TTIC 31190: Natural Language Processing

Kevin Gimpel

Spring 2018

Lecture 12: Sequence Labeling and Structured Prediction

Project Proposal

- project proposal due in one week

Midterm

- midterm on Wednesday, May 16^{th}
- we'll give you the formulas/definitions you will need

Roadmap

- words, morphology, lexical semantics
- text classification
- language modeling
- word embeddings
- recurrent/recursive/convolutional networks in NLP
- sequence labeling, HMMs, dynamic programming
- syntax and syntactic parsing
- semantics, compositionality, semantic parsing
- machine translation and other NLP tasks

Sequence Labeling Tasks in NLP

Part-of-Speech Tagging

determiner	verb (past)	prep.	proper noun	proper noun	poss.	adj.	noun
Some	questioned	if	Tim	Cook	’s	first	product

modal	verb	det.	adjective	noun	prep.	proper noun	punc.
would	be	a	breakaway	hit	for	Apple	.

Named Entity Recognition

O	O	O	B-PERSON	I-PERSON	O	O	O
Some	questioned	if	Tim	Cook	’s	first	product

O	O	O	O	O	O	B-ORGANIZATION	O
would	be	a	breakaway	hit	for	Apple	.

Penn Treebank tag set

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two*	TO	"to"	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential 'there'	*there*	VB	verb base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb gerund	*eating*
JJ	adjective	*yellow*	VBN	verb past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, sing.	*IBM*	$	dollar sign	*$*
NNPS	proper noun, plural	*Carolinas*	#	pound sign	*#*
PDT	predeterminer	*all, both*	"	left quote	' or "
POS	possessive ending	*'s*	"	right quote	' or "
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one's*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

Part-of-Speech

- **open-class**:
  - nouns, verbs, adjectives, adverbs
  - “open” because new words in these categories are often created
- **closed-class**:
  - function words like determiners and prepositions
  - new function words rarely catch on
  - (though new forms/variants of function words do appear, especially in “conversational text”)

POS Ambiguity

			WSJ		Brown	
Types:						
	Unambiguous	(1 tag)	44,432	(**86%**)	45,799	(**85%**)
	Ambiguous	(2+ tags)	7,025	(**14%**)	8,050	(**15%**)
Tokens:						
	Unambiguous	(1 tag)	577,421	(**45%**)	384,349	(**33%**)
	Ambiguous	(2+ tags)	711,780	(**55%**)	786,646	(**67%**)

Figure 10.2 The amount of tag ambiguity for word types in the Brown and WSJ corpora, from the Treebank-3 (45-tag) tagging. These statistics include punctuation as words, and assume words are kept in their original case.

- most word types have only one tag
  - frequent word types have more tags
  - rare words are often nouns or verbs

Universal Tag Set

- contains 12 tags:
  - noun, verb, adjective, adverb, pronoun, determiner, adposition, numeral, conjunction, particle, punctuation, other

sentence:	The	oboist	Heinz	Holliger	has	taken	a	hard	line	about	the	problems	.
original:	DT	NN	NNP	NNP	VBZ	VBN	DT	JJ	NN	IN	DT	NNS	.
universal:	DET	NOUN	NOUN	NOUN	VERB	VERB	DET	ADJ	NOUN	ADP	DET	NOUN	.

Figure 1: Example English sentence with its language specific and corresponding universal POS tags.

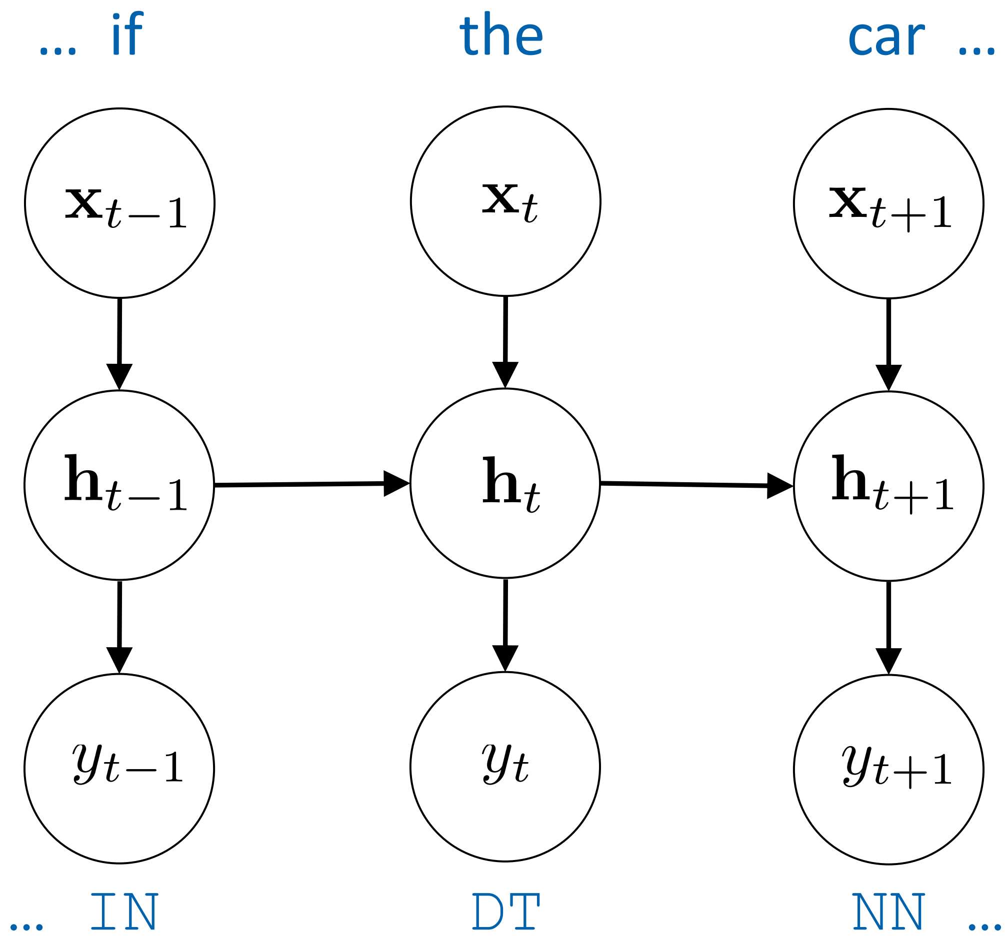
Petrov, Das, McDonald (2011)

Feed-Forward Networks for POS Tagging

- feed-forward networks are OK for tagging
- they tend to work best with very small contexts (e.g., 1 word to left & right)

- can also use convolutional networks defined on a window centered on the target word

RNNs for Part-of-Speech Tagging



RNN Taggers

- RNN POS taggers are simple and effective
- most common is to use some sort of bidirectional RNN, like a BiLSTM or BiGRU

RNN Taggers

- RNN taggers are not structured predictors

- yes, a structure is being predicted, but predictions for neighboring words are independent!

