Machine Learning 2

2. (Advanced) Support Vector Machines (SVMs)

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Institute for Computer Science
University of Hildesheim, Germany
# Syllabus

## A. Advanced Supervised Learning

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tue. 5.4.</td>
<td>(1) A.1 Generalized Linear Models</td>
</tr>
<tr>
<td>Tue. 12.4.</td>
<td>(2) A.2 Gaussian Processes</td>
</tr>
<tr>
<td>Tue. 19.4.</td>
<td>(3) A.2b Gaussian Processes (ctd.)</td>
</tr>
<tr>
<td>Tue. 26.4.</td>
<td>(4) A.3 Advanced Support Vector Machines</td>
</tr>
<tr>
<td>Tue. 3.5.</td>
<td>(5) A.4 Neural Networks</td>
</tr>
<tr>
<td>Tue. 10.5.</td>
<td>(6) A.5 Ensembles</td>
</tr>
<tr>
<td>Tue. 17.5.</td>
<td>(7) A.5b Ensembles (ctd.)</td>
</tr>
<tr>
<td>Tue. 24.5.</td>
<td>— Pentecoste Break —</td>
</tr>
<tr>
<td>Tue. 31.5.</td>
<td>(8) A.6 Sparse Linear Models — L1 regularization</td>
</tr>
<tr>
<td>Tue. 7.6.</td>
<td>(9) A.6b Sparse Linear Models — L1 regularization (ctd.)</td>
</tr>
<tr>
<td>Tue. 14.6.</td>
<td>(10) A.7. Sparse Linear Models — Further Methods</td>
</tr>
</tbody>
</table>

## B. Complex Predictors

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tue. 21.6.</td>
<td>(11) B.1 Latent Dirichlet Allocation (LDA)</td>
</tr>
<tr>
<td>Tue. 28.6.</td>
<td>(12) B.2 Deep Learning</td>
</tr>
<tr>
<td>Tue. 5.7.</td>
<td>(13) Questions and Answers</td>
</tr>
</tbody>
</table>
Outline

1. Stochastic (Sub)gradient Descent

2. Dual Coordinate Descent

3. The Adaptive Multi Hyperplane Machine
Outline

1. Stochastic (Sub)gradient Descent

2. Dual Coordinate Descent

3. The Adaptive Multi Hyperplane Machine
SVM Optimization Problem / Slack Variables

\[
\text{minimize } \frac{1}{2} \| \beta \|^2 + \gamma \sum_{n=1}^{N} \xi_n \\
\text{w.r.t. } y_i(\beta_0 + \beta^T x_n) \geq 1 - \xi_n, \quad n = 1, \ldots, N \\
\xi \geq 0 \\
\beta \in \mathbb{R}^p, \quad \beta_0 \in \mathbb{R}
\]
SVM Optimization Problem / Slack Variables

\[
\text{minimize } \frac{1}{2} \| \beta \|^2 + \gamma \sum_{n=1}^{N} \xi_n \\
\text{w.r.t. } y_i(\beta_0 + \beta^T x_n) \geq 1 - \xi_n, \quad n = 1, \ldots, N \\
\xi \geq 0 \\
\beta \in \mathbb{R}^p, \quad \beta_0 \in \mathbb{R}
\]

can be rewritten:

\[
\text{minimize } f(\beta) := \frac{1}{2} \| \beta \|^2 + \gamma \sum_{n=1}^{N} \max(0, 1 - y_n(\beta_0 + \beta^T x_n)) \\
\propto \frac{1}{N} \sum_{n=1}^{N} \max(0, 1 - y_n(\beta_0 + \beta^T x_n)) + \frac{1}{2} \lambda \| \beta \|^2, \quad \lambda := \frac{\gamma}{N}
\]
SVM Optimization Problem / Hinge Loss

can be rewritten (ctd.):

\[
\text{minimize } f(\beta) := \frac{1}{2} \|\beta\|^2 + \gamma \sum_{n=1}^{N} \max(0, 1 - y_n(\beta_0 + \beta^T x_n))
\]

