Inference and Representation: Lab 4

Yacine Jernite

September 25, 2014
Lecture plan

- Part 1: Variable elimination heuristics
- Part 2: Dynamic programming and inference: PCFG
Goal: efficient computing of

\[ Z = \sum \prod_{x \in C} \phi_c(x_c) \]

Find ordering \( \pi \) of variables that allows us to “get rid” of nodes:

\[ Z = \prod_{i=1}^{N} \sum_{x_{\pi(i)}} \prod_{c \ni x_{\pi(i)}} \phi_c(x_c) \]
Variable elimination in MRFs

Figure: Here, \( \sum_x \prod_{c \in C} \phi_c(x_c) = \sum_{x^A} \left( \prod_{c \in C \setminus (A,B)} \phi_c(x_c) \times \tau_1(x_B) \right) \), where \( \forall x_B, \tau_1(x_B) = \sum_{x_A} \phi_{(A,B)}(x_A, x_B) \phi_A(x_A) \).
Variable elimination in MRFs

Figure: Introducing fill edges

A - B - C - D - E - F - G - H

B - C - D - E - F - G - H

B - C - D - E - F

G - H - E - F

Yacine Jernite
Inference and Representation: Lab 4
Variable elimination in MRFs: ordering

- Different orderings induce different complexities
- Finding the best ordering is NP-hard
- Several heuristics, greedy and beam search with criteria:
  - **Min-neighbours**: Start with least number of neighbours.
  - **Min-fill**: Start with least number of necessary fill edges.
  - **Min-weight**: Alternative to **Min-neighbours**, where vertices are weighted by variable domain cardinality
  - **Weighted-min-fill**: Edges are weighted by the produce of the cardinalities of both variables
Variable elimination in MRFs: ordering

- No *a priori* way to choose
- Define a cost, run algorithms, compare
- Possible costs:
  - Computational complexity
  - Treewidth as a proxy
Eliminating variable $i$ with neighbours $\mathcal{N}(i)$:

$$C(i) = |X_i| \times \prod_{j \in \mathcal{N}(i)} |X_j|$$

If all variables have the same cardinality $c$:

$$C(i) = c^{|\mathcal{N}(i)+1|}$$

For $n$ nodes and an induced width $w$:

$$C \leq nc^w$$

Which is the most natural heuristic? Justify the others.
Variable elimination in MRFs: complexity

Figure: Comparing heuristics
Dynamic programming and inference: PCFG

- General inference problem
- Other probabilistic models can use dynamic programming
Introducing the problem: Syntactic Parsing

```
S
 /\  
NP  VP
 /\  /
N   NP
|   /\  
she V  the
  |  |   |
  killed Art  N
       |   /
       |   P
       |   /
       |   the
       |   Art
       |   N
       |   the
       |   tie
```
Introducing the problem: Syntactic Parsing.

- Two main approaches: dependency and constituency

- Devising a rule-based system

Grammar formulation

Problems with deterministic generative process

Probabilistic model
Grammar:
- Symbols $\mathcal{V}$
- Start symbols $S \in \mathcal{V}$
- Terminals $\mathcal{T} \in \mathcal{V}$
- Rules $\mathcal{R}$: where $R : \mathcal{V} \setminus \mathcal{T} \rightarrow (\mathcal{V} \setminus S)^k$

Probabilistic context free grammar:
- $R$ defines a probability distribution:
  $\forall v \in \mathcal{V} \setminus \mathcal{T}, \sum_w R(v, w) = 1$
- $R$ does not depend on context
PCFG: MAP inference problem

- Parse tree defined by structure and symbols
- Difficult to express as a graphical model

\[ P(tree) = \prod_{n \in \text{nodes}} R(n, \text{children}_n) \]

- Problem: exploring all trees
Solution: dynamic programming

CKY (Cocke-Kasami-Younger) algorithm:

\[
\max_{\text{tree}} P(\text{tree}) = \max_{\text{root,children}} R(\text{root,children}) \max_{\text{left subtree}} P(\text{left subtree}) \times \max_{\text{right subtree}} P(\text{right subtree})
\]
PCFG: MAP inference problem

Whiteboard