Language and Statistics II

Lecture 20: Contrastive estimation
Noah Smith
Administrivia

• Your drafts: hopefully by Thursday
• Email me a 3-best list for presentation times:
  o 11/28 3:00pm
  o 11/28 3:30pm
  o 11/30 3:00pm
  o 11/30 3:30pm
  o 12/5 3:00pm
  o 12/5 3:30pm
  o 12/7 3:00pm
  o 12/7 3:30pm
Today’s Lecture is a Bit Different

• Adapted from some talks in 2005
• Apologies for the heavy styling
“Red leaves don’t hide blue jays.”
What’s a sequence model?

Let $X$ be a random variable over $\Sigma^*$ ($x$ represents a value of $X$):

$$X = \text{red leaves don’t hide blue jays}$$

Can add hidden variables to the model, like labels, parse trees, etc. Call the hidden part $Y$.

These are all log-linear models.

- Markov ($n$-gram) models
- HMMs
- PCFGs
Sequence Models (Finite-State)

bigram

red -> leaves -> don't -> hide -> blue -> jays

trigram

red -> leaves -> don't -> hide -> blue -> jays

bigram HMM

JJ -> NNS -> MD -> VB -> JJ -> NNS

red -> leaves -> don't -> hide -> blue -> jays

bigram HMM

A -> B -> C -> D -> A -> B

red -> leaves -> don't -> hide -> blue -> jays
Sequence Models (Context-Free)

PCFG

S

NP
JJ  NNS  MD  VB
red  leaves  don’t  hide  blue  jays

NP  VP

JJ  NNS
red leaves don't hide blue jays
model class ≠ estimation method

• $n$-gram models
• HMMs
• “chain” MRFs
• WFSAs
• PCFGs
• WCFGs

• MLE
• conditional likelihood
• boosting
• perceptron
• maximum margin
Maximum Likelihood Estimation (Supervised)

\[ p(x, y) \]

\[
\max_{\theta} \sum_i \log \frac{p_\theta(x_i, y_i)}{\sum_{x, y} p_\theta(x, y)}
\]

\[ \sum^* \times \Lambda^* \]
Maximum Likelihood Estimation
(Unsupervised)

This is what EM does.

\[
\sum^* \times \Lambda^*
\]

\[
\max_{\theta} \sum_i \log \frac{\sum_y p_\theta(x_i, y)}{\sum_{x', y'} p_\theta(x', y')}
\]
Focusing Probability Mass

numerator

denominator
Conditional Estimation (Supervised)

\[ p \left\{ \begin{array}{ccccccc} JJ & NNS & MD & VB & JJ & NNS \\ red & leaves & don’t & hide & blue & jays \end{array} \right\} \]

\[ \max_{\theta} \sum_{i} \log \frac{p_{\theta}(x_{i}, y_{i})}{\sum_{y'} p_{\theta}(x_{i}, y')} \]

A different denominator!
## Objective Functions

<table>
<thead>
<tr>
<th>Objective</th>
<th>Numerator</th>
<th>Denominator</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLE</td>
<td>tags &amp; words</td>
<td>$\Sigma^* \times \Lambda^*$</td>
</tr>
<tr>
<td>MLE with hidden variables</td>
<td>words</td>
<td>$\Sigma^* \times \Lambda^*$</td>
</tr>
<tr>
<td>Conditional Likelihood</td>
<td>tags &amp; words</td>
<td>(words) $\times \Lambda^*$</td>
</tr>
<tr>
<td>Maximum Margin</td>
<td>$\approx$ tags &amp; words</td>
<td>$\approx$ hypothesized tags &amp; words</td>
</tr>
</tbody>
</table>
## Objective Functions

<table>
<thead>
<tr>
<th>Objective</th>
<th>Optimization Algorithm</th>
<th>Numerator</th>
<th>Denominator</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLE</td>
<td>Count &amp; Normalize*</td>
<td>tags &amp; words</td>
<td>$\Sigma^* \times \Lambda^*$</td>
</tr>
<tr>
<td>MLE with hidden variables</td>
<td>EM*</td>
<td>words</td>
<td>$\Sigma^* \times \Lambda^*$</td>
</tr>
<tr>
<td>Conditional Likelihood</td>
<td>Iterative Scaling</td>
<td>tags &amp; words</td>
<td>(words) $\times \Lambda^*$</td>
</tr>
<tr>
<td>Maximum Margin</td>
<td>Perceptron</td>
<td>$\approx$ tags &amp; words</td>
<td>$\approx$ hypothesized tags &amp; words</td>
</tr>
</tbody>
</table>
## Objective Functions

