Natural Language Processing and Information Retrieval

Statistical Learning Theory: Linear Classifiers

Alessandro Moschitti
Department of information and communication technology
University of Trento
Email: moschitti@dit.unitn.it
Outline

- Computational Learning theory
  - Introduction to Statistical Learning
  - Perceptron Learning
  - Margins
What is Statistical Learning?

- Statistical Methods – Algorithms that learn relations in the data from examples
- Simple relations are expressed by pairs of variables: \( \langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle, \ldots, \langle x_n, y_n \rangle \)
- Learning \( f \) such that evaluate \( y^* \) given a new value \( x^* \), i.e. \( \langle x^*, f(x^*) \rangle = \langle x^*, y^* \rangle \)
You have already tackled the learning problem.
Linear Regression

\[ Y = \beta X \]
Degree 2
Degree
Machine Learning Problems

- Overfitting
- How dealing with millions of variables instead of only two?
- How dealing with real world objects instead of real values?
Objectives: defining a well defined statistical framework

- What can we learn and how can we decide if our learning is effective?
- Efficient learning with many parameters
- Trade-off (generalization/and training set error)
- How to represent real world objects
Objectives: defining a well defined statistical framework

- What can we learn and how can we decide if our learning is effective?
- Efficient learning with many parameters
- Trade-off (generalization and training set error)
- How to represent real world objects
PAC Learning Definition (1)

- Let $c$ be the function (i.e. a concept) we want to learn
- Let $h$ be the learned concept and $x$ an instance (e.g. a person)
- $\text{error}(h) = \text{Prob } [c(x) \neq h(x)]$
- It would be useful if we could find:
- $\Pr(\text{error}(h) > \varepsilon) < \delta$
- Given a target error $\varepsilon$, the probability to make a larger error is less $\delta$
Definizione di PAC Learning (2)

- This methodology is called Probably Approximately Correct Learning
- The smaller $\varepsilon$ and $\delta$ are the better the learning is
- Problem:
  - Given $\varepsilon$ and $\delta$, determine the size $m$ of the training-set.
  - Such size may be independent of the learning algorithm
- Let us do it for a simple learning problem
Lower Bound on training-set size

Let us reconsider a first general bound:
- $h$ is bad: $\text{error}(h) > \varepsilon$
- $P(f(x)=h(x))$ for $m$ examples is lower than $(1 - \varepsilon)^m$
- Multiplying by the number of bad hypotheses we calculate the probability of selecting a bad hypothesis
  - $P(\text{bad hypothesis}) < N \cdot (1 - \varepsilon)^m < \delta$
  - $P(\text{bad hypothesis}) < N \cdot (e^{-\varepsilon})^m = N \cdot e^{-\varepsilon m} < \delta$

$\Rightarrow m > (1/\varepsilon) \left( \ln(1/\delta) + \ln(N) \right)$

This is a general lower bound
Example

- Suppose we want to learn a boolean function in $n$ variable

- The maximum number of different functions are $2^{2^n}$

\[ m > \frac{1}{\varepsilon} \left( \ln(1/\delta) + \ln(2^{2^n}) \right) = \]

\[ = \frac{1}{\varepsilon} \left( \ln(1/\delta) + 2^n \ln(2) \right) \]
Some Numbers

<table>
<thead>
<tr>
<th>n</th>
<th>epsilon</th>
<th>delta</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.1</td>
<td>0.1</td>
<td>245</td>
</tr>
<tr>
<td>5</td>
<td>0.1</td>
<td>0.01</td>
<td>268</td>
</tr>
<tr>
<td>5</td>
<td>0.01</td>
<td>0.1</td>
<td>2450</td>
</tr>
<tr>
<td>5</td>
<td>0.01</td>
<td>0.01</td>
<td>2680</td>
</tr>
</tbody>
</table>

| 10 | 0.1     | 0.1   | 7123|
| 10 | 0.1     | 0.01  | 7146|
| 10 | 0.01    | 0.1   | 71230|
| 10 | 0.01    | 0.01  | 71460|

==================================
The equation of a hyperplane is

\[ f(\vec{x}) = \vec{x} \cdot \vec{w} + b = 0, \quad \vec{x}, \vec{w} \in \mathbb{R}^n, b \in \mathbb{R} \]

\( \vec{x} \) is the vector representing the classifying example

\( \vec{w} \) is the gradient to the hyperplane

The classification function is

\[ h(x) = \text{sign}(f(x)) \]
Linear classifiers (2)

- Linear Functions are the simplest ones from an analytical point of view.

- The basic idea is to select a hypothesis with null error on the training-set.

- To learn a linear function a simple neural network of only one neuron is enough (Perceptron)
An animal neuron
The Perceptron

\[ \varphi(\vec{x}) = \text{sgn} \left( \sum_{i=1..n} w_i \times x_i + b \right) \]
Useful Concepts

- **Functional Margin** of an example with respect to a hyperplane: \( \gamma_i = y_i(\vec{w} \cdot \vec{x}_i + b) \)

- The distribution of functional margins of a hyperplane with respect to a training set \( S \) is the distribution of the margins of the examples in \( S \) wrt the hyperplane \((\vec{w}, b)\).

