Language and Statistics II

Lecture 15: Going Discriminative

Noah Smith
Lecture Outline

• Perceptron for training structured models
• Loss functions for structures
• Boosting
• Maximum margin training: intuition and the big idea
Beware

- In this lecture, I won’t say much about the form of the model.
- Assume discrete inputs and discrete outputs.
- If you like, think of parsing or tagging.
- Score = $\exp f(x, y) \cdot w$ unless otherwise noted.
- Nitpicky point, for correctness: assume $\forall x, \forall y, f_0(x, y) = 1$. ($w_0$ is a bias weight.)
- Subroutines you already know and love:
  - Sum scores over all $y$’s for a given $x$
  - Find maximum-scoring $y$ for a given $x$
General Idea

• MLE: $\max p(x, y) = p(x)p(y|x)$
• MCLE: $\max p(y|x)$

(Why model $x$?)

Indeed, why estimate densities at all?
Perceptron for Structured Models (Collins, 2002)

Unlike other training methods we have seen

• (Maximum likelihood
  • Maximum conditional likelihood)

the perceptron does not explicitly maximize a function.

Instead, it simply tries to learn a model that separates the right answer from the wrong answers.

It’s also really simple.
Perceptron for Structured Models (Collins, 2002)

• “Global linear model” over structures.
  – Prediction: \( \hat{y} = \arg \max_{y \in \text{GEN}(x)} f(x, y) \cdot w \)
  – \( x \) is the input
  – \( y \) is the output
  – \( \text{GEN} \) enumerates all possible \( y \) for a given \( x \)
  – \( f \) maps (input, output) to \( \mathbb{R}^d \)
  – \( w \) is the weight vector (\( \mathbb{R}^d \))

• Learning/training/estimation: pick \( w \)
Perceptron for Structured Models
(Collins, 2002)

- Nothing has changed! Just like log-linear models!

- Examples:
  - \( x \) is a sentence, \( y \) is a POS tag sequence
  - \( x \) is a sentence, \( y \) is an NP bracketing
  - \( x \) is a sentence, \( y \) is a parse tree
  - \( x \) is two sentences, \( y \) is a word alignment
  - \( x \) is a sentence, \( y \) is its translation
Perceptron for Structured Models (Collins, 2002)

- Input: \((x_i, y_i)\) for \(i = 1 \ldots n\); \(T\)
- Output: \(w\)

\[
\begin{align*}
\mathbf{w} & \leftarrow 0 \\
\text{for } t = 1 \ldots T \\
\quad & \text{for } i = 1 \ldots n \\
\quad & \quad y_{\text{hyp}} \leftarrow \arg\max_y f(x_i, y) \cdot \mathbf{w} \\
\quad & \quad \mathbf{w} \leftarrow \mathbf{w} + f(x_i, y_i) - f(x_i, y_{\text{hyp}}) \\
\text{return } \mathbf{w}
\end{align*}
\]
Intuition Behind Perceptron Updates

\[ \mathbf{w} \leftarrow \mathbf{w} + f(x_i, y_i) - f(x_i, y_{\text{hyp}}) \]

- If \( y_i = y_{\text{hyp}} \), no change.
- Otherwise, for each \( f_j \):
  - If \( f_j(x_i, y_i) > f_j(x_i, y_{\text{hyp}}) \), increase \( w_j \)
  - Else if \( f_j(x_i, y_i) < f_j(x_i, y_{\text{hyp}}) \), decrease \( w_j \)
  - Else \( f_j \) makes no difference on this example, so don’t change \( w_j \)
Duck, Duck, Goose

$w =$ decision boundary

Duck, Duck, Goose
Theorems

• If the training data \((x_i, y_i)\) for \(i = 1 \ldots n\) are separable with margin \(m\), and
\[
R \geq \| f(x_i, y_i) - f(x_i, y) \| \text{ for all } i = 1 \ldots n, \ y \in \text{GEN}(x_i) \setminus \{y_i\}
\]
then
\[
\text{#mistakes} \leq \frac{R^2}{m^2}
\]

