Dynamic Programming for Parsing and Estimation of Stochastic Unification-Based Grammars

Stuart Geman and Mark Johnson
Brown University

ACL’02, Philadelphia

Acknowledgements: Stefan Riezler (Parc)
NSF grants DMS 0074276 and ITR IIS 0085940
Talk outline

- Stochastic Unification-Based Grammars (SUBGs)
- Parsing and estimation of SUBGs
- Avoiding enumerating all parses
  - Feature locality
  - Parse weight is a product of functions of parse fragment weights
  - Graphical model calculation of argmax/sum

let us take the fifteen.

Tuesday, the

the fifteen
Stochastic Unification-based Grammars

- A **unification-based grammar** defines a **set of possible parses** $\mathcal{Y}(w)$ for each sentence $w$.

- **Features** $f_1, \ldots, f_m$ are real-valued functions on parses
  - Attachment location (high, low, argument, adjunct, etc.)
  - Head-to-head dependencies

- Probability defined by **conditional log-linear model**
  $$W(y) = \exp\left(\sum_{j=1}^{m} \lambda_j f_j(y)\right) = \prod_{j=1}^{m} \theta_j^{f_j(y)}$$

  $$\Pr(y|w) = W(y) / Z(w)$$

  where $\theta_j = e^{\lambda_j} > 0$ are **feature weights** and

  $Z(w) = \sum_{y \in \mathcal{Y}(w)} W(y)$ is the **partition function**.

Estimating feature weights

• Several algorithms for maximum conditional likelihood estimation
  – Various iterative scaling algorithms
  – Conjugate gradient and other optimization algorithms

• These algorithms are iterative
  ⇒ repeated reparsing of training data

• All of these algorithms require conditional expectations

\[ E[f_j | w] = \sum_{y \in \mathcal{Y}(w)} f_j(y) \Pr(y | w) \]

• Can we calculate these statistics and find the most likely parse without enumerating all parses \( \mathcal{Y}(w) \)? YES *
Maxwell and Kaplan packed parses

- A parse $y$ consists of set of fragments $\xi \in y$ (MK algorithm)
- A fragment is in a parse when its context function is true
- Context functions are functions of context variables $X_1, X_2, \ldots$
- The variable assignment must satisfy “not no-good” functions
- Each parse is identified by a unique context variable assignment

\[ \xi = \text{“the cat on the mat”} \]
\[ \xi_1 = \text{“with a hat”} \]
\[ X_1 \rightarrow \text{“attach } D \text{ to } B” \]
\[ \neg X_1 \rightarrow \text{“attach } D \text{ to } A” \]
Feature locality

- Features must be *local* to fragments: \( f_j(y) = \sum_{\xi \in y} f_j(\xi) \)
- May require changes to UBG to make all features local

\[ \xi = \text{“the cat on the mat”} \]
\[ \xi_1 = \text{“with a hat”} \]

\[ X_1 \rightarrow \text{“attach } D \text{ to } B” \land (\xi_1 \text{ ATTACH}) = \text{LOW} \]
\[ \neg X_1 \rightarrow \text{“attach } D \text{ to } A” \land (\xi_1 \text{ ATTACH}) = \text{HIGH} \]
Feature locality decomposes $W(y)$

- Feature locality: the weight of a parse is the product of weights of its fragments

\[
W(y) = \prod_{\xi \in y} W(\xi), \quad \text{where}
\]

\[
W(\xi) = \prod_{j=1}^{m} \theta_{f_j}^{f_j}(\xi)
\]

\[
W(\xi = \text{“the cat on the mat”})
\]

\[
W(\xi_1 = \text{“with a hat”})
\]

\[
X_1 \quad \rightarrow \quad W(\text{“attach D to B”} \land (\xi_1 \text{ ATTACH}) = \text{LOW})
\]

\[
\neg X_1 \quad \rightarrow \quad W(\text{“attach D to A”} \land (\xi_1 \text{ ATTACH}) = \text{HIGH})
\]
"Not no-goods" identify the variable assignments that correspond to parses

\[ \xi = \text{"I read a book"} \]
\[ \xi_1 = \text{"on the table"} \]
\[ X_1 \land X_2 \rightarrow \text{"attach D to B"} \]
\[ X_1 \land \neg X_2 \rightarrow \text{"attach D to A"} \]
\[ \neg X_1 \rightarrow \text{"attach D to C"} \]
\[ X_1 \lor X_2 \]

I read a book

on the table

A

B

C

D
Identify parses with variable assignments

- *Each variable assignment uniquely identifies a parse*

- For a given sentence $w$, let $W'(x) = W(y)$ where $y$ is the parse identified by $x$

  $\Rightarrow$ Argmax/sum/expectations over parses can be computed over context variables instead

**Most likely parse:** $\hat{x} = \text{argmax}_x W'(x)$

**Partition function:** $Z(w) = \sum_x W'(x)$

**Expectation:** $E[f_j|w] = \sum_x f_j(x)W'(x)/Z(w)$
$W'$ is a product of functions of $X$

- Then $W'(X) = \prod_{A \in A} A(X)$, where:
  - Each line $\alpha(X) \rightarrow \xi$ introduces a term $W(\xi)^{\alpha(X)}$
  - A “not no-good” $\eta(X)$ introduces a term $\eta(X)$

\[
\begin{align*}
\vdots & \quad \vdots \\
\alpha(X) & \rightarrow \xi \quad \times \quad W(\xi)^{\alpha(X)} \\
\vdots & \quad \times \quad \vdots \\
\eta(X) & \quad \times \quad \eta(X) \\
\vdots & \quad \times \quad \vdots 
\end{align*}
\]