Sequence Labeling as Structured Prediction

Modeling, Inference, Learning in Structured Prediction

inference: solve argmax

modeling: define score function

$$\text{classify}(\boldsymbol{x}, \boldsymbol{w}) = \underset{\boldsymbol{y}}{\text{argmax}} \ \text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w})$$

learning: choose $\boldsymbol{w}$

Modeling, Inference, Learning in Structured Prediction

modeling: define score function

$$\text{classify}(\boldsymbol{x}, \boldsymbol{w}) = \underset{\boldsymbol{y}}{\text{argmax}} \ \text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w})$$

- **Modeling**: How do we assign a score to an (***x***,***y***) pair using parameters ***w***?

Modeling, Inference, Learning in Structured Prediction

inference: solve argmax

modeling: define score function

$$\text{classify}(\boldsymbol{x}, \boldsymbol{w}) = \underset{\boldsymbol{y}}{\operatorname{argmax}} \; \text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w})$$

- **Inference**: How do we efficiently search over the space of all outputs?

Modeling, Inference, Learning in Structured Prediction

inference: solve argmax

modeling: define score function

$$\text{classify}(\boldsymbol{x}, \boldsymbol{w}) = \underset{\boldsymbol{y}}{\text{argmax}}\ \text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w})$$

learning: choose $\boldsymbol{w}$

- **Learning**: How do we choose the weights ***w***?

Learning in Structured Prediction

$$\text{classify}(\boldsymbol{x}, \boldsymbol{w}) = \underset{\boldsymbol{y}}{\text{argmax}} \;\; \text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w})$$

learning: choose $\boldsymbol{w}$

- learning in structured prediction is similar to learning in multi-class classification
- we can use the same loss functions

Loss Subgradients for Linear Models

- perceptron loss:

$$\text{loss}_{\text{perc}}(\boldsymbol{x}, y, \mathbf{w}) = -\sum_i w_i f_i(\boldsymbol{x}, y) \; + \; \max_{y' \in \mathcal{L}} \sum_i w_i f_i(\boldsymbol{x}, y')$$

- subderivative for a single parameter:

$$\frac{\partial \text{loss}_{\text{perc}}(\boldsymbol{x}, y, \mathbf{w})}{\partial w_j} = -f_j(\boldsymbol{x}, y) + f_j(\boldsymbol{x}, \text{classify}(\boldsymbol{x}, \mathbf{w}))$$

Loss Subgradients for Linear Models

- hinge loss:

$$\text{loss}_{\text{hinge}}(\boldsymbol{x}, y, \mathbf{w}) = -\sum_i w_i f_i(\boldsymbol{x}, y) + \max_{y' \in \mathcal{L}} \left(\sum_i w_i f_i(\boldsymbol{x}, y') + \text{cost}(y, y') \right)$$

- subderivative for a single parameter:

$$\frac{\partial \text{loss}_{\text{hinge}}(\boldsymbol{x}, y, \mathbf{w})}{\partial w_j} = -f_j(\boldsymbol{x}, y) + f_j(\boldsymbol{x}, \text{costClassify}(\boldsymbol{x}, y, \mathbf{w}))$$

Modeling, Inference, Learning in Structured Prediction

inference: solve argmax

modeling: define score function

$$\text{classify}(\boldsymbol{x}, \boldsymbol{w}) = \underset{\boldsymbol{y}}{\text{argmax}} \ \text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w})$$

learning: choose $\boldsymbol{w}$

- for learning, we can use the same loss functions
- but learning requires inference (classify/costClassify)
- inference becomes much more difficult in the structured setting

Applications of our Classifier Framework so far

task	**input (*x*)**	**output (*y*)**	**output space ($\mathcal{L}$)**	**size of $\mathcal{L}$**
text classification	a sentence	gold standard label for *x*	pre-defined, small label set (e.g., {positive, negative})	2-10

Applications of our Classifier Framework so far

task	**input (*x*)**	**output (*y*)**	**output space ($\mathcal{L}$)**	**size of $\mathcal{L}$**
text classification	a sentence	gold standard label for *x*	pre-defined, small label set (e.g., {positive, negative})	2-10
word sense disambiguation	instance of a particular word (e.g., *bass*) with its context	gold standard word sense of *x*	pre-defined sense inventory from WordNet for *bass*	2-30
learning skip-gram word embeddings	instance of a word in a corpus	a word in the context of *x* in a corpus	vocabulary	$\|V\|$
part-of-speech tagging	a sentence	gold standard part-of-speech tags for *x*	all possible part-of-speech tag sequences with same length as *x*	$\|P\|^{\|x\|}$

Applications of our Classifier Framework so far

task	input (x)	output (y)	output space ($\mathcal{L}$)	size of $\mathcal{L}$
text classification	a sentence	gold standard label for x	pre-defined, small label set (e.g., {positive, negative})	2-10
word sense disambiguation	instance of a particular word (e.g., *bass* … its cont…	gold standard …	pre-defined sense inventory from …	2-30
learning skip-gram word embeddings	instance … word in a c…			
part-of-speech tagging	a sentence	gold standard part-of-speech tags for x	all possible part-of-speech tag sequences with same length as x	$\|P\|^{\|x\|}$

exponential in size of input!