This extends a classification theorem by Freund & Schapire (1999).
Comments

• What if the training data are not separable?
  – See Collins (2002) for the bound. Not as tight!
  – Does it matter? Are NLP data separable?
• How long does it take?
  – You decide: \( T \) (finite convergence time for separable data guaranteed)
  – Remember: no function to optimize!
• Dealing with oscillation:
  – Averaging: average each iterate of \( \mathbf{w} \)
  – Voting: keep each iterate of \( \mathbf{w} \), let them all vote
Voted Perceptron

\[ w = \text{decision boundary} \]

Voted Perceptron
Voted Perceptron

\( w = \text{decision boundary} \)
## Estimation Methods: A Guide

<table>
<thead>
<tr>
<th>Name</th>
<th>Features?</th>
<th>Training?</th>
<th>Decoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum likelihood</td>
<td>Must fit stochastic “story”</td>
<td>Count &amp; Normalize®</td>
<td>( \max_y f(x,y) \cdot w )</td>
</tr>
<tr>
<td>Maximum Conditional likelihood</td>
<td>Relatively local</td>
<td>Convex optimization; ( \sum_y e^{f(x,y)} \cdot w ) (sum over ( y ))</td>
<td>( \max_y f(x,y) \cdot w )</td>
</tr>
<tr>
<td>Perceptron</td>
<td>Relatively local</td>
<td>Perceptron; ( \max_y f(x,y) \cdot w )</td>
<td>( \max_y f(x,y) \cdot w )</td>
</tr>
</tbody>
</table>
Loss Functions

• At this point it behooves us to talk about loss functions.

• Maximum conditional likelihood can be said to minimize $-\log p(y_i | x_i)$.
  – This is sometimes called the log loss.

• There are many other loss functions!
  – Some are easier to minimize than others, and some have loftier goals than others.
Loss Functions for Binary Classification

\[ p(y_i \mid x_i) \]

continuous? differentiable? convex?
Boosting and Exp Loss

• Usually refers to “AdaBoost,” another learning algorithm (Freund & Schapire, 1995).
  – The short version: aims* to minimize \( \text{exp loss} \):
    \[
    \sum_i \sum_{y \in \text{GEN}(x_i)} \exp(w \cdot f(x_i, y) - w \cdot f(x_i, y_i))
    \]
    \[
    = \frac{1}{p_w(y_i | x_i)} - 1
    \]
    *Actually minimizes a bound.

• Exp loss is an upper bound on \textbf{ranking loss} (the number of alternative \( y \) that beat \( y_i \)).
## Estimation Methods: A Guide

<table>
<thead>
<tr>
<th>Name</th>
<th>Features?</th>
<th>Training?</th>
<th>Decoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum likelihood</td>
<td>Must fit stochastic “story”</td>
<td>Count &amp; Normalize®</td>
<td>( \max_y f(x,y) \cdot w )</td>
</tr>
<tr>
<td>Maximum Conditional likelihood</td>
<td>Relatively local</td>
<td>Convex optimization; ( \sum_y e^{f(x,y) \cdot w} ) (sum over ( y ))</td>
<td>( \max_y f(x,y) \cdot w )</td>
</tr>
<tr>
<td>Perceptron</td>
<td>Relatively local</td>
<td>Perceptron; ( \max_y f(x,y) \cdot w )</td>
<td>( \max_y f(x,y) \cdot w )</td>
</tr>
<tr>
<td>Minimum exp-loss</td>
<td>Relatively local</td>
<td>Convex optimization or boosting; ( \sum_y e^{f(x,y) \cdot w} ) (sum over ( y ))</td>
<td>( \max_y f(x,y) \cdot w )</td>
</tr>
</tbody>
</table>
See Also

- Altun, Johnson, & Hoffman (EMNLP 2003)
  - Comparison among \{\text{log loss, exp loss}\} \times \{\text{sequence loss, pointwise loss}\}.
  - Sequence loss: pay for getting the whole tag sequence or tree wrong
  - Pointwise loss: pay for each word you got wrong
From Loss to Margin

• You can think of loss functions as trying to improve the score of \((x_i, y_i)\) as compared to scores of alternative outputs with \(x_i\).

