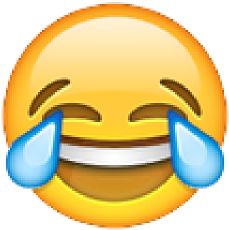
Sequence Models 1

CS 287

OED Word of the Year



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

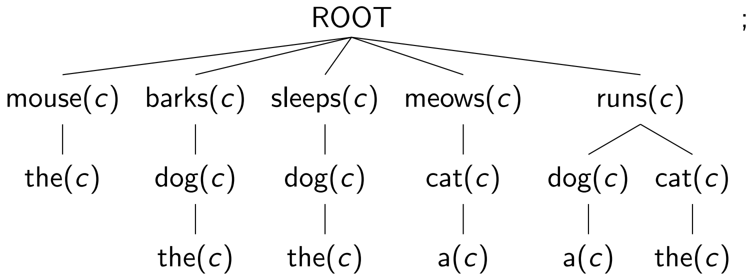
A young urban man who cultivates an appearance and style of dress (typified by a beard and check shirt) suggestive of a rugged outdoor lifestyle.

OxfordDictionaries.com

Homework: LM Terminology

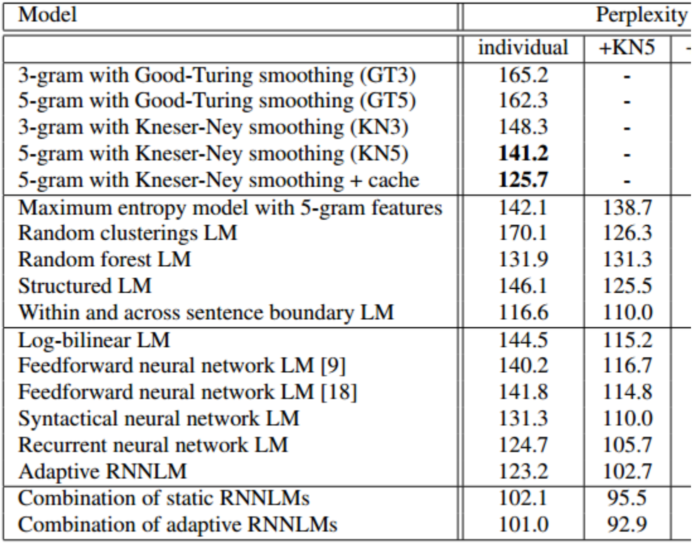
- Maximum-likelihood Estimation (MLE)
  - Smoothed count-based models are not MLE
  - (Un-regularized) NNLM models are MLE
  - Property of the estimation objective, not the model structure
- Markov Models
  - n-gram models are Markov models
  - windowed NNLM models are also Markov models
  - RNN-LM models 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)



Model	Perplexity		
	individual	+KN5	-
3-gram with Good-Turing smoothing (GT3)	165.2	-	
5-gram with Good-Turing smoothing (GT5)	162.3	-	
3-gram with Kneser-Ney smoothing (KN3)	148.3	-	
5-gram with Kneser-Ney smoothing (KN5)	**141.2**	-	
5-gram with Kneser-Ney smoothing + cache	**125.7**	-	
Maximum entropy model with 5-gram features	142.1	138.7	
Random clusterings LM	170.1	126.3	
Random forest LM	131.9	131.3	
Structured LM	146.1	125.5	
Within and across sentence boundary LM	116.6	110.0	
Log-bilinear LM	144.5	115.2	
Feedforward neural network LM [9]	140.2	116.7	
Feedforward neural network LM [18]	141.8	114.8	
Syntactical neural network LM	131.3	110.0	
Recurrent neural network LM	124.7	105.7	
Adaptive RNNLM	123.2	102.7	
Combination of static RNNLMs	102.1	95.5	
Combination of adaptive RNNLMs	101.0	92.9	

Review: RNNs for Language Modeling

- Recent popularization of RNNs has been based on language modeling (Mikolov, 2012)
- In particular RNNs allow for non-Markovian models

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

- Compare this to the windowed approach.

$$p(w_i | w_{i-n+1}, \ldots, w_{i-1}; \theta) = O(\mathbf{s}_i)$$

Shannon’s Markov Babblers I

4. Third-order approximation (trigram structure as in English). IN NO 1ST LAT WHEY CRATICT FROURE BIRS GROCID PONDENOME OF DEMONSTURES OF THE REPTAGIN IS REGOACTIONA OF CRE

5. First-Order Word Approximation. Rather than continue with tetragram, ... , II-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. REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE THESE.