<table>
<thead>
<tr>
<th>Objective</th>
<th>Optimization Algorithm</th>
<th>Numerator</th>
<th>Denominator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contrastive Estimation</td>
<td>generic numerical solvers (in this talk, LMVM L-BFGS)</td>
<td>observed data (in this talk, raw word sequence, sum over <em>all possible values of Y</em>)</td>
<td>?</td>
</tr>
</tbody>
</table>

- **Perceptron**
- **Iterative Scaling**
- **EM**
- **Count & Normalize**
- **Optimization Algorithm**
- **Maximum Margin** (words) $\times \Lambda$
- **Conditional Likelihood** $\Sigma \times \Lambda$
- **MLE with hidden variables** $\Sigma \times \Lambda$
- **Objective**
- **Contrastive Estimation**
- **Numerator**
- **Denominator**
- **Contrastive Estimation**
- **Optimization Algorithm**
- **Numerator**
- **Denominator**
- **Contrastive Estimation**
- **Optimization Algorithm**
- **Numerator**
- **Denominator**
- **Contrastive Estimation**
- **Optimization Algorithm**
- **Numerator**
- **Denominator**
This talk is about denominators ... in the unsupervised case.

A good denominator can improve accuracy and tractability.
MLE/EM as a Teacher

Red leaves don’t hide blue jays.

Mommy doesn’t love you.

Dishwashers are a dime a dozen.

Dancing granola doesn’t hide blue jays.
What We’d Like

• Focus on the model on the properties of the data that will lead to an explanation of syntax.

Red leaves don’t hide blue jays.
*Jays blue hide don’t leaves red.
*Blue don’t hide jays leaves red.
*Hide don’t blue jays red leaves.

• Idea: train model to explain order but not content.
Contrastive Estimation
(Smith & Eisner, 2005)

Σ*

observed sentences

implicitly negative sentences
Maximum Likelihood Estimation vs. Contrastive Estimation

**MLE/MAP:**
observed data are **Sentences**, neighborhood is \( S^* \)

\[
\max_{\theta} \left[ \prod_{i=1}^{n} \sum_y p_{\theta}(x_i, y) \right]
\]

**CE:**
observed data are **sentences**, neighborhood is \( \ldots \)?

\[
\max_{\theta} \left[ \prod_{i=1}^{n} \frac{\sum y \sum p_{\theta}(x, y)}{\sum_{x \in \mathcal{N}(x_i)} \sum p_{\theta}(x, y)} \right] = \max_{\theta} \left[ \prod_{i=1}^{n} p_{\theta}(X = x_i | X \in \mathcal{N}(x_i)) \right]
\]

Require numerical optimization
Partition Neighborhood = Conditional EM

$\Sigma^*$

observed sentences

implicitly negative sentences
Riezler’s (1999) Approximation

$\Sigma^*$ observed sentences
Analogy to Conditional Estimation
(Supervised)
CE for Syntax

$\Sigma^*$

observed sentences

Same content, syntactically ill-formed
CE as Teacher

Red leaves don’t hide blue jays.

Leaves red don’t hide blue jays.

Red don’t leaves hide blue jays.

Red leaves hide don’t blue jays.
What is a syntax model supposed to explain?

Each learning hypothesis corresponds to a denominator / neighborhood.
The Job of Syntax

“Explain why each word is necessary.”

→ \textbf{DEL1WORD} neighborhood

- red don’t hide blue jays
- leaves don’t hide blue jays
- red leaves don’t hide blue jays
- red leaves don’t hide blue
- red leaves don’t blue jays
- red leaves don’t hide jays
The Job of Syntax

“Explain the (local) order of the words.”

→ TRANS1 neighborhood

leaves red don’t hide blue jays

red leaves don’t hide blue jays

red leaves don’t hide jays blue

red leaves hide don’t blue jays

red leaves don’t blue hide jays
red leaves don’t hide blue jays

sentences in TRANS1 neighborhood
red leaves don’t hide blue jays

(sentences in \textsc{Trans1} neighborhood)

(with any tagging)
The New Modeling Imperative

A good sentence hints that a set of bad ones is nearby.

“Make the good sentence likely, at the expense of those bad neighbors.”
This talk is about **denominators** ... in the **unsupervised** case.