“structured prediction”

Inference for Text Classification

$$\text{classify}(\boldsymbol{x}, \boldsymbol{w}) = \operatorname*{argmax}_{y \in \{\text{positive}, \text{negative}\}} \text{score}(\boldsymbol{x}, y, \boldsymbol{w})$$

- trivial (loop over labels)

Inference for Structured Prediction

$$\text{classify}(\boldsymbol{x}, \boldsymbol{w}) = \operatorname*{argmax}_{\boldsymbol{y}} \ \text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w})$$

- how do we efficiently search over the space of all structured outputs?
- this space may have size exponential in the size of the input, or be unbounded

- complexity of inference is closely linked to the score function

Feature Locality

- **feature locality**: how "big" are your features?
- we need to be mindful of this to enable efficient inference
- features can be arbitrarily big in terms of the *input*
- but features **cannot** be arbitrarily big in terms of the *output*!

Features for Part-of-Speech Tagging

are these features big or small?

feature	big or small?
feature that counts instances of "*the*" in the input sentence, along with checking current tag	
feature that returns square root of counts of *am/is/was/were*, along with checking current tag	
feature that counts "`verb verb`" sequences	
feature that counts "`determiner noun verb verb`" sequences	
feature that returns 1 if and only if there are 5 nouns in a sentence	
feature that returns the ratio of nouns to verbs	

Features for Part-of-Speech Tagging

are these features big or small?

feature	big or small?
feature that counts instances of "*the*" in the input sentence, along with checking current tag	small
feature that returns square root of counts of *am/is/was/were*, along with checking current tag	small
feature that counts "`verb verb`" sequences	pretty small
feature that counts "`determiner noun verb verb`" sequences	pretty big!
feature that returns 1 if and only if there are 5 nouns in a sentence	big, but we can design specialized algorithms to handle them if they're the only big features
feature that returns the ratio of nouns to verbs	

Features for POS Tagging

- when designing features for structured prediction, focus on “small” features
- when considering larger features, start small
  - tag bigrams work well and are pretty efficient
  - tag trigrams are potentially powerful, but slow
  - what’s another problem with tag trigrams?
- remember: you can always define features based on the entire input… these are always “small”

Hidden Markov Models

- simple, useful, well-known model for sequence labeling: **Hidden Markov Model (HMM)**
- HMMs are used in NLP, speech processing, computational biology, and other areas
- good starting point for learning about **graphical models**

Markov Assumption


Andrei Markov

- simplifying assumption in language modeling:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

- or maybe:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$$

Hidden Markov Models

- *n*-gram language models define a probability distribution over word sequences ***x***

- HMMs define a joint probability distribution over input sequences ***x*** and output sequences ***y***

$$p(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{|\boldsymbol{x}|} p(x_i \mid x_1, ..., x_{i-1}, y_1, ..., y_i) p(y_i \mid x_1, ..., x_{i-1}, y_1, ..., y_{i-1})$$

- conditional independence assumptions ("Markov assumption") are used to factorize this joint distribution into small terms

*for now, we are omitting stopping probabilities for clarity

Independence

- **Independence**: two random variables *X* and *Y* are independent if:

$$P(X = x, Y = y) = P(X = x)P(Y = y)$$

(or $P(x, y) = P(x)P(y)$)

for all values *x* and *y*

we write this as: $X \perp Y$

Independence and Conditional Independence

- **Independence**: two random variables *X* and *Y* are independent if:

$$P(X = x, Y = y) = P(X = x)P(Y = y)$$

(or $P(x, y) = P(x)P(y)$)

$$X \perp Y$$

for all values *x* and *y*

- **Conditional Independence**: two random variables *X* and *Y* are conditionally independent given a third variable *Z* if

$$P(x, y \mid z) = P(x \mid z)P(y \mid z)$$

for all values of *x*, *y*, and *z*

$$X \perp Y \mid Z$$

(or $P(x \mid y, z) = P(x \mid z)$)

Markov Assumption

Andrei Markov

- simplifying assumption:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

- or maybe:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$$

$$W_t \perp W_{t-3}, W_{t-4}, \ldots, W_1 \mid W_{t-1}, W_{t-2}$$

J&M/SLP3

Random Variables

- let's define random variables for observations:
  - observation variable at time step *t*: X_t
  - its possible values: words in vocabulary $\mathcal{V}$

- and we'll define one "hidden" variable for each observation:
  - hidden variable at time *t*: Y_t
  - its possible values: discrete symbols in some set
  - for now, think of the set of possible POS tags

Conditional Independence Assumptions of HMMs

- two *Y'*s are conditionally independent given the *Y'*s between them:

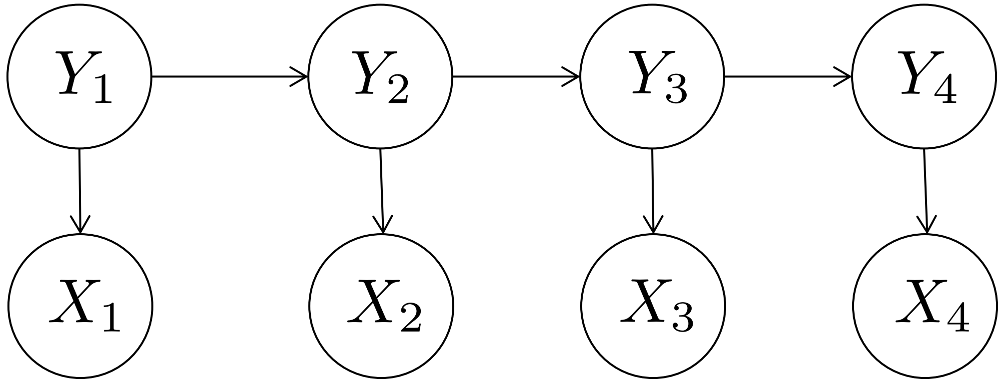
$$Y_{t-1} \perp Y_{t+1} \mid Y_t$$

- an *X* at position *t* is conditionally independent of other *Y'*s given the *Y* at position *t*:

$$X_t \perp Y_{t-1} \mid Y_t$$

Graphical Model for an HMM

(for a sequence of length 4)



a **graphical model** is a graph in which:

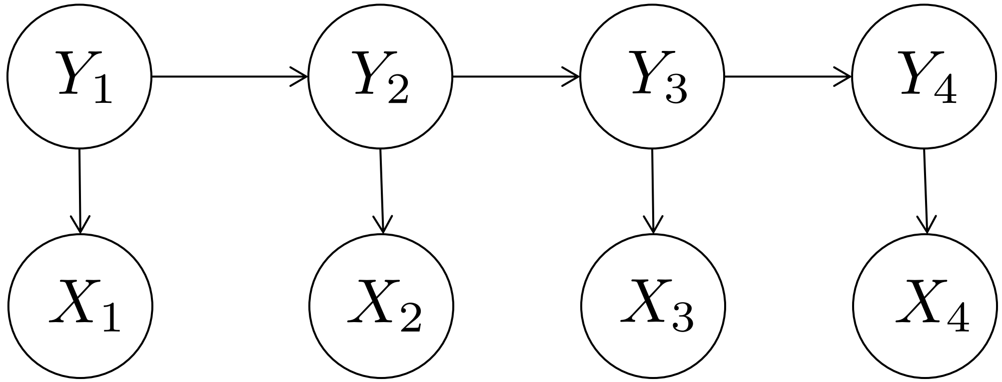
each node corresponds to a random variable

each directed edge corresponds to a conditional probability distribution of the target node given the source node

conditional independence statements among random variables are encoded by the edge structure

Graphical Model for an HMM

(for a sequence of length 4)



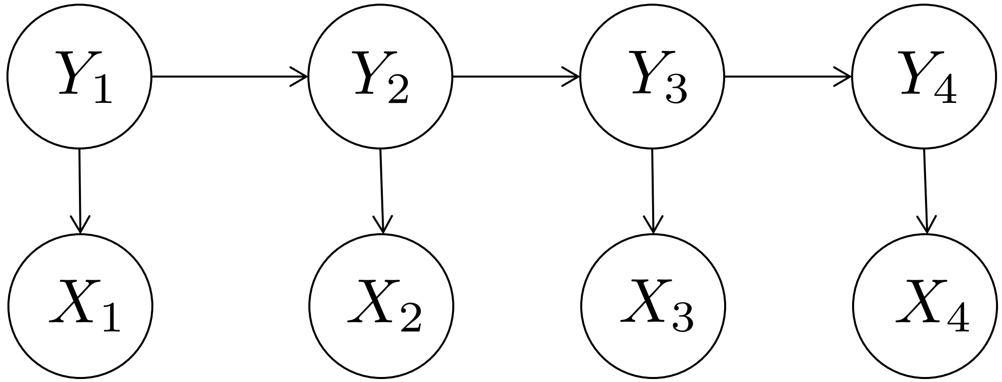
conditional independence statements among random variables are encoded by the edge structure:

$$Y_{t-1} \perp Y_{t+1} \mid Y_t$$

$$X_t \perp Y_{t-1} \mid Y_t$$

Graphical Model for an HMM

(for a sequence of length 4)



$$Y_{t-1} \perp Y_{t+1} \mid Y_t \qquad X_t \perp Y_{t-1} \mid Y_t$$

$$p(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{|\boldsymbol{x}|} p(x_i \mid x_1, ..., x_{i-1}, y_1, ..., y_i) p(y_i \mid x_1, ..., x_{i-1}, y_1, ..., y_{i-1})$$

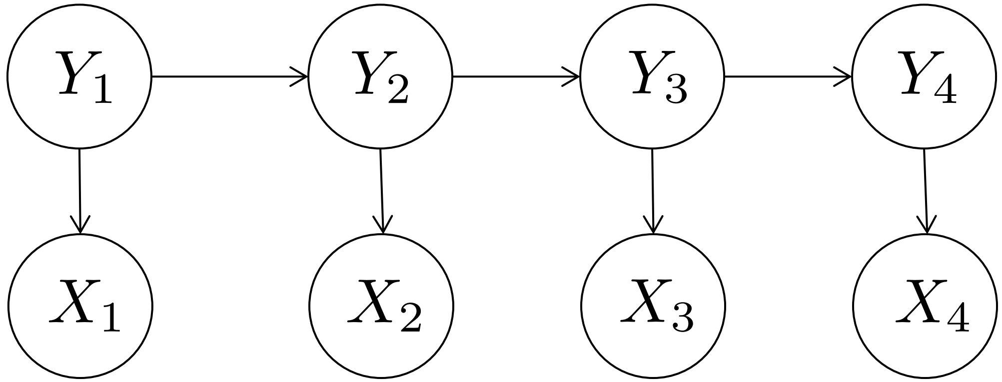
$$\Downarrow$$

$$p_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{|\boldsymbol{x}|} p_{\boldsymbol{\tau}}(y_i \mid y_{i-1}) p_{\boldsymbol{\eta}}(x_i \mid y_i)$$

*for now, we are omitting stopping probabilities for clarity

Graphical Model for an HMM

(for a sequence of length 4)



conditional independence statements encoded by edge structure → we only have to worry about **local distributions:**

