Trading Convexity for Scalability

Léon Bottou
leon@bottou.org

Ronan Collobert, Fabian Sinz, Jason Weston
ronan@collobert.com, fabee@tuebingen.mpg.de, jasonw@nec-labs.com

NEC Laboratories of America
A Word About Convexity

- **Advantages of Convexity:**
  - There is a **unique** solution
  - We know plenty of **good algorithms** for convex programming
  - Makes **theory easier**

- **Drawback:**
  - **Who said** convex problems are the only interesting ones? (except for lazy mathematicians)
Summary

I. Non-Convex SVMs

*Faster and sparser* than convex SVMs!

II. Fast Transductive SVMs

*Faster than any TSVM implementation, especially convex TSVM approaches*

*in noisy conditions*
Support Vector Machines

- **Decision function:**

\[ \hat{y}(x) = w \cdot \Phi(x) + b \]

- **Primal formulation:**

\[
\min_{w, b} \frac{1}{2} \|w\|^2 + C \sum_i H_1[y_i \hat{y}(x_i)]
\]

with the **Hinge Loss** \( H_s(z) = |s - z|_+ \)

- **Dual formulation:**

\[
\min_{\alpha} G(\alpha) = \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j \Phi(x_i) \cdot \Phi(x_j) - \sum_i y_i \alpha_i \quad \text{s.t.} \quad \left\{ \begin{array}{l} \sum_i \alpha_i = 0 \\ 0 \leq y_i \alpha_i \leq C \end{array} \right. 
\]
Part I

Non-Convex SVMs
**SVM Known Problem**

- The number of SVs increases **linearly** with $L$ (Steinwart, 04)

- The cost attributed to one example $(x, y)$ is

$$CH_1[y \hat{y}(x)]$$

Given $z = y \hat{y}(x)$, we have

**Outliers are SVs**

Our decision function is expressed with garbage
Non-Convex SVMs

Ramp Loss

Examples lying in the flat areas of the loss cannot be SVs

(Neural Networks) (Mason, 2000) (Shen, 2003)
The Concave-Convex Procedure (CCCP)

- Consider a cost function \( J(\theta) \)

- **Decompose** into a **convex** part and a **concave** part

\[
J(\theta) = J_{vex}(\theta) + J_{cav}(\theta)
\]

**Iterative** algorithm

\[
\theta^{t+1} = \arg\min_{\theta} \left\{ J_{vex}(\theta) + J'_{cav}(\theta^t) \cdot \theta \right\}
\]

(Le Thi, 94) (Yuille, 03)

- \( J(\theta^t) \) is guaranteed to **decrease at each iteration**

- Converges to a **local minima**
Ramp Algebra

\[ J^s(w, b) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{L} R_s[y_i \hat{y}(x_i)] \]

\[ = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{L} H_1[y_i \hat{y}(x_i)] - C \sum_{i=1}^{L} H_s[y_i \hat{y}(x_i)] \]

Convex

Concave
The Algorithm

1. Initialize $\beta = 0$ and choose $s$

2. Minimize

$$G(\alpha) = \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j \Phi(x_i) \cdot \Phi(x_j) - \sum_i y_i \alpha_i$$

with $\sum_i \alpha_i = 0$ and $-\beta_i \leq y_i \alpha_i \leq C - \beta_i$

3. Update $w$ and $b$

4. Update $\beta$

$$\beta_i \leftarrow \begin{cases} 
C & \text{if } y_i \hat{y}(x_i) < s \\
0 & \text{otherwise}
\end{cases}$$

5. Go back to step 2 until convergence
## Raw Results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Train</th>
<th>Test</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waveform$^1$</td>
<td>4000</td>
<td>1000</td>
<td>Artificial data, 21 dims.</td>
</tr>
<tr>
<td>Banana$^1$</td>
<td>4000</td>
<td>1300</td>
<td>Artificial data, 2 dims.</td>
</tr>
<tr>
<td>USPS+N$^2$</td>
<td>7329</td>
<td>2000</td>
<td>Class “0” vs. rest. with 10% training label noise.</td>
</tr>
<tr>
<td>Adult$^2$</td>
<td>32562</td>
<td>16282</td>
<td>As in (Platt, 1999).</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Dataset</th>
<th>SVM Error</th>
<th>$H_1$ SV</th>
<th>SVM Error</th>
<th>$R_S$ SV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waveform</td>
<td>8.8%</td>
<td>983</td>
<td>8.8%</td>
<td>865</td>
</tr>
<tr>
<td>Banana</td>
<td>9.5%</td>
<td>1039</td>
<td>9.5%</td>
<td>891</td>
</tr>
<tr>
<td>USPS+N</td>
<td>0.5%</td>
<td>3317</td>
<td>0.5%</td>
<td>601</td>
</tr>
<tr>
<td>Adult</td>
<td>15.1%</td>
<td>11347</td>
<td>15.0%</td>
<td>4588</td>
</tr>
</tbody>
</table>

All results are averaged using 10 random splits train-test
We do not need to initialize CCCP using the full dataset.
Convex vs Non-Convex SVMs

![Graphs showing comparison between SVM H and SVM R in terms of time and number of SVs for USPS+N and Adult datasets.](image-url)
Details on USPS+N

- Number of Training Examples vs. Number of Support Vectors

- Testing Error (%) vs. Number of Support Vectors
Details on Adult

Number of Support Vectors vs. Number of Training Examples

- SVM $H_1$
- SVM $R_{-1}$
- SVM $R_0$

Number of Support Vectors vs. Testing Error (%)