A good denominator can improve **accuracy** and **tractability**.
Log-Linear Models

score of \( x, y \)

\[
p(x,y) = \frac{\exp(f(x,y) \cdot \theta)}{Z(\theta)}
\]

Z may be infinite for some \( \theta \); computing it (if it is finite) may require solving a non-linear system.

partition function

\[
Z(\theta) = \sum_x \sum_y \exp(f(x,y) \cdot \theta)
\]

Sums over all possible taggings of all possible sentences!
Log-Linear Models

Computing $Z$ is undesirable!

- **Conditional Estimation (Supervised)**
  - 1 sentence: $Z(x)$

- **Contrastive Estimation (Unsupervised)**
  - a few sentences: $Z(N(x))$

**Score of $x$, $y$**

$$p(x, y) = \frac{\exp(f(x,y) \cdot \theta)}{Z(\theta)}$$

**Partition Function**

$$Z(\theta) = \sum_x \sum_y \exp(f(x,y) \cdot \theta)$$

Sums over all possible taggings of all possible sentences!
A Big Picture: Sequence Model Estimation

unannotated data

tractable sums

stochastic, EM: $p(x)$

stochastic, MLE: $p(x, y)$

E: Expected Counts

M: Normalize

Count and Normalize®
A Big Picture: Sequence Model Estimation

unannotated data

tractable sums

EM

Optimize function

Compute Z

log-linear, EM: \( p(x) \)

log-linear, conditional estimation: \( p(y \mid x) \)

log-linear, MLE: \( p(x, y) \)

GEM

Compute Z

Optimize function

Compute Z

MLE: \( p(x, y) \)
A Big Picture: Sequence Model Estimation

unannotated data

tractable sums

stochastic, EM: $p(x)$

stochastic, MLE: $p(x, y)$

log-linear, EM: $p(x)$

log-linear, conditional estimation: $p(y \mid x)$

log-linear, MLE: $p(x, y)$

log-linear, CE with lattice neighborhoods

Optimize function
Contrastive Neighborhoods

• **Guide** the learner toward models that do what syntax is *supposed* to do.

• Lattice representation $\rightarrow$ **efficient** algorithms.

There is an **art** to choosing neighborhood functions.
### Neighborhoods

<table>
<thead>
<tr>
<th>Neighborhood</th>
<th>Size</th>
<th>Lattice arcs</th>
<th>Perturbations</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEL1WORD</td>
<td>$n+1$</td>
<td>$O(n)$</td>
<td>delete up to 1 word</td>
</tr>
<tr>
<td>TRANS1</td>
<td>$n$</td>
<td>$O(n)$</td>
<td>transpose any bigram</td>
</tr>
<tr>
<td>DEL1ORTRANS1</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$\text{DEL1WORD} \cup \text{TRANS1}$</td>
</tr>
<tr>
<td>DEL1SUBSEQUENCE</td>
<td>$O(n^2)$</td>
<td>$O(n^2)$</td>
<td>delete any contiguous subsequence</td>
</tr>
<tr>
<td>Σ* (MLE)</td>
<td>$\infty$</td>
<td>-</td>
<td>replace each word with anything</td>
</tr>
</tbody>
</table>
Optimizing Contrastive Likelihood

\[ F(\tilde{\theta}) = \sum_{i=1}^{n} \log p_{\tilde{\theta}}(X = x_i) - \log p_{\tilde{\theta}}(X \in \mathcal{N}(x_i)) \]

\[ \frac{\partial F}{\partial \theta_r} = \sum_{i=1}^{n} \mathbb{E}_{p_{\tilde{\theta}}} [f_r(x_i, Y)] - \mathbb{E}_{p_{\tilde{\theta}}} [f_r(X,Y) | X \in \mathcal{N}(x_i)] \]
The Merialdo (1994) Task

Given **unlabeled text**

and a **POS dictionary**
(that tells **all** possible tags for **each** word type),

learn to tag.