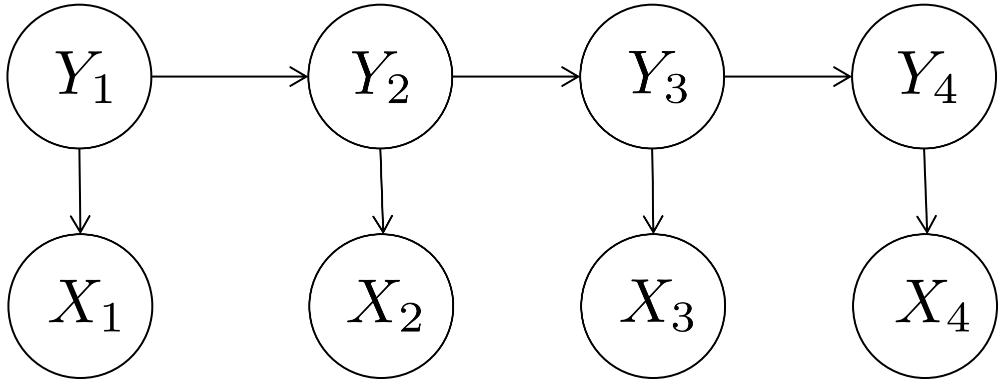
transition **parameters:** $p_{\boldsymbol{\tau}}(y_i \mid y_{i-1})$

emission **parameters:** $p_{\boldsymbol{\eta}}(x_i \mid y_i)$

$$p_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{|\boldsymbol{x}|} p_{\boldsymbol{\tau}}(y_i \mid y_{i-1}) p_{\boldsymbol{\eta}}(x_i \mid y_i)$$

Graphical Model for an HMM

(for a sequence of length 4)



$$p_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{|\boldsymbol{x}|} p_{\boldsymbol{\tau}}(y_i \mid y_{i-1}) p_{\boldsymbol{\eta}}(x_i \mid y_i)$$

transition parameters: $p_{\boldsymbol{\tau}}(y_i \mid y_{i-1})$

emission parameters: $p_{\boldsymbol{\eta}}(x_i \mid y_i)$

Important: Stopping Probabilities

$$p_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{|\boldsymbol{x}|} p_{\boldsymbol{\tau}}(y_i \mid y_{i-1}) p_{\boldsymbol{\eta}}(x_i \mid y_i)$$

$$p_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y}) = p_{\boldsymbol{\tau}}(</\mathrm{s}> \mid y_{|\boldsymbol{x}|}) \prod_{i=1}^{|\boldsymbol{x}|} p_{\boldsymbol{\tau}}(y_i \mid y_{i-1}) p_{\boldsymbol{\eta}}(x_i \mid y_i)$$

special end-of-sequence label

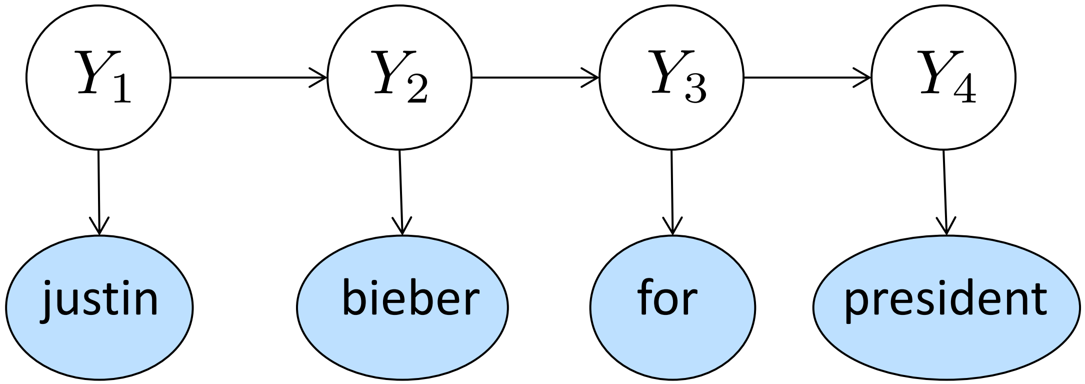
We also assume: $y_0 = <\mathrm{s}>$

special start-of-sequence label

why does this matter?

HMMs for Word Clustering

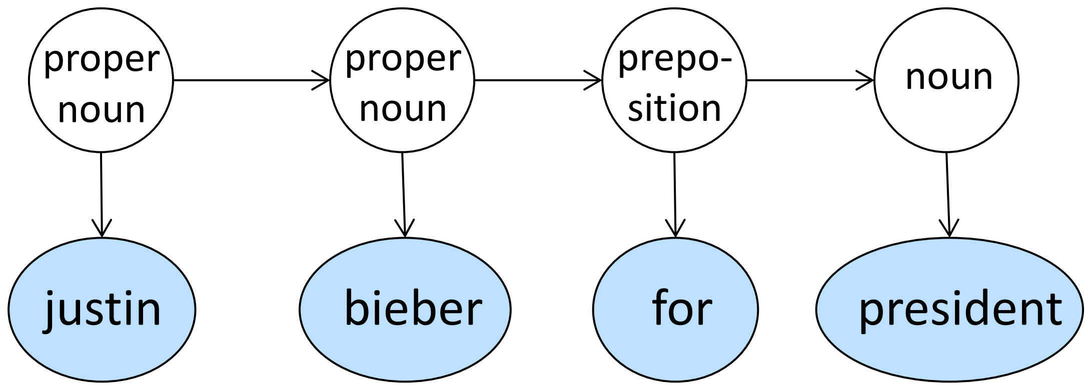
(Brown et al., 1992)



each $y_i \in \mathcal{L}$ is a cluster ID

so, label space is $\mathcal{L} = \{1, 2, ..., 100\}$

HMMs for Part-of-Speech Tagging



each $y_i \in \mathcal{L}$ is a part-of-speech tag
so, label space is $\mathcal{L} = \{\text{noun}, \text{verb}, \ldots\}$

what parameters need to be learned?

transition parameters: $p_{\boldsymbol{\tau}}(y_i \mid y_{i-1})$

emission parameters: $p_{\boldsymbol{\eta}}(x_i \mid y_i)$

How should we learn the HMM parameters?