- The functional margin of a hyperplane is the minimum margin of the distribution
If we normalize the hyperplane equation, i.e.
\[
\left( \frac{\vec{w}}{\| \vec{w} \|}, \frac{b}{\| \vec{w} \|} \right),
\]
we obtain the geometric margin.

The geometric margin measure the Euclidean distance between the target point and the hyperplane.

The training set Margin is the maximum geometric (functional) margin among all hyperplanes which separates the examples in S.

The hyperplane associated with the above quantity is called maximal margin hyperplane.
Basic Concepts

- From $\cos(\vec{x}, \vec{w}) = \frac{\vec{x} \cdot \vec{w}}{\|\vec{x}\| \cdot \|\vec{w}\|}$

- It follows that

$$\|\vec{x}\| \cos(\vec{x}, \vec{w}) = \frac{\vec{x} \cdot \vec{w}}{\|\vec{w}\|} = \vec{x} \cdot \frac{\vec{w}}{\|\vec{w}\|}$$

- Norm of $\vec{x}$ times the cosine between $\vec{x}$ and $\vec{w}$, i.e. the projection of $\vec{x}$ on $\vec{w}$
Geometric Margin
Geometric margins of 2 points and hyperplane margin
Maximal margin vs other margins
Perceptron training on a data set
(on-line algorithm)

\[ \begin{align*}
\tilde{w}_0 & \leftarrow 0; b_0 \leftarrow 0; k \leftarrow 0; R \leftarrow \max_{1 \leq i \leq l} \| \tilde{x}_i \| \\
\text{Repeat} \\
\quad \text{for } i = 1 \text{ to } m \\
\quad \quad \text{if } y_i (\tilde{w}_k \cdot \tilde{x}_i + b_k) \leq 0 \text{ then} \\
\quad \quad \quad \tilde{w}_{k+1} = \tilde{w}_k + \eta y_i \tilde{x}_i \\
\quad \quad \quad b_{k+1} = b_k + \eta y_i R^2 \\
\quad \quad \quad k = k + 1 \\
\quad \quad \text{endif} \\
\quad \text{endfor} \\
\text{until no error is found} \\
\text{return } k, (\tilde{w}_k, b_k)
\end{align*} \]
Novikoff’s Theorem

Let $S$ be a non-trivial training-set and let

$$R = \max_{i = 1, \ldots, m} \| x_i \| .$$

Let us suppose there is a vector $w^*, \| w^* \| = 1$ and

$$y_i (\langle w^*, x_i \rangle + b^*) \geq \gamma, \quad i = 1, \ldots, m,$$

with $\gamma > 0$. Then the maximum number of errors of the perceptron is:

$$t^* = \left( \frac{2R}{\gamma} \right)^2.$$
Observations

- The theorem states that independently of the margin size, if data is linearly separable the perceptron algorithm finds the solution in a finite amount of steps.
- This number is inversely proportional to the square of the margin.
- The bound is invariant with respect to the scale of the patterns (i.e. only the relative distances count).
- The learning rate is not essential for the convergence.
The decision function can be rewritten as:

\[ h(x) = \text{sgn}(\vec{w} \cdot \vec{x} + b) = \text{sgn}(\sum_{j=1..m} \alpha_j y_j \vec{x}_j \cdot \vec{x} + b) = \]

\[ \text{sgn}(\sum_{i=1..m} \alpha_j y_j \vec{x}_j \cdot \vec{x} + b) \]

as well as the updating function

\[ \text{if } y_i (\sum_{j=1..m} \alpha_j y_j \vec{x}_j \cdot \vec{x}_i + b) \leq 0 \text{ then } \alpha_i = \alpha_i + \eta \]

The learning rate \( \eta \) only affects the re-scaling of the hyperplane, it does not affect the algorithm, so we can fix \( \eta = 1 \).
First properties of SVMs

- **DUALITY** is the first feature of Support Vector Machines
- SVMs are learning machines using the following function:

\[
  f(x) = \text{sgn}(\vec{w} \cdot \vec{x} + b) = \text{sgn}(\sum_{j=1}^{m} \alpha_j y_j \vec{x}_j \cdot \vec{x} + b)
\]

- Note that data appears only as scalar product (for both testing and learning phases)
- The Matrix \( G = (\vec{x}_i \cdot \vec{x}_j)^m_{i,j=1} \) is called Gram matrix
Limits of Linear Classifiers

- Data must be linearly separable
- Noise (almost all classifier types)
- Data must be in vectorial format
Solutions

- **Multi-Layers Neural Network**: back-propagation learning algorithm.
- **SVMs**: kernel methods.
  The learning algorithm is decoupled by the application domain which is encoded by a kernel function.