- **exp loss**
  \[
  \max_w \sum_i \sum_{y \in \text{GEN}(x_i)} \exp(w \cdot f(x_i, y) - w \cdot f(x_i, y_i))
  \]

- **log loss**
  \[
  \max_w \sum_i \left( w \cdot f(x_i, y_i) - \log \sum_{y \in \text{GEN}(x_i)} \exp w \cdot f(x_i, y) \right)
  \]
$\log \text{loss} \quad \max_w \sum_i \left( w \cdot f(x_i, y_i) - \log \sum_{y \in \text{GEN}(x_i)} \exp w \cdot f(x_i, y) \right)$
softmax margin = margin to softmax \( y \mathbf{w} \cdot \mathbf{f}(x_i, y) \)

 softmax \((a,b)\) = \(\log(e^a + e^b)\) \(\frac{|a-b| \rightarrow \infty}{\rightarrow \infty}\) max\((a,b)\)

### Log loss

\[
\max_{\mathbf{w}} \sum_{i} \left( \mathbf{w} \cdot \mathbf{f}(x_i, y_i) - \log \sum_{y \in \text{GEN}(x_i)} \exp \mathbf{w} \cdot \mathbf{f}(x_i, y) \right)
\]
0-1 loss

\[ \max_w \sum_i \left( w \cdot f(x_i, y_i) - \max_{y \in \text{GEN}(x_i)} w \cdot f(x_i, y) \right) \]
0-1 loss

\[ \max_w \sum_i \left( w \cdot f(x_i, y_i) - \max_{y \in \text{GEN}(x_i)} w \cdot f(x_i, y) \right) \]
Desiderata

• Don’t really want 0-1 loss
  – Tagging accuracy
  – Parseval accuracy
  – MT evaluation scores

• Core idea of maximum margin methods:

Maximize the (hard) margin under a particular **loss** function.
(Multiclass) Support Vector Machines

First form:
Note constraint on $w$. This prevents us from cheating by using really big weights. (Can think of it as built-in regularization.)

Second form: change of variable.
Note that the objective is quadratic (indeed, psd!), and the constraints are linear.
# Estimation Methods: A Guide

<table>
<thead>
<tr>
<th>Name</th>
<th>Features?</th>
<th>Training?</th>
<th>Decoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum likelihood</td>
<td>Must fit stochastic “story”</td>
<td>Count &amp; Normalize®</td>
<td>$\max_y f(x,y) \cdot w$</td>
</tr>
<tr>
<td>Maximum Conditional</td>
<td>Relatively local</td>
<td>Convex optimization; $\sum_y e^{f(x,y) \cdot w}$ (sum over $y$)</td>
<td>$\max_y f(x,y) \cdot w$</td>
</tr>
<tr>
<td>likelihood</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perceptron</td>
<td>Relatively local</td>
<td>Perceptron; $\max_y f(x,y) \cdot w$</td>
<td>$\max_y f(x,y) \cdot w$</td>
</tr>
<tr>
<td>Minimum exp-loss</td>
<td>Relatively local</td>
<td>Convex optimization or boosting; $\sum_y e^{f(x,y) \cdot w}$ (sum over $y$)</td>
<td>$\max_y f(x,y) \cdot w$</td>
</tr>
<tr>
<td>Maximum margin (I)</td>
<td>Relatively local</td>
<td>Quadratic program (exponentially many constraints)</td>
<td>$\max_y f(x,y) \cdot w$</td>
</tr>
</tbody>
</table>
Coming Soon

• Maximum margin training:
  – Allowing for nonseparable data
  – Hinge loss
  – Making maximum margin training tractable
    • Dual form
    • Factored dual form
  – Sparsity and support vectors
  – Examples on NL tasks
  – Kernels

• Discriminative methods in general:
  – Bringing in “global” features
  – Reranking