Linear Support Vector Machines

T-61.6020 Popular Algorithms in Data Mining and Machine Learning

Stevan Keraudy
stevan.keraudy@tkk.fi

Helsinki University of Technology
April 16, 2008

Introduction

- SVMs are common machine learning techniques
- Used for classification and regression
- Here we describe Linear SVMs (LSVMs)
- Cutting-plane algorithm allows LSVMs to be trained in a linear time

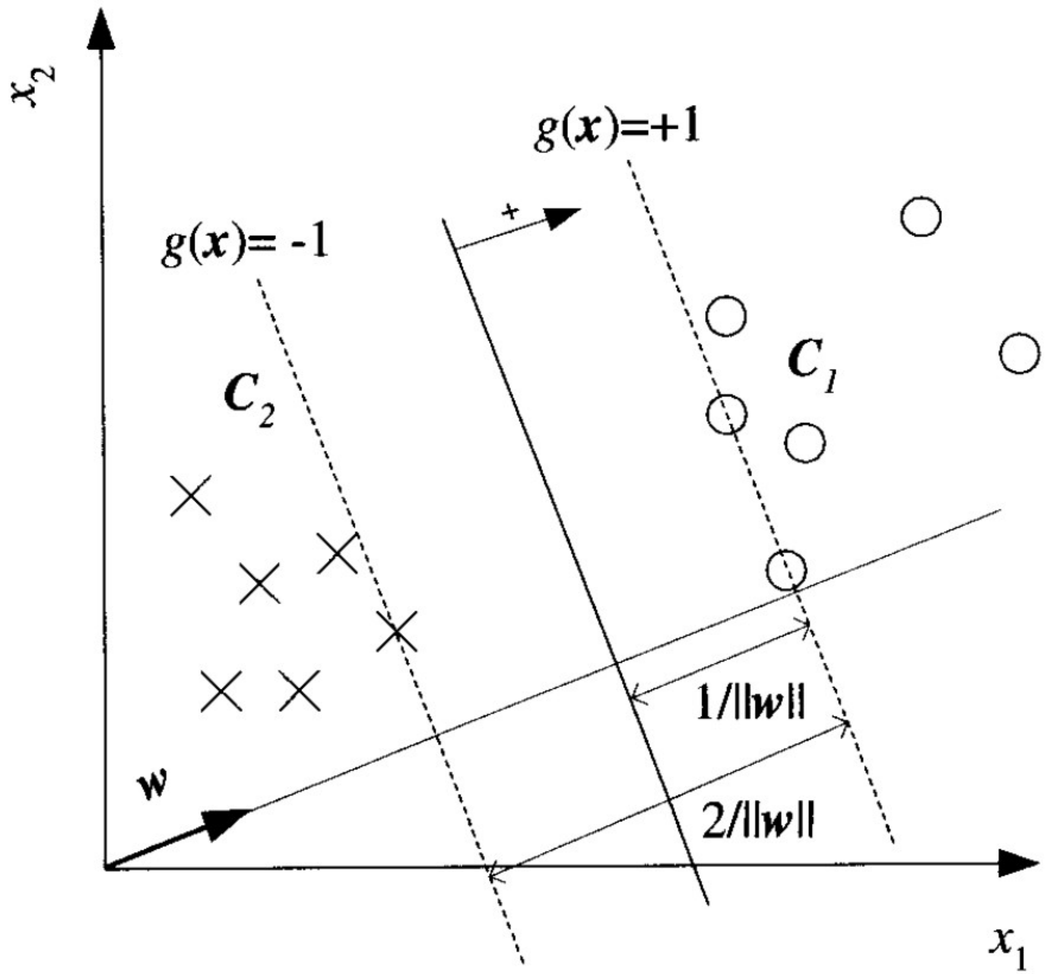
Outline

- LSVMs description
  - Separable case
  - Non-separable case
- Cutting-plane algorithm
- Experiments

Linear SVMs

- Supervised learning method
- In this section we describe the classification problem
- In a given space, find a separating hyperplane
- *Linear* means the decision function is linear
- 2 cases: separable and non-separable

Separable case



Source: Alpaydin

Separable case

- Training data: $\{x_i, y_i\}$, $y_i \in \{-1, 1\}$, $x_i \in \mathbb{R}^N$, $i = 1, \ldots, n$
- Separating hyperplane: $w^T \cdot x + b = 0$
- Margin: $m = 2 / \|w\|$
- For better generalization, we maximize the margin
- Task is to solve the quadratic program:

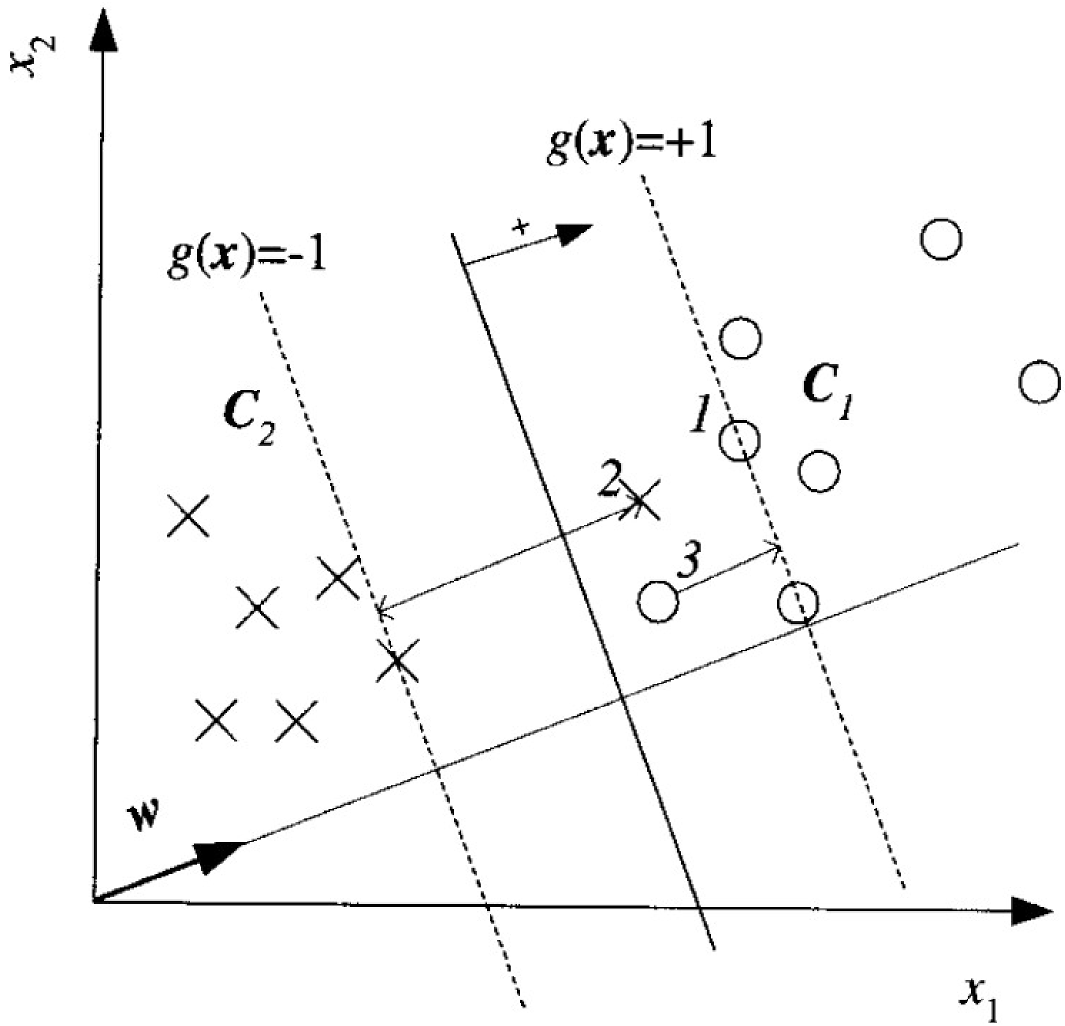
$$\min_w \frac{1}{2} w^T \cdot w$$

$$s.t.\ \ y_i (w^T \cdot x_i + b) \geqslant +1, for\ i = 1 \ldots n$$

Non-separable case

- Usually, data is not linearly separable
- Find the separating hyperplane with the minimum error
- Introduce *slack variables*, $\xi_i \geqslant 0$
- ξ_i store the deviation from the margin for each training point
- 3 cases: $\xi_i = 0$, $\xi_i \in [0,1]$, $\xi_i > 1$
- *Soft error*: $se = \sum_i \xi_i$

Non-separable case



Source: Alpaydin

- (1) $\xi = 0$
- (2) $\xi > 1$
- (3) $\xi \in [0,1]$

Non-separable case

- We add a penalty term to the primal equation which becomes:

$$\min_{w,\xi_i \geqslant 0} \frac{1}{2} w^T \cdot w + \frac{C}{n} \sum_{i=1}^{n} \xi_i$$

$$s.t.\ \ y_i (w^T \cdot x_i + b) \geqslant 1 - \xi_i,\ for\ i = 1 \ldots n$$

Where C is a penalty factor defined by the user

Linear SVMs

- One can improve the results using a non-linear decision function
- SVMs are efficient but their complexity (at least quadratic) reduce their scalability.
- T. Joachims developed the *cutting-plane algorithm* which allows to train a LSVM in linear time