Shannon's Markov Babblers II

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 $\hat{\mathbf{y}}$ to decide on next word.
- In particular RNNs allow for non-Markovian models

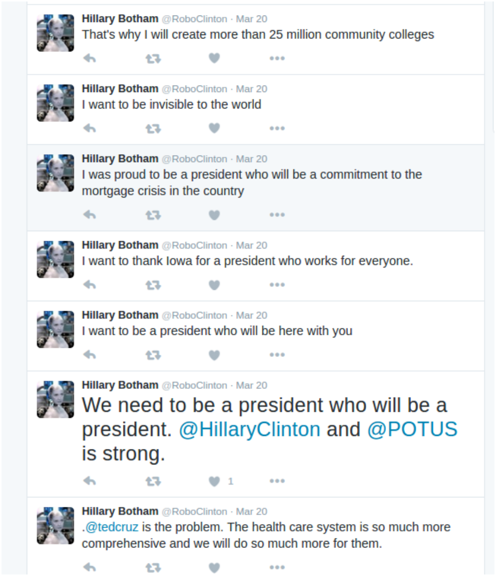
$$p(w_i | w_1, \dots, w_{i-1}; \theta) = \hat{\mathbf{y}}_i = O(\mathbf{s}_i)$$

- Greedy non-Markov babbler:

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

Babbling Visually

$$\begin{tikzcd}
\hat{\mathbf{y}}_1 \arrow[d, no head] \arrow[ddr, dashed] & \hat{\mathbf{y}}_2 \arrow[d, no head] \arrow[ddr, dashed] & \hat{\mathbf{y}}_3 \arrow[d, no head] \\
\mathbf{s}_1 \arrow[r, no head] \arrow[d, no head] & \mathbf{s}_2 \arrow[r, no head] \arrow[d, no head] & \mathbf{s}_3 \arrow[d, no head] \\
\mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3
\end{tikzcd}$$



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.
1

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.

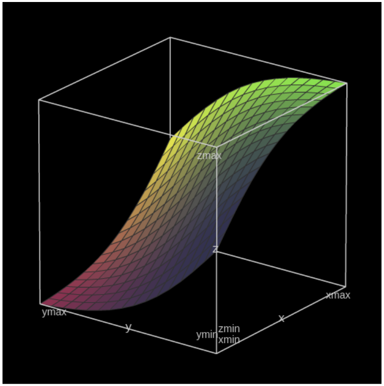
Diversity in Babbling

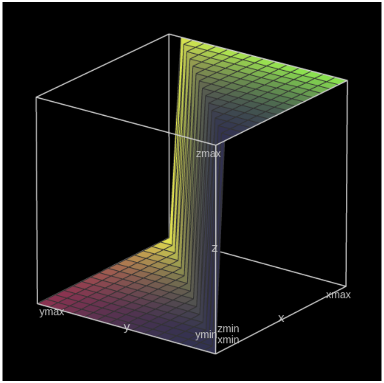
- Taking the greedy solution can lead to boring output.
- Instead can sample from soft-max distribution,

$$\hat{\mathbf{y}} \sim \mathrm{softmax}(\mathbf{s}_i \mathbf{W} + \mathbf{b})$$

- However this can lead to non-fluent output.

Review: Why is it called the softmax?





