\hat{J},y_i})^2 + \frac{1}{2}\sum_{i \in I^r_m}\sum_{p \in \bar{r}_i} (K_{i,\hat{J}}\alpha_{\hat{J},p} + 1 - K_{i,\hat{J}}\alpha_{\hat{J},r})^2 \tag{3.12}$$

Let e_J be the value of the objective function in (3.10) corresponding to the current set of basis vectors J and coefficients $\boldsymbol{\alpha}_J$ and $e_{\hat{J}}$ be the objective function value after adding j to the current set of basis vectors along with its optimized coefficients α_j. Defining $\hat{J} = (J; j)$, the new basis vector $j^\star$ is then selected as:

$$j^\star = \arg \max_{j \in \{1,\ldots,l\} \setminus J} (e_J - e_{\hat{J}}). \tag{3.13}$$

During the optimization of (3.12), if I^r and/or I^r_m change, the objective function value in (3.10) might increase. In such a case, the associated basis function j is not considered for inclusion into the basis vector set. It was observed in our experiments that the sets I^r and I^r_m get stabilized as iterations progress. Therefore, after a few initial iterations, the phenomenon of the change in I^r or I^r_m is rare. Moreover, if the objective function value in (3.10) does not decrease for any of the candidate basis vectors, then we terminate the algorithm. But, we did not encounter such a situation during our experiments. Algorithm 2 gives the details of this procedure.

Algorithm 2 : Basis Function Selection

```
Input: $J$, $\boldsymbol{\alpha}_{J,\cdot}$
Output: The new basis function $j^\star$, optimized coefficients
    $\alpha_{j^\star,\cdot}$
1: $Q = \{1, \ldots, l\} \setminus J$
2: for every $j \in Q$ do
3:     for $r = 1, \ldots, k$ do
4:         Solve the problem (3.12) w.r.t. $\alpha_{j,\cdot}$
5:     end for
6:     Compute $e_{\hat{J}} \begin{bmatrix} \boldsymbol{\alpha}_J \\ \alpha_j \end{bmatrix}$
7: end for
8: Select $j^*$ using (3.13), $\boldsymbol{\alpha}_{J,\cdot} = (\boldsymbol{\alpha}_{J,\cdot}; \alpha_{j^\star,\cdot})$
```

The computational complexity of the algorithm to select one basis function is $O(l^2k^2)$, so the complexity of selecting d_{max} number of basis vectors is $O(l^2k^2d_{max})$. After selecting $j^\star$ in step 3 of Algorithm 1 and updating J and $\boldsymbol{\alpha}_{J,\cdot}$ the problem (3.10) is solved w.r.t $\alpha_{J,\cdot}$. We discuss this procedure next.

3.3 Optimization of Basis Vector Coefficients $\boldsymbol{\alpha}_{J,\cdot}$

Given the set J, the solution of (3.10) gives optimal $\boldsymbol{\alpha}_{J,\cdot}$. The structure of this objective function can be effectively used to solve (3.10) in an iterative fashion by optimizing with respect to the variables $\alpha_{J,r}$ associated with the class r, repeating this procedure till some stopping condition is satisfied. Thus, the following problem is solved:

(3.14)

$$\min_{\alpha_{J,r}} \frac{\lambda}{2}\alpha_{J,r}^T K_{J,J}\alpha_{J,r} + \frac{1}{2}\sum_{i\in I^r}(K_{i,J}\alpha_{J,r} + 1 - K_{i,j}\alpha_{J,y_i})^2 + \frac{1}{2}\sum_{i\in I_m^r}\sum_{p\in\bar{r}_i}(K_{i,J}\alpha_{J,p} + 1 - K_{i,J}\alpha_{J,r})^2$$

where

(3.15)

$$\bar{r}_i = \{s \in Y : s \neq y_i \text{ and } \xi_{i,s} > 0\},\ \forall\, i \in \{1,\ldots,l\}.$$

Algorithm 3 gives the details of $\boldsymbol{\alpha}_{J,\cdot}$ optimization.

Algorithm 3 : $\boldsymbol{\alpha}_{J,\cdot}$ Optimization

```
Input: $J$, current $\boldsymbol{\alpha}_{J,\cdot}$
Output: Optimized $\boldsymbol{\alpha}_{J,\cdot}$
 1: Initialize $\boldsymbol{\alpha}^0_{J,\cdot} = \boldsymbol{\alpha}_{J,\cdot}$ , set $t := 0$
 2: while $\boldsymbol{\alpha}^t_{J,\cdot}$ is not optimal for (3.14) do
 3:     for $r = 1,\ldots,k$ do
 4:         Find $I^r$ and $I^r_m$
 5:         Find the generalized Hessian $\mathbf{H}$ [13] and gradient
            $\mathbf{g}$ of the objective function in (3.14)
 6:         Solve the problem in (3.14) w.r.t. $\alpha_{J,r}$ to get $\bar{\alpha}_{J,r}$
            as the result of a Newton step in the direction
            $\mathbf{d_N} = -\mathbf{H}^{-1}\mathbf{g}$
 7:         Find $\alpha^{t+1}_{J,r}$ to be the minimizer of (3.10) on the line
            segment joining $\alpha^t_{J,r}$ and $\bar{\alpha}_{J,r}$.
 8:     end for
 9:     $t := t + 1$
10: end while
```

Since the loss function in (3.14) is piecewise quadratic, Newton method converges in a finite number of iterations [17]. We observed that the number of iterations required to reach the exact solution of (3.14) was less than 4. Given a set of basis vectors J, the computational complexity of Algorithm 3 is $O\big(k(lk+|J|^3)\big)$, where the complexity of finding the Newton direction (computing the generalized Hessian inverse) in step 3 of Newton method is $O(|J|^3)$.

