
The Linear Programming Set Covering Machine

Zakria Hussain

ISIS Research Group
School of Electronics and Computer Science
University of Southampton
Southampton, SO17 1BJ, UK
zh03r@ecs.soton.ac.uk

Sandor Szedmak

ISIS Research Group
School of Electronics and Computer Science
University of Southampton
Southampton, SO17 1BJ, UK
ss03v@ecs.soton.ac.uk

John Shawe-Taylor

ISIS Research Group
School of Electronics and Computer Science
University of Southampton
Southampton, SO17 1BJ, UK
jst@ecs.soton.ac.uk

Abstract

The Set Covering Machine (SCM) was introduced by Marchand & Shawe-Taylor [6, 7] in which a minimum set cover of a class of examples was approximated to find a compact conjunction/disjunction of features for classification. Their approach was to solve the set cover problem using the *greedy* algorithm. In this paper we introduce an alternative method of solving the SCM by formulating it as a Linear Programme (LP). In this setting we can apply an LP solver to give us our set of data-dependent features and use a convex combination of these features in order to classify unseen data for both the conjunction and disjunction case. Our hope is to approximate better solutions to the set cover problem using an LP as opposed to the greedy method approach evaluated in [6, 7]. The LP formulation is motivated by the LPBoost algorithm and so we also apply boosting algorithms, LPBoost and AdaBoost, to our set of features in order to compare our results with the original SCM and Support Vector Machine(SVM) classifiers.

1 Introduction

The Set Covering Machine (SCM) evolved from work carried out by Valiant [9] and Haussler [5] during the 1980's. The former introduced a class of boolean valued functions called monomials. Within this class of functions he developed a learning algorithm known as The Standard Monomial Learning algorithm. Several years later David Haussler redefined Valiant's algorithm by showing that it could be reduced to the Minimum Set Cover problem famous in Combinatorial Optimisation. Although the problem is known to be NP-Complete there exists a *greedy algorithm* with a very good worst-case lower bound [2]. The idea involves creating a small conjunction/disjunction of monomials in order to classify unseen data. However, both algorithms suffer in two major areas: (1) they can only be

defined within the set of boolean valued functions (for example monomials) and so suffer from impracticality; (2) They are not immune to noisy data sets which make them unlikely candidates *even* for binary valued machine learning tasks.

Marchand & Shawe-Taylor [6, 7] improved upon Haussler’s algorithm by applying a technique known as data-dependent features in order to allow the algorithm to be used with real world data sets. They also overcame the problem of dealing with noisy data sets by introducing the idea of a penalty term p which would punish an example being misclassified (from the set not being covered). The algorithm also included a soft stopping criteria s which allowed the classifiers to be stopped early for better generalisation purposes (Occam’s Razor). The results given in the original set covering machine paper were encouraging and showed good competitiveness against the SVM algorithm and the Nearest Neighbour Classifier (NNC).

In this paper we will reformulate the SCM problem into a Linear Programme (LP) using ideas from LP boosting techniques given in [3]. We will show that our formulation is a soft set covering machine where the constraints of the set cover problem are loosened somewhat by the use of slack variables. Section 2 will describe the SCM algorithm, followed by a brief description of boosting using data-dependent balls as its set of weak learners in section 3. Next we will formulate the SCM problem into LP problems in section 4. Section 5 will describe the experimental results and we shall then conclude with section 6.

2 The Set Covering Machine

The Set Covering Machine (SCM) aims to produce a compact conjunction/disjunction of features in order to classify unseen data. This is very different to the idea behind Support Vector Machines (SVM) which tries to maximise the margin of the decision function. However it can be shown that many learning tasks are of a nature whereby it is more important to create a small number of relevant features to represent them, as opposed to a maximum margin approach. We will now give a brief explanation of the SCM algorithm that uses *data-dependent balls* as its set of features.