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

$$\arg\max([x\ y]) = \mathbf{1}(x > y)$$

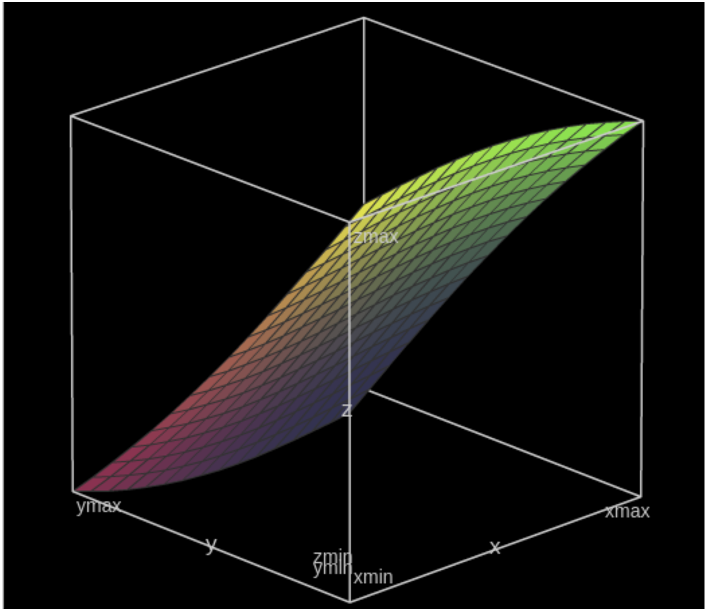
Temperature

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

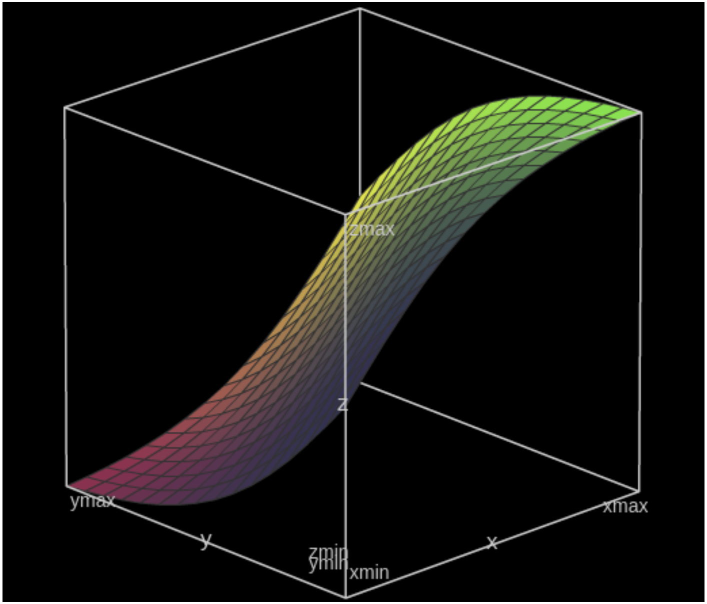
$$\hat{\mathbf{y}} \sim \text{softmax}((\mathbf{s}_i \mathbf{W} + \mathbf{b}) / t)$$

- What happens when the parameter is closer to the extremes?

