Announcements

• A1 due today
• A2 will be released later today (due: 3/6)
  • Covering HMMs, MEMMs, parsing (next two lectures)
• If you don’t know how to be added to Ed

What do you think of assignment 1?
A) Very easy  B) Easy  C) Moderate
D) Somewhat hard  E) Hard
Recap: Hidden Markov models

1. Set of states $S = \{1, 2, ..., K\}$ and set of observations $O = \{o_1, ..., o_n\}$

2. Initial state probability distribution $\pi(s_1)$

3. Transition probabilities $P(s_{t+1} | s_t)$

4. Emission probabilities $P(o_t | s_t)$

Strong assumptions
Recap: Hidden Markov models

1. **Markov assumption:**

   \[ P(s_{t+1} | s_1, \ldots, s_t) \approx P(s_{t+1} | s_t) \]

   1) assumes state sequences do not have very strong priors/long-range dependencies

2. **Output independence:**

   \[ P(o_t | s_1, \ldots, s_t) \approx P(o_t | s_t) \]

   2) assumes neighboring states don’t affect current observation
Recap: Viterbi decoding

\[ M[i, j] = \max_k M[i - 1, k] \ P(s_j | s_k) \ P(o_i | s_j) \quad 1 \leq k \leq K \quad 1 \leq i \leq n \]

**Backward:** Pick \( \max_k M[n, k] \) and backtrack using \( B \)
Trigram hidden Markov models

What we have seen so far is also called bigram HMM
Can be extended to trigram, 4-gram etc.

\[
P(S, O) = \prod_{i=1}^{n} P(s_i \mid s_{i-1}, s_{i-2})P(o_i \mid s_i)
\]

MLE estimate:

\[
P(s_i \mid s_{i-1}, s_{i-2}) = \frac{\text{Count}(s_i, s_{i-1}, s_{i-2})}{\text{Count}(s_{i-1}, s_{i-2})}
\]

Can add smoothing techniques to avoid zero probabilities!

Viterbi:

\[
M[i, j, k] = \max_{r} M[i-1, k, r] \cdot P(s_j \mid s_k, s_r) \cdot P(o_i \mid s_j) \quad 1 \leq j, k, r \leq K \quad 1 \leq i \leq n
\]

most probable sequence of states ending with state j at time i, and state k at i-1

Time complexity: \(O(nK^3)\)
Maximum Entropy Markov Models (MEMMs)

ICML 2000

Maximum Entropy Markov Models for Information Extraction and Segmentation

Andrew McCallum
Dayne Freitag
Just Research, 4616 Henry Street, Pittsburgh, PA 15213 USA

Fernando Pereira
AT&T Labs - Research, 180 Park Ave, Florham Park, NJ 07932 USA

MCCALLUM@JUSTRESEARCH.COM
DAYNE@JUSTRESEARCH.COM
PEREIRA@RESEARCH.ATT.COM
Generative vs discriminative models

- HMM is a *generative* model
- Can we model $P(s_1, \ldots, s_n \mid o_1, \ldots, o_n)$ directly?

<table>
<thead>
<tr>
<th>Generative</th>
<th>Discriminative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Text classification</strong></td>
<td><strong>Logistic Regression:</strong></td>
</tr>
<tr>
<td>Naive Bayes: $P(c)P(d \mid c)$</td>
<td>$P(c \mid d)$</td>
</tr>
<tr>
<td><strong>Sequence prediction</strong></td>
<td></td>
</tr>
<tr>
<td>HMM: $P(s_1, \ldots, s_n)P(o_1, \ldots, o_n \mid s_1, \ldots, s_n)$</td>
<td>MEMMM: $P(s_1, \ldots, s_n \mid o_1, \ldots, o_n)$</td>
</tr>
</tbody>
</table>
Maximum entropy Markov model (MEMM)

\[ P(S \mid O) = \prod_{i=1}^{n} P(s_i \mid s_{i-1}, s_{i-2}, \ldots, s_1, O) \]

\[ = \prod_{i=1}^{n} P(s_i \mid s_{i-1}, O) \]

\[ P(s_i = s \mid s_{i-1}, O) \propto \exp(w \cdot f(s_i = s, s_{i-1}, O, i)) \]

\[ O = \langle o_1, o_2, \ldots, o_n \rangle \]

