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: \( m = \frac{2}{\|w\|} \)
- For better generalization, we maximize the margin
- Task is to solve the quadratic program:
  \[
  \min_w \frac{1}{2} w^T \cdot w \\
  \text{s.t. } y_i (w^T \cdot x_i + b) \geq +1, \text{ for } i = 1 \ldots n
  \]
Non-separable case

- Usually, data is not linearly separable
- Find the separating hyperplane with the minimum error
- Introduce slack variables, $\xi_i \geq 0$
- $\xi_i$ 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: $se = \sum \xi_i$
Non-separable case

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

Source: Alpaydin
Non-separable case

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

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

\[
s.t. \quad y_i(w^T \cdot x_i + b) \geq 1 - \xi_i, \quad \text{for } i = 1 \ldots n
\]

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 $\xi$ instead of $n$
- Use of $2^n$ constraints instead of $n$
- The previous constant $b$ is dropped
- The user defines $C$ and the precision $\epsilon$
Classification

- Original classification problem:

\[
\min_{\mathbf{w}, \xi \geq 0} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{n} \xi_i \\
\text{s.t. } y_i (\mathbf{w}^T \mathbf{x}_i) \geq 1 - \xi_i, \text{ for } i = 1 \ldots n
\]

- Joachims' structural classification:

\[
\begin{align*}
\min_{\mathbf{w}, \xi \geq 0} & \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \xi \\
\text{s.t.} & \quad \frac{1}{n} \sum_{i=1}^{n} c_i y_i \mathbf{x}_i \geq \frac{1}{n} \sum_{i=1}^{n} c_i - \xi, \text{ for all } c \in \{0, 1\}^n
\end{align*}
\]
Classification

• Iteratively construct a *sufficient* subset of constraints $\mathcal{W}$

1. Compute the optimum over current $\mathcal{W}$

2. Find most violated constraint that requires the largest $\xi$ given current $w$:

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

3. Append $C$ to current $\mathcal{W}$
Classification

Algorithm 1 for training Classification SVMs via OP2.

1: Input: \( S = ((x_1, y_1), \ldots, (x_n, y_n)), C, \epsilon \)
2: \( W \leftarrow \emptyset \)
3: repeat
4: \((w, \xi) \leftarrow \arg\min_{w, \xi \geq 0} \frac{1}{2} w^T w + C \xi \)
   s.t. \( \forall c \in W: \frac{1}{n} w^T \sum_{i=1}^{n} c_i y_i x_i \geq \frac{1}{n} \sum_{i=1}^{n} c_i - \xi \)
5:  for \( i = 1, \ldots, n \) do
6: \( c_i \leftarrow \begin{cases} 
1 & y_i (w^T x_i) < 1 \\
0 & \text{otherwise}
\end{cases} \)
7:  end for
8: \( W \leftarrow W \cup \{c\} \)
9: until \( \frac{1}{n} \sum_{i=1}^{n} c_i - \frac{1}{n} \sum_{i=1}^{n} c_i y_i (w^T x_i) \leq \xi + \epsilon \)
10: return \((w, \xi)\)
# Experiments

## Training time in CPU-seconds

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

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

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

$O(n^{0.8})$  $O(n^{1.7})$

Source: Joachims
Experiments

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

Source: Joachims
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”