A form of supervision / domain knowledge.
Trigram Tagging Model

feature set:
tag trigrams
tag/word pairs from a POS dictionary
## Tagging Experiment

<table>
<thead>
<tr>
<th>Method</th>
<th>12K</th>
<th>24K</th>
<th>48K</th>
<th>96K</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRF (supervised)</td>
<td>100.0</td>
<td>99.8</td>
<td>99.8</td>
<td>99.5</td>
</tr>
<tr>
<td>HMM (supervised)</td>
<td>99.3</td>
<td>98.5</td>
<td>97.9</td>
<td>97.2</td>
</tr>
<tr>
<td>LENGTH</td>
<td>74.9</td>
<td>77.4</td>
<td>81.5</td>
<td>78.9</td>
</tr>
<tr>
<td>DEL1 Or TRANS1</td>
<td>70.8</td>
<td>70.8</td>
<td>78.3</td>
<td>79.1</td>
</tr>
<tr>
<td>TRANS1</td>
<td>72.7</td>
<td>72.7</td>
<td>77.2</td>
<td>78.1</td>
</tr>
<tr>
<td>EM</td>
<td>49.5</td>
<td>52.9</td>
<td>55.5</td>
<td>59.4</td>
</tr>
<tr>
<td>DEL1</td>
<td>55.4</td>
<td>55.6</td>
<td>60.3</td>
<td>59.9</td>
</tr>
<tr>
<td>DEL1 SUBSEQ</td>
<td>53.0</td>
<td>53.3</td>
<td>56.7</td>
<td>55.3</td>
</tr>
<tr>
<td>random expected</td>
<td>35.2</td>
<td>35.1</td>
<td>35.1</td>
<td>35.1</td>
</tr>
<tr>
<td>ambiguous words</td>
<td>6,244</td>
<td>12,923</td>
<td>25,879</td>
<td>51,521</td>
</tr>
</tbody>
</table>
So, why does LENGTH beat EM?

- the model is log-linear?

the objective function is better?
(don’t have to model # words)

functions essentially the same, but better search?
On Local Maxima

- Requiring weights to sum to one is simply a numerical constraint.

For bumpy functions, it’s preferable to have fewer constraints.
Trigram Tagging Model + Spelling

feature set:
- tag trigrams
- tag/word pairs from a POS dictionary
- 1- to 3-character suffixes, contains hyphen, digit
## Diluted Dictionary

### Tagging dictionary

| estimation     | model          | u-sel. | oracle | u-sel. | oracle | u-sel. | oracle | u-sel. | oracle | count | count |
|----------------|----------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| MAP/EM         | trigram        | 78.0   | 84.4   | 77.2   | 80.5   | 70.1   | 70.9   | 66.5   | 66.5   |        |        |
|                | trigram + spelling | 80.9 | 91.1   | 80.2   | 90.8   | 79.5   | 90.3   | 78.3   | 89.8   |        |        |
| CE/DEL1ORTRANS1| trigram        | 78.3   | 90.1   | 72.3   | 84.8   | 69.5   | 81.3   | 65.0   | 77.2   |        |        |
|                | trigram + spelling | 88.7 | 90.9   | 88.1   | 90.1   | 78.7   | 90.1   | 78.4   | 89.5   |        |        |
| CE/TRANS1      | trigram        | 90.4   | 90.4   | 80.8   | 82.9   | 77.0   | 78.6   | 71.7   | 73.4   |        |        |
|                | trigram + spelling | 87.8 | 90.4   | 68.1   | 78.3   | 65.3   | 75.2   | 62.8   | 72.3   |        |        |
| CE/LENGTH      | trigram        | 87.8   | 90.4   | 68.1   | 78.3   | 65.3   | 75.2   | 62.8   | 72.3   |        |        |
|                | trigram + spelling | 87.1 | 91.9   | 76.9   | 83.2   | 73.3   | 73.8   | 73.2   | 73.6   |        |        |
| random expected |                | 69.5   |        | 60.5   |        | 56.6   |        | 51.0   |        |        |        |
| ambiguous words |                | 13,150 |        | 13,841 |        | 14,780 |        | 15,996 |        |        |        |
| ave. tags/token |                | 2.3    |        | 3.7    |        | 4.4    |        | 5.5    |        |        |        |

(reduced, coarser tag set)
The sequence model need not be finite-state.

\( Y \) can range over trees.
Dependency Parsing

• Features (model from Klein and Manning, 2004):
  – (parent, child, direction) triples
  – “no children on left (right)”
  – “1 child on left (right)”
  – “multiple children on left (right)”

• Dynamic programming:
  – Eisner & Satta (1999) for inside algorithm
    (generalized for lattices)
• Dynamic programming saves the day again!