The complete details to design a sparse multiclass classifier (SMSVM), using the formulation in (3.10) are presented in Algorithm 4. The algorithm requires the user defined parameter d_{max} as its input. In our experiments, the algorithm was terminated if the relative decrease in the objective function was sufficiently small (less than 10^{-3}) or $|J|$ exceeded d_{max}. The computational complexity of the complete algorithm is $O\big(kd_{max}(l^2k + d^3_{max})\big)$.

Algorithm 4 : Algorithm SMSVM

```
Input: $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^l$, $d_{max}$
Output: $J$, $\boldsymbol{\alpha}_{J,\cdot}$
 1: Initialize $J$ to a randomly chosen example from $D$.
 2: Solve optimization problem (3.14) to find optimal $\boldsymbol{\alpha}_{J,\cdot}$
    by Newton method using Algorithm 3.
 3: while $|J| < d_{max}$ do
 4:     Find $j^\star$ and $\alpha_{j^*,\cdot}$ using Algorithm 2.
 5:     $J := J \cup \{j^*\}$
 6:     Find the optimal $\boldsymbol{\alpha}_{J,\cdot}$ using Algorithm 3.
 7: end while
```

Speed-up Heuristics: To speed up the basis function selection step (Algorithm 2), the set Q can be selected as a random subset of size κ from the set $\{1,\ldots,l\} \setminus J$ (Step 1). This reduces the complexity of adding a new basis function to $O(\kappa lk^2)$. After some experiments, we found that $\kappa = 25$ was a good choice. The basis function selection also requires efficient solution of (3.12) (Step 4). This can be done by caching the similarity scores $f_r(\cdot)$ of all the training set examples, useful in calculating $\xi_{i,r}$ and determining I^r and I^r_m. This cache can be easily modified as $f_r(\mathbf{x}_i) := f_r(\mathbf{x}_i) + \alpha_{j,r}K_{j,i}$ when a new basis function j is added to the set J. These computations can further be speeded up by maintaining kernel cache, kernel matrix rows corresponding to the set J. We also maintain the Cholesky decomposition of the Hessian matrix H (Algorithm 3, step 5), which can be easily updated using rank-one updates if a basis vector gets added to the set J.

The ideas mentioned in this section can also be used to solve the primal problem for multiclass classification given in [2] directly, by making use of the quadratic penalization of the error. Relevant details are provided in the supplementary material (Section 4). However, in our experiments, we observed that the corresponding classifier was sparser than ImVM and Sparse OVA; but it required more basis vectors than that obtained by solving (3.10). Therefore, the corresponding experimental results are not provided.

4 Multi-label Classification - Problem Formulation and Algorithm

In this section, we present the details of our proposed algorithm for sparse multi-label classifier design. We continue to use the same representation for $\mathbf{w}_r$ as in (3.6). Hariharan et al. [5] assumed that $f(\mathbf{x}, \mathbf{y}) = \mathbf{w}^T(\phi(\mathbf{x}) \otimes \psi(\mathbf{y}))$, where ϕ and ψ are the feature and label space mappings respectively, $\mathbf{w}$ is the weight vector, and $\otimes$ is the Kronecker product. Further, it was assumed that the labels have linear correlations. Prior knowledge about the labels was incorporated by choosing $\psi(\mathbf{y}) = P\mathbf{y}$ where an invertible matrix P encodes the prior knowledge. The loss function Δ was chosen as the Hamming loss, $\Delta(\mathbf{y}_i, \mathbf{y}) = \sum_r 1 - y_r y_{i,r}$. The advantage of using the Hamming loss function is that it can be decomposed over the labels. By introducing two new variables Z and R where $Z = WP$ (W is the $n \times k$ dimensional matrix form of vector $\mathbf{w}$) and $R = P^T P$, the multi-label classification problem was formulated as,

$$\begin{aligned} P &= \sum_{r=1}^{k} S_r \\ S_r &= \min_{z,\boldsymbol{\xi}} \frac{1}{2} z_r^T \sum_{m=1}^{k} R_{r,m}^{-1} z_m + C \sum_{i=1}^{l} \xi_{i,r} \\ &\text{s.t. } 2y_{i,r} z_r^T \phi(\mathbf{x}_i) \geq \Delta_r(\mathbf{y}_i, y_{i,r}) - \xi_{i,r} \ \forall i, r \\ &\quad\ \xi_{i,r} \geq \Delta_r(\mathbf{y}_i, y_{i,r}) \ \forall i, r. \end{aligned} \tag{4.16}$$

where z_r is the r^{th} column of Z. This formulation presents many attractive advantages. It has k correlated problems, each having only l constraints and is easier to solve. Second, the labels of a test point $\mathbf{x}$ can be found using $\text{sign}(Z^T \phi(\mathbf{x}))$ unlike the exponential number of evaluations required in the previous case. But, often the multi-label classifiers designed by solving this problem formulation are not sparse and not suitable for the applications which demand fast inference.

