Optimized Cutting Plane Algorithm for Support Vector Machines

Vojtěch Franc and Sören Sonnenburg
Fraunhofer FIRST.IDA, Berlin
Motivation

Large Scale Learning methods are needed

- Many Applications: huge amounts (e.g. millions) of data points
- needed to obtain state-of-the-art results

Applications

- Bioinformatics (Splice sites detection, Gene Boundaries, ...)
- IT-Security (Network traffic)
- Text-Classification (Spam vs. Non-Spam)
- Computer vision (Face detection)

Support Vector Machines (SVM) are powerful tools

Goal: Efficient solver for linear SVM classification
Learning the binary SVM classifier requires

\[
\mathbf{w}^* = \arg\min_{\mathbf{w} \in \mathbb{R}^n} \left( \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \max\{0, 1 - y_i \langle \mathbf{w}, \mathbf{x}_i \rangle\} \right)
\]

Standard approach: optimize the dual formulation of the problem

**Primal problem**

\( (n + m \text{ variables, } 2m \text{ constraints}) \)

\[
\mathbf{w}^*, \xi^* = \arg\min_{\mathbf{w}} \left( \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \xi_i \right)
\]

subject to

\[
\begin{align*}
y_i \langle \mathbf{w}_i, \mathbf{x}_i \rangle & \geq 1 - \xi_i, \quad i = 1, \ldots, m \\
\xi_i & \geq 0, \quad i = 1, \ldots, m
\end{align*}
\]

**Dual problem**

\( (m \text{ variables, } 2m \text{ constraints}) \)

\[
\alpha^* = \arg\max_{\alpha} \left( \langle \alpha, e \rangle - \frac{1}{2} \langle \alpha, \mathbf{H} \alpha \rangle \right)
\]

subject to

\[
\begin{align*}
\alpha_i & \geq 0, \quad i = 1, \ldots, m \\
\alpha_i & \leq C, \quad i = 1, \ldots, m
\end{align*}
\]

- High dimensional data \((n \gg 0)\) can be dealt with efficiently.
- Large number of examples \((m \gg 0)\) remains intractable because training time scales \(O(m^2) \approx O(m^3)\).
Master problem:

$$\mathbf{w}^* = \arg\min_{\mathbf{w} \in \mathbb{R}^n} F(\mathbf{w}) := \left( \frac{1}{2} \|\mathbf{w}\|^2 + C \cdot R(\mathbf{w}) \right)$$

Difficulty stems from the risk term $R(\mathbf{w}) = \sum_{i=1}^{m} \max\{0, 1 - y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \}$.

Idea: approximate the risk $R(\mathbf{w})$ by a simpler term

$$\hat{R}(\mathbf{w}) = \max_{i=1, \ldots, t} (\langle \mathbf{w}, \mathbf{a}_i \rangle + b_i)$$

where each $\langle \mathbf{w}, \mathbf{a}_i \rangle + b_i$ is the cutting plane at the point $\mathbf{w}_i$, i.e.,

$$R(\mathbf{w}) \geq R(\mathbf{w}_i) + \langle \mathbf{a}_i, \mathbf{w} - \mathbf{w}_i \rangle = \langle \mathbf{w}, \mathbf{a}_i \rangle + b_i, \quad \forall \mathbf{w} \in \mathbb{R}^n$$

and $\mathbf{a}_i \in \partial R(\mathbf{w}_i)$ is any subgradient of $R(\mathbf{w})$ at $\mathbf{w}_i$. 

---

Cutting plane algorithm [Joachims2006][Teo2007]
Master problem:

\[ w^* = \arg\min_{w \in \mathbb{R}^n} F(w) := \left( \frac{1}{2} \|w\|^2 + C \cdot R(w) \right) \]

Difficulty stems from the risk term \( R(w) = \sum_{i=1}^{m} \max\{0, 1 - y_i \langle w, x_i \rangle\}. \)

Idea: approximate the risk \( R(w) \) by a simpler term

\[ \hat{R}(w) = \max_{i=1,\ldots,t} \left( \langle w, a_i \rangle + b_i \right) \]

where each \( \langle w, a_i \rangle + b_i \) is the cutting plane at the point \( w_i \), i.e.,

\[ R(w) \geq R(w_i) + \langle a_i, w - w_i \rangle = \langle w, a_i \rangle + b_i , \quad \forall w \in \mathbb{R}^n \]

and \( a_i \in \partial R(w_i) \) is any subgradient of \( R(w) \) at \( w_i \).
Cutting plane algorithm: illustration

