A Pseudo-Boolean Set Covering Machine

Pascal Germain, Sébastien Giguère, Jean-Francis Roy, Brice Zirakiza, François Laviolette, and Claude-Guy Quimper

Département d'informatique et de génie logiciel, Université Laval, Québec, Canada
{sebastien.giguere.8, jean-francis.roy.1, brice.zirakiza.1}@ulaval.ca,
{pascal.germain, francois.laviolette, claude-guy.quimper}@ift.ulaval.ca

Abstract. The Set Covering Machine (SCM) is a machine learning algorithm that constructs a conjunction of Boolean functions. This algorithm is motivated by the minimization of a theoretical bound. However, finding the optimal conjunction according to this bound is a combinatorial problem. The SCM approximates the solution using a greedy approach. Even though SCM seems very efficient in practice, it is unknown how it compares to the optimal solution. To answer this question, we present a novel pseudo-Boolean optimization model that encodes the minimization problem. It is the first time a Constraint Programming approach addresses the combinatorial problem related to this machine learning algorithm. Using that model and recent pseudo-Boolean solvers, we empirically show that the greedy approach is surprisingly close to the optimal.

1 Introduction

Machine learning [2] studies algorithms that "learn" to perform a task by observing examples. In the classification framework, a learning algorithm is executed on a training set which contains examples. Each example is characterized by a description and a label. A learning algorithm's goal is to generalize the information contained in the training set to build a classifier, i.e. a function that takes as input an example description, and outputs a label prediction. A good learning algorithm produces classifiers of low risk, meaning a low probability of misclassifying a new example that was not used in the learning process.

Among all machine learning theories, *Sample Compression* [4] studies classifiers that can be expressed by a subset of the training set. This theory allows to compute bounds on a classifier's risk based on two main quantities: the size of the *compression set* (the number of training examples needed to describe the classifier) and the *empirical risk* (the proportion of misclassified training examples). This suggests that a classifier should realize a tradeoff between its complexity, quantified here by the compression set size, and its accuracy on the training set.

Based on this approach, the *Set Covering Machine* (SCM) is a learning algorithm motivated by a sample compression risk bound [8]. However, instead of finding the optimal value of the bound, the SCM algorithm is a greedy approach that aims to quickly find a good solution near the optimal bound's value.

In this paper, we address the following question: "How far to the optimal is the solution returned by the SCM algorithm?". To answer this question, one

needs to design a learning algorithm that directly minimizes the sample compression bound that inspired the SCM. This task is not a trivial one : unlike many popular machine learning algorithms that rely on the minimization of a convex function (as the famous Support Vector Machine [3]), this optimization problem is based on a combinatorial function. Although Hussain et al. [5] suggested a (convex) linear program version of the SCM, it remains a heuristic inspired by the bound. The present paper describes how to use Constraint Programming techniques to directly minimize the sample compression bound. More precisely, we design a pseudo-Boolean program that encodes the proper optimization problem, and finally show that the SCM is surprisingly accurate.

2 Problem Description

The Binary Classification problem in Machine Learning. An *example* is a pair $(\mathbf{x}, y)$, where $\mathbf{x}$ is a *description* and y is a *label*. In this paper, we consider binary classification, where the description is a vector of n real-valued attributes (i.e. $\mathbf{x} \in \mathbb{R}^n$) and the label is a Boolean value (i.e. $y \in \{0, 1\}$). We say that a 0-labeled example is a *negative example* and a 1-labeled is a *positive example*.

A *dataset* contains several examples coming from the observation of the same phenomenon. We denote S the *training set* of m examples used to "learn" this phenomenon. As the examples are considered to be independently and identically distributed (iid) following a probability distribution D on $\mathbb{R}^n \times \{0, 1\}$, we have:

$$S \stackrel{\text{def}}{=} \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\} \sim D^m .$$

A *classifier* receives as input the description of an example and predicts a label. Thus, a classifier is a function $h : \mathbb{R}^n \to \{0, 1\}$. The *risk* $R(h)$ of a classifier is the probability of misclassifying an example generated by the distribution D, and the *empirical risk* $R_S(h)$ of a classifier is the ratio of errors on its training set.

$$R(h) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} I(h(\mathbf{x}) \neq y) \quad \text{and} \quad R_S(h) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{(\mathbf{x},y)\in S} I(h(\mathbf{x}) \neq y) ,$$

where I is the indicator function: $I(a) = 1$ if a is true and $I(a) = 0$ otherwise.