We now present our proposed formulation for multi-label classification problem. Similar to the work in [5] we assume that the labels have at most linear correlation and incorporate prior knowledge about the labels using a positive definite matrix $P \in \mathbb{R}^{k \times k}$. Given an input $\mathbf{x}$, the score associated with label $\mathbf{y}$ is given by

$$f(\mathbf{x}, \mathbf{y}) = \mathbf{y}^T P^T \boldsymbol{\alpha}^T K_{J,\cdot} \tag{4.17}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{|J| \times k}$ and its r^{th} column, $\alpha_{J,r}$, stores linear expansion coefficients associated with r^{th} class label and J denotes the basis function set (common for all the labels). Given the matrix P, the multi-label classification problem amounts to finding an appropriate basis function set J and the linear expansion coefficients $\boldsymbol{\alpha}_{J,\cdot}$. This problem can be written as,

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \ & \frac{\lambda}{2} \text{Tr}(\boldsymbol{\alpha}^T K_{J,J} \boldsymbol{\alpha}) \\ & + \sum_{i=1}^{l} \max_{\mathbf{y}} \left(\Delta(\mathbf{y}_i, \mathbf{y}) + (\mathbf{y} - \mathbf{y}_i)^T P^T \boldsymbol{\alpha}^T K_{J,i} \right) \end{aligned} \tag{4.18}$$

where $\Delta(\cdot, \cdot)$ is a multi-label loss function, e.g. the Hamming loss function,

$$\Delta(\mathbf{y}_i, \mathbf{y}_j) = \sum_{r=1}^{k} \Delta_r(\mathbf{y}_i, \mathbf{y}_j) = \sum_{r=1}^{k} (1 - y_{i,r} y_{j,r}).$$

Defining $\boldsymbol{\beta} = \boldsymbol{\alpha} P$ and $R = P^T P$, the problem (4.18) is,

$$\begin{aligned} \min_{\boldsymbol{\beta}} \ & \frac{\lambda}{2} \text{Tr}(P^{-T} \boldsymbol{\beta}^T K_{J,J} \boldsymbol{\beta} P^{-1}) \\ & + \sum_{i=1}^{l} \max_{\mathbf{y}} \left(\Delta(\mathbf{y}_i, \mathbf{y}) + (\mathbf{y} - \mathbf{y}_i)^T \boldsymbol{\beta}^T K_{J,i} \right) \end{aligned} \tag{4.19}$$

Using $\text{Tr}(ABC) = \text{Tr}(CAB)$ and noting that the Hamming loss can be decomposed over the labels, we get the following equivalent problem:

$$\begin{aligned} \min_{\boldsymbol{\beta}} \ & \frac{\lambda}{2} \sum_{r=1}^{k} \sum_{m=1}^{k} R_{rm}^{-1} \beta_{J,r}^T K_{J,J} \beta_{J,m} \\ & + \sum_{i=1}^{l} \max_{\mathbf{y}} \left(\sum_{r=1}^{k} \Delta_r(y_{i,r}, y_r) + (y_r - y_{i,r}) {\beta_{J,r}}^T K_{J,i} \right) \end{aligned} \tag{4.20}$$

Note that the terms inside the maximization are independent over the k components of $\mathbf{y}$. Therefore, interchanging the maximization and summation, the above problem formulation becomes,

$$\begin{aligned} \min_{\boldsymbol{\beta}} \ & \frac{\lambda}{2} \sum_{r=1}^{k} \sum_{m=1}^{k} R_{rm}^{-1} \beta_{J,r}^T K_{J,J} \beta_{J,m} \\ & + \sum_{i=1}^{l} \sum_{r=1}^{k} (\max_{y_r \in \pm 1} \Delta_r(y_{i,r}, y_r) + (y_r - y_{i,r}) {\beta_{J,r}}^T K_{J,i}) \end{aligned} \tag{4.21}$$

For the basis function set J, let us define the misclassification error, $\xi_{i,r} = \max\left(0, 1 - y_{i,r} \beta_{J,r}^T K_{J,i}\right)$. Using quadratic penalization of this misclassification error, we define the new multi-label classification problem as

$$\begin{aligned} \min_{\boldsymbol{\beta}} \ & \frac{\lambda}{2} \sum_{r=1}^{k} \beta_{J,r}^T \sum_{m=1}^{k} R_{r,m}^{-1} K_{J,J} \beta_{J,m} \\ & + \sum_{i=1}^{l} \sum_{r=1}^{k} \left(\max(0, 1 - y_{i,r} {\beta_{J,r}}^T K_{J,i}) \right)^2 \end{aligned} \tag{4.22}$$

This primal problem is a summation of k correlated problems and can be solved directly to design a sparse multi-label classifier. The solution to the above problem is used to find the labels of a novel example $\mathbf{x}$ as the sign of $\boldsymbol{\beta}^T K_{J,\cdot}$. Algorithm 1 can be used to get the basis function set J and linear expansion coefficients $\boldsymbol{\beta}$ using the formulation (4.22). The steps involved are similar to those discussed in the previous section (See Algorithm 4). Hariharan et al. [5] proposed to solve the dual of the problem (4.16) using the optimization techniques for solving the standard SVM dual. Their

Dataset	l	l_{test}	n	k
Iris	150	$*$	4	3
Wine	178	$*$	13	4
Glass	214	$*$	9	7
Vowel	528	462	10	11
Vehicle	846	$*$	18	4
Segment	2310	$*$	19	7
DNA	2000	1186	180	3
Satimage	4435	2000	36	6
Shuttle	10000	14500	9	7
Letter	15000	5000	16	26
USPS	7291	2007	256	10

Table 1: Summary of Datasets (l, l_{test}, n and k denote the number of training examples, test examples, features and classes respectively. $*$ indicates that test data were not available for these data sets. 3-fold cross validation error is reported for these data sets.)

algorithm, however, required some book-keeping to handle the dense structure of the matrix R. Further, the resulting algorithm did not result in a sparse multi-label classifier.