Markov assumption: Bigram MEMM

Important: you can define features over entire word sequence \( O \)!
Use features and weights:

\[ P(s_i = s \mid s_{i-1}, O) \propto \exp(\mathbf{w} \cdot \mathbf{f}(s_i = s, s_{i-1}, O, i)) \]

- **Which of the following is the correct way to calculate this probability?**

A) \[ P(s_i = s \mid s_{i-1}, O) = \frac{\exp(\mathbf{w} \cdot \mathbf{f}(s_i = s, s_{i-1}, O, i))}{\sum_{s' = 1}^{K} \exp(\mathbf{w} \cdot \mathbf{f}(s_i = s, s_{i-1} = s', O, i))} \]

B) \[ P(s_i = s \mid s_{i-1}, O) = \frac{\exp(\mathbf{w} \cdot \mathbf{f}(s_i = s, s_{i-1}, O, i))}{\sum_{s' = 1}^{K} \exp(\mathbf{w} \cdot \mathbf{f}(s_i = s', s_{i-1}, O, i))} \]

C) \[ P(s_i = s \mid s_{i-1}, O) = \frac{\exp(\mathbf{w} \cdot \mathbf{f}(s_i = s, s_{i-1}, O, i))}{\sum_{O'} \exp(\mathbf{w} \cdot \mathbf{f}(s_i = s, s_{i-1}, O', i))} \]

*The answer is (B)*
Maximum entropy Markov model (MEMM)

- Bigram MEMM:
  \[ P(s_i = s | s_{i-1}, O) = \frac{\exp(w \cdot f(s_i = s, s_{i-1}, O, i))}{\sum_{s' = 1}^{K} \exp(w \cdot f(s_i = s', s_{i-1}, O, i))} \]

- Can be easily extended to trigram MEMM, 4-gram MEMM..
How to define features?

\[ f(s_i = s', s_{i-1}, s_{i-2}, O, i) \]

- \( t_i \) = tags (states)
- \( w_i \) = words (observations)

Feature templates:

\[
\langle t_i, w_{i-2} \rangle, \langle t_i, w_{i-1} \rangle, \langle t_i, w_i \rangle, \langle t_i, w_{i+1} \rangle, \langle t_i, w_{i+2} \rangle, \\
\langle t_i, t_{i-1} \rangle, \langle t_i, t_{i-2}, t_{i-1} \rangle, \\
\langle t_i, t_{i-1}, w_i \rangle, \langle t_i, w_{i-1}, w_i \rangle, \langle t_i, w_i, w_{i+1} \rangle,
\]

Features (binary):

- \( t_i = \text{VB and } w_{i-2} = \text{Janet} \)
- \( t_i = \text{VB and } w_{i-1} = \text{will} \)
- \( t_i = \text{VB and } w_i = \text{back} \)
- \( t_i = \text{VB and } w_{i+1} = \text{the} \)
- \( t_i = \text{VB and } w_{i+2} = \text{bill} \)
- \( t_i = \text{VB and } t_{i-1} = \text{MD} \)
- \( t_i = \text{VB and } t_{i-1} = \text{MD and } t_{i-2} = \text{NNP} \)
- \( t_i = \text{VB and } w_i = \text{back and } w_{i+1} = \text{the} \)
Features in an MEMM

Incorrect   DT JJ NN DT NN
Correct     DT NN VB DT NN

The old man the boat

\[ w_{i-1} \quad w_i \quad w_{i+1} \quad w_{i+2} \quad w_{i+3} \]

Which of these feature templates would help most to tag ‘old’ correctly?