A *learning algorithm* receives as input a training set and outputs a classifier. The challenge of a these algorithms is to generalize the information of the training set to produce a classifier of low risk. Since the data generating distribution D is unknown, a common practice to estimate the risk is to calculate the error ratio on a *testing set* containing examples that were not used in the training process.

Overview of the Sample Compression Theory. The sample compression theory, first expressed by Floyd et al. [4], focuses on classifiers that can be expressed by a subset of the training set.

Consider a classifier obtained by executing a learning algorithm on the training set S containing m examples. The *compression set* $S_{\mathbf{i}}$ refers to examples of

the training set that are needed to characterize the classifier.

$$S_{\mathbf{i}} \stackrel{\text{def}}{=} \{(\mathbf{x}_{i_1}, y_{i_1}), (\mathbf{x}_{i_2}, y_{i_2}), \ldots, (\mathbf{x}_{i_n}, y_{i_n})\} \subseteq S \quad \text{with } 1 \leq i_1 < i_2 < \ldots < i_n \leq m\,.$$

We sometimes use a *message string* μ that contains additional information[1]. The term *compressed classifier* refers to the classifier obtained solely with the compression set $S_{\mathbf{i}}$ and message string μ. Sample compression provides theoretical guarantees on a compressed classifier by upper-bounding its risk. Typically, those bounds suggest that a learning algorithm should favour classifiers of low empirical risk (accuracy) and that are expressed by a few training examples (sparsity). One can advocate for sparse classifiers because they are easy to understand by a human being.

The Set Covering Machine. Suggested by Marchand and Shawe-Taylor [8], the *Set Covering Machine* (SCM) is a learning algorithm directly motivated by the sample compression theory. It builds a conjunction or a disjunction of binary functions that rely on training set data. We focus here on the most studied case where each binary function is a *ball* $g_{i,j}$ characterized by two training examples, a center $(\mathbf{x}_i, y_i) \in S$ and a border $(\mathbf{x}_j, y_j) \in S$.

$$g_{i,j}(\mathbf{x}) \stackrel{\text{def}}{=} \begin{cases} y_i \text{ if } \|\mathbf{x}_i - \mathbf{x}\| < \|\mathbf{x}_i - \mathbf{x}_j\| \\ \neg y_i \text{ otherwise,} \end{cases} \tag{1}$$

where $\|\cdot\|$ is the Euclidean norm. For simplicity, we omit the case $\|\mathbf{x}_i - \mathbf{x}\| = \|\mathbf{x}_i - \mathbf{x}_j\|$ and consider that a ball correctly classifies its center ($g_{i,j}(\mathbf{x}_i) = y_i$) and its border ($g_{i,j}(\mathbf{x}_j) = y_j$).

We denote $\mathcal{H}_S$ the set of all possible balls on a particular dataset S, and $\mathcal{B}$ the set of balls selected by the SCM algorithm among $\mathcal{H}_S$. Thus, the classification function related to a conjunction of balls is expressed by:

$$h_{\mathcal{B}}(\mathbf{x}) \stackrel{\text{def}}{=} \bigwedge_{g \in \mathcal{B}} g(\mathbf{x})\,. \tag{2}$$

As the disjunction case is very similar to the conjunction case, we simplify the following discussion by dealing only with the latter[2]. Figure 1 illustrates an example of a classifier obtained by a conjunction of two balls.

The goal of the SCM algorithm is to choose balls among $\mathcal{H}_S$ to form the conjunction $h_{\mathcal{B}}$. By specializing the sample-compressed classifier's risk bound to the conjunction of balls, Marchand and Sokolova [9] proposed to minimize the risk bound given by Theorem 1 below. Note that the compression set $S_{\mathbf{i}}$ contains the examples needed to construct the balls of $h_{\mathcal{B}}$. Also, the message string μ identifies which examples of $S_{\mathbf{i}}$ are centers, and points out the border example associated with each center. In Theorem 1, the variables n_p and n_b encode the length of the message string μ.

[1] See [8] for further details about the *message* concept in sample compression theory.
[2] The disjunction case equations can be recovered by applying De Morgan's law.