\[
\approx \frac{1}{N} \sum_{n=1}^{N} \max(0, 1 - y_n(\beta_0 + \beta^T x_n)) + \frac{1}{2} \lambda \|\beta\|^2, \quad \lambda := \frac{\gamma}{N}
\]

\[
= \frac{1}{N} \sum_{n=1}^{N} \ell_{\text{hinge}}(y_n, \beta_0 + \beta^T x_n) + \frac{1}{2} \lambda \|\beta\|^2
\]

with

\[
\ell_{\text{hinge}}(y, \hat{y}) := \max(0, 1 - y\hat{y})
\]
(Sub)gradients

\[ f(\beta) := \frac{1}{N} \sum_{n=1}^{N} \max(0, 1 - y_n(\beta_0 + \beta^T x_n)) + \frac{1}{2} \lambda ||\beta||^2 \]

\[ = \frac{1}{N} \sum_{n=1}^{N} 1 - y_n(\beta_0 + \beta^T x_n) + \frac{1}{2} \lambda ||\beta||^2 \]

\[ y_n(\beta_0 + \beta^T x_n) < 1 \]
Machine Learning 2  1. Stochastic (Sub)gradient Descent

(Sub)gradients

\[ f(\beta) := \frac{1}{N} \sum_{n=1}^{N} \max(0, 1 - y_n(\beta_0 + \beta^T x_n)) + \frac{1}{2} \lambda \|\beta\|^2 \]

\[ = \frac{1}{N} \sum_{n=1}^{N} 1 - y_n(\beta_0 + \beta^T x_n) + \frac{1}{2} \lambda \|\beta\|^2 \]

subgradients:

\[ \frac{\partial f}{\partial \beta} = \frac{1}{N} \sum_{n=1}^{N} -y_n x_n + \lambda \beta \]

\[ y_n(\beta_0 + \beta^T x_n) < 1 \]
(Sub)gradients

\[
f(\beta) := \frac{1}{N} \sum_{n=1}^{N} \max(0, 1 - y_n(\beta_0 + \beta^T x_n)) + \frac{1}{2} \lambda \|\beta\|^2
\]

\[
= \frac{1}{N} \sum_{n=1}^{N} 1 - y_n(\beta_0 + \beta^T x_n) + \frac{1}{2} \lambda \|\beta\|^2
\]

subgradients:

\[
\frac{\partial f}{\partial \beta} = \frac{1}{N} \sum_{n=1}^{N} - y_n x_n + \lambda \beta
\]

stochastic subgradients:

\[
\frac{\partial f}{\partial \beta} \big|_{D(t)} = \frac{1}{|D(t)|} \sum_{(x,y) \in D(t)} y x + \lambda \beta, \quad D(t) \subseteq D^{\text{train}}, \text{iteration } t
\]
Bound on Parameter Norm

The optimal parameters are bound from above:

\[ ||\beta^*|| \leq \frac{1}{\sqrt{\lambda}} \]

Trivially,

\[ \frac{1}{2} \lambda ||\beta^*||^2 \leq f(\beta^*) \leq f(0) = 1 \]

\[ \Rightarrow ||\beta^*|| \leq \frac{\sqrt{2}}{\sqrt{\lambda}} \]

[SSSS07] have a more complex proof to show the tighter bound (p. 4, end of proof of theorem 1).
Primal Estimated Subgradient Solver for SVMs (Pegasos)

- use **stochastic (sub)gradient descent**

\[
\tilde{\beta}(t+1) := \beta(t) - \eta(t) \frac{\partial f}{\partial \beta}|_{D(t)}
\]

- use gradient sample size \(K\)
  - though no empirical evidence that \(K > 1\) has any benefits

- after each SGD step, **reproject/rescale** \(\beta\):

\[
\beta(t+1) := \tilde{\beta}(t+1) \frac{1}{\min(\sqrt{\lambda}, ||\tilde{\beta}(t+1)||)}
\]