Given a sample $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$ where $x_i \in X$ and $y_i \in \{-1, 1\}$, and a training set $X = \mathcal{P} \cup \mathcal{N}$ where \mathcal{P} is the set of *positive* training examples and \mathcal{N} the set of *negative* training examples such that

$$y_i = \begin{cases} 1 & \text{if } x_i \in \mathcal{P} \\ -1 & \text{if } x_i \in \mathcal{N} \end{cases}$$

and a set of functions $\mathcal{H} = \{h_j | h_j : X \rightarrow \{0, 1\}, j = 1, \dots, n, n = |\mathcal{H}|\}$ where \mathcal{H} is closed under its complement such that

$$\text{if } h_j \in \mathcal{H} \text{ then } \bar{h} = 1 - h_j \in \mathcal{H};$$

then our SCM classifier (hypothesis) will return a small subset $\mathcal{R} \subset \mathcal{H}$ of features, defined as a function $f(x)$ such that

$$f(x) = \begin{cases} \bigwedge_{j \in \mathcal{R}} h_j(x) & \text{for conjunction SCM} \\ \bigvee_{j \in \mathcal{R}} h_j(x) & \text{for disjunction SCM} \end{cases}$$

For the remainder of this paper we will use the following definition.

Definition 2.1 *Let \mathcal{P} be the set of positive (negative) training examples for the SCM constructing a conjunction (disjunction) of features from \mathcal{R} . Also, let \mathcal{N} be the set of negative (positive) training examples used for the SCM constructing a conjunction (disjunction) of features from \mathcal{R} .*

The SCM will try to cover as many negative (positive) training examples as is possible from the set \mathcal{N} . However a misclassification of \mathcal{P} examples must be accounted for. More formally let $|C|$ be the number of \mathcal{N} training examples covered by some feature h and let $|M|$ be the number of \mathcal{P} examples misclassified by h . The Utility (Usefulness) function is

$$U = |C| - p \times |M|$$

where the *penalty* parameter $p \in \mathcal{R}^+$. This definition allows us to find the feature that gives us the highest “*usefulness*” value rather than the highest coverage of a feature. Instead of picking a feature that covers the maximal number of \mathcal{N} examples we will now choose features that give us maximal value for U . This utility function now allows for errors occurring in the training set to be observed more accurately. Furthermore the SCM contains a “*soft stopping*” parameter s that stops the SCM early, so that a smaller conjunction/disjunction of features is produced, in the hope that it will generalise better than a more complex SCM.

Initially The SCM algorithm will take an empty hypothesis and look for features that cover the maximal number of \mathcal{N} examples by choosing features that yield maximal utility U . This greedy step will be repeated until there are no more \mathcal{N} examples left to cover or when s features have been chosen. In this way we are able to control the accuracy and complexity (size) of our classifiers generated by the SCM.

The set of features that we shall use for the SCM will be the set of data-dependent balls introduced in [6]. A formal definition follows.

Definition 2.2 For a training example x_i with label $y_i \in \{-1, 1\}$ and (real-valued) radius ρ , we define feature $h_{i,\rho}$ to be the following data-dependent ball centred on x_i :

$$h_{i,\rho}(x) = \begin{cases} y_i & \text{if } d(x, x_i) \leq \rho \\ \bar{y}_i & \text{otherwise} \end{cases}$$

where \bar{y}_i is the complement of y_i and $d(x, x_i)$ is the distance between x and x_i .

We will allow ball *centres* x_i to be defined by every example in our training set however a ball *border* x_j will only be defined from the set of \mathcal{P} examples. The radius

$$\rho = \begin{cases} d(x_i, x_j) + \epsilon & \text{if } x_i \in \mathcal{P} \\ d(x_i, x_j) - \epsilon & \text{if } x_i \in \mathcal{N} \end{cases}$$

where ϵ is a small positive real number.