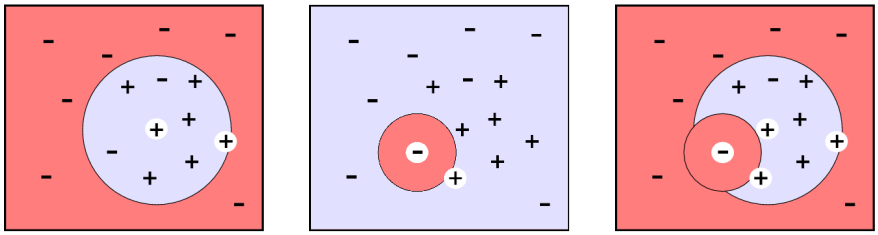

Fig. 1: On a 2-dimensional dataset of 16 examples, from left to right: a positive ball, a negative ball, and the conjunction of both balls. Examples in the light blue region and the red region will be respectively classified positive and negative.

Theorem 1 *(Marchand and Sokolova [9]) For any data-generating distribution D for which we observe a dataset S of m examples, and for each $\delta \in (0, 1]$:*

$$\Pr_{S\sim D^m}\Big(\forall \mathcal{B} \subseteq \mathcal{H}_S \;:\; R(h_{\mathcal{B}}) \;\leq\; \varepsilon\big(\mathcal{B}\big)\Big) \;\geq\; 1-\delta\,,$$

where:

$$\varepsilon(\mathcal{B}) \stackrel{\text{def}}{=} 1-\exp\left(\frac{-1}{m-(|S_{\mathbf{i}}|+k)}\ln\left[\binom{m}{|S_{\mathbf{i}}|+k}\cdot\binom{|S_{\mathbf{i}}|+k}{k}\cdot\binom{n_p}{n_b}\cdot\frac{1}{\zeta(n_b)\zeta(|S_{\mathbf{i}}|)\zeta(k)\delta}\right]\right), \quad (3)$$

and where k is the number of errors that $h_{\mathcal{B}}$ does on training set S, n_p is the number of positive examples in compression set $S_{\mathbf{i}}$, n_b is the number of different examples used as a border, and $\zeta(a) \stackrel{\text{def}}{=} \frac{6}{\pi^2}(a+1)^{-2}$.

This theorem suggests to minimize the expression of $\varepsilon(\mathcal{B})$ in order to find a good balls conjunction. For fixed values of m and δ, this expression tends to decrease with decreasing values of $|S_{\mathbf{i}}|$ and k, whereas $n_b \leq n_p \leq |S_{\mathbf{i}}|$. Moreover, even if the expression of $\varepsilon(\mathcal{B})$ contains many terms, we notice that the quantity $\left[\binom{n_p}{n_b}\cdot\frac{1}{\zeta(n_b)\zeta(|S_{\mathbf{i}}|)\zeta(k)\delta}\right]$ is very small. If we neglect this term, it is easy to see that minimizing $\epsilon(\mathcal{B})$ boils down to find the minimum of Equation (4), which is the sum of the compression set size and the number of empirical errors.

$$\mathcal{F}(\mathcal{B}) \;\stackrel{\text{def}}{=}\; |S_{\mathbf{i}}| + k. \quad (4)$$

This consideration leads us to the SCM algorithm. We say that a ball belonging to a conjunction *covers an example* whenever it classifies it negatively. Note that a balls conjunction $h_{\mathcal{B}}$ negatively classifies an example $\mathbf{x}$ if and only if at least one ball of $\mathcal{B}$ covers $\mathbf{x}$. This implies that if one wants to add a new ball to an existing balls conjunction, he can only change the classification outcome on uncovered examples. A good strategy for choosing a ball to add to a conjunction is then to cover as few positive examples as possible to avoid misclassifying them. This observation underlies the heuristic of the SCM algorithm.