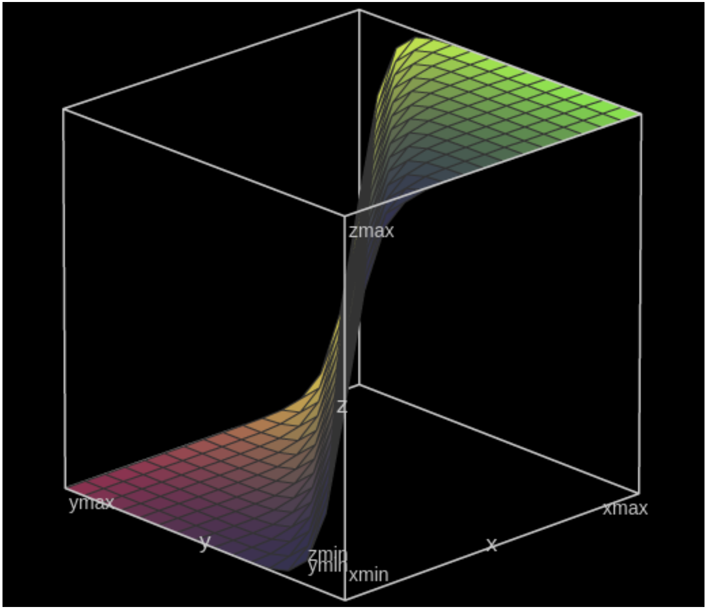
Temperature $t = 0.99$



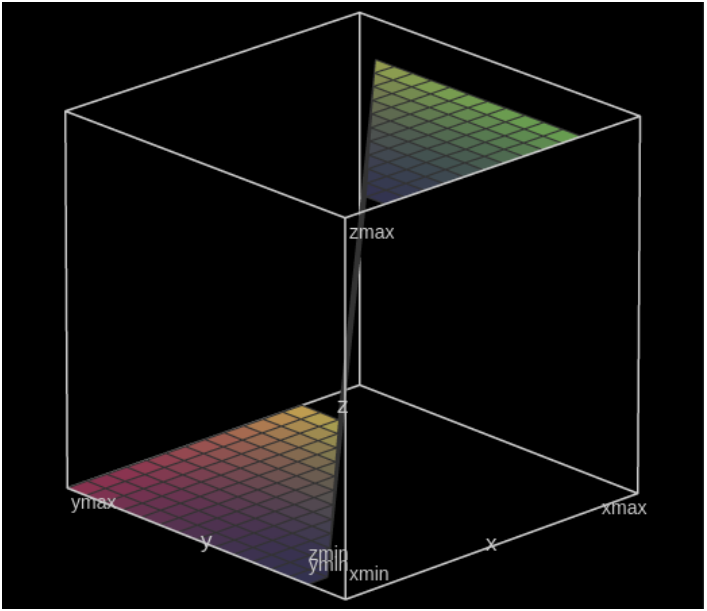
Temperature $t = 0.5$



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

```
t h e d o g w a l k s t o t h e p a r k
```

You should output,

`the dog walks to the park`.

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

So Far: Multiclass Classification

Every problem:

- Input encoding $\mathbf{x}$ (sparse or dense)
- Output encoding $\mathbf{y}$ (sparse index $\mathcal{C}$)
- Prediction $\hat{\mathbf{y}}$
- Classification

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

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, hasdigits, etc.
- Also include features on previous tags

Example: Rare word tagging with context

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

$$\begin{aligned}\mathbf{x} &= \delta(\texttt{last:CD}) + \delta(\texttt{prefix:3:reg}) + \delta(\texttt{prefix:2:re}) \\ &+ \delta(\texttt{prefix:1:r}) + \delta(\texttt{has-hyphen}) \\ &+ \delta(\texttt{lower-case}) + \delta(\texttt{suffix:3:son}) \ldots\end{aligned}$$

Standard Approach

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{x}\mathbf{W} + \mathbf{b})$$

- Let $\mathcal{C}$ to all possible tag sequences.
- Let $\mathbf{x}$ to be a representation of whole sentence.
- Let $\hat{\mathbf{y}}$ 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 $d_{\mathrm{out}} = O(\mathcal{T}^n)$ sequences!
- Just naively computing the softmax is exponential in length.
- Even if you could compute the softmax, $\mathbf{W} \in \mathbb{R}^{d_{\mathrm{in}} \times d_{\mathrm{out}}}$ would be impossible to train.

Predicting Structure

Instead we will try to maximize over the sequence,

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

- Note: This is a test-time optimization, different from training.
- We will discuss many different scoring functions f.

Encoding Output

We will assume that,

- Assume true output is a sequence,

$$\mathbf{y}_1, \ldots \mathbf{y}_n$$

- Each is a sparse one-hot vector $\mathbf{y}_1 = c_1, \ldots \mathbf{y}_n = c_n$

- Training data consists of labeled sequences

$$(\mathbf{x}^{(1)}_{1:n}, \mathbf{y}^{(1)}_{1:n}), \ldots (\mathbf{x}^{(J)}_{1:n}, \mathbf{y}^{(J)}_{1:n})$$

.

Contents

Hidden Markov Model

- Natural extension of naive Bayes to sequences.
- Allows us to efficiently solve argmax,

$$\arg\max_{\hat{\mathbf{y}} \in \mathcal{Y}} f(\mathbf{x}, \hat{\mathbf{y}}; \theta)$$

Review: Multinomial Naive Bayes

Reminder, joint probability chain rule,

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}|\mathbf{y})p(\mathbf{y})$$

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

$$\begin{aligned}
p(\mathbf{x}, \mathbf{y}) &= p(x_{f_1} = 1, \ldots, x_{f_k} = 1 | \mathbf{y} = \delta(c)) p(\mathbf{y} = \delta(c)) = \\
&= \prod_{i=1}^{k} p(x_{f_i} = 1 | x_{f_1} = 1, \ldots, x_{f_{i-1}} = 1, \mathbf{y} = \delta(c)) p(\mathbf{y} = \delta(c)) \approx \\
&= \prod_{i=1}^{k} p(x_{f_i} = 1 | \mathbf{y}) p(\mathbf{y})
\end{aligned}$$