A) \( \langle t_i, t_{i-1}, w_i, w_{i-1}, w_{i+1} \rangle \)
B) \( \langle t_i, t_{i-1}, w_i, w_{i-1} \rangle \)
C) \( \langle t_i, w_i, w_{i-1}, w_{i+1} \rangle \)
D) \( \langle t_i, w_i, w_{i-1}, w_{i+1}, w_{i+2} \rangle \)

The answer is (D)

t_i = \text{tags (states)}

w_i = \text{words (observations)}
MEMMs: Decoding

- Bigram MEMM:

\[
\hat{S} = \arg \max_{S} P(S \mid O) = \arg \max_{S} \prod_i P(s_i \mid s_{i-1}, O)
\]

- Greedy decoding:

\[
\hat{s}_1 = \arg \max_{s} P(s_i = s \mid \emptyset, O) = \arg \max_{s} \mathbf{w} \cdot \mathbf{f}(s_i = s, s_{i-1} = \emptyset, O) = \text{DT}
\]
MEMMs: Decoding

- Bigram MEMM:

\[
\hat{S} = \arg \max_S P(S \mid O) = \arg \max_S \prod_i P(s_i \mid s_{i-1}, O)
\]

- Greedy decoding:

\[
\hat{s}_2 = \arg \max_S P(s_i = s \mid DT, O) = NN
\]
MEMMs: Decoding

- Bigram MEMM:

\[ \hat{S} = \arg \max_S P(S \mid O) = \arg \max_S \prod_i P(s_i \mid s_{i-1}, O) \]

- Greedy decoding:

\[ \hat{s}_i = \arg \max_s P(s_i = s \mid \hat{s}_{i-1}, O) \]
Viterbi decoding for MEMMs

$M[i, j]$ stores joint probability of most probable sequence of states ending with state $j$ at time $i$

\[
M[i, j] = \max_k M[i - 1, k] P(s_i = j | s_{i-1} = k, O) \quad 1 \leq k \leq K \quad 1 \leq i \leq n
\]

**Backward:** Pick $\max_k M[n, k]$ and backtrack using $B$
MEMM: Decoding

How would you compare the computational complexity of Viterbi decoding for bigram MEMMs compared to decoding for bigram HMMs?
A) More operations in MEMM  
B) More operations in HMM  
C) Equal  
D) Depends on number of features in MEMM

The answer is (D)

MEMM:  
\[ M[i,j] = \max_k M[i-1,k] \underbrace{P(s_i = j | s_{i-1} = k, O)}_{1 \leq k \leq K \ 1 \leq i \leq n} \]

HMM:  
\[ M[i,j] = \max_k M[i-1,k] \ P(s_j \mid s_k) \ P(o_i \mid s_j) \quad 1 \leq k \leq K \ 1 \leq i \leq n \]
MEMM: Learning

• **Gradient descent**: similar to logistic regression!

\[ P(s_i = s \mid s_{i-1}, O) = \frac{\exp(w \cdot f(s_i = s, s_{i-1}, O, i))}{\sum_{s'} \exp(w \cdot f(s_i = s', s_{i-1}, O, i))} \]

• **Given**: annotated pairs of \((S, O)\) where each \(S = \langle s_1, s_2, \ldots, s_n \rangle\)

   Loss for one sequence, \(L = - \sum_{i=1}^{n} \log P(s_i \mid s_{i-1}, O)\)

• Compute gradients with respect to weights \(w\) and update
MEMM vs HMM

- HMM models the joint $P(S, O)$ while MEMM models the required prediction $P(S | O)$
- MEMM has more expressivity
  - accounts for dependencies between neighboring states and **entire observation** sequence
  - allows for **more flexible features**
- HMM may hold an advantage if the dataset is small
Conditional Random Fields (CRFs)

ICML 2001

Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data

John Lafferty†
Andrew McCallum‡
Fernando Pereira‡

†WhizBang! Labs–Research, 4616 Henry Street, Pittsburgh, PA 15213 USA
‡School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA
§Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104 USA

LAFFERTY@CS.CMU.EDU
MCCALLUM@WHIZBANG.COM
FPEREIRA@WHIZBANG.COM
Conditional Random Field

- Model $P(s_1, \ldots, s_n \mid o_1, \ldots, o_n)$ directly
- No Markov assumption
  - Map entire sequence of states $S$ and observations $O$ to a global feature vector
- Normalize over entire sequences

$$P(S \mid O) = \frac{\exp(w \cdot f(S, O))}{\sum_{S'} \exp(w \cdot f(S', O))} = \frac{\exp(w \cdot f(S, O))}{Z(O)}$$
Features

- Each $F_k$ in $\mathbf{f}$ is a \textbf{global} feature function

\[
P(S | O) = \frac{\exp(\sum_{k=1}^{m} w_k \cdot F_k(S, O))}{\sum_{S'} \exp(\sum_{k=1}^{m} w_k \cdot F_k(S', O))}
\]