Algorithm 1 SCM (*dataset S, penalties $\{p_1, \ldots, p_n\}$, selection function f*)

1: Consider all possible balls: $\mathcal{H}_S \leftarrow \{g_{i,j} \mid (\mathbf{x}_i, \cdot) \in S, (\mathbf{x}_j, 1) \in S, \mathbf{x}_i \neq \mathbf{x}_j\}$.
2: Initialize: $\mathcal{B}^* \leftarrow \emptyset$.
3: **for** $p \in \{p_1, p_2, \ldots, p_n\}$ **do**
4: Initialize: $\mathcal{N} \leftarrow \{\mathbf{x} \mid (\mathbf{x}, 0) \in S\}$, $\mathcal{P} \leftarrow \{\mathbf{x} \mid (\mathbf{x}, 1) \in S\}$ and $\mathcal{B} \leftarrow \emptyset$.
5: **while** $\mathcal{N} \neq \emptyset$ **do**
6: Choose the best ball according to the following heuristic:
$$g \leftarrow \underset{g \in \mathcal{H}_S}{\operatorname{argmax}} \left\{ \left| \{\mathbf{x} \in \mathcal{N} \mid g(\mathbf{x}) = 0\} \right| - p \cdot \left| \{\mathbf{x} \in \mathcal{P} \mid g(\mathbf{x}) = 0\} \right| \right\} .$$
7: Add this ball to current conjunction: $\mathcal{B} \leftarrow \mathcal{B} \cup \{g\}$.
8: Clean covered examples: $\mathcal{N} \leftarrow \{\mathbf{x} \in \mathcal{N} \mid g(\mathbf{x}) = 1\}$, $\mathcal{P} \leftarrow \{\mathbf{x} \in \mathcal{P} \mid g(\mathbf{x}) = 1\}$.
9: Retain the best conjunction : **if** $f(\mathcal{B}) < f(\mathcal{B}^*)$ **then** $\mathcal{B}^* \leftarrow \mathcal{B}$.
10: **end while**
11: **end for**
12: **return** $\mathcal{B}^*$

Given a training set S, the SCM algorithm (see Algorithm 1) is a greedy procedure for selecting a small subset $\mathcal{B}$ of all possible balls[3] so that a high number of negative examples of S are covered by at least one ball belonging to $\mathcal{B}$. At each step of the algorithm, the tradeoff between the number of covered negative examples and the number of covered positive examples is due to a heuristic (Line 6 of Algorithm 1) that depends on a penalty parameter $p \in [0, \infty)$. We initialize the algorithm with a selection of penalty values, allowing it to create a variety of balls conjunctions. The algorithm returns the best conjunction according to a *model selection function* of our choice.

Several model selection functions can be used along with the SCM algorithm. The function ε given by Equation (3) leads to excellent empirical results. In other words, by running the algorithm with a variety of penalty parameters, selecting from all generated balls conjunctions the one with the lowest bound value allows to obtain a low risk classifier. This method as been shown by Marchand and Shawe-Taylor [8] to be as good as cross-validation[4]. It is exceptional for a risk bound to have such property.

As we explain, the bound relies mainly on the sum $|S_{\mathbf{i}}| + k$, and our extensive experiments with the SCM confirms that the simple model selection function $\mathcal{F}$ given by Equation (4) gives equally good results. We are then interested to know if the SCM algorithm provides a good approximation of this function.

To answer this question, next section presents a pseudo-Boolean optimization model that directly finds the set $\mathcal{B}$ that minimizes the function $\mathcal{F}$.

[3] More precisely, the heuristic function (Line 6 of Algorithm 1) makes it possible to consider only balls whose borders are defined by positive examples (see [8]).

[4] Cross-validation is a widely used method for estimating reliability of a model, but substantially increases computational needs (see section 1.3 of [2]).

3 A pseudo-Boolean optimization model

A pseudo-Boolean problem consists of linear inequality constraints with integer coefficients over binary variables. One can also have a linear objective function.

To solve our machine learning problem with a pseudo-Boolean solver, the principal challenge is to translate the original problem into this particular form. The main strategy to achieve this relies on the following observation:

Observation. As the classification function $h_{\mathcal{B}}$ is a conjunction (see Equation (2)), we observe that $h_{\mathcal{B}}$ misclassifies a positive example iff a negative ball covers it. Similarly, $h_{\mathcal{B}}$ misclassifies a negative example iff no ball covers it.

