Decoding in Latent Conditional Models: A Practically Fast Solution for an NP-hard Problem

Xu Sun (孫栩)
University of Tokyo
2010.06.16
Outline

• Introduction

• Related Work & Motivations

• Our proposals

• Experiments

• Conclusions
Latent dynamics

• Latent-structures (latent dynamics here) are important in information processing
  – Natural language processing
  – Data mining
  – Vision recognition

• Modeling latent dynamics: Latent-dynamic conditional random fields (LDCRF)
Latent dynamics

- **Latent-structures** (latent dynamics here) are important in information processing

Parsing: Learn refined grammars with latent info

He heard the voice

\[
S \\
NP \quad VP \\
PRP \quad VBD \quad NP \\
He \quad heard \quad DT \quad NN \\
the \quad voice
\]
Latent dynamics

- **Latent-structures** (latent dynamics here) are important in information processing.

Parsing: Learn refined grammars with latent info.

```
S-x
   NP-x  VP-x
      PRP-x VBD-x NP-x
  He  heard DT-x NN-x
      the voice
```
More common cases: linear-chain latent dynamics

• The previous example is a tree-structure
• More common cases could be linear-chain latent dynamics
  – Named entity recognition
  – Phrase segmentation
  – Word segmentation

These are her flowers.

Phrase segmentation [Sun+ COLING 08]
A solution without latent annotation: Latent-dynamic CRFs

A solution: Latent-dynamic conditional random fields (LDCRFs)
[Morency+ CVPR 07]
* No need to annotate latent info

Phrase segmentation [Sun+ COLING 08]
Current problem & our target

A solution: Latent-dynamic conditional random fields (LDCRFs)
[Morency+ CVPR 07]
* No need to annotate latent info

Current problem:
Inference (decoding) is an NP-hard problem.

Our target:
An *almost exact* inference method with fast speed.
Outline

• Introduction

• Related Work & Motivations

• Our proposals

• Experiments

• Conclusions
Traditional methods

• Traditional sequential labeling models

  – Hidden Markov Model (HMM)
    [Rabiner IEEE 89]
  – Maximum Entropy Model (MEM)
    [Ratnaparkhi EMNLP 96]
  – Conditional Random Fields (CRF)
    [Lafferty+ ICML 01]
  – Collins Perceptron
    [Collins EMNLP 02]

Arguably the most accurate one.
We will use it as one of the baseline.
Conditional random field (CRF)  
[Lafferty+ ICML 01]

\[
P(y \mid x, \theta) = \frac{1}{Z(x, \theta)} \exp \left( \sum_k \theta_k F_k (y, x) \right)
\]

Problem: CRF does not model latent info
Latent-Dynamic CRFs

[Morency+ CVPR 07]
Latent-Dynamic CRFs

[Morency+ CVPR 07]

We can think (informally) it as “CRF + unsup. learning on latent info”
Latent-Dynamic CRFs

[Morency+ CVPR 07]

\[
P(y \mid x, \theta) = \sum_{h : \forall h_j \in \mathcal{H}_{y_j}} P(h \mid x, \theta) = \sum_{h : \forall h_j \in \mathcal{H}_{y_j}} \frac{1}{Z(x, \theta)} \exp \left( \sum_k \theta_k F_k (h, x) \right)
\]

Good performance reports

* Outperforming HMM, MEMM, SVM, CRF, etc.
* Syntactic parsing [Petrov+ NIPS 08]
* Syntactic chunking [Sun+ COLING 08]
* Vision object recognition [Morency+ CVPR 07; Quattoni+ PAMI 08]
Outline

• Introduction

• Related Work & Motivations

• Our proposals

• Experiments

• Conclusions
Inference problem

- Prob: Exact inference (find the sequence with max probability) is **NP-hard**!
  - no fast solution existing

Recent fast solutions are only approximation methods:
- *Best Hidden Path* [Matsuzaki+ ACL 05]
- *Best Marginal Path* [Morency+ CVPR 07]
Related work 1: Best hidden path (BHP)  

[Matuzaki+ ACL 05]

<table>
<thead>
<tr>
<th>Seg-0</th>
<th>○</th>
<th>○</th>
<th>○</th>
<th>○</th>
<th>○</th>
<th>○</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seg-1</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Seg-2</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>noSeg-0</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>noSeg-1</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>noSeg-2</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