- SVM $H_1$
- SVM $R_s$
**Objective Function vs Iterations**

Fast convergence of the **CCCP** procedure
Part II

Transductive SVMs
Transductive SVMs
Losses for Transduction

- \((x_i, y_i)_{1 \leq i \leq N}\) labeled examples, \((x_i)_{N+1 \leq i \leq N+U}\) unlabeled examples

- Cost to be minimized

\[
J(\theta) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{L} H[\hat{y}(x_i)] + C^* \sum_{i=L+1}^{L+U} J_U[\hat{y}(x_i)],
\]

- Possible losses \(J_U\) for unlabeled
Ramp Algebra for Transduction

- **Loss considered** given an unlabeled \( x \) and \( z = \hat{y}(x) \)

\[
J_U(z) = R_s(z) + R_s(-z)
\]

- Ramp Loss on unlabeled appearing twice with both possible labels

\[
J(\theta) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^L H_1[y_i \hat{y}(x_i)] + C^* \sum_{i=L+1}^{L+U} J_U(\hat{y}(x_i))
\]

\[
= \frac{1}{2} \|w\|^2 + C \sum_{i=1}^L H_1[y_i \hat{y}(x_i)] + C^* \sum_{i=L+1}^{L+2U} R_s[y_i \hat{y}(x_i)].
\]

- Decompose again the Ramp into two Hinges and apply CCCP
Balancing Constraint

- Transductive SVMs fail without balancing constraint (Joachims, 1999)

- **Constraint:** (Chapelle & Zien, 2005)

\[
\frac{1}{U} \sum_{i=L+1}^{L+U} \hat{y}(x_i) = \frac{1}{L} \sum_{i=1}^{L} y_i
\]

- CCCP remains valid

- Extra example

\[
\Phi(x_0) = \frac{1}{U} \sum_{i=L+1}^{L+U} \Phi(x_i)
\]

- **Efficiency:** compute the kernel column only once

\[
\Phi(x_0) \cdot \Phi(x_j) = \frac{1}{U} \sum_{i=L+1}^{L+U} \Phi(x_i) \cdot \Phi(x_j) \quad \forall j
\]
The Algorithm

1. Initialize \( w \) and \( b \) with the SVM solution

2. Choose \( s \), initialize \( \beta \) as in (1), set \( \xi_i = y_i \) \((i \neq 0)\) \(\xi_0 = \frac{1}{L} \sum_{j=1}^{L} y_j \)

3. Minimize

\[
G(\alpha) = \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j \Phi(x_i) \cdot \Phi(x_j) - \sum_i \xi_i \alpha_i
\]

with \( \sum_i \alpha_i = 0 \) and \( -\beta_i \leq y_i \alpha_i \leq C - \beta_i \)

4. Update \( w \) and \( b \)

5. Update \( \beta \)

\[
\beta_i \leftarrow \begin{cases} 
C & \text{if } y_i \hat{y}(x_i) < s \text{ and } i \geq L + 1 \\
0 & \text{otherwise}
\end{cases}
\]

6. Go back to step 2 until convergence
## Raw Results

<table>
<thead>
<tr>
<th>data set</th>
<th>classes</th>
<th>dims</th>
<th>points</th>
<th>labeled</th>
</tr>
</thead>
<tbody>
<tr>
<td>g50c</td>
<td>2</td>
<td>50</td>
<td>500</td>
<td>50</td>
</tr>
<tr>
<td>Coil20</td>
<td>20</td>
<td>1024</td>
<td>1440</td>
<td>40</td>
</tr>
<tr>
<td>Text</td>
<td>2</td>
<td>7511</td>
<td>1946</td>
<td>50</td>
</tr>
<tr>
<td>Uspst</td>
<td>10</td>
<td>256</td>
<td>2007</td>
<td>50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Coil20</th>
<th>g50c</th>
<th>Text</th>
<th>Uspst</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>24.6</td>
<td>8.3</td>
<td>18.9</td>
<td>23.2</td>
</tr>
<tr>
<td>SVMLight-TSVM</td>
<td>26.3</td>
<td>6.9</td>
<td>7.4</td>
<td>26.5</td>
</tr>
<tr>
<td>∇TSVM</td>
<td>17.6</td>
<td>5.8</td>
<td>5.7</td>
<td>17.6</td>
</tr>
<tr>
<td>CCCP-TSVM</td>
<td>16.7</td>
<td>5.6</td>
<td>8.0</td>
<td>16.6</td>
</tr>
<tr>
<td>CCCP-TSVM (</td>
<td>s=0)_{UC^*=LC}</td>
<td>15.9</td>
<td>3.9</td>
<td>4.9</td>
</tr>
</tbody>
</table>

As in (Chapelle & Zien, 2005)
**CCCP-TSVM vs SVMLight vs ▽TSVM**

![Graphs showing the comparison between CCCP-TSVM, SVMLight, and ▽TSVM.](image)

- **g50c**
- **Text**
Large Scale Datasets: Reuters and MNIST

![Graph of Test Error vs. Number of Unlabeled Examples for Reuters RCV1 and MNIST datasets.](image)
Large Scale Datasets: Scaling

![Graph 1: 0.1k training set](image1)

- Optimization time [sec]
- Number of unlabeled examples [k]

- 0.1k training set
- CCCP-TSVM
- Quadratic fit

![Graph 2: Reuters RCV1 MNIST](image2)

- Time (Hours)
- Number of Unlabeled Examples

Quadratic Tendency
I. Two non-convex algorithms with advantages over convex alternatives

II. **CCCP** is one good way to handle non-convex problems

III. Why **limiting** ourselves to **convex algorithms**?