transition parameters: $p_{\boldsymbol{\tau}}(y_i \mid y_{i-1})$

$p_{\boldsymbol{\tau}}(\text{verb} \mid \text{noun})$

$p_{\boldsymbol{\tau}}(\text{verb} \mid \text{adjective})$

$\ldots$

emission parameters: $p_{\boldsymbol{\eta}}(x_i \mid y_i)$

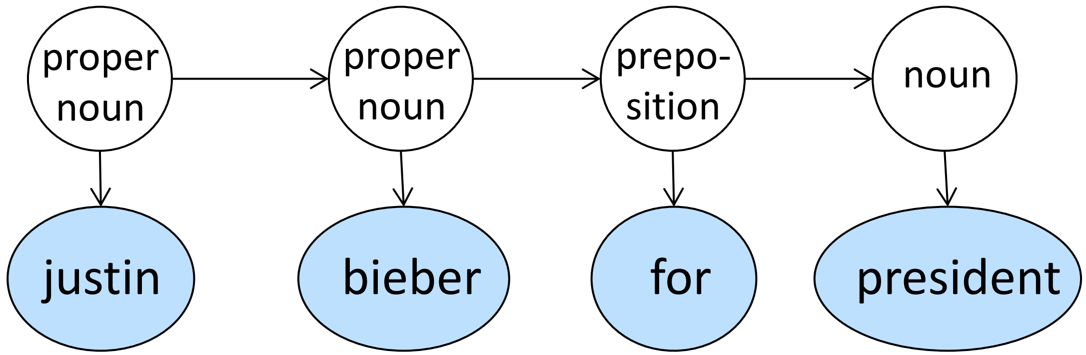
$p_{\boldsymbol{\eta}}(\mathit{for} \mid \text{verb})$

$p_{\boldsymbol{\eta}}(\mathit{walk} \mid \text{verb})$

$\ldots$

Supervised HMMs

- given a dataset of input sequences and annotated outputs:



- to estimate transition/emission distributions, use maximum likelihood estimation (count and normalize):

$$p_{\boldsymbol{\tau}}(y \mid y') \leftarrow \frac{\text{count}(y'\ y)}{\text{count}(y')} \qquad p_{\boldsymbol{\eta}}(x \mid y) \leftarrow \frac{\text{count}(y, x)}{\text{count}(y)}$$

$$p_{\boldsymbol{\tau}}(\text{verb} \mid \text{noun}) \leftarrow \frac{\text{count}(\text{noun}\ \ \text{verb})}{\text{count}(\text{noun})} \qquad p_{\boldsymbol{\eta}}(\textit{walk} \mid \text{verb}) \leftarrow \frac{\text{count}(\text{verb}, \textit{walk})}{\text{count}(\text{verb})}$$

Estimates of Tag Transition Probabilities

	proper noun	modal verb	infinitive verb	adjective	noun	adverb	determiner
	NNP	**MD**	**VB**	**JJ**	**NN**	**RB**	**DT**
$<s>$	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

Figure 9.5 The A transition probabilities $P(t_i|t_{i-1})$ computed from the WSJ corpus without smoothing. Rows are labeled with the conditioning event; thus $P(VB|MD)$ is 0.7968.

$$p_{\boldsymbol{\tau}}(y \mid y') \leftarrow \frac{\text{count}(y'\ y)}{\text{count}(y')}$$

Estimates of Emission Probabilities

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0.000097	0
NN	0	0.000200	0.000223	0.000006	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Figure 9.6 Observation likelihoods *B* computed from the WSJ corpus without smoothing.

$$p_{\boldsymbol{\eta}}(x \mid y) \leftarrow \frac{\mathrm{count}(y, x)}{\mathrm{count}(y)}$$

Inference in HMMs

$$\text{classify}(\boldsymbol{x}, \boldsymbol{w}) = \operatorname*{argmax}_{\boldsymbol{y}} \; p_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y})$$

$$= \operatorname*{argmax}_{\boldsymbol{y}} \; p_{\boldsymbol{\tau}}(</\text{s}> \mid y_{|\boldsymbol{x}|}) \prod_{i=1}^{|\boldsymbol{x}|} p_{\boldsymbol{\tau}}(y_i \mid y_{i-1}) p_{\boldsymbol{\eta}}(x_i \mid y_i)$$

- since the output is a sequence, this argmax requires iterating over an exponentially-large set
- we can use **dynamic programming (DP)** to solve these problems exactly
- for HMMs (and other sequence models), the algorithm for solving this is the **Viterbi algorithm**

Dynamic Programming (DP)

- what is dynamic programming?
  - a family of algorithms that break problems into smaller pieces and reuse solutions for those pieces
  - only applicable when the problem has certain properties (**optimal substructure** and **overlapping sub-problems**)

- we can often use DP to iterate over exponentially-large output spaces in polynomial time
- we focus on a particular type of DP algorithm: **memoization**

Viterbi Algorithm

- recursive equations + memoization:

base case:
returns probability of sequence starting with label *y* for first word

$$V(1, y) = p_{\boldsymbol{\eta}}(x_1 \mid y)\, p_{\boldsymbol{\tau}}(y \mid \text{<s>})$$

$$V(m, y) = \max_{y' \in \mathcal{L}} \left(p_{\boldsymbol{\eta}}(x_m \mid y)\, p_{\boldsymbol{\tau}}(y \mid y')\, V(m-1, y') \right)$$

recursive case:
computes probability of max-probability label sequence that ends with label *y* at position *m*

final value is in: $goal(\boldsymbol{x}) = \max_{y' \in \mathcal{L}} \left(p_{\boldsymbol{\tau}}(\text{</s>} \mid y')\, V(|\boldsymbol{x}|, y') \right)$