3 Boosting of “weak” data-dependent balls

In recent years the idea of boosting has been researched in great detail and a number of algorithms have been developed. Boosting is a general purpose method of taking some “weak learners” and boosting their performance on the training data. The idea stems from the question of whether or not a classifier that performs slightly better than random guessing can be boosted into a strong learning rule. Let h_1, \dots, h_n be the set of “weak” hypotheses such that a combination of them can be created

$$f(x) = \sum_{j=1}^n a_j h_j(x), \quad \text{where } \sum_{j=1}^n a_j = 1, \quad a_j \geq 0, \quad j = 1, \dots, n$$

and so a_j is the weight of “importance” put on hypothesis h_j when classifying example x . Hence in boosting we would like to take a linear combination (ensemble) of weak learners (hypotheses) and boost their performance. In general these boosted algorithms can perform very well on real world data sets and are the focus of research in many machine learning

tasks. Our interest in these algorithms is that they can be defined in a game theoretic framework which in turn can exploit Linear Programming techniques. This is where our interest lies for this paper and so we shall now give a brief description of the set of weak learners used by both the AdaBoost and LPBoost algorithms in order to compare the results of our LP solution of the SCM using data-dependent balls as its set of features.

3.1 AdaBoost

The AdaBoost algorithm (Freund & Schapire [4]) was one of the first boosting algorithms to be developed. It calls a weak (base) learning algorithm for a number of rounds T and at each iteration calculates the new weight a_j for weak learner h_j using a gradient descent method. The set of weak learners we will use with AdaBoost will be the set of data-dependent balls described in definition 2.2.

3.2 LPBoost

Here is a formulation of the boosting problem in a linear programming framework from [3] and [8]:

$$\begin{aligned} \min_{\rho, a, \xi} \quad & \rho + C \sum_{i=1}^m \xi_i \\ \text{subject to} \quad & y_i \sum_{j=1}^n h_{ij} a_j + \xi_i \geq \rho, \quad i = 1, \dots, m, \\ & \sum_{j=1}^n a_j = 1, \\ & \xi_i \geq 0, a_j \geq 0, \quad j = 1, \dots, n \end{aligned} \quad (1)$$

where $C > 0$. The dual LP can be expressed in the form

$$\begin{aligned} \max_{\beta, u} \quad & \beta \\ \text{subject to} \quad & \sum_{i=1}^m u_i y_i h_{ij} \leq \beta, \quad j = 1, \dots, n \\ & \sum_{i=1}^m u_i = 1, \\ & 0 \leq u_i \leq C, \quad i = 1, \dots, m, \end{aligned} \quad (2)$$

4 Linear Programming Formulation

In this section we will derive the LP formulation of the SCM, firstly, by setting up a hard case problem where the constraints will restrict us to an Integer Programming (IP) solution. By way of loosening these constraints, using slack, we will show the soft set covering machine formulation that can be solved using an LP. However, as mentioned earlier the set covering machine estimates a maximal set covering of a class of examples and so to motivate our soft SCM we will now give a standard IP formulation of the set cover problem.

4.1 The set cover problem

Given a set of points $X = \{x_1, \dots, x_m\}$, a finite set of subsets $V = \{V_1, \dots, V_n\}$ satisfying the following conditions

$$V_j \cap X \neq \emptyset \quad \text{and} \quad X \subseteq \bigcup_{j=1}^n V_j \quad \text{where} \quad j = 1, \dots, n$$

and a matrix $H = \{h_{ij}\}$ where

$$h_{ij} = \begin{cases} 1 & \text{if } x_i \in V_j \\ 0 & \text{otherwise} \end{cases}$$

then the *set cover problem* is defined as the minimum cardinality subset R of V such that all points of X are covered.