$R(w)$
Cutting plane algorithm: illustration

\[ R(w) \]

\[ \langle w, a_1 \rangle + b_1 \]

\[ w_1 \]
Cutting plane algorithm: illustration

\[ R(w) \]

\[ \langle w, a_1 \rangle + b_1 \]

\[ \langle w, a_2 \rangle + b_2 \]

\[ w_2 \]

\[ w_1 \]
Cutting plane algorithm: illustration

\[ R(w) = \langle w, a_1 \rangle + b_1 \]

\[ \hat{R}(w) \]

\[ \langle w, a_2 \rangle + b_2 \]
Cutting plane algorithm: illustration

\[ R(w) = \langle w, a_2 \rangle + b_2 \]
\[ \hat{R}(w) = \langle w, a_1 \rangle + b_1 \]
Cutting plane algorithm

Master problem

$$w^* = \arg\min_{w \in \mathbb{R}^n} F(w) := \left(\frac{1}{2} \|w\|^2 + C \cdot R(w)\right)$$

Reduced problem

$$w_t = \arg\min_{w \in \mathbb{R}^n} \hat{F}(w) := \left(\frac{1}{2} \|w\|^2 + C \cdot \hat{R}(w)\right)$$

where $$\hat{R}(w) = \max_{i=1,...,t} \left(\langle w, a_i \rangle + b_i\right)$$

Cutting Plane Algorithm

Initialize $$t = 0$$

Loop

1. Solve the reduced problem $$w_t = \arg\min_{w \in \mathbb{R}^n} \hat{F}(w)$$
2. Add a new cutting plane at the point $$w_t$$, i.e. compute $$a_{t+1} \in \partial R(w_t)$$ and $$b_t = R(w_t) - \langle a_{t+1}, w_t \rangle$$.

until $$F(w_t) - \hat{F}(w_t) \leq \varepsilon$$
Our proposal to accelerate the CP algorithm

Source of inefficiency in the CP algorithm

- CP algorithm monotonically increases the reduced objective $\hat{F}(w_t)$ while the master objective $F(w_t)$ can heavily fluctuate.
- CP algorithm selects a new cutting plane at the point $w_t$ which can be farther from the optimal $w^*$ than a previous solution $w_{t'}$, $t' < t$.

$\Rightarrow$ a lot of cutting planes do not contribute to approximation of $F(w)$ around the optimal $w^*$.

To accelerate the CP algorithm we propose

1. Additional step to the CP algorithm
   Optimization of the master objective $F(w)$ by a line-search.

2. Modification of the CP selection strategy
   New cutting plane selected in a vicinity of the best so far solution.
Our proposal to accelerate the CP algorithm

Source of inefficiency in the CP algorithm

- CP algorithm monotonically increases the reduced objective $\hat{F}(\mathbf{w}_t)$ while the master objective $F(\mathbf{w}_t)$ can heavily fluctuate.
- CP algorithm selects a new cutting plane at the point $\mathbf{w}_t$ which can be farther from the optimal $\mathbf{w}^*$ than a previous solution $\mathbf{w}_{t'}$, $t' < t$.

$\Rightarrow$ a lot of cutting planes do not contribute to approximation of $F(\mathbf{w})$ around the optimal $\mathbf{w}^*$.

To accelerate the CP algorithm we propose

1. Additional step to the CP algorithm
   Optimization of the master objective $F(\mathbf{w})$ by a line-search.

2. Modification of the CP selection strategy
   New cutting plane selected in a vicinity of the best so far solution.
Optimized Cutting Plane Algorithm (OCAS)

Initialize \( t = 0 \) and \( w_{t}^{\text{best}} = 0 \)

Loop

1. Solve the reduced problem \( w_{t} = \arg\min_{w \in \mathbb{R}^n} \hat{F}(w) \)

2. Compute a new best so far solution by a line-search

\[
\begin{align*}
   w_{t}^{\text{best}} &= \arg\min_{k \geq 0} F(w_{t-1}^{\text{best}}(1 - k) + w_{t}k) .
\end{align*}
\]

3. Add a new cutting plane at the point

\[
\begin{align*}
   w_{t}^{\text{cut}} &= w_{t}^{\text{best}}(1 - \lambda) + w_{t}\lambda , \quad \text{where} \quad \lambda \in (0, 1] ,
\end{align*}
\]

and compute \( a_{t+1} \in \partial R(w_{t}^{\text{cut}}) \), \( b_{t+1} = R(w_{t}^{\text{cut}}) - \langle a_{t}^{\text{cut}} , w_{t}^{\text{cut}} \rangle \).

