Lecture 3: Inside-Outside

Jelle Zuidema
ILLC, Universiteit van Amsterdam

Unsupervised Language Learning, 2013
Recap

Challenge
Inference
Generative models
Probabilistic Grammars

Inside-Outside
The Challenge (lecture 1)

Developing algorithms for learning about the syntax (semantics, pragmatics, phonology) of natural language from unlabeled data.

- Classic, hard problem in artificial intelligence
- Many unsuccessful attempts to develop heuristic algorithms for grammar induction.
Statistical Inference (lecture 1, assignment 1, readings)

- Statistical Models can deal with the noise and uncertainties (hidden information, “latent variables”) inherent in real world data;
- Statistical Inference offers a flexible toolbox of techniques and concepts:
  - Probabilities: likelihood, prior, posterior, data prior;
  - Criteria: maximum likelihood, MAP, minimum risk;
  - Optimization techniques: grid search, stochastic hillclimbing, EM, MCMC;
- Conceptual separation of objective function and optimization technique.
Generative models

- “Generative story”: we define probability distributions by describing the mechanism by which the data could have been generated.
- “Graphical models”: often graphs are used to define the statistical dependencies in the generative story.
Statistical inference

\[ P(G|D) = \frac{P(D|G)P(G)}{P(D)} \]
Statistical inference
Bayesian inversion
Generative model

Recap
Inside-Outside
Stochastic hillclimbing

$P(G|D)$
Stochastic hillclimbing
Stochastic hillclimbing
Stochastic hillclimbing
Stochastic hillclimbing

$P(G|D)$
Stochastic hillclimbing
Local optimum

\[ P(G|D) \]
Statistical inference

Bayesian inversion

Generative model

\[ P(G|D) \]

Recap
Inside-Outside

P(G|D)
Bayesian inversion
Generative model
P(D|G)
Statistical inference

Bayesian inversion

Generative model
Statistical inference

Bayesian inversion

Generative model
Probabilistic Grammars

- For natural language grammar, generative models are instantiated with probabilistic grammars
- The extended Chomsky hierarchy for symbolic grammars, is mirrored by a hierarchy of probabilistic grammars.
- Classes on the hierarchy are proper subsets of each other;
  - Corrolary: everything lower in the hierarchy can in principled be modelled by formalisms for probabilistic grammars higher in the hierarchy;
  - E.g., PCFGs can model ngrams and HMMs (and probabilistic bilexical dependency grammars).
Probabilistic Context Free Grammars

- Add probabilities to the rules of a context-free grammar;
- The PCFG now defines a probability distribution \((G_S)\) over trees (with \(S\) as root, and words as leaves);
- It also defines probability distributions over sentences;
- It also defines probability distributions \((G_A)\) over trees rooted in any other nonterminal \(A\);
- \(G_S\) can be defined using the probabilities of all rules with \(S\) as left-hand side and \(G_A \ldots G_Z\).
PCFGs as recursive mixtures

The distributions over strings induced by a PCFG in Chomsky-normal form (i.e., all productions are of the form $A \rightarrow BC$ or $A \rightarrow x$, where $A, B, C \in N$ and $x \in T$) is $G_S$ where:

$$G_A = \sum_{A \rightarrow BC \in R_A} pA \rightarrow BCG_B \cdot G_C + \sum_{A \rightarrow w \in R_A} pA \rightarrow x\delta_x$$

$$(P \cdot Q)(z) = \sum_{xy=z} P(x)Q(y)$$

$$\delta_x(w) = 1 \text{ if } w = x \text{ and } 0 \text{ otherwise}$$

In fact, $G_A(w) = P(A \Rightarrow^* w|\theta)$, the sum of the probability of all trees with root node $A$ and yield $w$. 
Things we want to compute with PCFGs

Given a PCFG $G$ and a string $w \in T^*$,

- (parsing): the most likely tree for $w$,
  \[ \arg\max_{\psi \in \Psi_G(w)} P_G(\psi) \]

- (language modeling): the probability of $w$,
  \[ P_G(w) = \sum_{\psi \in \Psi_G(w)} P_G(\psi) \]

Learning rule probabilities from data:

- (maximum likelihood estimation from visible data): given a corpus of trees $d = (\psi_1, \ldots, \psi_n)$, which rule probabilities $p$ makes $d$ as likely as possible?

- (maximum likelihood estimation from hidden data): given a corpus of strings $w = (w_1, \ldots, w_n)$, which rule probabilities $p$ makes $w$ as likely as possible?
Parsing and language modeling

The probability $P_G(\psi)$ of a tree $\psi \in \Psi_G(w)$ is:

$$P_G(\psi) = \prod_{r \in R} p(r)^{f_r(\psi)}$$

Suppose the set of parse trees $\Psi_G(w)$ is finite, and we can enumerate it.