- use **fixed hyperbola schedule as learning rate**:

\[
\eta(t) := \frac{1}{\lambda t}
\]

- see [SSSS07]
Performance Comparison

Table 1. Training time in CPU-seconds

<table>
<thead>
<tr>
<th></th>
<th>Pegasos</th>
<th>SVM-Perf</th>
<th>SVM-Light</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCAT</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-ph</td>
<td>2</td>
<td>5</td>
<td>80</td>
</tr>
</tbody>
</table>

Figure 2. Comparisons of Pegasos to Norma (left) and Pegasos to stochastic gradient descent with a fixed learning rate (right) on the Astro-Physics dataset. In the left plot, the solid lines designate the objective value and the dashed lines depict the loss on the test set.

[SSSS07]
Performance Comparison (2/2)

Figure 3. The effect of $k$ on the objective value of Pegasos on the Astro-Physics dataset. Left: $T$ is fixed. Right: $kT$ is fixed.
(Non-Linear) Kernels
SGD in the primal first works for linear kernels.

Any linear model can be kernelized by representing instances in terms of kernel features:

original feature representation:
\[ x_n \in \mathbb{R}^M, \quad n \in \{1, \ldots, N\} \]

kernel feature representation:
\[ \tilde{x}_n \in \mathbb{R}^N, x_{n,m} := k(x_n, x_m), \quad m \in \{1, \ldots, N\} \]

then:
\[
\hat{y}_{\text{linear}}(\tilde{x}_n; \beta) = \beta^T \tilde{x}_n = \sum_{m=1}^{N} \beta_m \tilde{x}_{n,m}
\]

\[
= \sum_{m=1}^{N} \alpha_m k(x_m, x_n) = \hat{y}_{\text{kernel}} k(x_n; \alpha), \quad \alpha_m := \beta_m
\]
Outline

1. Stochastic (Sub)gradient Descent

2. Dual Coordinate Descent

3. The Adaptive Multi Hyperplane Machine
Dual Problem

Remember, the dual problem was:

$$\text{minimize } f(\alpha) := \frac{1}{2} \alpha^T Q \alpha - 1^T \alpha, \quad Q_{n,m} := y_n y_m x_n^T x_m$$

w.r.t. $\alpha \in [0, \frac{1}{N\lambda}]$
Dual Problem

Remember, the dual problem was:

\[ \text{minimize } f(\alpha) := \frac{1}{2} \alpha^T Q \alpha - 1^T \alpha, \quad Q_{n,m} := y_n y_m x_n^T x_m \]

w.r.t. \( \alpha \in [0, \frac{1}{N\lambda}] \)

coordinate descent w.r.t. coordinate \( \alpha_n \):

\[ f_n(\alpha_n) := f(\alpha_n; \alpha_{-n}) \propto \frac{1}{2} Q_{n,n} \alpha_n^2 + Q_{n,-n} \alpha_{-n} \alpha_n - \alpha_n \]

\[ \frac{\partial f_n}{\partial \alpha_n} = Q_{n,n} \alpha_n + Q_{n,-n} \alpha_{-n} - 1 = 0 \]

\[ \Rightarrow \alpha_n = \frac{1 - Q_{n,-n} \alpha_{-n}}{Q_{n,n}} \]

possibly clip \( \alpha_n \):

\[ \alpha_n = \max(0, \min(\frac{1}{N\lambda}, \frac{1 - Q_{n,-n} \alpha_{-n}}{Q_{n,n}})) \]
Avoid Computing $Q_{n,-n} \alpha_{-n}$

\[ \alpha_n^{(t+1)} := \frac{1 - Q_{n,-n} \alpha_n^{(t)}}{Q_{n,n}} \]
\[ = \frac{1 - Q_{n,.} \alpha(t) + Q_{n,n} \alpha_n^{(t)}}{Q_{n,n}} \]
\[ = \alpha_n^{(t)} - \frac{Q_{n,.} \alpha(t) - 1}{Q_{n,n}} \]
\[ = \alpha_n^{(t)} - \frac{y_n \hat{y}_n - 1}{Q_{n,n}} \]
Avoid Computing $Q_{n,-n}\alpha_{-n}$