These are her flowers.
Related work 1: Best hidden path (BHP) [Matsuzaki+ ACL 05]

Result:
Seg Seg Seg Seg NoSeg Seg
Related work 2: Best marginal path (BMP)

[Morency+ CVPR 07]

<table>
<thead>
<tr>
<th></th>
<th>Seg-0</th>
<th>Seg-1</th>
<th>Seg-2</th>
<th>noSeg-0</th>
<th>noSeg-1</th>
<th>noSeg-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>These</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>are</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>her</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>flowers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

These are her flowers.
Related work 2: Best marginal path (BMP)

[Morency+ CVPR 07]

<table>
<thead>
<tr>
<th></th>
<th>Seg-0</th>
<th>Seg-1</th>
<th>Seg-2</th>
<th>noSeg-0</th>
<th>noSeg-1</th>
<th>noSeg-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seg-0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.4</td>
<td>0.2</td>
<td>0.2</td>
<td>0.0</td>
</tr>
<tr>
<td>Seg-1</td>
<td>0.6</td>
<td>0.1</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.1</td>
</tr>
<tr>
<td>Seg-2</td>
<td>0.2</td>
<td>0.5</td>
<td>0.0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.5</td>
</tr>
<tr>
<td>noSeg-0</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
<td>0.1</td>
<td>0.0</td>
<td>0.2</td>
</tr>
<tr>
<td>noSeg-1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.7</td>
<td>0.7</td>
<td>0.0</td>
</tr>
<tr>
<td>noSeg-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Result:
Seg Seg Seg Seg NoSeg Seg
Our target

- Prob: Exact inference (find the sequence with max probability) is NP-hard!

  1) Exact inference
  2) Comparable speed to existing approximation methods

Challenge/Difficulty:
Exact & practically-fast solution on an NP-hard problem
Outline

• Introduction
• Related Work & Motivations
  • Our proposals
• Experiments
• Conclusions
Essential ideas
[Sun+ EACL 09]

- **Fast & exact inference** from a key observation
  - A key observation on prob. Distribution
  - **Dynamic** top-n search
  - Fast decision on optimal result from top-n candidates
Key observation

• Natural problems (e.g., NLP problems) are not completely ambiguous

• Normally, **Only a few** result candidate are highly probable

• Therefore, probability distribution on latent models could be **sharp**
Key observation

- Probability distribution on latent models is *sharp*

<table>
<thead>
<tr>
<th>These</th>
<th>are</th>
<th>her</th>
<th>flowers</th>
<th>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>seg</td>
<td>noSeg</td>
<td>seg</td>
<td>seg</td>
<td>seg</td>
</tr>
<tr>
<td>seg</td>
<td>seg</td>
<td>seg</td>
<td>noSeg</td>
<td>seg</td>
</tr>
<tr>
<td>seg</td>
<td>seg</td>
<td>seg</td>
<td>seg</td>
<td>seg</td>
</tr>
<tr>
<td>seg</td>
<td>seg</td>
<td>noSeg</td>
<td>noSeg</td>
<td>seg</td>
</tr>
<tr>
<td>seg</td>
<td>noSeg</td>
<td>seg</td>
<td>noSeg</td>
<td>seg</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- \( P = 0.2 \)
- \( P = 0.3 \)
- \( P = 0.2 \)
- \( P = 0.1 \)
- \( P = \ldots \)
- \( P = \ldots \)

\[0.8\] prob
Key observation

- Probability distribution on latent models is sharp.

- Challenge: the number of probable candidates are unknown & changing.

- Need a method which can automatically adapt itself on different cases.

<table>
<thead>
<tr>
<th>seg</th>
<th>noSeg</th>
<th>seg</th>
<th>seg</th>
<th>seg</th>
</tr>
</thead>
<tbody>
<tr>
<td>seg</td>
<td>seg</td>
<td>seg</td>
<td>noSeg</td>
<td>seg</td>
</tr>
<tr>
<td>seg</td>
<td>seg</td>
<td>seg</td>
<td>seg</td>
<td>seg</td>
</tr>
<tr>
<td>seg</td>
<td>seg</td>
<td>noSeg</td>
<td>noSeg</td>
<td>seg</td>
</tr>
<tr>
<td>seg</td>
<td>noSeg</td>
<td>seg</td>
<td>noSeg</td>
<td>seg</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

P = 0.2
P = 0.3
P = 0.2
P = 0.1
P = ...