Naive parsing/language modeling algorithms for PCFG $G$ and string $w \in T^*$:

1. Enumerate the set of parse trees $\Psi_G(w)$
2. Compute the probability of each $\psi \in \Psi_G(w)$
3. Argmax/sum as appropriate
A CFG is in *Chomsky Normal Form (CNF)* iff all productions are of the form $A \rightarrow B C$ or $A \rightarrow x$, where $A, B, C \in N$ and $x \in T$.

PCFGs *without epsilon productions* $A \rightarrow \epsilon$ can always be put into CNF.

Key step: *binarize* productions with more than two children by introducing new nonterminals.
Substrings and string positions

Let $w = w_1w_2 \ldots w_n$ be a string of length $n$.

A **string position** for $w$ is an integer $i \in 0, \ldots, n$ (informally, it identifies the position between words $w_{i-1}$ and $w_i$).

```
- the - dog - chases - cats -
0 1 2 3 4
```

A **substring** of $w$ can be specified by beginning and ending string positions.

$w_{i,j}$ is the substring starting at word $i + 1$ and ending at word $j$.

- $w_{0,4} = \text{the dog chases cats}$
- $w_{1,2} = \text{dog}$
- $w_{2,4} = \text{chases cats}$
Language modeling using dynamic programming

- **Goal:** To compute $P_G(w) = \sum_{\psi \in \Psi_G(w)} P_G(\psi) = P_G(S \Rightarrow^* w)$

- **Data structure:** A table called a *chart* recording $P_G(A \Rightarrow^* w_{i,k})$ for all $A \in N$ and $0 \leq i < k \leq |w|$

- **Base case:** For all $i = 1, \ldots, n$ and $A \rightarrow w_i$, compute:

  $$P_G(A \Rightarrow^* w_{i-1,i}) = p(A \rightarrow w_i)$$

- **Recursion:** For all $k - i = 2, \ldots, n$ and $A \in N$, compute:

  $$P_G(A \Rightarrow^* w_{i,k}) = \sum_{j=i+1}^{k-1} \sum_{A \rightarrow B C \in R(A)} p(A \rightarrow B C) P_G(B \Rightarrow^* w_{i,j}) P_G(C \Rightarrow^* w_{j,k})$$
Dynamic programming recursion

\[ P_G(A \Rightarrow^* w_{i,k}) = \sum_{j=i+1}^{k-1} \sum_{A \rightarrow B C \in R(A)} p(A \rightarrow B C) P_G(B \Rightarrow^* w_{i,j}) P_G(C \Rightarrow^* w_{j,k}) \]

\[ P_G(A \Rightarrow^* w_{i,k}) \text{ is called the inside probability of } A \text{ spanning } w_{i,k}. \]
Example PCFG string probability calculation

\[ w = \text{George hates John} \]

\[ R = \begin{cases} 
1.0 & \text{S} \rightarrow \text{NP VP} \\
0.7 & \text{NP} \rightarrow \text{George} \\
0.5 & \text{V} \rightarrow \text{likes} \\
1.0 & \text{VP} \rightarrow \text{V NP} \\
0.3 & \text{NP} \rightarrow \text{John} \\
0.5 & \text{V} \rightarrow \text{hates} 
\end{cases} \]
Intermediate Summary

- PCFGs define probability distributions over trees, subtrees and strings. These distributions can be viewed as recursive mixtures.
- The probability of a (sub)string can be calculated the naive way by summing the probabilities of all the trees that contain it;
- We can make use of the recursive nature of PCFG distributions to calculate string probabilities more efficiently: the inside algorithm.
Computational complexity of PCFG parsing

\[ P_G(A \Rightarrow^* w_{i,k}) = \sum_{j=i+1}^{k-1} \sum_{A \to B \in R(A)} p(A \to B C) P_G(B \Rightarrow^* w_{i,j}) P_G(C \Rightarrow^* w_{j,k}) \]

For each production \( r \in R \) and each \( i, k \), we must sum over all intermediate positions \( j \Rightarrow O(n^3 |R|) \) time
Estimating (learning) PCFGs from data

Estimating productions and production probabilities from visible data (corpus of parse trees) is straight-forward:

- The productions are identified by the local trees in the data
- Maximum likelihood principle: select production probabilities in order to make corpus as likely as possible
- Bayesian estimators often produce more useful estimates