5 Experimental Evaluations

In this section, we discuss the experimental evaluations of proposed sparse multiclass and multi-label classifiers.
Experimental Setup: We used Gaussian kernel function, $K(\mathbf{x}_i, \mathbf{x}_j) \equiv e^{-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2}$ where $\gamma > 0$, for all the experiments. The kernel parameter γ and regularization hyper-parameter λ (in (3.10)) for multiclass classification and (in (4.22)) for multi-label classification were tuned using cross-validation. All the experiments were performed using MATLAB implementations on a 1.7GHz dual quad core machine with 10GB of main memory under Linux.

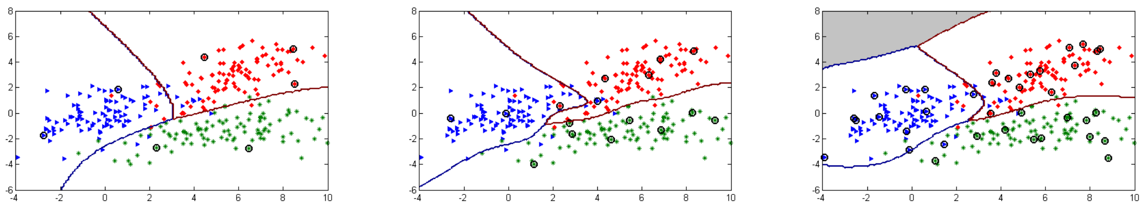
5.1 Multiclass Classification We compare our proposed method, SMSVM, against other sparse and full-model multiclass classifiers on various real-world datasets. Eleven benchmark datasets, summarized in Table 1, were used for this purpose.[3] We report the average cross-validation accuracy for the datasets *iris, wine, glass, vehicle* and *segment* as test data are not available for these datasets. For the dataset *shuttle*, the training set used consisted of 10000 randomly chosen examples from the original training set, keeping the distribution of classes the same as the one in the original training set. We compare the following methods: (1) SMSVM: our proposed sparse multiclass SVM classifiers discussed in Sections 3.1, (2) ImVM: sparse multiclass classifier, Import Vector Machine (ImVM) proposed in [20], (3) Sparse-OVA: Sparse binary classifier in [8], applied to a multiclass problem in OVA form[4], (4) WW: Multiclass SVM proposed by [19] (corresponding to the problem formulation in (2.2)), and (5) CS: Multiclass SVM proposed by [2]. For the full-model classifiers, WW and CS, all the results are reported from [6]. For fair comparison of sparse classifiers, we modified the original ImVM by applying the speed up heuristic of sampling κ random examples for import vector selection as discussed in Section 3.

5.1.1 ***Synthetic Dataset - Decision boundary comparison:*** To compare the decision boundaries and the number of basis vectors used by different sparse classifiers, a synthetic dataset was generated. The training set for two classes ($\triangleright$ and $\Diamond$) was generated using two different Gaussian distributions while the examples for the third class ($*$) were generated using a mixture of Gaussians. The classification boundaries formed by three sparse classifiers, SMSVM, ImVM and Sparse-OVA, are shown in Fig. 1. The points on the boundaries have equal similarity scores (given in (**??**)) for the multiple (two or three) neighboring classes. From Fig. 1, it is clear that SMSVM requires the least number of basis functions to achieve the desired test set accuracy (94.66%). Due to the one-vs-rest nature of Sparse-OVA, the decision boundaries formed by Sparse-OVA are not natural, as compared with those formed by SMSVM and ImVM. Note that in Fig. 1(c), the test points generated (not shown) in the shaded region are classified as belonging to the class of '$*$'.

5.1.2 ***Comparison with respect to sparsity, training and inference time:*** Different multiclass classifiers can be compared with respect to the number of basis vectors and test set performance using the details reported in Table 2. The performance of non-sparse classifiers (WW and CS), given in the last two columns, is reported from [6]. Note that the results for the USPS dataset were not available in [6]. For the sparse classifiers, the number of basis vectors required to achieve the same generalization performance as that given by SMSVM, is reported. If a classifier is not able to achieve the generalization performance achieved by the other classifiers, its best generalization performance along with the number of required basis vectors is reported. For fair comparison, we have reported the number of unique basis vectors across multiple independent binary classifiers in case of Sparse-OVA algorithm. It is clear from this table that the test set performance of the proposed methods is

[3]These datasets are available at
http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html

[4]The program for binary classification, available at http://olivier.chapelle.cc/primal/, was used to design Sparse-OVA classifier.



(a) SMSVM (b) ImVM (c) Sparse-OVA

Figure 1: Decision functions obtained by different sparse classifiers on a synthetic dataset: $l = 300$, $k = 3$, Test Accuracy-94.66%. The number of basis vectors are (a) 7, (b) 16 and (c) 37. In (c), the test points generated (not shown) in the shaded region are classified as belonging to the class denoted by '$*$'.