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

Review: Multinomial Naive Bayes

Reminder, joint probability chain rule,

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}|\mathbf{y})p(\mathbf{y})$$

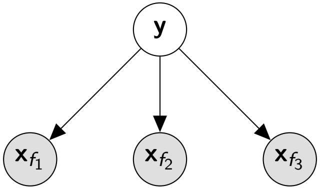
For a sparse features, with observed classes we can write as,

$$\begin{aligned}
p(\mathbf{x}, \mathbf{y}) &= p(x_{f_1} = 1, \ldots, x_{f_k} = 1|\mathbf{y} = \delta(c))p(\mathbf{y} = \delta(c)) = \\
&= \prod_{i=1}^{k} p(x_{f_i} = 1|x_{f_1} = 1, \ldots, x_{f_{i-1}} = 1, \mathbf{y} = \delta(c))p(\mathbf{y} = \delta(c)) \approx \\
&= \prod_{i=1}^{k} p(x_{f_i} = 1|\mathbf{y})p(\mathbf{y})
\end{aligned}$$

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.

Generative Sequence Model

- Define the function f to be the joint probability,

$$f(\mathbf{x}, c_1, \ldots, c_n) = p(\mathbf{y}_1 = \delta(c_1), \ldots, \mathbf{y}_n = \delta(c_n), \mathbf{x}_1, \ldots, \mathbf{x}_n)$$

- Aim will be to maximize this,

$$\operatorname*{arg\,max}_{c_1 \in \mathcal{C}, \ldots, c_n \in \mathcal{C}} f(\mathbf{x}, c_1, \ldots, c_n; \theta)$$

Hidden Markov Model

$$\begin{aligned}
f(\mathbf{x}, c_{1:n}) &= p(\mathbf{y}_1 = \delta(c_1), \ldots, \mathbf{y}_n = \delta(c_n), \mathbf{x}_{1:n}) \\
&= p(\mathbf{x}_{1:n} | \mathbf{y}_{1:n}) p(\mathbf{y}_{1:n}) \\
&= \prod_{i=1}^{n} p(\mathbf{x}_i | \mathbf{x}_{1:i-1}, \mathbf{y}_{1:n}) p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_1 = \delta(c_1) \ldots \mathbf{y}_{i-1} = \delta(c_{i-} \\
&\approx \prod_{i=1}^{n} p(\mathbf{x}_i | \mathbf{x}_{1:i-1}, \mathbf{y}_{1:n}) p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_{i-1} = \delta(c_{i-1})) \\
&\approx \prod_{i=1}^{n} p(\mathbf{x}_i | \mathbf{y}_i = \delta(c_i)) p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_{i-1} = \delta(c_{i-1}))
\end{aligned}$$

Hidden Markov Model

$$
\begin{aligned}
f(\mathbf{x}, c_{1:n}) &= p(\mathbf{y}_1 = \delta(c_1), \ldots, \mathbf{y}_n = \delta(c_n), \mathbf{x}_{1:n}) \\
&= p(\mathbf{x}_{1:n} | \mathbf{y}_{1:n}) p(\mathbf{y}_{1:n}) \\
&= \prod_{i=1}^{n} p(\mathbf{x}_i | \mathbf{x}_{1:i-1}, \mathbf{y}_{1:n}) p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_1 = \delta(c_1) \ldots \mathbf{y}_{i-1} = \delta(c_{i-} \\
&\approx \prod_{i=1}^{n} p(\mathbf{x}_i | \mathbf{x}_{1:i-1}, \mathbf{y}_{1:n}) p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_{i-1} = \delta(c_{i-1})) \\
&\approx \prod_{i=1}^{n} p(\mathbf{x}_i | \mathbf{y}_i = \delta(c_i)) p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_{i-1} = \delta(c_{i-1}))
\end{aligned}
$$