until \( F(w_{t}) - \hat{F}(w_{t}) \leq \varepsilon \)
Solving the line-search efficiently in $m \log m$ time

The line-search requires optimizing $F(w_{t-1}^{\text{best}}(1 - k) + w_t k) = F(k)$

$$k^* = \arg\min_{k \geq 0} F(k) := \left( \frac{1}{2} k^2 \cdot A_0 + k \cdot B_0 + \sum_{i=1}^{m} \max\{0, k \cdot B_i + C_i\} \right)$$

The optimality condition $0 \in \partial F(k^*) = \sum_{i=0}^{m} \partial F_i(k^*)$ leads to

$$0 \in k \cdot A_0 + B_0 + \sum_{i=1}^{m} \partial F_i(k), \quad \partial F_i(k) = \begin{cases} 0 & \text{if } k \cdot B_i + C < 0 \\ B_i & \text{if } k \cdot B_i + C > 0 \\ [0, B_i] & \text{if } k \cdot B_i + C = 0 \end{cases}$$
Solving the line-search efficiently in $m \log m$ time

The line-search requires optimizing $F(w_{t-1}^{\text{best}}(1 - k) + w_t k) = F(k)$

$$k^* = \arg\min_{k \geq 0} F(k) := \left( \frac{1}{2} k^2 \cdot A_0 + k \cdot B_0 + \sum_{i=1}^{m} \max\{0, k \cdot B_i + C_i\} \right)$$

The optimality condition $0 \in \partial F(k^*) = \sum_{i=0}^{m} \partial F_i(k^*)$ leads to

$$0 \in k \cdot A_0 + B_0 + \sum_{i=1}^{m} \partial F_i(k), \quad \partial F_i(k) = \left\{ \begin{array}{ll} 0 & \text{if } k \cdot B_i + C < 0 \\ B_i & \text{if } k \cdot B_i + C > 0 \\ [0, B_i] & \text{if } k \cdot B_i + C = 0 \end{array} \right.$$
Solving the line-search efficiently in $m \log m$ time

The line-search requires optimizing $F(w_{t-1}^{\text{best}}(1-k) + w_t k) = F(k)$

$$k^* = \arg\min_{k \geq 0} F(k) := \left( \frac{1}{2}k^2 \cdot A_0 + k \cdot B_0 + \sum_{i=1}^{m} \max\{0, k \cdot B_i + C_i\} \right)$$

The optimality condition $0 \in \partial F(k^*) = \sum_{i=0}^{m} \partial F_i(k^*)$ leads to

$$0 \in k \cdot A_0 + B_0 + \sum_{i=1}^{m} \partial F_i(k), \quad \partial F_i(k) = \begin{cases} 0 & \text{if } k \cdot B_i + C < 0 \\ B_i & \text{if } k \cdot B_i + C > 0 \\ [0, B_i] & \text{if } k \cdot B_i + C = 0 \end{cases}$$
Proposed method: stops if $F(w) - F(w^*) \leq \varepsilon$

- Optimized Cutting Plane Algorithm for SVM (OCAS).

Competing methods

1. **Accurate solvers:** stop if $\varepsilon$-KKT satisfied, $F(w) - F(w^*) \leq \varepsilon$
   - SVM$^{light}$, SVM$^{perf}$ [Joachims 1999, 2006], BMRM [Smola 2007].

2. **Approximative solvers:** stop after fixed number of iterations
   - Pegasos [Schwartz 2007], SGD [Bottou 2008].

Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Examples</th>
<th>Dim</th>
<th>Sparsity [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>70,000</td>
<td>784</td>
<td>19</td>
</tr>
<tr>
<td>Astro</td>
<td>99,757</td>
<td>62,369</td>
<td>0.08</td>
</tr>
<tr>
<td>Artificial</td>
<td>150,000</td>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>Cov1</td>
<td>581,012</td>
<td>54</td>
<td>22</td>
</tr>
<tr>
<td>CCAT</td>
<td>804,414</td>
<td>47,236</td>
<td>0.16</td>
</tr>
<tr>
<td>Worm</td>
<td>1,026,036</td>
<td>804</td>
<td>25</td>
</tr>
<tr>
<td>Human</td>
<td>15,028,326</td>
<td>564</td>
<td>25</td>
</tr>
</tbody>
</table>
Experiments: Learning binary linear SVM classifier

**Proposed method:** stops if \( F(w) - F(w^*) \leq \varepsilon \)
- Optimized Cutting Plane Algorithm for SVM (OCAS).

**Competing methods**

1. **Accurate solvers:** stop if \( \varepsilon \)-KKT satisfied, \( F(w) - F(w^*) \leq \varepsilon \)
   - SVM\(_{\text{light}}\) , SVM\(_{\text{perf}}\) [Joachims 1999, 2006], BMRM [Smola 2007].