Dataset	SMSVM		ImVM		Sparse-OVA		WW		CS	
	bv	acc	bv	acc	bv	acc	sv	acc	sv	acc
Iris	**7**	**97.33**	20	**97.33**	12	**97.33**	16	**97.33**	28	**97.33**
Wine	**14**	98.15	15	98.15	19	98.15	55	**98.88**	42	**98.88**
Glass	**19**	**73.83**	80	**73.83**	103	**73.83**	124	71.03	143	71.96
Vowel	**11**	**98.92**	35	**98.92**	55	**98.92**	279	98.49	391	98.67
Vehicle	**54**	86.64	171	86.64	225	86.64	264	86.0	265	**87.76**
Segment	**18**	96.25	103	96.08	316	96.25	358	**97.58**	970	97.32
DNA	**98**	95.03	860	94.01	308	95.03	951	95.61	945	**95.87**
Satimage	**68**	90.10	353	90.10	390	90.10	1426	91.25	2670	**92.35**
Shuttle	**13**	99.74	57	99.74	84	99.74	*	99.91	*	**99.94**
Letter	**180**	94.18	1023	94.18	4327	94.18	7627	**97.76**	6374	97.68
USPS	**83**	**94.32**	5213	**94.32**	767	**94.32**	N.A.		N.A.	

Table 2: Comparison of Multiclass Classifiers. The columns *bv* and *sv* represent the number of the basis vectors and support vectors respectively. The column *acc* represents the test set accuracy. In the case of Sparse-OVA, we report the number of *unique* basis vectors across multiple independent binary classifiers. The results of WW and CS classifiers are reported from [6] and are not available for the USPS dataset. * : **sv** are not mentioned for Shuttle dataset, because WW and CS classifiers use the complete training dataset (43500 training examples) whereas the sparse classifiers use the subset of training dataset (10000 training examples).

Dataset	SMSVM		ImVM		Sparse-OVA	
	Training time	Inference time	Training time	Inference time	Training time	Inference time
Iris	0.65	**0.01**	41.33	**0.01**	**0.23**	**0.01**
Wine	0.89	**0.01**	37.08	**0.01**	**0.58**	**0.01**
Glass	3.99	**0.01**	272.51	0.03	**2.48**	0.03
Vowel	5.60	**0.01**	355.36	0.02	**2.34**	0.03
Vehicle	15.55	**0.01**	1491.73	0.04	**9.84**	0.07
Segment	**9.60**	**0.01**	3231.43	0.06	14.82	0.16
DNA	50.27	**0.15**	24566.84	1.32	**46.17**	0.78
Satimage	88.90	**0.07**	119926.57	0.37	**72.35**	0.68
Shuttle	**15.40**	**0.09**	4572.15	0.38	17.11	0.78
Letter	8437.66	**0.57**	1576352.76	3.13	**608.02**	15.90
USPS	422.71	**0.16**	753789.53	0.97	**255.72**	1.78

Table 3: Time Comparisons of Sparse Multiclass Classifiers (*time in seconds*)

comparable with that of full model (non-sparse) methods. Further, it is achieved using significantly smaller number of basis vectors. From the table, it is also clear that SMSVM requires smaller number of basis vectors to achieve the same test set performance compared to other sparse classifiers like ImVM and Sparse-OVA. The number of basis vectors needed by SMSVM is about an order of magnitude smaller than those needed by Sparse-OVA on the datasets such as Segment, Letter and USPS. On the DNA dataset, ImVM required significantly more number of basis vectors than any of the other sparse classifiers. Thus, SMSVM requires lesser memory compared to other sparse classification methods as smaller number of basis vectors need to be stored. Use of common basis vector set J reduces kernel computations, thereby reducing the inference time.

The training and inference time comparison of the sparse classifiers is reported in Table 3. For every classifier, the training time required to achieve the same test set performance (given in Table 2) is mentioned. From

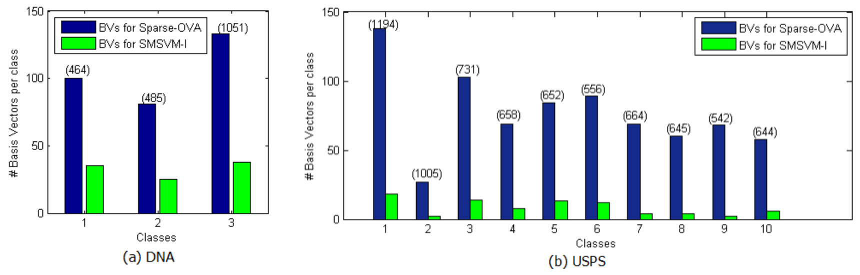


Figure 2: Distribution of basis vectors over classes- (a)DNA (b)USPS datasets. (The numbers in parentheses, on top of the bars, indicate the number of training points belonging to that particular class.)

the results, it is clear that the training of ImVM is very slow. This is mainly due to its non-quadratic (multi-logit regression) formulation. Although, Sparse-OVA is faster than the proposed methods in terms of training, the inference time required by the proposed methods is significantly lesser than that required by Sparse-OVA. This is due to the sparser classification model designed by the proposed methods. Therefore, SMSVM is suitable in applications where sparser classifiers are needed for fast inference, as inference time is directly proportional to the number of unique basis vectors. These methods may require more training time; but this extra training time is worth the effort as training is often a one time process and inference needs to be done multiple times. Thus, SMSVM is a powerful alternative for sparse multiclass classifier design for faster inference.