$\Rightarrow \ W'$ is a Markov Random Field over the context variables $X$
$W'$ is a product of functions of $X$

$W'(X_1) = W(\xi = "\text{the cat on the mat"})$
$\times W(\xi_1 = "\text{with a hat"})$
$\times W ("\text{attach } D \text{ to } B" \land (\xi_1 \text{ ATTACH}) = \text{LOW})^{X_1}$
$\times W ("\text{attach } D \text{ to } A" \land (\xi_1 \text{ ATTACH}) = \text{HIGH})^{\neg X_1}$

![Diagram](attachment:diagram.png)
Product expressions and graphical models

- MRFs are products of terms, each of which is a function of (a few) variables
- Graphical models provide *dynamic programming algorithms* for Markov Random Fields (MRF) (Pearl 1988)
- These algorithms implicitly *factorize the product*
- They generalize the Viterbi and Forward-Backward algorithms to arbitrary graphs (Smyth 1997)

\[\Rightarrow\] Graphical models provide *dynamic programming techniques for parsing and training Stochastic UBGs*
Factorization example

\[ W'(X_1) = \]
\[ \times W(\xi_1 = "with a hat") \]
\[ \times W("attach D to B" \land (\xi_1 \text{ ATTACH}) = \text{LOW})^{X_1} \]
\[ \times W("attach D to A" \land (\xi_1 \text{ ATTACH}) = \text{HIGH})^{\neg X_1} \]

\[ \max_{X_1} W'(X_1) = \]
\[ \times W(\xi = "the cat on the mat") \]
\[ \times W(\xi_1 = "with a hat") \]
\[ \times \max_{X_1} \left( W("attach D to B" \land (\xi_1 \text{ ATTACH}) = \text{LOW})^{X_1}, \right. \]
\[ \left. W("attach D to A" \land (\xi_1 \text{ ATTACH}) = \text{HIGH})^{\neg X_1} \right) \]
Dependency structure graph $G_A$

\[
Z(w) = \sum_x W'(x) = \sum_x \prod_{A \in A} A(x)
\]

- $G_A$ is the dependency graph for $A$
  - context variables $X$ are vertices of $G_A$
  - $G_A$ has an edge $(X_i, X_j)$ if both are arguments of some $A \in A$

\[
A(X) = a(X_1, X_3)b(X_2, X_4)c(X_3, X_4, X_5)d(X_4, X_5)e(X_6, X_7)
\]
Graphical model computations

\[
Z = \sum_x a(x_1, x_3)b(x_2, x_4)c(x_3, x_4, x_5)d(x_4, x_5)e(x_6, x_7)
\]

\[
Z_1(x_3) = \sum_{x_1} a(x_1, x_3)
\]

\[
Z_2(x_4) = \sum_{x_2} b(x_2, x_4)
\]

\[
Z_3(x_4, x_5) = \sum_{x_3} c(x_3, x_4, x_5)Z_1(x_3)
\]

\[
Z_4(x_5) = \sum_{x_4} d(x_4, x_5)Z_2(x_4)Z_3(x_4, x_5)
\]

\[
Z_5 = \sum_{x_5} Z_4(x_5)
\]

\[
Z_6(x_7) = \sum_{x_6} e(x_6, x_7)
\]

\[
Z_7 = \sum_{x_7} Z_6(x_7)
\]

\[
Z = Z_5Z_7
\]

\[
= (\sum_{x_5} Z_4(x_5)) (\sum_{x_7} Z_6(x_7))
\]

See: Pearl (1988) *Probabilistic Reasoning in Intelligent Systems*
Graphical model for Homecentre example

Use a damp, lint-free cloth to wipe the dust and dirt buildup from the scanner plastic window and rollers.
Computational complexity

- Polynomial in $m = \text{the maximum number of variables in the dynamic programming functions} \geq \text{the number of variables in any function } A$

- $m$ depends on the ordering of variables (and $G$)

- Finding the variable ordering that minimizes $m$ is NP-complete, but there are good heuristics

$\Rightarrow$ Worst case exponential (no better than enumerating the parses), but average case might be much better

  - Much like UBG parsing complexity
Conclusion

- There are DP algorithms for parsing and estimation from packed parses that avoid enumerating parses
  - Generalizes to all Truth Maintenance Systems (not grammar specific)

- Features must be local to parse fragments
  - May require adding features to the grammar

- Worst-case exponential complexity; average case?

- Makes available techniques for graphical models to packed parse representations
  - MCMC and other sampling techniques
Future directions

- Reformulate “hard” grammatical constraints as “soft” stochastic features
  - Underlying grammar permits all possible structural combinations
  - Grammatical constraints reformulated as stochastic features
- Is this computation tractable?
- Comparison with Miyao and Tsujii (2002)