Equivalence rules. Let's first state two general rules that will be useful to express the problem with pseudo-Boolean constraints. For any positive integer $n \in \mathbb{N}^*$ and Boolean values $\alpha_1, \ldots, \alpha_n, \beta \in \{0, 1\}$, the conjunction and disjunction of the Boolean values α_i can be encoded with these linear inequalities:

$$\alpha_1 \wedge \ldots \wedge \alpha_n = \beta \;\Leftrightarrow\; n - 1 \;\geq\; \alpha_1 + \ldots + \alpha_n - n \cdot \beta \;\geq\; 0\,, \tag{5}$$

$$\alpha_1 \vee \ldots \vee \alpha_n = \beta \;\Leftrightarrow\; 0 \;\geq\; \alpha_1 + \ldots + \alpha_n - n \cdot \beta \;\geq\; 1 - n\,. \tag{6}$$

Program variables. Let $P \stackrel{\text{def}}{=} \{i \mid (\mathbf{x}_i, 1) \in S\}$ and $N \stackrel{\text{def}}{=} \{i \mid (\mathbf{x}_i, 0) \in S\}$ be two disjoint sets, containing indices of positive and negative examples respectively. We define m sets B_i, each containing the indices of the borders that can be associated to center $\mathbf{x}_i$, and m sets C_j, each containing the indices of the centers that can be associated to border $\mathbf{x}_j$. As Marchand and Shawe-Taylor [8], we only consider balls with positive borders. Thus, for $i, j \in \{1, \ldots, m\}$, we have:

$$B_i \;\stackrel{\text{def}}{=}\; \{j \mid j \in P,\, j \neq i\} \quad \text{and} \quad C_j \;\stackrel{\text{def}}{=}\; \{i \mid i \in P \cup N,\, j \in B_i\}\,.$$

In other words, B_k is the set of example indices that can be the border of a ball centered on x_k. Similarly, C_k is the set of example indices that can be the center of a ball whose border is x_k. Necessarily, we have $j \in B_k \iff k \in C_j$.

Given the above definitions of B_i and C_j, the solver have to determine the value of Boolean variables s_i, r_i and $b_{i,j}$ described below:

For every $i \in \{1, \ldots, m\}$:
- s_i is equal to 1 iff the example $\mathbf{x}_i$ belongs to the compression set.
- r_i is equal to 1 iff the $h_{\mathcal{B}}$ misclassifies the example $\mathbf{x}_i$.
- For every $j \in B_i$, $b_{i,j}$ is equal to 1 iff the example $\mathbf{x}_i$ is the center of a ball and $\mathbf{x}_j$ if the border of that same ball.

Objective function. The function to optimize (see Equation (4)) becomes:

$$\min \sum_{i=1}^{m} (r_i + s_i)\,. \tag{7}$$

Program constraints. If an example $\mathbf{x}_i$ is the center of a ball, we want exactly one example $\mathbf{x}_j$ to be its border. Also, if $\mathbf{x}_i$ is not the center of any ball, we don't

want any example $\mathbf{x}_j$ to be its border. Those two conditions are encoded by:

$$\sum_{j \in B_i} b_{i,j} \;\leq\; 1 \quad \text{for } i \in \{1, \ldots, m\}\,. \tag{8}$$

An example belongs to the compression set iff it is a center or a border. We then have $s_k = \left[\bigvee_{i \in C_k} b_{i,k}\right] \vee \left[\bigvee_{j \in B_k} b_{k,j}\right]$. Equivalence rule (6) gives:

$$1 - |B_k \cup C_k| \;\leq\; -|B_k \cup C_k| \cdot s_k + \sum_{i \in C_k} b_{i,k} + \sum_{j \in B_k} b_{k,j} \;\leq\; 0 \;\text{ for } k \in \{1, \ldots, m\}\,. \tag{9}$$

We denote by $D_{i,j}$ the distance between examples $\mathbf{x}_i$ and $\mathbf{x}_j$. Therefore, D is a square matrix of size $m \times m$. For each example index $k \in \{1, \ldots, m\}$, let E_k be the set of all balls that cover (i.e. negatively classify) the example $\mathbf{x}_k$:

$$E_k \;\stackrel{\text{def}}{=}\; \{b_{i,j} \mid i \in P, j \in B_i, D_{i,j} < D_{i,k}\} \cup \{b_{i,j} \mid i \in N, j \in B_i, D_{i,j} > D_{i,k}\}\;.$$

First, suppose that $\mathbf{x}_k$ is a positive example (thus, $k \in P$). Then, recall that the conjunction misclassifies the example $\mathbf{x}_k$ iff a ball covers it (see "observation" above). Therefore, $r_k = \bigvee_{b_{i,j} \in E_k} b_{i,j}$. Using Equivalence Rule (6), we obtain:

$$1 - |E_k| \;\leq\; -|E_k| \cdot r_k + \sum_{b_{i,j} \in E_k} b_{i,j} \;\leq\; 0 \quad \text{for } k \in P\,. \tag{10}$$