To compare the inference time of SMSVM and Sparse-OVA on data sets with large test set size, we conducted an experiment on *poker*[5] data set, with 10 classes and 10 features, where training and test set sizes are $25,010$ and $1,000,000$ respectively. The classifier, designed using the proposed SMSVM method, used only 100 basis functions and required 29.6 seconds for inference on all the test set examples. On the other hand, the classifier designed using Sparse-OVA used 1000 basis vectors (907 unique basis vectors) and required 655.2 seconds for the same inference. Although kernel evaluations are to be done only for unique basis vectors, computing the score involves multiplications of coefficients with kernel values for all the basis vectors. The additional kernel evaluations result in significantly slower inference for OVA methods.

5.1.3 *Distribution of basis vectors across different classes:*

To get an idea about the distribution of basis vectors over different classes, we compared SMSVM-I and Sparse-OVA on two datasets and the results are shown in Fig. 2. The values in the parentheses on the top of the bars indicate the number of training set examples belonging to the respective classes. In Sparse-OVA, the binary classifiers designed are independent of each other and hence, often use different sets of basis vectors for design of different classifiers. On the other hand, both the proposed methods use a common set of basis vectors for different classifiers. Therefore, the number of unique basis vectors needed by Sparse-OVA is higher than those needed by SMSVM-I or SMSVM-II. This is also evident from Fig. 2. In all the cases, Sparse-OVA required significantly larger number of basis vectors from every class than SMSVM-I. Further, for the USPS dataset, the number of basis vectors required by Sparse-OVA for all the classes is an order of magnitude higher than those required by SMSVM-I.

5.1.4 *Statistical comparisons of sparse multi-class classifiers:*

In the experiments discussed above, we compared different sparse multiclass classifiers (SMSVM, ImVM and Sparse-OVA) on multiple benchmark datasets, using various criteria such as sparsity, training time and inference time. We now evaluate ranks of these classifiers using the statistical comparison test (Friedman test) discussed in [3] to compare multiple classifiers over multiple datasets. Table 4 presents the results of Friedman test on sparse classifiers. From these results, it is clear that the *p-value* in all the cases is less than or equal to 0.0002. Hence, according to Friedman test, the null hypothesis (that the classifiers are comparable with each other) is rejected. Further, SMSVM is ranked *first* according to the sparsity and the inference time criteria and is ranked *second* according to the training time criterion. Though, Sparse-OVA ranks first as per the training time criterion, it does not use any systematic approach to select common basis vectors across multiple classes and often does not provide natural multiclass classification boundary (details available in the supplementary material). Thus, considering var-

[5]Poker dataset is available at
http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html

Criterion	Average Ranks			p-value
	SMSVM	ImVM	Sparse-OVA	
Sparsity	**1**	2.27	2.73	< 0.0001
Training time	1.82	3	**1.36**	< 0.0001
Inference time	**1.18**	1.95	2.77	0.0002

Table 4: Results of Friedman test on sparse multiclass classifiers

Name	$l_{training}$	l_{test}	n	k
Yeast	1500	917	103	14
Scene	1211	1196	294	6
Mediamill	5000	12914	120	101
Animal	5000	6180	256	85
fMRI-Word	2592	648	28244	25

Table 5: Summary of Multi-label Benchmark Datasets : $l_{training}$, l_{test}, n and k denote the number of training examples, test examples, attributes and classes respectively.

ious evaluation criteria, SMSVM is preferred for sparse multiclass classifier design.

5.2 Multi-label Classification: We now compare the proposed sparse max-margin multi-label method (described in Section 4) against the full model max-margin multi-label classifier [5] on various real-world benchmark datasets. The datasets are summarized in Table 5. We compared the following methods: (1) SM3L: our proposed sparse multi-label classifier without using the prior knowledge about label correlations, (2) M3L: Max-margin multi-label classifier in [5] (corresponding to the problem formulation in (4.16)) without using the prior knowledge about label correlations, (3) SM3L-P: our proposed sparse multi-label classifier which incorporates the prior knowledge about label correlations P, and (4) M3L-P: Max-margin multi-label classifier in [5] (corresponding to the problem formulation in (4.16)) which incorporates the prior knowledge about label correlations, P.

5.2.1 ***Usefulness of Common Basis Vectors:*** To demonstrate the usefulness of common basis vectors in a multi-label problem setting where the training and test set label correlations are similar or different, we conducted an experiment. Subsets of three datasets containing only two labels with different training and test set label correlations were obtained. Characteristics of these datasets are given in Table 6. We compared the performance of SM3L, M3L, SM3L-P and M3L-P over these datasets, on the basis of test set Hamming loss and the number of basis vectors, as shown in Table 6. As the experimental results suggest, the sparse classifiers SM3L and SM3L-P are comparable with their full-model (non-sparse) counterparts, M3L and M3L-P, with respect to the generalization performance. Further, SM3L and SM3L-P use lesser number of basis vectors than M3L and M3L-P, to achieve the same generalization performance. This demonstrates that, as in the case of sparse multiclass classifier design, the common basis vectors in SM3L and SM3L-P are sufficient to model the multi-label classifiers under the widely varying label correlation conditions.