\[
\alpha_n^{(t+1)} = \alpha_n^{(t)} - \frac{y_n\hat{y}_n - 1}{Q_{n,n}}
\]

with

\[
\hat{y}_n = \beta^T x_n
\]

and due to

\[
\beta = \sum_{n=1}^{N} \alpha_n y_n x_n
\]

as only $\alpha_n$ changes:

\[
\beta^{(t+1)} := \beta^{(t)} + (\alpha_n^{(t+1)} - \alpha_n^{(t)}) y_n x_n
\]

- accelerates from $O(N)$ to $O(M)$
- even $O(M_{nz})$ for sparse predictor vectors $x$
  ($M_{nz}$ being the average number of nonzeros)
Performance Comparison

Table 2. On the right training time for a solver to reduce the primal objective value to within 1% of the optimal value; see (20). Time is in seconds. The method with the shortest running time is boldfaced. Listed on the left are the statistics of data sets: \( l \) is the number of instances and \( n \) is the number of features.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Data statistics</th>
<th>Linear L1-SVM</th>
<th>Linear L2-SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( l ) ( n ) # nonzeros</td>
<td>DCDL1 Pegasos SVM(^{perf})</td>
<td>DCDL2 PCD TRON</td>
</tr>
<tr>
<td>a9a</td>
<td>32,561 123 451,592</td>
<td>0.2 1.1 6.0</td>
<td>0.4 0.1 0.1</td>
</tr>
<tr>
<td>astro-physic</td>
<td>62,369 99,757 4,834,550</td>
<td>0.2 2.8 2.6</td>
<td>0.2 0.5 1.2</td>
</tr>
<tr>
<td>real-sim</td>
<td>72,309 20,958 3,709,083</td>
<td>0.2 2.4 2.4</td>
<td>0.1 0.2 0.9</td>
</tr>
<tr>
<td>news20</td>
<td>19,996 1,355,191 9,097,916</td>
<td>0.5 10.3 20.0</td>
<td>0.2 2.4 5.2</td>
</tr>
<tr>
<td>yahoo-japan</td>
<td>176,203 832,026 23,506,415</td>
<td>1.1 12.7 69.4</td>
<td>1.0 2.9 38.2</td>
</tr>
<tr>
<td>rcv1</td>
<td>677,399 47,236 49,556,258</td>
<td>2.6 21.9 72.0</td>
<td>2.7 5.1 18.6</td>
</tr>
<tr>
<td>yahoo-korea</td>
<td>460,554 3,052,939 156,436,656</td>
<td>8.3 79.7 656.8</td>
<td>7.1 18.4 286.1</td>
</tr>
</tbody>
</table>
Performance Comparison (2/2)

Figure 1. Time versus the relative error \((20)\). DCDL1-S, DCDL2-S are DCDL1, DCDL2 with shrinking. The dotted line indicates the relative error 0.01. Time is in seconds.

Figure 2. Time versus the difference of testing accuracy between the current model and the reference model (obtained using strict stopping conditions). Time is in seconds.
Outline

1. Stochastic (Sub)gradient Descent

2. Dual Coordinate Descent

3. The Adaptive Multi Hyperplane Machine
Multi-Class SVM

multi-class SVM:

\[ \hat{y}(x) := \arg \max_{y \in \mathcal{Y}} s_y(x) \]

\[ s_y(x; \beta) := \beta_y^T x, \quad \beta_y \in \mathbb{R}^M \quad \forall y \in \mathcal{Y} = \{y_1, \ldots, y_L\} \]

\[ f(\beta) := \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, x_n) + \frac{\lambda}{2} \|\beta\|^2, \quad \beta := (\beta_{y_1}, \beta_{y_2}, \ldots, \beta_{y_L}) \]

