Sequence Models 1

CS 287
OED Word of the Year

Laughing emoji
Which words made the shortlist?

- **sharing economy**: An economic system in which assets or services are shared between private individuals, either for free or for a fee, typically by means of the Internet.

- **they**: Used to refer to a person of unspecified sex.

- **on fleek**: Extremely good, attractive, or stylish.

- **ad blocker**: A piece of software designed to prevent advertisements from appearing on a web page.

- **refugee**: A person who has been forced to leave their country in order to escape war, persecution, or natural disaster.

- **Brexit**: A term for the potential or hypothetical departure of the United Kingdom from the European Union.

- **Dark Web**: The part of the World Wide Web that is only accessible by means of special software, allowing users and website operators to remain anonymous.

- **lumbersexual**: A young urban man who cultivates an appearance and style of dress (typified by a beard and check shirt) suggestive of a rugged outdoor lifestyle.
Maximum-likelihood Estimation (MLE)
- Smoothed count-based models are not MLE
- (Un-regularized) NNLM models are MLE
- Property of the estimation objective, not the model structure

Markov Models
- n-gram models are Markov models
- windowed NNLM models are also Markov models
- RNN-LM models are not Markovian models
- Property of the model structure, not the estimation
Homework: Reverse Trie Data structure

Used in several standard language modeling toolkits.
Homework: Results (Mikolov et al, 2011)

<table>
<thead>
<tr>
<th>Model</th>
<th>Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>individual</td>
</tr>
<tr>
<td>3-gram with Good-Turing smoothing (GT3)</td>
<td>165.2</td>
</tr>
<tr>
<td>5-gram with Good-Turing smoothing (GT5)</td>
<td>162.3</td>
</tr>
<tr>
<td>3-gram with Kneser-Ney smoothing (KN3)</td>
<td>148.3</td>
</tr>
<tr>
<td>5-gram with Kneser-Ney smoothing (KN5)</td>
<td>141.2</td>
</tr>
<tr>
<td>5-gram with Kneser-Ney smoothing + cache</td>
<td>125.7</td>
</tr>
<tr>
<td>Maximum entropy model with 5-gram features</td>
<td>142.1</td>
</tr>
<tr>
<td>Random clusterings LM</td>
<td>170.1</td>
</tr>
<tr>
<td>Random forest LM</td>
<td>131.9</td>
</tr>
<tr>
<td>Structured LM</td>
<td>146.1</td>
</tr>
<tr>
<td>Within and across sentence boundary LM</td>
<td>116.6</td>
</tr>
<tr>
<td>Log-bilinear LM</td>
<td>144.5</td>
</tr>
<tr>
<td>Feedforward neural network LM [9]</td>
<td>140.2</td>
</tr>
<tr>
<td>Feedforward neural network LM [18]</td>
<td>141.8</td>
</tr>
<tr>
<td>Syntactical neural network LM</td>
<td>131.3</td>
</tr>
<tr>
<td>Recurrent neural network LM</td>
<td>124.7</td>
</tr>
<tr>
<td>Adaptive RNNLM</td>
<td>123.2</td>
</tr>
<tr>
<td>Combination of static RNNLMs</td>
<td>102.1</td>
</tr>
<tr>
<td>Combination of adaptive RNNLMs</td>
<td>101.0</td>
</tr>
</tbody>
</table>
Recent popularization of RNNs has been based on language modeling (Mikolov, 2012)

In particular RNNs allow for non-Markovian models

\[
p(w_i|w_1, \ldots, w_{i-1}; \theta) = O(s_i)
\]

Compare this to the windowed approach.

\[
p(w_i|w_{i-n+1}, \ldots, w_{i-1}; \theta) = O(s_i)
\]
4. Third-order approximation (trigram structure as in English).

5. First-Order Word Approximation. Rather than continue with tetragram, ... , 2-gram structure it is easier and better to jump at this point to word units. Here words are chosen independently but with their appropriate frequencies.
6. Second-Order Word Approximation. The word transition probabilities are correct but no further structure is included.

THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH 'RITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED

The resemblance to ordinary English text increases quite noticeably at each of the above steps.
Non-Markov Babblers

- Use previous prediction $\hat{y}$ to decide on next word.

- In particular RNNs allow for non-Markovian models

  $$p(w_i | w_1, \ldots, w_{i-1}; \theta) = \hat{y}_i = O(s_i)$$

- Greedy non-Markov babbler:

  $$w_{i+1} = \arg \max_w \hat{y}_{i,w}$$
Babbling Visually

\[ \hat{y}_1 \quad \hat{y}_2 \quad \hat{y}_3 \]

\[ s_1 \quad s_2 \quad s_3 \]

\[ x_1 \quad x_2 \quad x_3 \]
Hillary Botham @RoboClinton · Mar 20
That's why I will create more than 25 million community colleges

Hillary Botham @RoboClinton · Mar 20
I want to be invisible to the world

Hillary Botham @RoboClinton · Mar 20
I was proud to be a president who will be a commitment to the mortgage crisis in the country

Hillary Botham @RoboClinton · Mar 20
I want to thank Iowa for a president who works for everyone.

Hillary Botham @RoboClinton · Mar 20
I want to be a president who will be here with you

Hillary Botham @RoboClinton · Mar 20
We need to be a president who will be a president. @HillaryClinton and @POTUS is strong.

Hillary Botham @RoboClinton · Mar 20
@tedcruz is the problem. The health care system is so much more comprehensive and we will do so much more for them.
Diversity in Babbling

- Taking the greedy solution can lead to boring output.

- Instead can sample from soft-max distribution,

\[ \hat{y} \sim \text{softmax}(s_iW + b) \]

- However this can lead to non-fluent output.
Review: Why is it called the softmax?

\[
\text{softmax}([x \ y]) = \frac{\exp(x)}{\exp(x) + \exp(y)}
\]

\[
\arg \max([x \ y]) = 1(x > y)
\]
Temperature

- Can use a temperature parameter $t \in (0, 1]$ to modulate
- Temperature-based sampling.

$$\hat{y} \sim \text{softmax}( (s_iW + b) / t )$$

- What happens when the parameter is closer to the extremes?
Temperature $t = 0.99$
Temperature $t = 0.5$
Temperature $t = 0.01$
Temperature $t = 0.001$
Quiz: Babbling for Segmentation

The upcoming homework asks you to insert spaces in a non-segmented stream of characters using an RNN transducer. For example given

```
thedogwalkstothepark
```

You should output,

```
thedogwalkstothepark
```

How might you do this efficiently with a RNN transducer? What does the training look like and how do you generate the output with a greedy babbler?
Contents

Structured Models

Hidden Markov Model

Maximum-Entropy Markov Models

Forward Pass
So Far: Multiclass Classification

Every problem:

- Input encoding $\mathbf{x}$ (sparse or dense)

- Output encoding $\mathbf{y}$ (sparse index $C$)

- Prediction $\hat{y}$

- Classification

$$c = \arg \max_{i \in C} y_i$$
Predicting Structure

What if we want to predict:

- Parse trees
- Named-entities
- Semantic Roles
- Translations

These are all combinatorial structures
Example: BIO Tagging

**B-TYPE** Stop current mention and begin new mention

**I-TYPE** Continue adding to current mention

**O** Not part of a mention.

Example:

[PER George Bush ] [LOC U.S. ] president is traveling to [LOC Baghdad ] .
Example: BIO Tagging

B-TYPE Stop current mention and begin new mention

I-TYPE Continue adding to current mention

O Not part of a mention.

Example:

[PER George Bush ] [LOC U.S. ] president is traveling to [LOC Baghdad ] .
Better Tag Features: Tag Sequence

Representation can use specific aspects of text.

- \( \mathcal{F} \); Prefixes, suffixes, hyphens, first capital, all-capital, has digits, etc.