• If the set $N(x)$ is represented as a lattice, we can apply the usual Inside-Outside algorithm with a slight change.
<table>
<thead>
<tr>
<th></th>
<th>German test accuracy</th>
<th>English test accuracy</th>
<th>Bulgarian test accuracy</th>
<th>Mandarin test accuracy</th>
<th>Turkish test accuracy</th>
<th>Portuguese test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>directed</td>
<td>undirected</td>
<td>directed</td>
<td>undirected</td>
<td>directed</td>
<td>undirected</td>
</tr>
<tr>
<td>ATTACH-LEFT</td>
<td>8.2</td>
<td>59.1</td>
<td>22.6</td>
<td>62.1</td>
<td>37.2</td>
<td>61.0</td>
</tr>
<tr>
<td>ATTACH-RIGHT</td>
<td>47.0</td>
<td>55.2</td>
<td>39.5</td>
<td>62.1</td>
<td>23.8</td>
<td>61.0</td>
</tr>
<tr>
<td>Σ* (MAP/EM)</td>
<td>19.8</td>
<td>55.2</td>
<td>41.6</td>
<td>62.2</td>
<td>44.6</td>
<td>63.1</td>
</tr>
<tr>
<td>DEL1</td>
<td>26.5</td>
<td>43.7</td>
<td>18.3</td>
<td>33.9</td>
<td>13.1</td>
<td>31.5</td>
</tr>
<tr>
<td>TRANS1</td>
<td>17.9</td>
<td>53.0</td>
<td>29.4</td>
<td>57.8</td>
<td>23.8</td>
<td>61.0</td>
</tr>
<tr>
<td>DEL1OrTRANS1</td>
<td>59.3</td>
<td>72.6</td>
<td>47.3</td>
<td>63.6</td>
<td>24.2</td>
<td>60.0</td>
</tr>
<tr>
<td>LENGTH</td>
<td>49.2</td>
<td>64.1</td>
<td>45.5</td>
<td>64.9</td>
<td>27.0</td>
<td>60.1</td>
</tr>
<tr>
<td>DYNASEARCH</td>
<td>16.0</td>
<td>53.0</td>
<td>39.7</td>
<td>61.9</td>
<td>23.8</td>
<td>61.0</td>
</tr>
<tr>
<td></td>
<td>German test accuracy</td>
<td>English test accuracy</td>
<td>Bulgarian test accuracy</td>
<td>Mandarin test accuracy</td>
<td>Turkish test accuracy</td>
<td>Portuguese test accuracy</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>------------------------</td>
<td>------------------------</td>
<td>----------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td></td>
<td>directed</td>
<td>undirected</td>
<td>directed</td>
<td>undirected</td>
<td>directed</td>
<td>undirected</td>
</tr>
<tr>
<td><strong>ATTACH-LEFT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8.2</td>
<td>59.1</td>
<td>22.6</td>
<td>62.1</td>
<td>37.2</td>
<td>61.0</td>
</tr>
<tr>
<td><strong>ATTACH-RIGHT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>47.0</td>
<td>55.2</td>
<td>39.5</td>
<td>62.1</td>
<td>23.8</td>
<td>61.0</td>
</tr>
<tr>
<td><em><em>Σ</em> (MAP/EM)</em>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>54.4</td>
<td>71.9</td>
<td>41.6</td>
<td>62.2</td>
<td>45.6</td>
<td>63.6</td>
</tr>
<tr>
<td><strong>DEL1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>34.4</td>
<td>49.3</td>
<td>39.7</td>
<td>53.5</td>
<td>17.7</td>
<td>33.8</td>
</tr>
<tr>
<td><strong>TRANS1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>45.6</td>
<td>59.0</td>
<td>41.2</td>
<td>62.5</td>
<td>40.1</td>
<td>57.9</td>
</tr>
<tr>
<td><strong>DEL1ORTRANS1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>63.4</td>
<td>66.5</td>
<td>57.6</td>
<td>69.0</td>
<td>40.5</td>
<td>61.5</td>
</tr>
<tr>
<td><strong>LENGTH</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>57.3</td>
<td>65.1</td>
<td>45.5</td>
<td>64.9</td>
<td>38.3</td>
<td>63.4</td>
</tr>
<tr>
<td><strong>DYNASEARCH</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>45.7</td>
<td>58.6</td>
<td>47.6</td>
<td>65.3</td>
<td>34.0</td>
<td>58.0</td>
</tr>
<tr>
<td>s-sel. (N)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>63.4</td>
<td>66.5</td>
<td>57.6</td>
<td>69.0</td>
<td>40.5</td>
<td>61.5</td>
</tr>
</tbody>
</table>
Summing Up (Ha Ha)

• Contrastive estimation = designing a negative evidence class that keeps part of the data the same (e.g., semantics) but damages the part you want your model to learn (e.g., syntax).

• Idea of “implicit negative evidence” is central.