Cutting-plane algorithm

- Consider a large data set with n examples, N features and sparsity $s \ll N$
- Sparsity is defined as the number of non-zero features
- Cutting-plane training time is independent of N:
  - Classification: $O(sn)$
  - Ordinal regression: $O(sn\log(n))$

Cutting-plane algorithm

- Performance is achieved by modifying primal equations for classification and regression
- Use of a single slack variable ξ instead of n
- Use of 2^n constraints instead of n
- The previous constant b is dropped
- The user defines C and the precision ϵ

Classification

- Original classification problem:

$$\min_{w,\xi_i \geqslant 0} \frac{1}{2} w^T \cdot w + \frac{C}{n} \sum_{i=1}^{n} \xi_i$$
$$s.t.\ \ y_i(w^T \cdot x_i) \geqslant 1 - \xi_i,\ for\ i = 1 \ldots n$$

- Joachims' structural classification:

2^n constraints c

1 slack variable ξ

$$\min_{w,\xi \geqslant 0} \frac{1}{2} w^T \cdot w + C \xi$$
$$s.t.\ \frac{1}{n} w^T \sum_{i=1}^{n} c_i y_i x_i \geqslant \frac{1}{n} \sum_{i=1}^{n} c_i - \xi,\ for\ all\ c \in \{0,1\}^n$$

Classification

- Iteratively construct a *sufficient* subset of constraints W

1. Compute the optimum over current W
2. Find most violated constraint that requires the largest ξ given current w:

$$c = \underset{c \in \{0,1\}^n}{argmax} \left\{ \frac{1}{n} \sum_{i=1}^{n} c_i - \frac{1}{n} \sum_{i=1}^{n} c_i y_i \left(w^T x_i \right) \right\}$$

3. Append c to current W

Classification

Algorithm 1 for training Classification SVMs via OP2.

1: Input: $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n))$, C, ϵ
2: $\mathcal{W} \leftarrow \emptyset$
3: **repeat**
4: $\quad (\mathbf{w}, \xi) \leftarrow \mathrm{argmin}_{\mathbf{w}, \xi \geq 0} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi$
$\qquad$ s.t. $\forall \mathbf{c} \in \mathcal{W}$: $\frac{1}{n}\mathbf{w}^T\sum_{i=1}^{n} c_i y_i \mathbf{x}_i \geq \frac{1}{n}\sum_{i=1}^{n} c_i - \xi$
5: $\quad$ **for** i=1,...,n **do**
6: $\qquad c_i \leftarrow \begin{cases} 1 & y_i(\mathbf{w}^T\mathbf{x}_i) < 1 \\ 0 & otherwise \end{cases}$
7: $\quad$ **end for**
8: $\quad \mathcal{W} \leftarrow \mathcal{W} \cup \{\mathbf{c}\}$
9: **until** $\frac{1}{n}\sum_{i=1}^{n} c_i - \frac{1}{n}\sum_{i=1}^{n} c_i y_i(\mathbf{w}^T\mathbf{x}_i) \leq \xi + \epsilon$
10: return$(\mathbf{w}, \xi)$