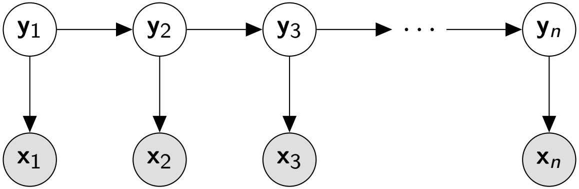
Hidden Markov Model

$$
\begin{aligned}
f(\mathbf{x}, c_{1:n}) &= p(\mathbf{y}_1 = \delta(c_1), \ldots, \mathbf{y}_n = \delta(c_n), \mathbf{x}_{1:n}) \\
&= p(\mathbf{x}_{1:n} | \mathbf{y}_{1:n}) p(\mathbf{y}_{1:n}) \\
&= \prod_{i=1}^{n} p(\mathbf{x}_i | \mathbf{x}_{1:i-1}, \mathbf{y}_{1:n}) p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_1 = \delta(c_1) \ldots \mathbf{y}_{i-1} = \delta(c_{i-} \\
&\approx \prod_{i=1}^{n} p(\mathbf{x}_i | \mathbf{x}_{1:i-1}, \mathbf{y}_{1:n}) p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_{i-1} = \delta(c_{i-1})) \\
&\approx \prod_{i=1}^{n} p(\mathbf{x}_i | \mathbf{y}_i = \delta(c_i)) p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_{i-1} = \delta(c_{i-1}))
\end{aligned}
$$

Hidden Markov Model

$$\begin{aligned}
f(\mathbf{x}, c_{1:n}) &= p(\mathbf{y}_1 = \delta(c_1), \ldots, \mathbf{y}_n = \delta(c_n), \mathbf{x}_{1:n}) \\
&= p(\mathbf{x}_{1:n} | \mathbf{y}_{1:n}) p(\mathbf{y}_{1:n}) \\
&= \prod_{i=1}^{n} p(\mathbf{x}_i | \mathbf{x}_{1:i-1}, \mathbf{y}_{1:n}) p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_1 = \delta(c_1) \ldots \mathbf{y}_{i-1} = \delta(c_{i-} \\
&\approx \prod_{i=1}^{n} p(\mathbf{x}_i | \mathbf{x}_{1:i-1}, \mathbf{y}_{1:n}) p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_{i-1} = \delta(c_{i-1})) \\
&\approx \prod_{i=1}^{n} p(\mathbf{x}_i | \mathbf{y}_i = \delta(c_i)) p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_{i-1} = \delta(c_{i-1}))
\end{aligned}$$

Example: Hidden Markov Model



Hidden Markov Model

Hidden Markov model requires two distributions,

- Transition distribution

$$p(\mathbf{y}_i | \mathbf{y}_{i-1}; \theta)$$

- Emission distribution

$$p(\mathbf{x}_i | \mathbf{y}_i; \theta)$$

- How many total parameters?

Maximum Likelihood Estimation

- Sequences are assumed given.
- Therefore we can fit in the standard way

$$\mathcal{L}(\theta) = -\sum_{j=1}^{J}\sum_{i=1}^{n} \log p(\mathbf{x}_i^{(j)}|\mathbf{y}_i^{(j)};\theta) + \log p(\mathbf{y}_i^{(j)}|\mathbf{y}_{i-1}^{(j)};\theta)$$

- Note: training is very similar to multiclass.

Multinomial Hidden Markov Model

(Should look familiar)

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

Multinomial Hidden Markov Model

(Should look familiar)

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

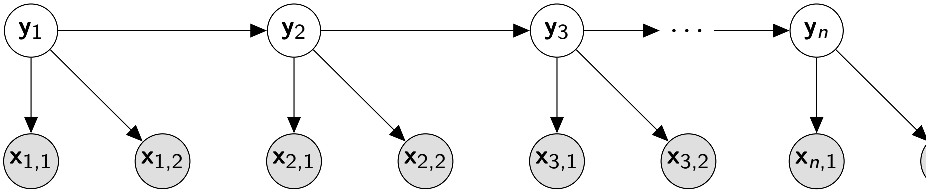
Example: Named-Entity Recognition

U.N.	NNP	I-NP	I-ORG
official	NN	I-NP	O
Ekeus	NNP	I-NP	I-PER
heads	VBZ	I-VP	O
for	IN	I-PP	O
Baghdad	NNP	I-NP	I-LOC
.	.	O	O

- F is constructed by counting up rows
- T is constructed by iterating over column
- Standard multinomial MLE issues apply (smoothing, rare-words)

Feature HMM

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?