- Can be computed as a combination of local features:

\[
F_k = \sum_{i=1}^{n} f_k(s_{i-1}, s_i, O, i)
\]

- Each local feature only depends on previous and current states

\[
\begin{aligned}
\{x_i = \text{the}, y_i = \text{DET}\} \\
\{y_i = \text{PROP}, x_{i+1} = \text{Street}, y_i-1 = \text{NUM}\} \\
\{y_i = \text{VERB}, y_i-1 = \text{AUX}\}
\end{aligned}
\]
CRF: Decoding

\[ \hat{S} = \arg \max_{S} P(S \mid O) = \arg \max_{S} \frac{\exp(w \cdot f(S, O))}{Z(O)} \]

\[ = \arg \max_{S} \exp(w \cdot f(S, O)) \]

\[ = \arg \max_{S} \sum_{k=1}^{m} \sum_{i=1}^{n} w_k f_k(s_{i-1}, s_i, O, i) \]

- Use Viterbi similar to HMM and MEMM
CRF: Learning

\[
P(S \mid O) = \frac{\exp\left( \sum_{k=1}^{m} \sum_{i=1}^{n} w_k f_k(s_{i-1}, s_i, O, i) \right)}{Z(O)}
\]

\[
= \frac{\exp\left( \sum_{k=1}^{m} \sum_{i=1}^{n} w_k f_k(s_{i-1}, s_i, O, i) \right)}{\sum_{s'_1, \ldots, s'_n} \exp\left( \sum_{k=1}^{m} \sum_{i=1}^{n} w_k f_k(s'_{i-1}, s'_i, O, i) \right)}
\]

\[-\log P(S \mid O) = - \sum_{k=1}^{m} \sum_{i=1}^{n} w_k f_k(s_{i-1}, s_i, O, i) + \log \sum_{s'_1, \ldots, s'_n} \exp\left( \sum_{k=1}^{m} \sum_{i=1}^{n} w_k f_k(s'_{i-1}, s'_i, O, i) \right)
\]

\[-\frac{\partial \log P(S \mid O)}{\partial w_k} \text{ can be done efficiently using dynamic programming}
\]
CRF vs MEMM

- MEMM models the required prediction $P(S \mid O)$ using the Markov assumption, while the CRF does not
- CRF uses global features while MEMM features are localized
- Feature design is flexible in both models
- CRF is computationally more complex
History of CRFs

• Very popular in the 2000s
• Wide variety of applications:
  • Information extraction
  • Summarization
  • Image labeling/segmentation
History of CRFs

- Very popular in the 2000s
- Wide variety of applications:
  - Information extraction
  - Summarization
  - Image labeling/segmentation

Software

This is a partial list of software that implement generic CRF tools.

- RNNSharp\(^\text{®}\) CRFs based on recurrent neural networks (C#, .NET)
- CRF-ADF\(^\text{®}\) Linear-chain CRFs with fast online ADF training (C#, .NET)
- CRFSharp\(^\text{®}\) Linear-chain CRFs (C#, .NET)
- GCO\(^\text{®}\) CRFs with submodular energy functions (C++, Matlab)
- DGM\(^\text{®}\) General CRFs (C++)
- GRMM\(^\text{®}\) General CRFs (Java)
- factorie\(^\text{®}\) General CRFs (Scala)
- CRFall\(^\text{®}\) General CRFs (Matlab)
- Sarawagi's CRF\(^\text{®}\) Linear-chain CRFs (Java)
- HCRF library\(^\text{®}\) Hidden-state CRFs (C++, Matlab)
- Accord.NET\(^\text{®}\) Linear-chain CRF, HCRF and HMMs (C#, .NET)
- Wapiti\(^\text{®}\) Fast linear-chain CRFs (C)\(^{[18]}\)
- CRFSuite\(^\text{®}\) Fast restricted linear-chain CRFs (C)
- CRF++\(^\text{®}\) Linear-chain CRFs (C++)
- FlexCRFs\(^\text{®}\) First-order and second-order Markov CRFs (C++)
- crf-chain1\(^\text{®}\) First-order, linear-chain CRFs (Haskell)
- imageCRF\(^\text{®}\) CRF for segmenting images and image volumes (C++)
- MALLET\(^\text{®}\) Linear-chain for sequence tagging (Java)
Empirical performance