- Also include features on previous tags

Example: Rare word tagging with context

\[
\mathbf{x} = \delta(\text{last:CD}) + \delta(\text{prefix:3:reg}) + \delta(\text{prefix:2:re}) + \delta(\text{prefix:1:r}) + \delta(\text{has-hyphen}) + \delta(\text{lower-case}) + \delta(\text{suffix:3:son}) \ldots
\]
Standard Approach

\[ \hat{y} = \text{softmax}(xW + b) \]

- Let \( \mathcal{C} \) to all possible tag sequences.
- Let \( x \) to be a representation of whole sentence.
- Let \( \hat{y} \) be distribution over tag sequences.
- What breaks down?
Issues with Multiclass for Sequences

- Say there are $\mathcal{T}$ tags and sequence length is $n$
- There are $d_{\text{out}} = O(\mathcal{T}^n)$ sequences!
- Just naively computing the softmax is exponential in length.
- Even if you could compute the softmax, $\mathbf{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ would be impossible to train.
Predicting Structure

Instead we will try to maximize over the sequence,

$$\arg \max_{c_1 \in C, \ldots, c_n \in C} f(x, c_1, \ldots, c_n; \theta)$$

- Note: This is a test-time optimization, different from training.

- We will discuss many different scoring functions $f$. 
We will assume that,

- Assume true output is a sequence, 
  \[ y_1, \ldots, y_n \]

- Each is a sparse one-hot vector \( y_1 = c_1, \ldots, y_n = c_n \)

- Training data consists of labeled sequences
  \[ (x_1^{(1)}, y_1^{(1)}), \ldots, (x_1^{(J)}, y_1^{(J)}) \]
Hidden Markov Model

- Natural extension of naive Bayes to sequences.

- Allows us to efficiently solve \( \text{arg max} \),

\[
\text{arg max}_{c_1 \in C, \ldots, c_n \in C} f(x, c_1, \ldots, c_n; \theta)
\]

- Classical model, not really used as-is in practice (but important to understand)
Review: Multinomial Naive Bayes

Reminder, joint probability chain rule,

\[ p(x, y) = p(x|y)p(y) \]

For a sparse features, with observed classes we can write as,

\[
p(x, y) = p(x_{f_1} = 1, \ldots, x_{f_k} = 1|y = \delta(c))p(y = \delta(c)) = \\
= \prod_{i=1}^{k} p(x_{f_i} = 1|x_{f_1} = 1, \ldots, x_{f_{i-1}} = 1, y = \delta(c))p(y = \delta(c)) \approx \\
= \prod_{i=1}^{k} p(x_{f_i} = 1|y)p(y)
\]

First is by chain-rule, second is by independence assumption.
Review: Multinomial Naive Bayes

Reminder, joint probability chain rule,

\[ p(x, y) = p(x|y)p(y) \]

For a sparse features, with observed classes we can write as,

\[ p(x, y) = p(x_{f_1} = 1, \ldots, x_{f_k} = 1|y = \delta(c))p(y = \delta(c)) = \]
\[ = \prod_{i=1}^{k} p(x_{f_i} = 1|x_{f_1} = 1, \ldots, x_{f_{i-1}} = 1, y = \delta(c))p(y = \delta(c)) \approx \]
\[ = \prod_{i=1}^{k} p(x_{f_i} = 1|y)p(y) \]

First is by chain-rule, second is by independence assumption.
Naive Bayes

Graphical model representation,

For simplicity, assume that there is only one space feature in $x$. 
Define the function $f$ to be the joint probability,

$$f(x, c_1, \ldots, c_n) = p(y_1 = \delta(c_1), \ldots, y_n = \delta(c_n), x_1, \ldots, x_n)$$