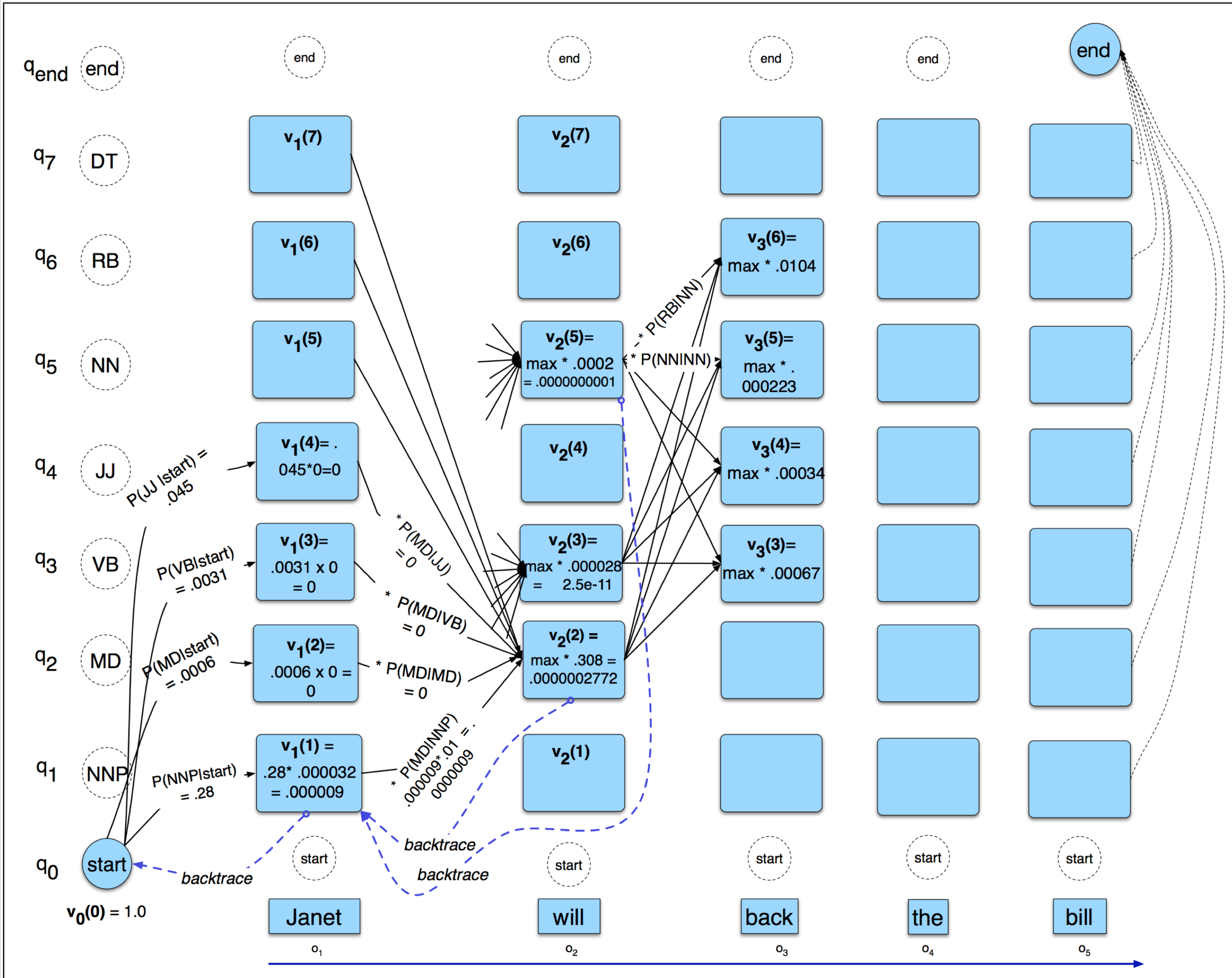
Example:

Janet	*will*	*back*	*the*	*bill*
proper noun	modal verb	infinitive verb	determiner	noun

Janet	*will*	*back*	*the*	*bill*
proper noun	modal verb	infinitive verb	determiner	noun

	NNP	**MD**	**VB**	**JJ**	**NN**	**RB**	**DT**
<s>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

	Janet	**will**	**back**	**the**	**bill**
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0.000097	0
NN	0	0.000200	0.000223	0.000006	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0



q_end end
q_7 DT
q_6 RB
q_5 NN
q_4 JJ
q_3 VB
q_2 MD
q_1 NNP
q_0 start
v_0(0) = 1.0
end
v_1(7)
v_1(6)
v_1(5)
v_1(4)= . 045*0=0
v_1(3)= .0031 x 0 = 0
v_1(2)= .0006 x 0 = 0
v_1(1) = .28* .000032 = .000009
P(JJ |start) = .045
P(VB|start) = .0031
P(MD|start) = .0006
P(NNP|start) = .28
* P(MD|JJ) = 0
* P(MD|VB) = 0
* P(MD|MD) = 0
* P(MD|NNP) .000009*.01 = .0000009
v_2(7)
v_2(6)
v_2(5)= max * .0002 = .0000000001
v_2(4)
v_2(3)= max * .000028 = 2.5e-11
v_2(2) = max * .308 = .0000002772
v_2(1)
* P(RB|NN)
* P(NN|NN)
v_3(6)= max * .0104
v_3(5)= max * . 000223
v_3(4)= max * .00034
v_3(3)= max * .00067
backtrace
backtrace
backtrace
start
Janet
will
back
the
bill
o_1
o_2
o_3
o_4
o_5

Viterbi Algorithm

- space and time complexity?
- can be read off from the recursive equations:

space complexity:
size of memoization table, which is # of unique indices of recursive equations

length of sentence * **number of labels**

$$V(m, y) = \max_{y' \in \mathcal{L}} \left(p_{\boldsymbol{\eta}}(x_m \mid y)\, p_{\boldsymbol{\tau}}(y \mid y')\, V(m-1, y') \right)$$

so, space complexity is O(|*x*| |*L*|)

Viterbi Algorithm

- space and **time** complexity?
- can be read off from the recursive equations:

time complexity:
size of memoization table * complexity of computing each entry

length of sentence * **number of labels** * **each entry requires iterating through the labels**

$$V(m, y) = \max_{y' \in \mathcal{L}} \left(p_{\boldsymbol{\eta}}(x_m \mid y)\, p_{\boldsymbol{\tau}}(y \mid y')\, V(m-1, y') \right)$$

so, time complexity is O(|*x*| |*L*| |*L*|) = O(|*x*| |*L*|^2)

Linear Sequence Models

- we can generalize HMMs and talk about linear models for scoring label sequences in our classifier framework

- but first, how do we know that the HMM scoring function is a linear model?

HMM as a Linear Model?