The Integer Programming (IP) formulation of the set cover problem is

$$\begin{aligned} \min & \sum_{j=1}^n z_j \\ \text{subject to} & \sum_{j=1}^n h_{ij} z_j \geq 1, \quad i = 1, \dots, m \\ & z_j \in \{0, 1\}, \quad j = 1, \dots, n \end{aligned} \quad (3)$$

where z_j indicates whether or not the set V_j occurs in the minimum cardinality subset R . The objective function minimises the number of subsets in R while the constraint ensures a subset V_j covers at least one point in X .

4.2 The Integer Programming SCM

From earlier let our sample $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$. Also let $h_j(x_i) = h_{ij}$ where $h_j \in \{0, 1\}$ is our feature (ball) and x_i our example. Now that we have the building block for our IP we can now redefine the constraints from 4.1 to include the SCM in our problem as follows.

For the *SCM using a disjunction of features* we have

$$\begin{aligned} \min & \sum_{j=1}^n z_j \\ \text{subject to} & \sum_{j=1}^n h_{ij} z_j \geq 1, \quad i = \mathcal{I}_+ \\ & \sum_{j=1}^n h_{ij} z_j = 0, \quad i = \mathcal{I}_- \\ & z_j \in \{0, 1\}, \quad j = 1, \dots, n \end{aligned} \quad (4)$$

where z_j indicates whether or not feature h_j is included in our hypothesis $\mathcal{R} \subset \mathcal{H}$ and \mathcal{I}_+ and \mathcal{I}_- are the set of indices from the set \mathcal{P} and \mathcal{N} , respectively. The objective function minimises the number of features in \mathcal{R} subject to at least one \mathcal{P} example being covered and zero \mathcal{N} examples being misclassified.

The *SCM using a conjunction of features* gives:

$$\begin{aligned} \min & \sum_{j=1}^n z_j \\ \text{subject to} & \sum_{j=1}^n h_{ij} z_j \geq 1, \quad i = \mathcal{I}_- \\ & \sum_{j=1}^n h_{ij} z_j = 0, \quad i = \mathcal{I}_+ \\ & z_j \in \{0, 1\}, \quad j = 1, \dots, n \end{aligned} \quad (5)$$

again the objective function minimises the number of features in \mathcal{R} subject to at least one \mathcal{N} example being covered and zero \mathcal{P} examples being misclassified. Note that the disjunction and conjunction IP's are equivalent but with the indices \mathcal{I}_+ switched to \mathcal{I}_- from the first inequalities and indices \mathcal{I}_- switched to \mathcal{I}_+ from the second.

We can make our IP formulations more compact by taking $H_+ = \{h_{ij} | i \in \mathcal{I}_+\}$ and $H_- = \{h_{ij} | i \in \mathcal{I}_-\}$ which are the matrices containing the positive and negative examples respectively. Let $z = \{z_1, \dots, z_n\}$ be the vector containing the selection variables of a particular feature h_j and let $e = (1, \dots, 1)$ be a column vector of size n .

The *SCM using a disjunction of features* can be reformulated as follows:

$$\begin{aligned} \min_z & e^T z \\ \text{subject to} & H_+ z \geq 1, \\ & H_- z = 0, \\ & z \in \{0, 1\}^n, \end{aligned} \quad (6)$$

and the *SCM using a conjunction of features* can also be reformulated as:

$$\begin{aligned} \min_z & e^T z \\ \text{subject to} & H_- z \geq 1, \\ & H_+ z = 0, \\ & z \in \{0, 1\}^n, \end{aligned} \quad (7)$$

However both these formulations are IP problems which are known to be NP-hard and so we would like to somehow relax the constraints in order to form a Linear Programming formulation of the SCM.

4.3 The Linear Programming SCM

In this soft case we allow the constraints of the SCM to be fathomed somewhat but include penalty parameter D to account for the margin of error. As above we will state first the disjunction case followed by the conjunction case.