margin-based loss:

\[ \ell(y, x; \beta) := \max(0, 1 + \max_{y' \in \mathcal{Y}, y' \neq y} s_{y'}(x) - s_y(x)) \]
Multi-Hyperplane Machine

multi-hyperplane score function:
\[ s_y(x; \beta) := \max_{k=1,\ldots,K} \beta_{y,k}^T x, \quad \beta_{y,k} \in \mathbb{R}^M, \ k \in \{1, \ldots, K\} \]

margin-based loss:
\[ \ell(y, x; \beta) := \max(0, 1 + \max_{y' \in \mathcal{Y}, y' \neq y} s_{y'}(x) - s_y(x)) \]

relaxation / convex upper bound:
\[ \ell(y_n, x_n; \beta, z_n) := \max(0, 1 + \max_{y' \in \mathcal{Y}, y' \neq y_n} s_{y'}(x_n) - \beta_{y_n,z_n}^T x_n) \]

- block coordinate descent / EM type training \((\beta, z)\)
- use SGD to train \(\beta\).
SGD for Training the Multi-Hyperplane Machine

relaxation / convex upper bound:

$$\ell(y_n, x_n; \beta, z_n) := \max(0, 1 + \max_{y' \in \mathcal{Y}, y' \neq y_n} s_{y'}(x_n) - \beta_{y_n, z_n}^T x_n)$$

gradient:

$$\frac{\partial \ell}{\partial \beta_{y,k}}(y_n, x_n; z_n) = \begin{cases} x_n, & \text{if } (y, k) = \arg \max_{y' \in \mathcal{Y}, y' \neq y_n} \beta_{y', k'}^T x_n \\ -x_n, & \text{if } (y, k) = (y_n, z_n) \\ 0, & \text{otherwise} \end{cases}$$

**Adaptive** Multi-Hyperplane Machine:

- initialize $\beta \equiv 0$.
- if all $\beta_{y', k'} x < 0^T x = 0$, create a new hyperplane $K + 1$ with $\beta_{y, K+1} = 0$.
  (conceptually infinite number of hyperplanes)
Performance Comparison

Table 3: Error rate and training time comparison with large-scale algorithms (RBF SVM is solved by LibSVM unless specified otherwise. Poly2 and LibSVM results are from [5]).

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Error rate (%)</th>
<th>Training time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AMM batch</td>
<td>AMM online</td>
</tr>
<tr>
<td>a9a</td>
<td>15.03±0.11</td>
<td>16.44±0.23</td>
</tr>
<tr>
<td>ijcnn</td>
<td>2.40±0.11</td>
<td>3.02±0.14</td>
</tr>
<tr>
<td>webspam</td>
<td>4.50±0.24</td>
<td>6.14±1.08</td>
</tr>
<tr>
<td>mnist_bin</td>
<td>0.53±0.05</td>
<td>0.54±0.03</td>
</tr>
<tr>
<td>mnist_mc</td>
<td>3.20±0.16</td>
<td>3.36±0.20</td>
</tr>
<tr>
<td>rcv1_bin</td>
<td>2.20±0.01</td>
<td>2.21±0.02</td>
</tr>
<tr>
<td>url</td>
<td>1.34±0.21</td>
<td>2.87±1.49</td>
</tr>
</tbody>
</table>

1 excludes data loading time.
2 achieved by parallel training P-packSVMs on 512 processors; results from [28].
3 achieved by LaSVM; results from [12].
Outlook

See [DLVW13] for

- two further scalable learning algorithms for non-linear SVMs,
- an implementation, and
- an evaluation
Further Readings

See the cited original papers.
References

BudgetedSVM: A toolbox for scalable SVM approximations.

A dual coordinate descent method for large-scale linear SVM.

S. Shalev-Shwartz, Y. Singer, and N. Srebro.
Pegasos: Primal estimated sub-gradient solver for svm.

Zhuang Wang, Nemanja Djuric, Koby Crammer, and Slobodan Vucetic.