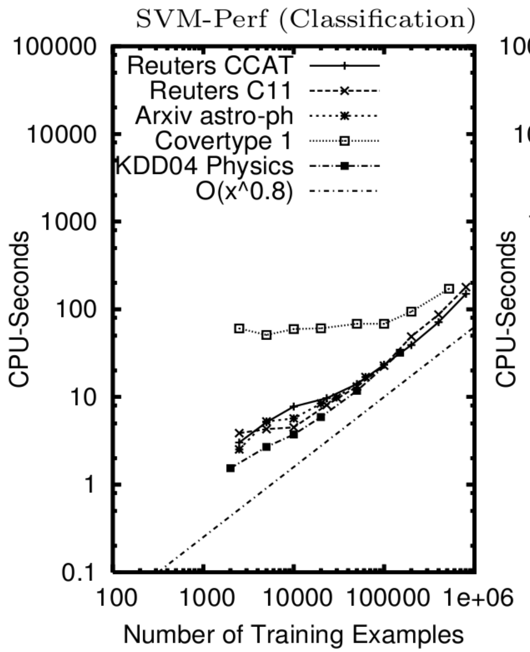
Experiments

Training time in CPU-seconds

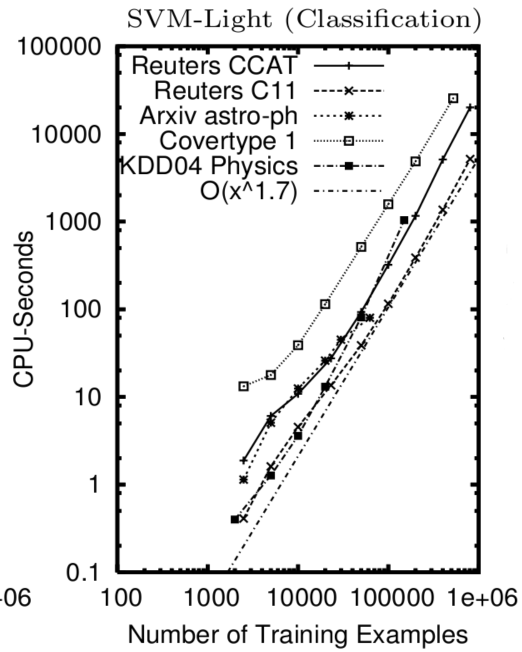
	n	N	s	Classification		Ordinal Regression	
				SVM-Perf	SVM-Light	SVM-Perf	SVM-Light
Reuters CCAT	804,414	47,236	0.16%	149.7	20,075.5	304.1	NA
Reuters C11	804,414	47,236	0.16%	178.9	5,187.4	499.1	NA
Arxiv astro-ph	62,369	99,757	0.08%	16.9	80.1	26.1	NA
Covertype 1	522,911	54	22.22%	171.7	25,514.3	1,109.1	NA
KDD04 Physics	150,000	78	38.42%	31.9	1,040.2	132.5	NA

- Parameters used:
  - $\epsilon = 0.001$
  - C : setting that achieves the best perfomance on test set (from 10,000 to 1,000,000)

Experiments



$O(n^{0.8})$

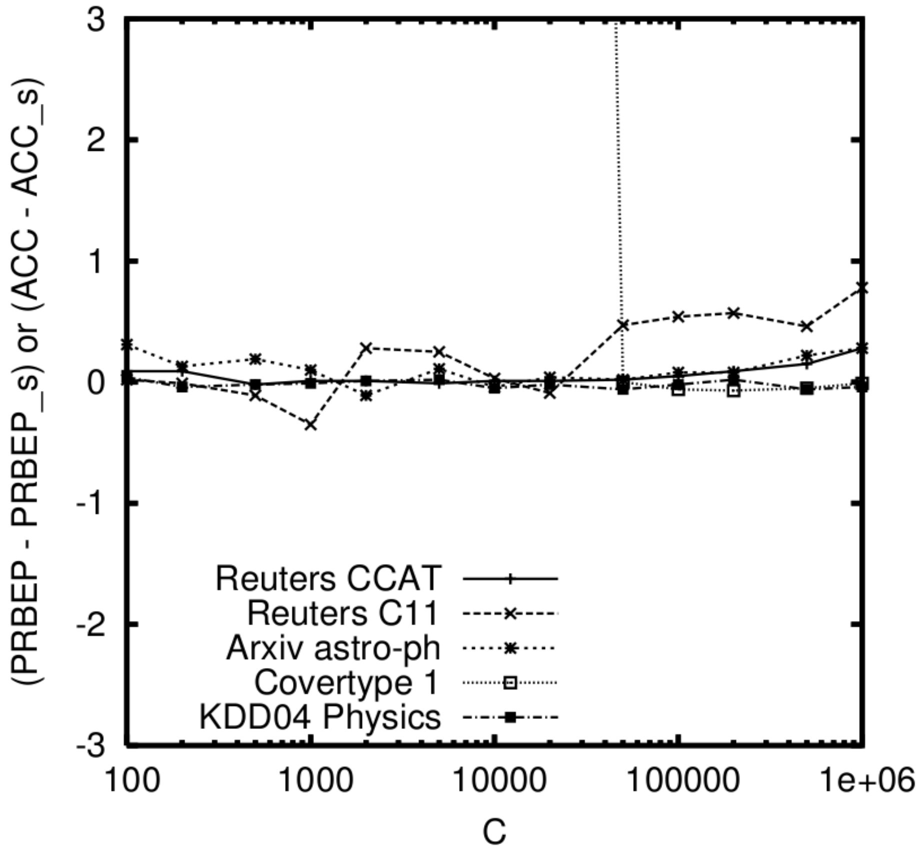


$O(n^{1.7})$

Source: Joachims

- Training time in function of training set size
- SVM-Perf scales better than SVM-Light

Experiments



Source: Joachims

- Difference in prediction accuracy in function of C
- Percentage points are shown
- Performances are similar

Conclusion

- Joachims' algorithm is much faster than other SVM implementations
- Its speed depends on the sparsity of the data
- Not very much more complicated than original SVM to implement
- Very useful for huge data sets

Project tip

- In the project, use Algorithm 1 where you replace the primal quadratic program in line 4 by the dual OP3
- OP3 can be found in Joachims' paper at:
  `http://www.cs.cornell.edu/People/tj/publications/joachims_06a.pdf`

References

- Thorsten Joachims, "*Training Linear SVMs in Linear Time*"
- Ethem Alpaydin, "*Introduction to Machine Learning*"
- Christopher J.C. Burges, "*A tutorial on Support Vector Machines for Pattern Recognition*"