The *soft SCM using a disjunction of features* gives:

$$\begin{aligned} \min_{z,\xi,\eta} \quad & e^T z + De^T \xi + De^T \eta \\ \text{subject to} \quad & H_+ z \geq 1 - \xi, \\ & H_- z \leq 0 + \eta, \\ & z \in \{0, 1\}^n, \xi \geq 0, \eta \geq 0 \end{aligned} \quad (8)$$

the *soft SCM using a conjunction of features* gives:

$$\begin{aligned} \min_{z,\xi,\eta} \quad & e^T z + De^T \xi + De^T \eta \\ \text{subject to} \quad & H_- z \geq 1 - \xi, \\ & H_+ z \leq 0 + \eta, \\ & z \in \{0, 1\}^n, \xi \geq 0, \eta \geq 0 \end{aligned} \quad (9)$$

Where ξ and η are our slack variables that allow classification in a non-separable case. To avoid the integer optimisation problem we relax the constraint $z \in \{0, 1\}^n$ to $z \geq 0$ instead. We will now use substitutions

$$e^T z = \frac{1}{\rho}, \quad a = \rho z$$

in order to derive an LP formulation similar to that given in LPBoost in section 3.2.

The LP solution for the *soft SCM using a disjunction of features* can now be given as:

$$\begin{aligned} \max_{\rho,\xi,\eta} \quad & \rho - De^T \xi - De^T \eta \\ \text{subject to} \quad & H_+ a \geq \rho - \xi, \\ & H_- a \leq 0 + \eta, \\ & e^T a = 1, \\ & a \geq 0, \xi \geq 0, \eta \geq 0, \end{aligned} \quad (10)$$

from symmetry we have the LP of the *soft SCM using a conjunction of features*:

$$\begin{aligned} \max_{\rho,\xi,\eta} \quad & \rho - De^T \xi - De^T \eta \\ \text{subject to} \quad & H_- a \geq \rho - \xi, \\ & H_+ a \leq 0 + \eta, \\ & e^T a = 1, \\ & a \geq 0, \xi \geq 0, \eta \geq 0. \end{aligned} \quad (11)$$

Finally we have our LP SCM which is our SCM formulated as a Linear Programming problem and so we can now apply a Linear Programme solver in order to generate our subset of data-dependent balls which will now be used as a convex combination in order to classify unseen data. So now the LP SCM classifier $f(x)$ will classify data like so:

$$f(x) = \begin{cases} 1 & \text{if } Ha \geq t \\ -1 & \text{otherwise} \end{cases}$$

where $H = \begin{pmatrix} H_+ \\ H_- \end{pmatrix}$, $t \geq 0$ and learned on training set X .

5 Experimental results

Experiments were conducted on the same data sets used by Marchand and Shawe–Taylor in [6, 7] where all examples with contradictory labels were removed as well as features containing unknown values. These modifications were also carried out by Marchand and Shawe–Taylor using the original data sets taken from the UCI repository [1].

We conducted 10–fold cross validation on all data sets using the SCM, AdaBoost, LPBoost and LP SCM algorithms. The NNC and SVM results reported in table 1 are those given in [7]. The SVM was equipped with a radial basis kernel γ for which the values have not been reported here. The values for C have also not been reported for the SVM as the reader is referred to [7] for both these values. We have only reported the size of the classifiers (the number of support vectors) on average for the 10–fold cross validation procedure. The error reported is the sum of errors from each test set within 10–fold cross validation.

The results of the SCM in Table 1 were those that gave the smallest empirical error on the data from an *exhaustive scan of many values for penalty parameter p* . The type column is the type of machine that gave the smallest error where a $c = \textit{conjunction}$ machine and a $d = \textit{disjunction}$ machine. The s column refers to the soft stopping parameter and so is the size of the classifier returned by the SCM. The error is again the empirical error on all test sets during 10–fold cross validation.