HMM: $$p_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y}) = p_{\boldsymbol{\tau}}(\text{</s>} \mid y_{|\boldsymbol{x}|}) \prod_{i=1}^{|\boldsymbol{x}|} p_{\boldsymbol{\tau}}(y_i \mid y_{i-1}) p_{\boldsymbol{\eta}}(x_i \mid y_i)$$

linear model: $$\text{score}(\boldsymbol{x}, \boldsymbol{y}, \mathbf{w}) = \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) = \sum_i w_i f_i(\boldsymbol{x}, \boldsymbol{y})$$

$$p_{\mathbf{w}}(\boldsymbol{x}, \boldsymbol{y}) \propto \exp\{\text{score}(\boldsymbol{x}, \boldsymbol{y}, \mathbf{w})\}$$

- what are the feature templates and weights?

HMM as a Linear Model

HMM: $p_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y}) = p_{\boldsymbol{\tau}}(\text{</s>} \mid y_{|\boldsymbol{x}|}) \prod_{i=1}^{|\boldsymbol{x}|} p_{\boldsymbol{\tau}}(y_i \mid y_{i-1}) p_{\boldsymbol{\eta}}(x_i \mid y_i)$

linear model: $\text{score}(\boldsymbol{x}, \boldsymbol{y}, \mathbf{w}) = \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) = \sum_i w_i f_i(\boldsymbol{x}, \boldsymbol{y})$

feature templates and weights:

$$f_{\boldsymbol{\tau}(y', y'')}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{|\boldsymbol{x}|} \mathbb{I}[(y_{i-1} = y') \wedge (y_i = y'')] \qquad w_{\boldsymbol{\tau}(y', y'')} = \log p_{\boldsymbol{\tau}}(y'' \mid y')$$

$$f_{\boldsymbol{\eta}(y', x')}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{|\boldsymbol{x}|} \mathbb{I}[(y_i = y') \wedge (x_i = x')] \qquad w_{\boldsymbol{\eta}(y', x')} = \log p_{\boldsymbol{\eta}}(x' \mid y')$$

Linear Sequence Models

- so, an HMM is:
  - a linear sequence model
  - with particular features on label transitions and label-observation emissions
  - and uses maximum likelihood estimation (count & normalize) for learning
- but we could use any feature functions we like, and use any of our loss functions for learning!

(Chain) Conditional Random Fields

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

John Lafferty[†*] LAFFERTY@CS.CMU.EDU
Andrew McCallum[*†] MCCALLUM@WHIZBANG.COM
Fernando Pereira[*‡] FPEREIRA@WHIZBANG.COM
[*]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

Abstract

We present *conditional random fields*, a framework for building probabilistic models to segment and label sequence data. Conditional random fields offer several advantages over hidden Markov models and stochastic grammars for such tasks, including the ability to relax strong independence assumptions made in those models. Conditional random fields also avoid a fundamental limitation of maximum entropy Markov models (MEMMs) and other discrimi-

mize the joint likelihood of training examples. To define a joint probability over observation and label sequences, a generative model needs to enumerate all possible observation sequences, typically requiring a representation in which observations are task-appropriate atomic entities, such as words or nucleotides. In particular, it is not practical to represent multiple interacting features or long-range dependencies of the observations, since the inference problem for such models is intractable.

This difficulty is one of the main motivations for looking at conditional models as an alternative. A conditional model

(Chain) Conditional Random Fields

$$\text{classify}(\boldsymbol{x}, \mathbf{w}) = \underset{\boldsymbol{y}}{\text{argmax}} \ \text{score}(\boldsymbol{x}, \boldsymbol{y}, \mathbf{w})$$

$$\text{score}(\boldsymbol{x}, \boldsymbol{y}, \mathbf{w}) = \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) = \sum_i w_i f_i(\boldsymbol{x}, \boldsymbol{y})$$

- linear sequence model
- arbitrary features of input are permitted
- test-time inference uses Viterbi Algorithm
- learning done by minimizing log loss (DP algorithms used to compute gradients)

Max-Margin Markov Networks

Max-Margin Markov Networks

Ben Taskar **Carlos Guestrin** **Daphne Koller**
{*btaskar,guestrin,koller*}*@cs.stanford.edu*
Stanford University

Abstract

In typical classification tasks, we seek a function which assigns a label to a single object. Kernel-based approaches, such as support vector machines (SVMs), which maximize the margin of confidence of the classifier, are the method of choice for many such tasks. Their popularity stems both from the ability to use high-dimensional feature spaces, and from their strong theoretical guarantees. However, many real-world tasks involve sequential, spatial, or structured data, where multiple labels must be assigned. Existing kernel-based methods ignore structure in the problem, assigning labels independently to each object, losing much useful information. Conversely, probabilistic graphical models, such as Markov networks, can represent correlations between labels, by exploiting problem structure, but cannot handle high-dimensional feature spaces, and lack strong theoretical generalization guarantees. In this paper, we present a new framework that combines the advantages of both approaches: *Maximum margin Markov (M^3) networks* incorporate both kernels, which efficiently deal with high-dimensional features, and the ability to capture correlations in structured data. We present an efficient algorithm for learning M^3 networks based on a

Maximum-Margin Markov Networks

$$\text{classify}(\boldsymbol{x}, \mathbf{w}) = \underset{\boldsymbol{y}}{\text{argmax}} \;\; \text{score}(\boldsymbol{x}, \boldsymbol{y}, \mathbf{w})$$

$$\text{score}(\boldsymbol{x}, \boldsymbol{y}, \mathbf{w}) = \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) = \sum_i w_i f_i(\boldsymbol{x}, \boldsymbol{y})$$

- linear sequence model
- arbitrary features of input are permitted
- test-time inference uses Viterbi Algorithm
- learning done by minimizing **hinge** loss (DP algorithm used to compute **subgradients**)

Feature Locality

- features can be arbitrarily big in terms of the input sequence, but not output sequence
- the features in HMMs are small in both the input and output sequences (only two pieces at a time)
- features in (chain) CRFs and max-margin Markov networks:
  - small in output sequence (<= 2 labels at a time)
  - but can be large in input sequence

- why do HMMs use such small features of the input sequence?
- what benefit does this give us?
  - HMM parameters can be estimated with closed-form via MLE (count & normalize)
  - for CRFs/M3Ns, we need to do inference during training