Aim will be to maximize this,

$$\arg\max_{c_1 \in C, \ldots, c_n \in C} f(x, c_1, \ldots, c_n; \theta)$$
Hidden Markov Model

\[ f(x, c_{1:n}) = p(y_1 = \delta(c_1), \ldots, y_n = \delta(c_n), x_{1:n}) = p(x_{1:n} | y_{1:n}) p(y_{1:n}) \]

\[ = \prod_{i=1}^{n} p(x_i | x_{1:i-1}, y_{1:n}) p(y_i = \delta(c_i) | y_1 = \delta(c_1) \ldots y_{i-1} = \delta(c_{i-1})) \approx \prod_{i=1}^{n} p(x_i | y_i = \delta(c_i)) p(y_i = \delta(c_i) | y_{i-1} = \delta(c_{i-1})) \]
Hidden Markov Model

\[ f(x, c_{1:n}) = p(y_1 = \delta(c_1), \ldots, y_n = \delta(c_n), x_{1:n}) \]

\[ = p(x_{1:n} | y_{1:n}) p(y_{1:n}) \]

\[ = \prod_{i=1}^{n} p(x_i | x_{1:i-1}, y_{1:n}) p(y_i = \delta(c_i) | y_1 = \delta(c_1) \ldots y_{i-1} = \delta(c_{i-1})) \]

\[ \approx \prod_{i=1}^{n} p(x_i | x_{1:i-1}, y_{1:n}) p(y_i = \delta(c_i) | y_{i-1} = \delta(c_{i-1})) \]

\[ \approx \prod_{i=1}^{n} p(x_i | y_i = \delta(c_i)) p(y_i = \delta(c_i) | y_{i-1} = \delta(c_{i-1})) \]
Hidden Markov Model

\[ f(x, c_{1:n}) = p(y_1 = \delta(c_1), \ldots, y_n = \delta(c_n), x_{1:n}) \]
\[ = p(x_{1:n} | y_{1:n}) p(y_{1:n}) \]
\[ = \prod_{i=1}^{n} p(x_i | x_{1:i-1}, y_{1:n}) p(y_i = \delta(c_i) | y_1 = \delta(c_1) \ldots y_{i-1} = \delta(c_{i-1})) \]
\[ \approx \prod_{i=1}^{n} p(x_i | x_{1:i-1}, y_{1:n}) p(y_i = \delta(c_i) | y_{i-1} = \delta(c_{i-1})) \]
\[ \approx \prod_{i=1}^{n} p(x_i | y_i = \delta(c_i)) p(y_i = \delta(c_i) | y_{i-1} = \delta(c_{i-1})) \]
Hidden Markov Model

\[ f(\mathbf{x}, c_{1:n}) = p(\mathbf{y}_1 = \delta(c_1), \ldots, \mathbf{y}_n = \delta(c_n), \mathbf{x}_{1:n}) \]

\[ = p(\mathbf{x}_{1:n} | \mathbf{y}_{1:n}) p(\mathbf{y}_{1:n}) \]

\[ = \prod_{i=1}^{n} p(\mathbf{x}_i | \mathbf{x}_{1:i-1}, \mathbf{y}_{1:n}) p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_1 = \delta(c_1) \ldots \mathbf{y}_{i-1} = \delta(c_{i-1})) \]

\[ \approx \prod_{i=1}^{n} p(\mathbf{x}_i | \mathbf{y}_i = \delta(c_i)) p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_{i-1} = \delta(c_{i-1})) \]
Example: Hidden Markov Model
Hidden Markov Model

Hidden Markov model requires two distributions,

- Transition distribution
  \[ p(y_i|y_{i-1}; \theta) \]

- Emission distribution
  \[ p(x_i|y_i; \theta) \]

- How many total parameters?
Maximum Likelihood Estimation

- Sequences are assumed given.