5.2.2 ***Sparsity and test set performance:*** We compared the performance of sparse classifiers, SM3L and SM3L-P, and full model classifiers, M3L and M3L-P, on five multi-label benchmark datasets (Table 7). It is evident from this table that the Hamming loss performance improves if the label correlations are used, as observed in [5]. Further, it is observed that the test set performance (Hamming loss) of the SM3L and SM3L-P methods are comparable with their full model counterparts. Also, SM3L and SM3L-P result in sparser classifiers. The number of basis vectors needed by the proposed sparse classifiers to achieve the same generalization performance is an order of magnitude smaller than those needed by their full model counterparts on many data sets.

5.2.3 ***Zero-shot Learning:*** There are practical problems in the field of computer vision or image search, where the classifier is required to recognize example instances from previously unseen test categories. These problems are referred to as zero-shot learning problems. A zero-shot learning problem can be solved by using knowledge about common attributes among different example categories [5]. The classifiers can be built based on common attributes in the training set categories. These learnt classifiers can then be used to identify the same attributes for the instances in unseen test categories. The learning problem can be posed as a multi-label classification problem where the attributes are treated as labels. But, in this framework, the training and test set label correlations are often very different.

We compared the performance of four multi-label classifiers discussed earlier on two real-world zero-shot learning datasets, Animal and fMRI-Words data described in Table 8. In Animal dataset [9], 50 different animals like polar bear, zebra, horse etc. are described using 85 common attributes like ‘is black’, ‘have stripes’ etc. The examples (images) of 40 animal categories were used for training and the examples of remaining 10 animal categories were used for testing. We used only 200 examples for training in this experiment. Similarly, in fMRI-Words dataset [14], 60 different words like ‘celery’, ‘airplane’, ‘hammer’ etc., are described using 25 common verbs like ‘eat’, ‘taste’ or ‘fill’. The fMRI examples from 48 words were used for training and the examples from the remaining 12 words were used for

Name	Label Pair	$R_{training}$	R_{test}	SM3L-P		M3L-P		SM3L		M3L	
				bv	H.L.	bv	H.L.	bv	H.L.	bv	H.L.
Animal	1 : 43	0.2000	-0.7660	**23**	**20.87**	179	**20.87**	27	30.01	183	30.01
Mediamill	45 : 68	0.9188	-0.5293	**13**	**1.41**	811	**1.41**	**13**	1.69	798	1.67
Mediamill	48 : 49	0.9356	1	**8**	**0.98**	1011	**0.98**	9	1.01	989	1.01
Scene	2 : 3	0.4021	0.3328	39	**19.11**	1197	**19.11**	**37**	**19.11**	1201	**19.11**
Scene	4 : 6	0.3079	0.2742	**43**	18.75	1206	**18.73**	47	18.76	1197	18.73

Table 6: Results of multi-label datasets with label pairs : $R_{training}$ and R_{test}, **bv** and **H.L.** denote the number of training correlation, test correlation, number of basis vectors and test set Hamming loss respectively. Note the improvement in the Hamming loss when the training and test set correlations are significantly different.

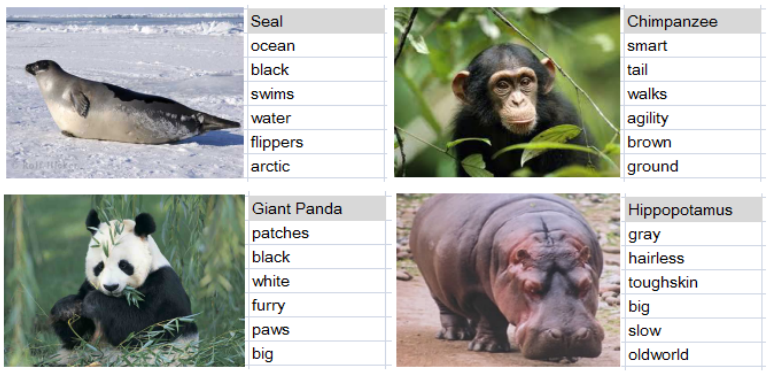


Figure 3: Results of SM3L-P on Animal Dataset: Important class labels for four test set animal categories are listed, as provided by SM3L-P classifier.

Name	SM3L-P		M3L-P		SM3L		M3L	
	bv	H.L.	bv	H.L.	bv	H.L.	bv	H.L.
Yeast	**183**	18.78	1474	**18.65**	198	18.71	1468	18.67
Scene	**41**	**17.61**	1211	**17.61**	44	17.63	1215	17.62
Mediamill	98	**3.16**	4991	**3.16**	**95**	**3.16**	4997	3.17
Animal	36	26.31	5000	**26.11**	**24**	29.02	4713	29.02
fMRI-Word	18	**49.37**	2137	**49.37**	21	55.31	2587	55.31

Table 7: Results of Multi-label Benchmark Datasets : bv and H.L. denote the number of basis vectors and test set Hamming loss respectively.

Name	$l_{training}$	$c_{training}$	l_{test}	c_{test}	n	k
Animal	200	40	6180	10	256	85
fMRI-Words	400	48	648	12	28244	25

Table 8: Summary of Zero-shot Learning Datasets: $l_{training}$, $c_{training}$, l_{test}, c_{test} n and k denote the number of training examples, training categories, test examples, test categories, features and classes respectively.

