# A Pseudo-Boolean Set Covering Machine

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

GRAAL
(Université Laval, Québec city)

October 9, 2012

# Plan

1. Binary classification and Machine learning (ML)
2. Set covering machines (SCM)
3. Using a CP approach to answer a ML question
4. Empirical results

# Binary Classification and Machine Learning (ML)

## Example

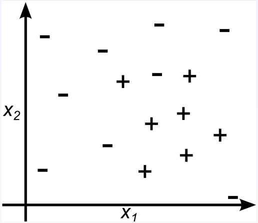
Each example $(\mathbf{x}, y)$ is a **description-label pair**:

- The **description** $\mathbf{x} \in \mathbb{R}^n$ is a feature vector.
- The **label** $y \in \{0, 1\}$ is a boolean value.

## Dataset

A dataset $S$ is a **collection of several examples**.

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



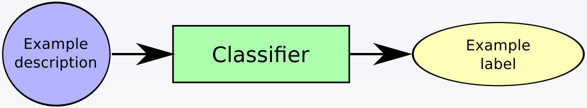
# Binary Classification and Machine Learning (ML)

## Learning Algorithm $A(S) \rightarrow h$

The goal of a learning algorithm is to **study a dataset** and **build a classifier**.


## Classifier $h(\mathbf{x}) \rightarrow y$

A classifier is a function that **takes a description** of an example as input, and **outputs a label** prediction.



# Set Covering Machines (SCM) [Marchand and Shawe-Taylor, 2002]

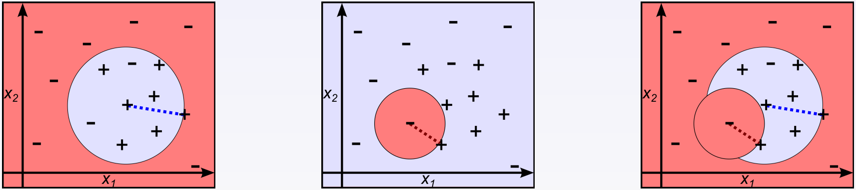
## Data-Dependent Ball

A ball $g_{i,j}$ is defined by a **center** $(\mathbf{x}_i, y_i) \in S$ and a **border** $(\mathbf{x}_j, y_j) \in S$.

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

## Conjunction of Data-Dependent Balls

Given a set of balls $\mathcal{B}$, the SCM classifier is $\quad \mathrm{h}_{\mathcal{B}}(\mathbf{x}) \stackrel{\text{def}}{=} \bigwedge_{g_{i,j} \in \mathcal{B}} g_{i,j}(\mathbf{x})\,.$




Positive ball Negative ball Conjunction of balls

## Sample Compression Theory

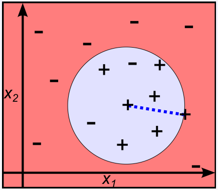
The theory suggests to **minimize the following cost function** :

$$f(\mathcal{B}) \stackrel{\text{def}}{=} 2\times \boxed{\texttt{number of balls}} + \boxed{\texttt{number of training errors}}$$

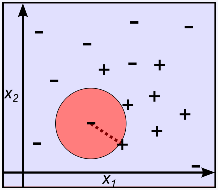
## SCM is a Greedy Algorithm

The SCM is a fast algorithm **driven by a parameterized heuristic**.

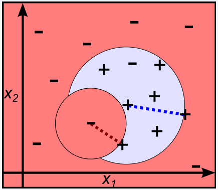
- At each greedy step, the heuristic chooses a ball to add to the conjunction $\mathcal{B}$.
- The search is restarted several times with different heuristic parameters.
- The cost function $f(\mathcal{B})$ selects the best conjunction among all restarts.




$f(\mathcal{B}) = 2\times 1 + 2 = 4$




$f(\mathcal{B}) = 2\times 1 + 8 = 10$




$f(\mathcal{B}) = 2\times 2 + 1 = 5$

# Using a CP approach to answer a ML question

## How Good is the Greedy Strategy?

**How far to the optimal** $f(\mathcal{B}^*)$ is the solution found by the SCM?

## Finding the global minimum is hard

Finding the optimal $f(\mathcal{B}^*)$ is a **combinatorial NP-hard problem**.

## CP to the rescue!

We designed a **Pseudo-Boolean program** that directly minimizes $f(\mathcal{B})$ and compare the solution to the one obtained by the SCM.