- Therefore we can fit in the standard way

\[
\mathcal{L}(\theta) = - \sum_{j=1}^{J} \sum_{i=1}^{n} \log p(x_{i}^{(j)} | y_{i}^{(j)}; \theta) + \log p(y_{i}^{(j)} | y_{i-1}^{(j)}; \theta)
\]

- Note: training is very similar to multiclass.
Multinomial Hidden Markov Model

(Should look familiar)

- Both $p(y_i|y_{i-1})$ and $p(x_i|y_i)$ are parameterized as multinomials.
- Fit first using count matrix $T$,
  - Let $T_{c',c}$ be the counts of class $c'$ preceding class $c$.
  - 
    $$p(y_i = \delta(c_i)|y_{i-1} = \delta(c_{i-1})) = \frac{T_{c',c}}{T_{c'}}.$$  
- Fit second using count matrix $F$ ,
  - Let $F_{f,c}$ be the counts of emission $f$ with class $c$.
  - Then,
    $$p(x_i = \delta(f)|y_i = \delta(c)) = \frac{F_{f,c}}{F_{c'}}.$$  

Multinomial Hidden Markov Model

(Should look familiar)

▶ Both $p(y_i|y_{i-1} = 1)$ and $p(x_i|y_i)$ are parameterized as multinomials.

▶ Fit first using count matrix $T$,
  
  ▶ Let $T_{c',c}$ be the counts of class $c'$ preceding class $c$.
  
  $p(y_i = \delta(c_i)|y_{i-1} = \delta(c_{i-1})) = \frac{T_{c',c}}{T_{c',c}}$

▶ Fit second using count matrix $F$,

  ▶ Let $F_{f,c}$ be the counts of emission $f$ with class $c$.

  Then,

  $p(x_i = \delta(f)|y_i = \delta(c)) = \frac{F_{f,c}}{F_{.,c}}$
### Example: Named-Entity Recognition

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>U.N.</td>
<td>NNP</td>
<td>I-NP</td>
<td>I-ORG</td>
</tr>
<tr>
<td>official</td>
<td>NN</td>
<td>I-NP</td>
<td>0</td>
</tr>
<tr>
<td>Ekeus</td>
<td>NNP</td>
<td>I-NP</td>
<td>I-PER</td>
</tr>
<tr>
<td>heads</td>
<td>VBZ</td>
<td>I-VP</td>
<td>0</td>
</tr>
<tr>
<td>for</td>
<td>IN</td>
<td>I-PP</td>
<td>0</td>
</tr>
<tr>
<td>Baghdad</td>
<td>NNP</td>
<td>I-NP</td>
<td>I-LOC</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- $F$ is constructed by counting up rows
- $T$ is constructed by iterating over column
- Standard multinomial MLE issues apply (smoothing, rare-words)
Can extend $F$ to multiple features as in naive Bayes.
Versus Multiclass

Exercise: How does this model improve upon previous setup?

- Number Parameters?
- Producing a solution?
- Types of features?
Structured Models

Hidden Markov Model

Maximum-Entropy Markov Models

Forward Pass
Discriminative Sequence Models

- Just as with multiclass, we can instead fit the conditional probability.
- (In fact, we can also fit a neural network model)
- However we will see that this approach has some issues.
Review: Multiclass Logistic Regression

- Direct estimation of conditional $p(y = c|x; \theta)$

$$\hat{y} = p(y = c|x; \theta) = \text{softmax}(xW + b)$$

- $W \in \mathbb{R}^{d_{in} \times d_{out}}$, $b \in \mathbb{R}^{1 \times d_{out}}$; model parameters
- “Regression” of the distribution.
- Classification still done as,

$$\hat{c} = \arg \max_{c \in C}(xW + b)_c$$
Review: Multiclass Logistic Regression: A Model with Many Names

- Multinomial Logistic Regression
- Log-Linear Model (particularly in NLP)

\[
\text{log softmax}(z) = z - \log \sum_c \exp(z_c)
\]