Contents

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 $p(\mathbf{y} = c|\mathbf{x}; \theta)$

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

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

$$\hat{c} = \arg\max_{c \in \mathcal{C}} (\mathbf{x}\mathbf{W} + \mathbf{b})_c$$

Review: Multiclass Logistic Regression: A Model with Many Names

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

$$\log \operatorname{softmax}(\mathbf{z}) = \mathbf{z} - \log \sum_{c} \exp(z_c)$$

- 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

$$
\begin{aligned}
f(\mathbf{x}, c_1, \ldots, c_n; \theta) &= p(\mathbf{y}_1 = \delta(c_1), \ldots, \mathbf{y}_n = \delta(c_n) | \mathbf{x}_{1:n}) \\
&= \prod_{i=1}^{n} p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_1 = \delta(c_1) \ldots \mathbf{y}_{i-1} = \delta(c_{i-1}), \mathbf{x}_{1:n}) \\
&\approx \prod_{i=1}^{n} p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_{i-1} = \delta(c_{i-1}), \mathbf{x}_{1:n})
\end{aligned}
$$

Discriminative Markov Model

$$\begin{aligned}
f(\mathbf{x}, c_1, \ldots, c_n; \theta) &= p(\mathbf{y}_1 = \delta(c_1), \ldots, \mathbf{y}_n = \delta(c_n) | \mathbf{x}_{1:n}) \\
&= \prod_{i=1}^{n} p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_1 = \delta(c_1) \ldots \mathbf{y}_{i-1} = \delta(c_{i-1}), \mathbf{x}_{1:n}) \\
&\approx \prod_{i=1}^{n} p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_{i-1} = \delta(c_{i-1}), \mathbf{x}_{1:n})
\end{aligned}$$

Discriminative Markov Model

$$\begin{aligned}
f(\mathbf{x}, c_1, \ldots, c_n; \theta) &= p(\mathbf{y}_1 = \delta(c_1), \ldots, \mathbf{y}_n = \delta(c_n) | \mathbf{x}_{1:n}) \\
&= \prod_{i=1}^{n} p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_1 = \delta(c_1) \ldots \mathbf{y}_{i-1} = \delta(c_{i-1}), \mathbf{x}_{1:n}) \\
&\approx \prod_{i=1}^{n} p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_{i-1} = \delta(c_{i-1}), \mathbf{x}_{1:n})
\end{aligned}$$

Maximum-Entropy Markov Models

$$\begin{tikzcd}
\mathbf{y}_1 \arrow[r] & \mathbf{y}_2 \arrow[r] & \mathbf{y}_3 \arrow[r] & \cdots \arrow[r] & \mathbf{y}_n
\end{tikzcd}$$

Maximum Entropy Markov Model

MEMM estimates only a transition distribution,

- Transition distribution (also conditioned on input)

$$p(\mathbf{y}_i|\mathbf{y}_{i-1} = \delta(c_{i-1}), \mathbf{x}_1, \ldots, \mathbf{x}_n) = \text{softmax}(\textit{feat}(\mathbf{x}, c_{i-1})\mathbf{W} + \mathbf{b})$$

- Here *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

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

$$
\begin{aligned}
\mathbf{x} &= \delta(\texttt{last:CD}) + \delta(\texttt{prefix:3:reg}) + \delta(\texttt{prefix:2:re}) \\
&+ \delta(\texttt{prefix:1:r}) + \delta(\texttt{has-hyphen}) \\
&+ \delta(\texttt{lower-case}) + \delta(\texttt{suffix:3:son}) \ldots
\end{aligned}
$$

Contents

Using Markov Model

- Assume that we have $c_1, \dots c_{i-1}$.
- Following previous notation let $\hat{\mathbf{y}}_i$ 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

$$\hat{\mathbf{y}}_i(c_{i-1}) = \text{softmax}(\log p(\mathbf{y}_i|\mathbf{y}_{i-1} = \delta(c_{i-1})) + \log p(\mathbf{y}_i|\mathbf{x}_i))$$

- MEMM

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

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

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

Next Time

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