Table 2 shows the results of AdaBoost, LPBoost and LP SCM. The results of the boosting algorithms, AdaBoost and LPBoost, are those for which the base learner was the set of data–dependent balls described in section 2.2. For AdaBoost the T column is the number of rounds that the base learning algorithm was called upon and so corresponds to the size of the classifier which gave the smallest generalisation error from 10–fold cross validation. The size columns for LPBoost and LP SCM refer to the average size given over the cross validation process and so these are not always integers as these algorithms do not have an explicit stopping criterion defined.

The results of the AdaBoost algorithm show good competitiveness with both the SCM and SVM algorithms and actually give smaller generalisation error for the Votes, Haberman and Credit data sets. The most significant result is that of Haberman where we achieve nearly a 20% improvement on the error in comparison to the SVM and SCM. AdaBoost also shows very good sparsity where most data sets can be classified by boosting just 1 data–dependent ball.

LPBoost shows good competitive behaviour with the NNC, SVM and SCM and has smaller generalisation error for the Credit data set. The LP SCM is also competitive and gives approximately a 40% reduction in overall error for the Credit data set against the SVM and SCM algorithms. It can also be seen that the size of the classifiers are quite large (complex) for the LPBoost and LP SCM classifiers in comparison to the SCM algorithm which generates smaller (simple) classifiers.

6 Conclusions

We have formulated a linear programming problem of the set covering machine (LP SCM) that can be used with the set of data–dependent balls in order to classify new data. The results for the LP SCM, AdaBoost and LPBoost using data–dependent balls as features are promising and show good competitive behaviour against more famous algorithms such as NNC and SVM’s. Clearly the LP SCM is a good alternative to the greedy method approach of the original SCM and so should be the subject of further research.

Data Set	no. of ex		NNC	SVM (finite C')		SCM (finite s, p)		
	pos	neg	error	size	error	type	s	error
BreastW	239	444	29	57.7	19	c	2	15
Votes	18	34	7	18.2	3	c	1	6
Pima	269	499	247	526.2	203	c	3	189
Haberman	219	75	107	146.4	71	d	11	72
Bupa	145	200	124	265.9	107	d	9	108
Glass	87	76	36	91.8	34	c	5	31
Credit	296	357	214	423.2	190	d	4	194

Table 1: NNC, SVM and SCM results for 10-fold cross validation

Data Set	AdaBoost		LPBoost		LP SCM	
	T	error	size	error	size	error
BreastW	1	17	2.8	17	36.8	21
Votes	2	2	1	8	2	5
Pima	1	217	38.5	219	67.6	208
Haberman	1	59	14.2	79	24.8	81
Bupa	1	116	20.2	136	2.6	130
Glass	1	32	7	51	19	39
Credit	2	189	58	152	70.2	116

Table 2: AdaBoost, LPBoost and LP SCM results for 10-fold cross validation

References

- [1] C.L. Blake and Merz C.J. UCI repository of machine learning databases, 1998.
- [2] V. Chvátal. A greedy heuristic for the set covering problem. *Mathematics of Operations Research*, 4:233–235, 1979.
- [3] A. Demiriz, K.P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Journal of Machine Learning Research*, 46(13):225–254, 2002.
- [4] Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *Thirteenth International Joint Conference on Machine Learning*, 1996.
- [5] D. Haussler. Quantifying inductive bias: AI learning algorithms and valiant’s learning framework. In *Artificial Intelligence*, 36:177–221, 1988.
- [6] M. Marchand and J. Shawe-Taylor. Learning with the set covering machine. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 345–352, 2001.
- [7] M. Marchand and J. Shawe-Taylor. The set covering machine. *Journal of Machine Learning Research*, 2002.
- [8] G. Rätsch, B. Schölkopf, A. Smola, S. Mika, T. Onoda, and K. R. Müller. Robust ensemble learning. In *Advances in Large Margin Classifiers*, pages 208–222. Cambridge, MA. MIT Press, 1999.
- [9] L.G. Valiant. A theory of the learnable. In *Communications of the Association of Computing Machinery*, 27(11):1134–1142, November 1984.