# Pseudo-Boolean Set Covering Machine

Given a dataset $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\}$ of $m$ examples.

$$f(\mathcal{B}^*) = \min \sum_{i=1}^{m} (r_i + s_i) \qquad \text{subject to } 5 \times m \text{ linear constraints.}$$

## Program Variables $\sim m^2$

For every $i, j \in \{1, \ldots, m\}$:

- $s_i$ is equal to 1 iff the example $\mathbf{x}_i$ belongs to a ball.
- $r_i$ is equal to 1 iff $h_{\mathcal{B}^*}$ misclassifies the example $\mathbf{x}_i$.
- $b_{i,j}$ is equal to 1 iff the ball $g_{i,j}$ belongs to $\mathcal{B}^*$.

**We compare the original SCM to three pseudo-Boolean solvers:**

- PWBO, *Lynce (2011)*
- BSOLO, *Vasco Manquinho and Marques-Silva (2006)*
- SCIP, *Achterberg (2004)*

# Empirical results (common benchmarks in Machine Learning community)

| Dataset | | SCM | | PWBO | | SCIP | | BSOLO | |
|---|---|---|---|---|---|---|---|---|---|
| **name** | **size** | $\mathcal{F}$ | **time** | $\mathcal{F}$ | **time** | $\mathcal{F}$ | **time** | $\mathcal{F}$ | **time** |
| breastw | 25 | **2** | 0.04 | **2** | 0.03 | **2** | 0.71 | **2** | 0.05 |
| | 50 | **2** | 0.07 | **2** | 0.06 | **2** | 3.7 | **2** | 0.64 |
| | 100 | **2** | 0.16 | **2** | 0.43 | **2** | 0.05 | **2** | 20 |
| bupa | 25 | 8 | 0.31 | **7** | 0.31 | **7** | 4.1 | **7** | 0.64 |
| | 50 | 14 | 1.32 | **12** | 589 | **12** | 47 | **12** | 989 |
| | 100 | **27** | 11 | 32 | **T/O** | 30 | **T/O** | 34 | **T/O** |
| credit | 25 | **4** | 0.11 | **4** | 0.08 | **4** | 2 | **4** | 0.22 |
| | 50 | 6 | 0.25 | **5** | 9.3 | **5** | 21 | **5** | 30.1 |
| | 100 | 12 | 1.3 | 11 | **T/O** | **10** | 798 | 18 | **T/O** |
| glass | 25 | **5** | 0.11 | **5** | 0.03 | **5** | 12 | **5** | 0.2 |
| | 50 | 9 | 0.49 | **8** | 10.3 | **8** | 35 | **8** | 28 |
| | 100 | 18 | 2.9 | **17** | **T/O** | **17** | **T/O** | 22 | **T/O** |
| haberman | 25 | **5** | 0.17 | **5** | 0.03 | **5** | 3.6 | **5** | 0.18 |
| | 50 | **10** | 0.94 | **10** | 34 | **10** | 30 | **10** | 65 |
| | 100 | 21 | 4.5 | **20** | **T/O** | **20** | **T/O** | 23 | **T/O** |
| pima | 25 | **8** | 0.33 | **8** | 0.36 | **8** | 4 | **8** | 0.94 |
| | 50 | 15 | 0.9 | **13** | 2204 | **13** | 37 | **13** | 1985 |
| | 100 | 25 | 7.4 | 26 | **T/O** | **23** | **T/O** | 30 | **T/O** |
| USvotes | 25 | **3** | 0.07 | **3** | 0.011 | **3** | 0.21 | **3** | 0.08 |
| | 50 | 5 | 0.17 | **4** | 0.141 | **4** | 2.4 | **4** | 1.1 |
| | 100 | 6 | 0.35 | **4** | 1.21 | **4** | 100 | **4** | 80 |

# Conclusion

**Thanks to pseudo-Boolean techniques**

- For the first time, we show empirically the **effectiveness of the SCM**.
- This is a very surprising result given the **simplicity** and the **low complexity** of the greedy algorithm.

**Final word from Anonymous Reviewer #3**

*This is one of those disconcerting results that show that simple, low-complexity algorithms can be enough to solve combinatorially hard problems that appear to need heavier-weight approaches.*

# Conclusion

**Thanks to pseudo-Boolean techniques**

- For the first time, we show empirically the **effectiveness of the SCM**.
- This is a very surprising result given the **simplicity** and the **low complexity** of the greedy algorithm.

**Final word from Anonymous Reviewer #3**

*This is one of those disconcerting results that show that simple, low-complexity algorithms can be enough to solve combinatorially hard problems that appear to need heavier-weight approaches.*