Lecture 7: Support Vector Machines

Isabelle Guyon
guyoni@inf.ethz.ch

References

• An training algorithm for optimal margin classifiers
  Boser-Guyon-Vapnik, COLT, 1992
• Book chapters 1 and 12
• Software LibSVM
  http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Perceptron Learning Rule

\[ w_j \leftarrow w_j + y_i x_{ij} \]

If \( x \) is misclassified

\[ w_j = \sum \alpha_i y_i x_{ij} \]

Converges to “one” solution classifying well all the examples.

Optimum Margin Perceptron

\[ w_j \leftarrow w_j + y_i x_{ij} \]

If \( x \) is the least well classified

\[ w_j = \sum \alpha_i y_i x_{ij} \]

Converges to the optimum margin Perceptron (Mezard-Krauth, 1988).

QP formulation (vapnik, 1962).
**Optimum Margin Solution**

- Unique solution.
- Depends only on support vectors (SV)
  \[ w_j = \sum_{i \in SV} \alpha_i y_i x_{ij} \]
- SVs are examples closest to the boundary.
- Bound on leave-one-out error: \( \text{LOO} \leq n_{SV}/m \)
- Most "stable", good from MDL point of view.
- But: sensitive to outliers and works only for linearly separable case.

**Negative Margin**

Multiple negative margin solutions, which all give the same number of errors.

---

**Soft-Margin**

Examples within the margin area incur a penalty and become non-marginal support vectors. Unique solution again (Cortes-Vapnik, 1995):

\[ w_j = \sum_{i \in SV} \alpha_i y_i x_{ij} \]

**Non-Linear Perceptron**

\[ f(x) = w \cdot \Phi(x) + b \]
**Kernel “Trick”**

- \( f(x) = w \cdot \Phi(x) \)
- \( w = \sum_i \alpha_i y_i \Phi(x_i) \)

**Dual forms**

\( f(x) = \sum_i \alpha_i y_i k(x_i, x) \)

\( k(x_i, x) = \Phi(x_i) \cdot \Phi(x) \)

---

**Kernel Method**

\( f(x) = \sum_i \alpha_i k(x_i, x) + b \)

\( k(\cdot, \cdot) \) is a similarity measure or “kernel”.

---

**Some Kernels (reminder)**

A kernel is a dot product in some feature space:

\( k(s, t) = \Phi(s) \cdot \Phi(t) \)

- **Examples:**
  - \( k(s, t) = s \cdot t \) Linear kernel
  - \( k(s, t) = \exp \cdot \gamma ||s-t||^2 \) Gaussian kernel
  - \( k(s, t) = \exp \cdot \gamma ||s-t|| \) Exponential kernel
  - \( k(s, t) = (1 + s \cdot t)^q \) Polynomial kernel
  - \( k(s, t) = (1 + s \cdot t)^q \exp \cdot \gamma ||s-t||^2 \) Hybrid kernel

---

**Support Vector Classifier**

\( f(x) = \sum_{k=SV} \alpha_k y_k k(x_k, x) \)

Boser-Guyon-Vapnik-1992
Margin and $||w||$

Maximizing the margin is equivalent to minimizing $||w||$.

Quadratic Programming

- **Hard margin:**
  
  \[
  \min ||w||^2 \\
  \text{such that} \\
  y_j(w \cdot x_j + b) \geq 1 
  \]

- **Soft margin:**
  
  \[
  \min ||w||^2 + C \sum_j \xi_j^\beta \\
  \xi_j \geq 0, \beta = 1, 2 \\
  \text{such that} \\
  y_j(w \cdot x_j + b) \geq (1 - \xi_j) 
  \]

Dual Formulation

- Non-linear case: $x \rightarrow \Phi(x)$
- \[
  \min ||w||^2 + C \sum_j \xi_j^\beta \\
  \xi_j \geq 0, \beta = 1, 2 \\
  \text{such that} \\
  y_j(w \cdot \Phi(x_j) + b) \geq (1 - \xi_j) 
  \]
- \[
  \max -\frac{1}{2} \alpha^\top K \alpha + \alpha \cdot \mathbf{1} \\
  K = [y_j k(x_i, x_j)] + (1/C) \delta_j \\
  \text{such that} \\
  \alpha^\top y = 0; 0 \leq \alpha \leq C 
  \]

“Ridge SVC”

- **Soft margin:**
  
  \[
  \min ||w||^2 + C \sum_j \xi_j^\beta \\
  \xi_j \geq 0, \beta = 1, 2 \\
  \text{such that} \\
  y_j(w \cdot \Phi(x_j) + b) \geq (1 - \xi_j) 
  \]

- **Ridge SVC:**
  
  \[
  \text{Loss } L(x_j) = \max (0, 1 - y_j f(x_j))^\beta \\
  \text{Risk } \sum_i L(x_i) \\
  \min (1/C) ||w||^2 + \sum_j L(x_j) \\
  \text{regularized risk} 
  \]
**Ridge Regression (reminder)**