<table>
<thead>
<tr>
<th>Model</th>
<th>F score</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM combination (Kudo et al., 2001)</td>
<td>94.39%</td>
</tr>
<tr>
<td>CRF</td>
<td>94.38%</td>
</tr>
<tr>
<td>Generalized winnow (Zhang et al., 2002)</td>
<td>93.89%</td>
</tr>
<tr>
<td>Voted perceptron</td>
<td>94.09%</td>
</tr>
<tr>
<td>MEMM</td>
<td>93.70%</td>
</tr>
</tbody>
</table>

Table 2: NP chunking F scores

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRF vs. SVM</td>
<td>0.469</td>
</tr>
<tr>
<td>CRF vs. MEMM</td>
<td>0.00109</td>
</tr>
<tr>
<td>CRF vs. voted perceptron</td>
<td>0.116</td>
</tr>
<tr>
<td>MEMM vs. voted perceptron</td>
<td>0.0734</td>
</tr>
</tbody>
</table>

Table 4: McNemar’s tests on labeling disagreements

(Sha and Pereira, 2003): Shallow Parsing with Conditional Random Fields
CRFs in deep learning era

- Use CRFs on top of neural representations (instead of features and weights)
- Joint sequence prediction without the need for defining features!
- Recent architectures such as seq2seq w/ attention or Transformer may implicitly do the job

Conditional Random Fields as Recurrent Neural Networks


Neural Architectures for Named Entity Recognition

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, Chris Dyer
Carnegie Mellon University, NLP Group, Pompeu Fabra University
{glample, sandeeps, kkawakam, cdyer}@cs.cmu.edu, miguel.ballesteros@upf.edu

Bidirectional LSTM-CRF Models for Sequence Tagging

Zhiheng Huang, Wei Xu, Kai Yu
Baidu research
huangzhiheng@baidu.com, xuwei06@baidu.com, yuikai@baidu.com
Named entity recognition (NER)
Named entity recognition

Barack Hussein Obama II (born August 4, 1961) is an American attorney and politician who served as the 44th President of the United States from January 20, 2009, to January 20, 2017. A member of the Democratic Party, he was the first African American to serve as president. He was previously a United States Senator from Illinois and a member of the Illinois State Senate.
Named entities

• Named entity, in its core usage, means anything that can be referred to with a proper name.

• NER is the task of 1) finding spans of text that constitute proper names; 2) tagging the type of the entity

• Most common 4 tags:
  • **PER** (Person): “Marie Curie”
  • **LOC** (Location): “New York City”
  • **ORG** (Organization): “Princeton University”
  • **MISC** (Miscellaneous): nationality, events, ..
Only France and Britain backed Fischler’s proposal.

Steve Jobs founded Apple with Steve Wozniak.

O = not an entity
If multiple words constitute a named entity, they will be labeled with the same tag.
NER: BIO Tagging

[PER Jane Villanueva] of [ORG United], a unit of [ORG United Airlines Holding], said the fare applies to the [LOC Chicago] route.

<table>
<thead>
<tr>
<th>Words</th>
<th>BIO Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jane</td>
<td>B-PER</td>
</tr>
<tr>
<td>Villanueva</td>
<td>I-PER</td>
</tr>
<tr>
<td>of</td>
<td>O</td>
</tr>
<tr>
<td>United</td>
<td>B-ORG</td>
</tr>
<tr>
<td>Airlines</td>
<td>I-ORG</td>
</tr>
<tr>
<td>Holding</td>
<td>I-ORG</td>
</tr>
<tr>
<td>discussed</td>
<td>O</td>
</tr>
<tr>
<td>the</td>
<td>O</td>
</tr>
<tr>
<td>Chicago</td>
<td>B-LOC</td>
</tr>
<tr>
<td>route</td>
<td>O</td>
</tr>
<tr>
<td>.</td>
<td>O</td>
</tr>
</tbody>
</table>

B: token that begins a span
I: tokens that inside a span
O: tokens outside of a span