Name	SM3L-P		M3L-P		SM3L		M3L	
	bv	H.L.	bv	H.L.	bv	H.L.	bv	H.L.
Animal	**24**	**26.62**	200	**26.62**	**24**	29.31	193	29.31
fMRI-Words	**15**	**51.05**	400	**51.05**	18	56.87	400	56.87

Table 9: Results on Zero-shot Learning Datasets : bv represents the number of basis vectors and H.L. represents the test set Hamming Loss.

testing.

Table 9 presents the results of different multi-label classifiers on these datasets with respect to the number of basis vectors and test set Hamming loss. It is observed that the sparse classifiers, SM3L-P and SM3L achieve the same Hamming loss as compared to the corresponding full model classifiers, M3L-P and M3L respectively, using a smaller number of basis vectors. Further, the test set Hamming loss decreases significantly in SM3L-P and M3L-P methods, due to the incorporation of prior knowledge about the test set label correlations. It is clear from Table 9 that the proposed sparse multi-label classification algorithm results in sparser classifiers compared to those designed using M3L or M3L-P methods. The number of basis vectors required is observed to be an order of magnitude smaller than those required by M3L or M3L-P.

6 Conclusion

There has been an increasing interest in the machine learning community to design multiclass and multi-label classifiers for online applications where real-time inference is needed. In this work, we suggested new problem formulations and efficient algorithms to build sparse multiclass and multi-label classifiers. The proposed algorithms are fast and scalable. Further, we demonstrated that the resulting classifiers achieved nearly the same generalization performance as that achieved by full model or other sparse classifiers. On many data sets it is observed that the proposed algorithms result in using significantly smaller number of basis vectors compared to other sparse classifiers. Thus, the proposed algorithms are recommended for sparse multiclass and multi-label classifier design when fast inference is needed. We are currently investigating the extension of these ideas to other problems like structured output prediction problems. Sparse classifier design using other measures like F-measure will also be investigated in future.

References

[1] E. J. Bredensteiner and K. P. Bennett, *Multicategory classification by support vector machines*, *Computational Optimization and Applications,*, 12 (1999), pp. 53–79.

[2] K. Crammer and Y. Singer, *On the algorithmic implementation of multiclass kernel-based vector machines*, *Journal of Machine Learning Research,*, 2 (2001), pp. 265–292.

[3] J. Demsar, *Statistical comparisons of classifiers over multiple data sets*, Journal of Machine Learning Research, 7 (2006), pp. 1–30.

[4] Theodoros Evgeniou and Massimiliano Pontil, *Regularized multi-task learning*, in Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '04, 2004, pp. 109–117.

[5] B. Hariharan, S. V. N. Vishwanathan, and M. Varma, *Efficient max-margin multi-label classification with applications to zero-shot learning*, Machine Learning, 88 (2012), pp. 127–155.

[6] C. W. Hsu and C.J. Lin, *A comparison of methods for multiclass support vector machines*, *IEEE Transactions on Neural Networks,*, 13 (2002), pp. 415–425.

[7] Thorsten Joachims and Chun-Nam John Yu, *Sparse kernel svms via cutting-plane training*, Machine Learning, 76 (2009), pp. 179–193.

[8] S. S. Keerthi, O. Chapelle, and D. Decoste, *Building support vector machines with reduced classifier complexity*, *Journal of Machine Learning Research,*, 7 (2006), pp. 1493–1515.

[9] C. H. Lampert, H. Nickisch, and S. Harmeling, *Learning to detect unseen object classes by between-class attribute transfer*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[10] N. D. Lawrence, M. Seeger, and R. Herbrich, *Fast sparse Gaussian process methods: The informative vector machine*, in *Advances in Neural Information Processing Systems (NIPS)*, vol. 15, 2002, pp. 609–616.

[11] Sangkyun Lee and Stephen J. Wright, *Sparse nonlinear support vector machines via stochastic approximation*, in Proceedings of ACM SIGKDD international conference on Knowledge discovery and data mining, 2010.

[12] Y. Lee, Y. Lin, and G. Wahba, *Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data*, *Journal of the American Statistical Association,*, 99 (2004), pp. 67–81.

[13] O. L. Mangasarian, *A finite Newton method for classification*, *Optimization Methods and Software,*, 17 (2002), pp. 913–929.

[14] T. Mitchell, S. Shinkareva, A. Carlson, K.-M. Chang, V. Malave, R. Mason, and A. Just, *Predicting human brain activity associated with the meanings of nouns*, Science, 320 (2008), pp. 1191–1195.

[15] I. Psorakis, T. Damoulas, and M. A. Girolami, *Multiclass relevance vector machines: Sparsity and accuracy*, *IEEE Transactions On Neural Networks,*, 21 (2010), pp. 1588–1598.

[16] I. Steinwart, *Sparseness of support vector machines - some asymptotically sharp bounds*, in Proceedings of the 16th NIPS Conference, 2004.

[17] J. Sun, *On piecewise quadratic newton and trust region problems*, *Mathematical Programming,*, 76 (1997), pp. 451–467.

[18] G. Tsoumakas and I. Katakis, *Multi-label classification: An overview*, *International Journal of Data Warehousing and Mining,*, 3 (2007), pp. 81–113.

[19] J. Weston and C. Watkins, *Support vector machines for multi-class pattern recognition*, in *Proceedings of the 7th European Symposium On Artificial Neural Networks*, vol. 4, 1999, pp. 219–224.

[20] J. Zhu and T. Hastie, *Kernel logistic regression and the import vector machine*, *Journal of Computational and Graphical Statistics*, 14 (2005), pp. 185–205.