Allowing for slack: “Soft margin” SVM

minimize
\[ w, b \quad w \cdot w + C \sum_j \xi_j \]
\[
\left( w \cdot x_j + b \right) y_j \geq 1 - \xi_j, \quad \forall j \quad \xi_j \geq 0
\]

“slack variables”

What is the (optimal) value of \( \xi_j \) as a function of \( w \) and \( b \)?

If \( (w \cdot x_j + b) y_j \geq 1 \), then \( \xi_j = 0 \)

If \( (w \cdot x_j + b) y_j < 1 \), then \( \xi_j = 1 - (w \cdot x_j + b) y_j \)

Sometimes written as
\[
\left( 1 - (w \cdot x_j + b) y_j \right)_+
\]

\( \xi_j = \max(0, 1 - (w \cdot x_j + b) y_j) \)
Equivalent hinge loss formulation

\[
\min_{w,b} \quad w \cdot w + C \sum_{j} \xi_j \\
(w \cdot x_j + b) y_j \geq 1 - \xi_j, \quad \forall j \quad \xi_j \geq 0
\]

Substituting \( \xi_j = \max(0, 1 - (w \cdot x_j + b) y_j) \) into the objective, we get:

\[
\min ||w||^2 + C \sum_{j} \max(0, 1 - (w \cdot x_j + b) y_j)
\]

The **hinge loss** is defined as \( L(y, \hat{y}) = \max(0, 1 - \hat{y}y) \)

\[
\min_{w,b} ||w||^2 + C \sum_{j} L(y_j, w \cdot x_j + b)
\]

This is called **regularization**; used to prevent overfitting! This part is empirical risk minimization, using the hinge loss.
Hinge loss vs. 0/1 loss

Hinge loss:

\[ L(y, \hat{y}) = \max \left( 0, 1 - \hat{y}y \right) \]

0-1 Loss:

\[ L(y, \hat{y}) = 1[\hat{y} \neq y] \]

Hinge loss upper bounds 0/1 loss!
How to deal with imbalanced data?

- In many practical applications we may have **imbalanced** data sets
- We may want errors to be equally distributed between the positive and negative classes
- A slight modification to the SVM objective does the trick!

\[
\min_{w,b} \frac{1}{2} ||w||^2 + \frac{C}{N_+} \sum_{j : y_j = +1} \xi_j + \frac{C}{N_-} \sum_{j : y_j = -1} \xi_j
\]

Class-specific weighting of the slack variables
How do we do multi-class classification?
One versus all classification

Learn 3 classifiers:
• - vs \{o,+\}, weights \( w_- \)
• + vs \{o,-\}, weights \( w_+ \)
• o vs \{+,-\}, weights \( w_o \)

Predict label using:
\[
\hat{y} \leftarrow \arg \max_k \ w_k \cdot x + b_k
\]

Any problems?
Could we learn this dataset? →
Multi-class SVM

Simultaneously learn 3 sets of weights:

• How do we guarantee the correct labels?
• Need new constraints!

The “score” of the correct class must be better than the “score” of wrong classes:

\[ w^{(y_j)} \cdot x_j + b^{(y_j)} > w^{(y)} \cdot x_j + b^{(y)} \quad \forall j, y \neq y_j \]
Multi-class SVM

As for the SVM, we introduce slack variables and maximize margin:

\[
\begin{align*}
\text{minimize}_{w,b} & \quad \sum_y w(y).w(y) + C \sum_j \xi_j \\
& \quad w(y_j).x_j + b(y_j) \geq w(y').x_j + b(y') + 1 - \xi_j, \quad \forall y' \neq y_j, \quad \forall j \\
& \quad \xi_j \geq 0, \quad \forall j
\end{align*}
\]

To predict, we use:
\[
\hat{y} \leftarrow \arg \max_k w_k \cdot x + b_k
\]

Now can we learn it? →
Software

- **SVM\textsuperscript{light}**: one of the most widely used SVM packages. Fast optimization, can handle very large datasets, C++ code.
- **LIBSVM** (used within Python’s scikit-learn)
- Both of these handle multi-class, weighted SVM for imbalanced data, etc.
- There are several new approaches to solving the SVM objective that can be much faster:
  - Stochastic subgradient method (up next!)
  - Distributed computation
PEGASOS [ICML 2007]
Primal Efficient sub-GrAdient SOlver for SVM

Shai Shalev-Shwartz
Yoram Singer
Nati Srebro

The Hebrew University
Jerusalem, Israel

Google
Support Vector Machines

QP form:
\[
\begin{align*}
\argmin_{w, \xi_i \geq 0} & \quad \frac{1}{2}||w||^2 + C \sum_{i=1}^{m} \xi_i \\
\text{s.t.} & \quad \forall i, y_i \langle w, x_i \rangle \geq 1 - \xi_i
\end{align*}
\]

More “natural” form:
\[
\begin{align*}
\argmin_w f(w) & \quad \text{where:} \\
f(w) & \overset{\text{def}}{=} \frac{\lambda}{2}||w||^2 + \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i \langle w, x_i \rangle\}
\end{align*}
\]
A_t = S

Subgradient method

\[ |A_t| = 1 \]

Stochastic gradient

Number of iterations \( T \)

**INITIALIZE.** Choose \( w_1 \) s.t. \( \|w_1\| \leq 1 / \sqrt{\lambda} \)

**FOR** \( t = 1, 2, \ldots, T \)

- Choose \( A_t \subseteq S \)
  \[ A_t^+ = \{(x, y) \in A_t : y \langle w_t, x \rangle < 1\} \]

- \( \nabla_t = \lambda w_t - \frac{1}{|A_t|} \sum_{(x, y) \in A_t^+} y x \)

- \( \eta_t = \frac{1}{t \lambda} \)

- \( w'_t = w_t - \eta_t \nabla_t \)

- \( w_{t+1} = \min \left\{ 1, \frac{1 / \sqrt{\lambda}}{\|w'_t\|} \right\} w'_t \)

**OUTPUT:** \( w_{T+1} \)
Run-Time of Pegasos

• Choosing $|A_t|=1$
  ➔ Run-time required for Pegasos to find $\varepsilon$ accurate solution w.p. $\geq 1-\delta$

$$\tilde{O} \left( \frac{n}{\delta \lambda \varepsilon} \right)$$

• Run-time does not depend on #examples
• Depends on “difficulty” of problem ($\lambda$ and $\varepsilon$)
Experiments

- **3 datasets** (provided by Joachims)
  - Reuters CCAT (800K examples, 47k features)
  - Physics ArXiv (62k examples, 100k features)
  - Covertype (581k examples, 54 features)

Training Time (in seconds):

<table>
<thead>
<tr>
<th></th>
<th>Pegasos</th>
<th>SVM-Perf</th>
<th>SVM-Light</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reuters</td>
<td>2</td>
<td>77</td>
<td>20,075</td>
</tr>
<tr>
<td>Covertype</td>
<td>6</td>
<td>85</td>
<td>25,514</td>
</tr>
<tr>
<td>Astro-Physics</td>
<td>2</td>
<td>5</td>
<td>80</td>
</tr>
</tbody>
</table>
What’s Next!

• Learn one of the most interesting and exciting recent advancements in machine learning
  – The “kernel trick”
  – High dimensional feature spaces at no extra cost!

• But first, a detour
  – Constrained optimization!