Estimating production probabilities from hidden data (corpus of terminal strings) is much more difficult:

- The Expectation-Maximization (EM) algorithm finds probabilities that locally maximize likelihood of corpus
- The Inside-Outside algorithm runs in time polynomial in length of corpus
- Bayesian estimators have recently been developed

Estimating the productions from hidden data is an open problem.
Estimating PCFGs from visible data

Data: A treebank of parse trees Ψ = ψ₁, ..., ψₙ.

\[
L(p) = \prod_{i=1}^{n} P_G(\psi_i) = \prod_{A \rightarrow \alpha \in R} p(A \rightarrow \alpha)^{f_{A \rightarrow \alpha}(\Psi)}
\]

Introduce \(|N|\) Lagrange multipliers \(c_B, B \in N\) for the constraints \(\sum_{B \rightarrow \beta \in R(B)} p(B \rightarrow \beta) = 1:\)

\[
\frac{\partial}{\partial p(A \rightarrow \alpha)} \left( L(p) - \sum_{B \in N} c_B \left( \sum_{B \rightarrow \beta \in R(B)} p(B \rightarrow \beta) - 1 \right) \right) = \frac{L(p) f_r(\Psi)}{p(A \rightarrow \alpha)} - c_A
\]

Setting this to 0,

\[
p(A \rightarrow \alpha) = \frac{f_{A \rightarrow \alpha}(\Psi)}{\sum_{A \rightarrow \alpha' \in R(A)} f_{A \rightarrow \alpha'}(\Psi)}
\]
Visible PCFG estimation example

\[ \Psi = \]

\[
\begin{cases}
S \\
NP & \text{rice grows} \\
VP
\end{cases}
\]

\[
\begin{cases}
S \\
NP & \text{rice grows} \\
VP
\end{cases}
\]

\[
\begin{cases}
S \\
NP & \text{corn grows} \\
VP
\end{cases}
\]

<table>
<thead>
<tr>
<th>Rule</th>
<th>Count</th>
<th>Rel Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>NP → rice</td>
<td>2</td>
<td>2/3</td>
</tr>
<tr>
<td>NP → corn</td>
<td>1</td>
<td>1/3</td>
</tr>
<tr>
<td>VP → grows</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
P \left( \begin{array}{c}
S \\
NP & \text{rice grows} \\
VP
\end{array} \right) = \frac{2}{3}
\]

\[
P \left( \begin{array}{c}
S \\
NP & \text{corn grows} \\
VP
\end{array} \right) = \frac{1}{3}
\]
Estimating production probabilities from hidden data

Data: A corpus of sentences \( \mathbf{w} = w_1, \ldots, w_n \).

\[
L(\mathbf{w}) = \prod_{i=1}^{n} P_G(w_i). \quad P_G(\mathbf{w}) = \sum_{\psi \in \Psi_G(\mathbf{w})} P_G(\psi).
\]

\[
\frac{\partial L(\mathbf{w})}{\partial p(A \rightarrow \alpha)} = \frac{L(\mathbf{w}) \sum_{i=1}^{n} E_G(f_{A \rightarrow \alpha} | w_i)}{p(A \rightarrow \alpha)}
\]

Setting this equal to the Lagrange multiplier \( c_A \) and imposing the constraint \( \sum_{B \rightarrow \beta \in R(B)} p(B \rightarrow \beta) = 1 \):

\[
p(A \rightarrow \alpha) = \frac{\sum_{i=1}^{n} E_G(f_{A \rightarrow \alpha} | w_i)}{\sum_{A \rightarrow \alpha' \in R(A)} \sum_{i=1}^{n} E_G(f_{A \rightarrow \alpha'} | w_i)}
\]

This is an iteration of the expectation maximization algorithm!
The EM algorithm for PCFGs

Input: a corpus of strings \( \mathbf{w} = \mathbf{w}_1, \ldots, \mathbf{w}_n \)

Guess initial production probabilities \( p^{(0)} \)

For \( t = 1, 2, \ldots \) do:

1. Calculate expected frequency \( \sum_{i=1}^{n} E_{p^{(t-1)}}(f_{A \rightarrow \alpha} | \mathbf{w}_i) \) of each production:

\[
E_{p}(f_{A \rightarrow \alpha} | \mathbf{w}) = \sum_{\psi \in \Psi_G(\mathbf{w})} f_{A \rightarrow \alpha}(\psi) p(\psi)
\]