P(unknown) ≤ 0.2
A demo on lattice

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Seg-0</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Seg-1</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Seg-2</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>noSeg-0</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>noSeg-1</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>noSeg-2</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

These are her flowers.
### (1) Admissible heuristics for A* search

<table>
<thead>
<tr>
<th>Seg-0</th>
<th>○</th>
<th>○</th>
<th>○</th>
<th>○</th>
<th>○</th>
<th>○</th>
<th>○</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seg-1</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Seg-2</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>noSeg-0</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>noSeg-1</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>noSeg-2</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

These are her flowers.
(1) Admissible heuristics for A* search

These are her flowers.

Viterbi algo. (Right to left)
(1) Admissible heuristics for A* search

<table>
<thead>
<tr>
<th>Seg-0</th>
<th>h00</th>
<th>h10</th>
<th>h20</th>
<th>h30</th>
<th>h40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seg-1</td>
<td>h01</td>
<td>h11</td>
<td>h21</td>
<td>h31</td>
<td>h41</td>
</tr>
<tr>
<td>Seg-2</td>
<td>h02</td>
<td>h12</td>
<td>h22</td>
<td>h32</td>
<td>h42</td>
</tr>
<tr>
<td>noSeg-0</td>
<td>h03</td>
<td>h13</td>
<td>h23</td>
<td>h33</td>
<td>h43</td>
</tr>
<tr>
<td>noSeg-1</td>
<td>h04</td>
<td>h14</td>
<td>h24</td>
<td>h34</td>
<td>h44</td>
</tr>
<tr>
<td>noSeg-2</td>
<td>h05</td>
<td>h15</td>
<td>h25</td>
<td>h35</td>
<td>h45</td>
</tr>
</tbody>
</table>

These are her flowers.
These are her flowers.
(3) Get $y_1$ & $P(y_1)$:
Forward-Backward algo.

Seg-0  ○h00  ○h10  ○h20  ○h30  ○h40

Seg-1  ○h01  ○h11  ○h21  ○h31  ○h41

Seg-2  ○h02  ○h12  ○h22  ○h32  ○h42

noSeg-0 ○h03  ○h13  ○h23  ○h33  ○h43

noSeg-1 ○h04  ○h14  ○h24  ○h34  ○h44

noSeg-2 ○h05  ○h15  ○h25  ○h35  ○h45

These are her flowers.
(3) Get $y_1$ & $P(y_1)$: Forward-Backward algo.

$P(seg, noSeg, seg, seg, seg) = 0.2$

$P(y^*) = 0.2$

$P(unknown) = 1 - 0.2 = 0.8$

$P(y^*) > P(unknown)$?
(4) Find 2nd latent path h2: 
A* search

Seg-0 〇h00 〇h10 〇h20 〇h30 〇h40
Seg-1 〇h01 〇h11 〇h21 〇h31 〇h41
Seg-2 〇h02 〇h12 〇h22 〇h32 〇h42

noSeg-0 〇h03 〇h13 〇h23 〇h33 〇h43
noSeg-1 〇h04 〇h14 〇h24 〇h34 〇h44
noSeg-2 〇h05 〇h15 〇h25 〇h35 〇h45

These are her flowers.
(5) Get $y_2$ & $P(y_2)$: Forward-backward algo.

These are her flowers.
(5) Get $y_2$ & $P(y_2)$: Forward-backward algo.

$$P(\text{seg, seg, seg, noSeg, seg}) = 0.3$$
$$P(y^*) = 0.3$$
$$P(\text{unknown}) = 0.8 - 0.3 = 0.5$$
$$P(Y^*) > P(\text{unknown})?$$

These are her flowers.
Data flow: the inference algo.

1. Search for the top-n ranked latent sequence: $h_n$
2. Compute its label sequence: $y_n$
3. Compute $p(y_n)$ and remaining probability
4. Find the existing $y$ with max prob: $y^*$

Decision:
- Yes: Optimal results = $y^*$
- No:

cycle n
Key: make this exact method as fast as previous approx. methods!