Now, suppose that $\mathbf{x}_k$ is a negative example (thus, $k \in N$). Then, recall that the conjunction misclassifies $\mathbf{x}_k$ iff no ball covers it (see "observation" above). We have $r_k = \bigwedge_{b_{i,j} \in E_k} \neg\, b_{i,j}$. By using Equivalence Rule (5) and $\alpha = \neg\beta \Leftrightarrow \alpha = 1 - \beta$ (where $\alpha, \beta \in \{0, 1\}$), we obtain the following constraints:

$$\begin{aligned} 0 \;&\leq\; -|E_k| \cdot r_k + \sum_{b_{i,j} \in E_k} (1 - b_{i,j}) \;\leq\; |E_k| - 1 \\ \Leftrightarrow 1 \;&\leq\; |E_k| \cdot r_k + \sum_{b_{i,j} \in E_k} b_{i,j} \;\leq\; |E_k| \quad \text{for } k \in N\,. \end{aligned} \tag{11}$$

4 Empirical Results on Natural Data

The optimization problem of minimizing Equation (7) under Constraints (8, 9, 10, 11) gives a new learning algorithm that we call *PB-SCM*. To evaluate this new algorithm, we solve several learning problems using three well-known pseudo-Boolean solvers, PWBO [6], SCIP [1] and BSOLO [7], and compare the obtained results to the SCM (the greedy approach described by Algorithm 1).

We use the same seven datasets than [8] and [9], which are common benchmark datasets in the machine learning community. For each dataset, we repeat the following experimental procedure four times with training set sizes $m = |S|$ of 25, 50, 75 and 100 examples. First, we randomly split the dataset examples in a training set S of m examples and a testing set T containing all remaining

Table 1: Empirical results comparing the objective value $\mathcal{F}$ obtained by SCM and PB-SCM algorithms, the test risk of obtained classifiers and required running time ("T/O" means that the pseudo-Boolean solver reaches the time limit).

Dataset		**SCM**			**PB-SCM** (pwbo)			**PB-SCM** (scip)			**PB-SCM** (bsolo)		
name	**size**	$\mathcal{F}$	**risk**	**time**	$\mathcal{F}$	**risk**	**time**	$\mathcal{F}$	**risk**	**time**	$\mathcal{F}$	**risk**	**time**
breastw	25	**2**	0.046	0.04	**2**	0.081	0.03	**2**	0.064	0.71	**2**	0.046	0.05
	50	**2**	0.047	0.07	**2**	0.046	0.06	**2**	0.049	3.7	**2**	0.047	0.64
	75	**2**	0.044	0.12	**2**	0.041	0.16	**2**	0.044	7.4	**2**	0.044	3.7
	100	**2**	0.046	0.16	**2**	0.046	0.43	**2**	0.05	38	**2**	0.046	20
bupa	25	8	0.403	0.31	**7**	0.45	0.31	**7**	0.45	4.1	**7**	0.419	0.64
	50	14	0.431	1.32	**12**	0.495	589	**12**	0.495	47	**12**	0.464	989
	75	21	0.404	4.1	21	0.463	T/O	**19**	0.467	1763	24	0.419	T/O
	100	**27**	0.355	11	32	0.494	T/O	30	0.396	T/O	34	0.367	T/O
credit	25	**4**	0.202	0.11	**4**	0.202	0.08	**4**	0.202	2	**4**	0.202	0.22
	50	6	0.239	0.25	**5**	0.257	9.3	**5**	0.209	21	**5**	0.257	30.1
	75	9	0.216	0.61	**8**	0.266	1920	**8**	0.263	138	**8**	0.268	1862
	100	12	0.233	1.3	11	0.237	T/O	**10**	0.242	798	18	0.302	T/O
glass	25	**5**	0.333	0.11	**5**	0.261	0.03	**5**	0.297	12	**5**	0.261	0.2
	50	9	0.265	0.49	**8**	0.265	10.3	**8**	0.265	35	**8**	0.265	28
	75	16	0.307	1.5	**15**	0.273	T/O	**15**	0.227	736	**15**	0.227	T/O
	100	18	0.222	2.9	**17**	0.222	T/O	**17**	0.206	T/O	22	0.19	T/O
haberman	25	**5**	0.305	0.17	**5**	0.305	0.03	**5**	0.305	3.6	**5**	0.312	0.18
	50	**10**	0.246	0.94	**10**	0.332	34	**10**	0.332	30	**10**	0.246	65
	75	15	0.237	2.5	**14**	0.324	T/O	**14**	0.324	436	16	0.279	T/O
	100	21	0.278	4.5	**20**	0.289	T/O	**20**	0.33	T/O	23	0.289	T/O
pima	25	**8**	0.408	0.33	**8**	0.381	0.36	**8**	0.385	4	**8**	0.381	0.94
	50	15	0.312	0.9	**13**	0.306	2204	**13**	0.311	37	**13**	0.306	1985
	75	20	0.375	3.8	20	0.342	T/O	**19**	0.339	2641	24	0.336	T/O
	100	25	0.326	7.4	26	0.316	T/O	**23**	0.338	T/O	30	0.379	T/O
USvotes	25	**3**	0.112	0.07	**3**	0.11	0.011	**3**	0.107	0.21	**3**	0.12	0.08
	50	5	0.14	0.17	**4**	0.114	0.141	**4**	0.127	2.4	**4**	0.127	1.1
	75	5	0.119	0.28	**3**	0.131	0.183	**3**	0.131	54	**3**	0.131	33
	100	6	0.084	0.35	**4**	0.146	1.21	**4**	0.107	100	**4**	0.137	80