- Softmax Regression
- Max-Entropy (MaxEnt)
Maximum-Entropy Markov Model

- Idea: Softmax regression over next tag conditioned on input.
- Benefits allows features on input and previous decisions
- Does not require any of the sequential independence assumptions of HMM
Discriminative Markov Model

\[ f(x, c_1, \ldots, c_n; \theta) = p(y_1 = \delta(c_1), \ldots, y_n = \delta(c_n) | x_{1:n}) \]

\[ = \prod_{i=1}^{n} p(y_i = \delta(c_i) | y_1 = \delta(c_1) \ldots y_{i-1} = \delta(c_{i-1}), x_{1:n}) \]

\[ \approx \prod_{i=1}^{n} p(y_i = \delta(c_i) | y_{i-1} = \delta(c_{i-1}), x_{1:n}) \]
Discriminative Markov Model

\[
f(x, c_1, \ldots, c_n; \theta) = p(y_1 = \delta(c_1), \ldots, y_n = \delta(c_n)|x_{1:n})
\]

\[
= \prod_{i=1}^{n} p(y_i = \delta(c_i)|y_1 = \delta(c_1) \ldots y_{i-1} = \delta(c_{i-1}), x_{1:n})
\]

\[
\approx \prod_{i=1}^{n} p(y_i = \delta(c_i)|y_{i-1} = \delta(c_{i-1}), x_{1:n})
\]
Discriminative Markov Model

\[ f(\mathbf{x}, c_1, \ldots, c_n; \theta) = p(\mathbf{y}_1 = \delta(c_1), \ldots, \mathbf{y}_n = \delta(c_n) | \mathbf{x}_{1:n}) \]

\[ = \prod_{i=1}^{n} p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_1 = \delta(c_1) \ldots \mathbf{y}_{i-1} = \delta(c_{i-1}), \mathbf{x}_{1:n}) \]

\[ \approx \prod_{i=1}^{n} p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_{i-1} = \delta(c_{i-1}), \mathbf{x}_{1:n}) \]
Maximum-Entropy Markov Models

$y_1 \rightarrow y_2 \rightarrow y_3 \rightarrow \cdots \rightarrow y_n$
MEMM estimates only a transition distribution,

- Transition distribution (also conditioned on input)
  \[ p(y_i | y_{i-1} = \delta(c_{i-1}), x_1, \ldots, x_n) = \text{softmax}(\text{feat}(x, c_{i-1})W + b) \]

- Here \textit{feat} is a deterministic combination of the input and the previous \( c_{i-1} \)

- How many total parameters?
Better Tag Features: Tag Sequence

Representation can use specific aspects of text.

- $\mathcal{F}$; Prefixes, suffixes, hyphens, first capital, all-capital, hasdigits, etc.

- Also include features on previous tags

Example: Rare word tagging with context

\[
\text{x} = \delta(\text{last:CD}) + \delta(\text{prefix:3:reg}) + \delta(\text{prefix:2:re}) \\
+ \delta(\text{prefix:1:r}) + \delta(\text{has-hyphen}) \\
+ \delta(\text{lower-case}) + \delta(\text{suffix:3:son})...
\]

in 130/CD regular-season/* games ,
Contents

Structured Models

Hidden Markov Model

Maximum-Entropy Markov Models

Forward Pass
Using Markov Model

- Assume that we have $c_1, \ldots c_{i-1}$.
- Following previous notation let $\hat{y}_i$ be distribution at this point.
- How do we predict $c_i$? What is it a function of?
Answers

Becomes just a standard linear model, if we take $c_{i-1}$ as an argument

- **HMM**

$$\hat{y}_i(c_{i-1}) = \text{softmax}(\log p(y_i|y_{i-1} = \delta(c_{i-1}))) + \log p(y_i|x_i))$$

- **MEMM**

$$\hat{y}_i(c_{i-1}) = \text{softmax}(\text{feat}(x, c_{i-1})W + b)$$

- We will see that any parameterize function of $c_{i-1}$ works

$$\hat{y}_i(c_{i-1})$$
Next Time

- How to compute predictions on a full sequence.
- Approximations for sequence prediction.
- Further algorithms.