1. **Cycle n**
   - Search for the top-n ranked latent sequence: $h_n$
   - Compute its label sequence: $y_n$
   - Compute $p(y_n)$ and remaining probability
   - Find the existing $y$ with max prob: $y^*$

Efficient top-n search: "A* Search"

Speed up the summation: dynamic programming

Decision

Yes

Optimal results = $y^*$
Key: make this exact method as fast as previous approx. methods!

- **Speeding up:** by simply setting a threshold on the search step, \( n \)
Conclusions

• Inference on LDCRFs is an NP-hard problem (even for linear-chain latent dynamics)!
• Proposed an exact inference method on LDCRFs.
• The proposed method achieves good accuracies yet with fast speed.
Latent variable perceptron for structured classification

Xu Sun (孫 榮)
University of Tokyo
2010.06.16
A new model for fast training

[Sun+ IJCAI 09]

Conditional latent variable model:

\[
y^* = \arg\max_y \sum_{h: \text{Proj}(h) = y} P(h \mid x, \theta)
\]

Normally, batch training
(do weight update after go over all samples)

Our proposal, a new model (Sun et al., 2009):

\[
h^* = \arg\max_h P'(h \mid x, \theta)
\]

Online training
(do weight update on each sample)
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Seg-0</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Seg-1</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Seg-2</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>noSeg-0</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>noSeg-1</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>noSeg-2</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>These</td>
<td>are</td>
<td>her</td>
<td>flowers</td>
<td>.</td>
<td></td>
</tr>
</tbody>
</table>
Our proposal:
latent perceptron training

\[
\theta^{i+1} = \theta^i + f[\arg \max_h F(h | y^*_i, x_i, \theta^i), x_i] - f[\arg \max_h F(h | x_i, \theta^i), x_i]
\]
Convergence analysis: separability

[Sun+ IJCAI 09]

• With latent variables, is data space still separable?  Yes

**Theorem 1.** Given the latent feature mapping \( m = (m_1, \ldots, m_n) \), for any sequence of training examples \((x_i, y_i^*)\) which is separable with margin \( \delta \) by a vector \( U \) represented by \((\alpha_1, \ldots, \alpha_n)\) with \( \sum_{i=1}^{n} \alpha_i^2 = 1 \), the examples then will also be latently separable with margin \( \bar{\delta} \), and \( \bar{\delta} \) is bounded below by

\[
\bar{\delta} \geq \delta / T,
\]

where \( T = (\sum_{i=1}^{n} m_i \alpha_i^2)^{1/2} \).
Convergence
[Sun+ IJCAI 09]

• Is latent perceptron training convergent?  Yes

**Theorem 2.** For any sequence of training examples \((x_i, y_i^*)\) which is separable with margin \(\delta\), the number of mistakes of the latent perceptron algorithm in Figure 1 is bounded above by

\[
\text{number of mistakes} \leq 2T^2 M^2 / \delta^2
\]

Comparison to traditional perceptron:

\[
\text{number of mistakes} \leq R^2 / \delta^2
\]
A difficult case: inseparable data
[Sun+ IJCAI 09]

• Are errors tractable for inseparable data?
  
  #mistakes per iteration is up-bounded

**Theorem 3.** For any training sequence \((x_i, y_i^*)\), the number of mistakes made by the latent perceptron training algorithm is bounded above by

\[
\text{number of mistakes} \leq \min_{U, \delta} \left( \sqrt{2M + D_{U, \delta}} \right)^2 / \delta^2
\]
Summarization: convergence analysis

• Latent perceptron is convergent
  – By adding any latent variables, a separable data will still be separable
  – Training is not endless (will stop on a point)
  – Converge speed is fast (similar to traditional perceptron)
  – Even for a difficult case (inseparable data), mistakes are tractable (up-bounded on #mistake-per-iter)
References & source code


• X. Sun & J. Tsujii. Sequential labeling with latent variables. In *EACL 2009*.

• Source code (Latent-dynamic CRF, LDI inference, Latent-perceptron) can be downloaded from my homepage: http://www.ibis.t.u-tokyo.ac.jp/XuSun