- Sum of squares:
  \[ R = \sum_i (f(x_i) - y_i)^2 = \sum_i (1-y_i f(x_i))^2 \]

- Add "regularizer":
  \[ R = \sum_i (1-y_i f(x_i))^2 + \lambda \|w\|^2 \]

- Compare with SVC:
  \[ R = \sum_i \max(0, 1-y_i f(x_i)) + \lambda \|w\|^2 \]

**Structural Risk Minimization**

- Nested subsets of models, increasing complexity/capacity:
  \[ S_1 \subset S_2 \subset \ldots \subset S_N \]  
  \[ \text{Vapnik-1984} \]

- Example, rank with \( \|w\|^2 \)
  \[ S_k = \{ w | \|w\|^2 < A_k \}, A_1 < A_2 < \ldots < A_k \]  

- Minimization under constraint:
  \[ \min R_{\text{emp}}[f] \text{ s.t. } \|w\|^2 < A_k \]

- Lagrangian:
  \[ R_{\text{reg}}[f] = R_{\text{emp}}[f] + \lambda \|w\|^2 \]

- Radius-margin bound:
  \[ \text{LOOcv} = 4 r^2 \|w\|^2 \]  
  \[ \text{Vapnik-Chapelle-2000} \]

**Loss Functions**

- Decision boundary
- Margin
- \( z = y f(x) \)
- \( L(y, f(x)) \)

**Regularizers**

- \( \|w\|^2 = \sum_i w_i^2 \) : 2-norm regularization (ridge regression, original SVM)
- \( \|w\|_1 = \sum_i |w_i| \) : 1-norm regularization (Lasso Tibshirani 1996, 1-norm SVM 1965)
- \( \|w\|_0 = \text{length}(w) \) : 0-norm (Weston et al., 2003)
Regression SVM

• Epsilon insensitive loss:
  \[ | y_i - f(x_i) |^\varepsilon \]

Unsupervised learning

SVMs for:
• density estimation: Fit \( F(x) \) (Vapnik, 1998)
• finding the support of a distribution (one-class SVM) (Schoelkopf et al, 1999)
• novelty detection (Schoelkopf et al, 1999)
• clustering (Ben Hur, 2001)

Summary

• For statistical model inference, two ingredients needed:
  – A loss function: defines the residual error, i.e. what is not explained by the model; characterizes the data uncertainty or “noise”.
  – A regularizer: defines our “prior knowledge”, biases the solution; characterizes our uncertainty about the model. We usually bet on simpler solutions (Ockham's razor).

Exercise Class
**Homework 7**

1. Download the software for homework 7.
2. Inspiring yourself by the examples, write a new feature ranking filter object. Choose one in Chapter 3 or invent your own.
3. Provide the p-value and FDR (using a tabulated distribution or the probe method).
4. Email a zip file your object and a plot of the FDR to guyoni@inf.ethz.ch with subject "Homework7" no later than: Tuesday December 13th.

---

**Dexter**

### DEXTER filters texts

<table>
<thead>
<tr>
<th>Size</th>
<th>Type</th>
<th>Features</th>
<th>Training</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9</td>
<td>MB</td>
<td>sparse</td>
<td>Ber</td>
<td>0.33</td>
<td>0.55</td>
</tr>
<tr>
<td>0.7</td>
<td>int</td>
<td></td>
<td></td>
<td>0.05</td>
<td>0.33</td>
</tr>
<tr>
<td>0.5</td>
<td>int</td>
<td></td>
<td></td>
<td>0.05</td>
<td>0.33</td>
</tr>
</tbody>
</table>

Best entries:
BER~3.3-3.9% AUC~0.97-0.99
Frac_fet~1.5% Frac_probe~50%

---

**Baseline Dexter**

- `my_classif=svc({'coef0=1', 'degree=1', 'gamma=0', 'shrinkage=0.5'})`
- `my_model=chain($(s2n('f_max=300'), normalize, my_classif))`

### Results:

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Valid</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>BER</td>
<td>0.33</td>
<td>0.05</td>
<td>0.33</td>
</tr>
<tr>
<td>AUC</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>Frac_feature</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>Frac_probe</td>
<td>0.33</td>
<td>0.33</td>
<td>0.33</td>
</tr>
</tbody>
</table>
**Evaluation of pval and FDR**

- **Ttest object:**
  - computes pval analytically
  - FDR ~ pval * n\_sp / n

- **probe object:**
  - takes any feature ranking object as an argument (e.g. s2n, relief, Ttest)
  - pval ~ n\_sp / n
  - FDR ~ pval * n\_sp / n

**Analytic vs. probe**

- Red analytic – Blue probe

**Relief**

- Relief vs. Ttest (Dashed line: Ttest with probes)