2. Set \( p^{(t)} \) to the relative expected frequency of each production

\[
p^{(t)}(A \rightarrow \alpha) = \frac{\sum_{i=1}^{n} E_{p^{(t-1)}}(f_{A \rightarrow \alpha} | \mathbf{w}_i)}{\sum_{A \rightarrow \alpha'} \sum_{i=1}^{n} E_{p^{(t-1)}}(f_{A \rightarrow \alpha'} | \mathbf{w}_i)}
\]

It is as if \( p^{(t)} \) were estimated from a visible corpus \( \Psi_G \) in which each tree \( \psi \) occurs \( \sum_{i=1}^{n} p^{(t-1)}(\psi | \mathbf{w}_i) \) times.
Dynamic programming for $E_p(f_{A \rightarrow B C} | w)$

\[
E_p(f_{A \rightarrow B C} | w) = \sum_{0 \leq i < j < k \leq n} P(S \Rightarrow^* w_{1,i} A w_{k,n}) p(A \rightarrow B C) P(B \Rightarrow^* w_{i,j}) P(C \Rightarrow^* w_{j,k})
\]

\[
P_G(w)
\]
Calculating “outside probabilities”

Construct a table of “outside probabilities”
\[ P_G(S \Rightarrow^* w_{0,i} A w_{k,n}) \] for all \( 0 \leq i < k \leq n \) and \( A \in N \)

Recursion from *larger to smaller* substrings in \( w \).

**Base case:** \( P(S \Rightarrow^* w_{0,0} S w_{n,n}) = 1 \)

**Recursion:** \( P(S \Rightarrow^* w_{0,j} C w_{k,n}) = \)

\[
\sum_{i=0}^{j-1} \sum_{A,B \in N, A \rightarrow B \in R} P(S \Rightarrow^* w_{0,i} A w_{k,n}) p(A \rightarrow B) P(B \Rightarrow^* w_{i,j})
\]

\[
+ \sum_{l=k+1}^{n} \sum_{A,D \in N, A \rightarrow C \in R} P(S \Rightarrow^* w_{0,j} A w_{l,n}) p(A \rightarrow C D) P(D \Rightarrow^* w_{k,l})
\]
Recursion in $P_G(S \Rightarrow^* w_{0,i} A w_{k,n})$

$$P(S \Rightarrow^* w_{0,j} C w_{k,n}) = \sum_{i=0}^{j-1} \sum_{A,B \in N \atop A \rightarrow BC \in R} P(S \Rightarrow^* w_{0,i} A w_{k,n}) p(A \rightarrow BC) P(B \Rightarrow^* w_{i,j})$$

$$+ \sum_{l=k+1}^{n} \sum_{A,D \in N \atop A \rightarrow CD \in R} P(S \Rightarrow^* w_{0,j} A w_{l,n}) p(A \rightarrow CD) P(D \Rightarrow^* w_{k,l})$$
Example: The EM algorithm with a toy PCFG

Initial rule probs

<table>
<thead>
<tr>
<th>rule</th>
<th>prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>VP → V</td>
<td>0.2</td>
</tr>
<tr>
<td>VP → V NP</td>
<td>0.2</td>
</tr>
<tr>
<td>VP → NP V</td>
<td>0.2</td>
</tr>
<tr>
<td>VP → V NP NP</td>
<td>0.2</td>
</tr>
<tr>
<td>VP → NP NP V</td>
<td>0.2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Det → the</td>
<td>0.1</td>
</tr>
<tr>
<td>N → the</td>
<td>0.1</td>
</tr>
<tr>
<td>V → the</td>
<td>0.1</td>
</tr>
</tbody>
</table>

“English” input
the dog bites
the dog bites a man
a man gives the dog a bone
...

“pseudo-Japanese” input
the dog a man bites
a man the dog a bone gives
...
Probability of "English"

Average sentence probability vs. Iteration.
Rule probabilities from “English”

![Graph showing rule probabilities over iterations](image-url)
Probability of “Japanese”
Rule probabilities from “Japanese”

![Graph showing rule probabilities for different grammar rules over iterations, with axes labeled as follows: Rule probability on the y-axis, Iteration on the x-axis. The graph includes lines for different grammar rules, such as VP → V NP, VP → NP V, VP → V NP NP, VP → NP NP V, Det → the, N → the, V → the, plotted over iterations.]
Learning in statistical paradigm

- The likelihood is a differentiable function of rule probabilities
  ⇒ learning can involve small, incremental updates
- Learning structure (rules) is hard, but . . .
- Parameter estimation can approximate rule learning
  - start with “superset” grammar
  - estimate rule probabilities
  - discard low probability rules
- Non-parametric Bayesian estimators combine parameter and rule estimation