examples[5]. Then, we execute the four learning algorithms (SCM algorithm and PB-SCM with three different solvers) on the same training set S, and compute the risk on the testing set T.

To obtain SCM results, the algorithm is executed with a set of 41 penalty values $\{10^{a/20} \mid a = 0, 1, \ldots, 40\}$ and the model selection function $\mathcal{F}$ given by Equation (4). The PB-SCM problem is solved with the three different solvers. For each solver, we fix the time limit to 3600 seconds and keep the solver's default values for other parameters. When a solver fails to converge in 3600 seconds, we consider the best solution so far. Using the solution of the SCM to provide an initial upper bound to the pseudo-Boolean solvers provided no speed-up.

Table 1 shows the obtained results. Of course, except for T/O situations, the minimal value of the heuristic $\mathcal{F}$ is always obtained by solving the PB-SCM problem. However, it is surprising that the SCM often reaches the same minimum value. Moreover, the SCM sometimes (quickly) finds a best value of $\mathcal{F}$

[5] Training sets are smalls because of the extensive computational power needed by pseudo-Boolean solvers.

when the pseudo-Boolean programs time out, and there is no clear amelioration of the testing risk when PB-SCM finds a slightly better solution than SCM. We conclude that the greedy strategy of SCM is particularly effective.

5 Conclusion

We have presented a pseudo-Boolean model that encodes the core idea behind the combinatorial problem related to the Set Covering Machine. Extensive experiments have been done using three different pseudo-Boolean solvers. For the first time, empirical results show the effectiveness of the greedy approach of Marchand and Shawe-Taylor [8] at building SCM of both small compression set and empirical risk. This is a very surprising result given the simplicity and the low complexity of the greedy algorithm.

References

1. Achterberg, T.: SCIP-a framework to integrate constraint and mixed integer programming. Konrad-Zuse-Zentrum für Informationstechnik (2004)
2. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006)
3. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning 20(3), 273–297 (1995)
4. Floyd, S., Warmuth, M.: Sample compression, learnability, and the vapnik-chervonenkis dimension. Machine Learning 21, 269–304 (1995)
5. Hussain, Z., Szedmak, S., Shawe-Taylor, J.: The linear programming set covering machine (2004)
6. Lynce, R.: Parallel search for boolean optimization (2011)
7. Manquinho, V., Marques-Silva, J.: On using cutting planes in pseudo-boolean optimization. Journal on Satisfiability, Boolean Modeling and Computation 2, 209–219 (2006)
8. Marchand, M., Shawe-Taylor, J.: The set covering machine. Journal of Machine Learning Research 3, 723–746 (2002)
9. Marchand, M., Sokolova, M.: Learning with decision lists of data-dependent features. J. Mach. Learn. Res. 6, 427–451 (December 2005)