2. **Approximative solvers:** stop after fixed number of iterations
   - Pegasos [Schwartz 2007], SGD [Bottou 2008].

**Datasets**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Examples</th>
<th>Dim</th>
<th>Sparsity [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>70,000</td>
<td>784</td>
<td>19</td>
</tr>
<tr>
<td>Astro</td>
<td>99,757</td>
<td>62,369</td>
<td>0.08</td>
</tr>
<tr>
<td>Artificial</td>
<td>150,000</td>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>Cov1</td>
<td>581,012</td>
<td>54</td>
<td>22</td>
</tr>
<tr>
<td>CCAT</td>
<td>804,414</td>
<td>47,236</td>
<td>0.16</td>
</tr>
<tr>
<td>Worm</td>
<td>1,026,036</td>
<td>804</td>
<td>25</td>
</tr>
<tr>
<td>Human</td>
<td>15,028,326</td>
<td>564</td>
<td>25</td>
</tr>
</tbody>
</table>
Performance measure: Area Under Receiver Operator Characteristic.

Time: total time for model selection, i.e. learning classifiers for range of $C \in \{0.001, 0.005, \ldots, 5\}$.

### Performance of accurate vs. approximative solvers

<table>
<thead>
<tr>
<th></th>
<th>Astro</th>
<th>CCAT</th>
<th>Cov1</th>
<th>MNIST</th>
<th>Worm</th>
<th>Artificial</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg perform for accurate solvers</td>
<td>98.15</td>
<td>98.51</td>
<td>83.92</td>
<td>95.86</td>
<td>99.45</td>
<td>86.38</td>
</tr>
<tr>
<td></td>
<td>±0.00</td>
<td>±0.01</td>
<td>±0.01</td>
<td>±0.01</td>
<td>±0.00</td>
<td>±0.02</td>
</tr>
<tr>
<td>Pegasos</td>
<td>98.15</td>
<td>98.51</td>
<td>83.89</td>
<td>95.84</td>
<td>99.27</td>
<td>78.35</td>
</tr>
<tr>
<td>SGD</td>
<td>98.13</td>
<td>98.52</td>
<td>83.88</td>
<td>95.71</td>
<td>99.43</td>
<td>80.88</td>
</tr>
</tbody>
</table>

Accurate solvers: OCAS, SVM_{light}, SVM_{perf}, BMRM
Experiments: Performance versus time

Speedup factor with respect to the proposed OCAS solver

- astro
- CCAT
- Cov1
- MNIST
- Worm
- Artificial

SLOWER

FASTER

- BMRM
- SVM\textsuperscript{perf}
- SVM\textsuperscript{light}
- Pegasos
- SGD

NC

NC

NC

NC
Experiments: Time versus objective value

Astro

Worm

MNIST

Artificial
**Dataset:** 15 million human splice dataset (itself 32GB in size)

**Computer:** 2.4GHz 16-Core AMD Opteron.

Saturation at the speed up 4.5 most likely due to the memory load: For 8 CPUs output computation ≈ load of 28GB/s on memory reads.
**Dataset:** 15 million human splice dataset (itself 32GB in size)

**Computer:** 2.4GHz 16-Core AMD Opteron.

Saturation at the speed up 4.5 most likely due to the memory load: For 8 CPUs output computation $\approx$ load of 28GB/s on memory reads.
Conclusions

We have developed

**Optimized Cutting Plane Algorithm for SVM (OCAS)**

- **OCAS outperforms the state-of-the-art solvers** by several orders of magnitude.
- **OCAS is faster even in early stages** of the optimization - previously domain of the approximative methods.
- **OCAS can be efficiently parallelized.**

**Future research**

- Extension to further problems (multi-class SVM, logistic regression, novelty detection, ...).
- Extension to learning with kernels.

**Implementation available at**

- [www.shogun-toolbox.org](http://www.shogun-toolbox.org)

Workshop: **Pascal Large Scale Learning Challenge**, Tomorrow, 8:30
Thank you!