Natural Language Processing
Lecture 05 Sequence Labeling; POS Tagging; HMM; MEMM; CRF

Qun Liu, Valentin Malykh
Huawei Noah's Ark Lab

Spring 2020
A course delivered at MIPT, Moscow

Content

1. Sequence labeling problems
2. Word window classification
3. Hidden Markov models (HMMs)
4. Graphical models for sequence labeling

Content

1 Sequence labeling problems

2 Word window classification

3 Hidden Markov models (HMMs)

4 Graphical models for sequence labeling

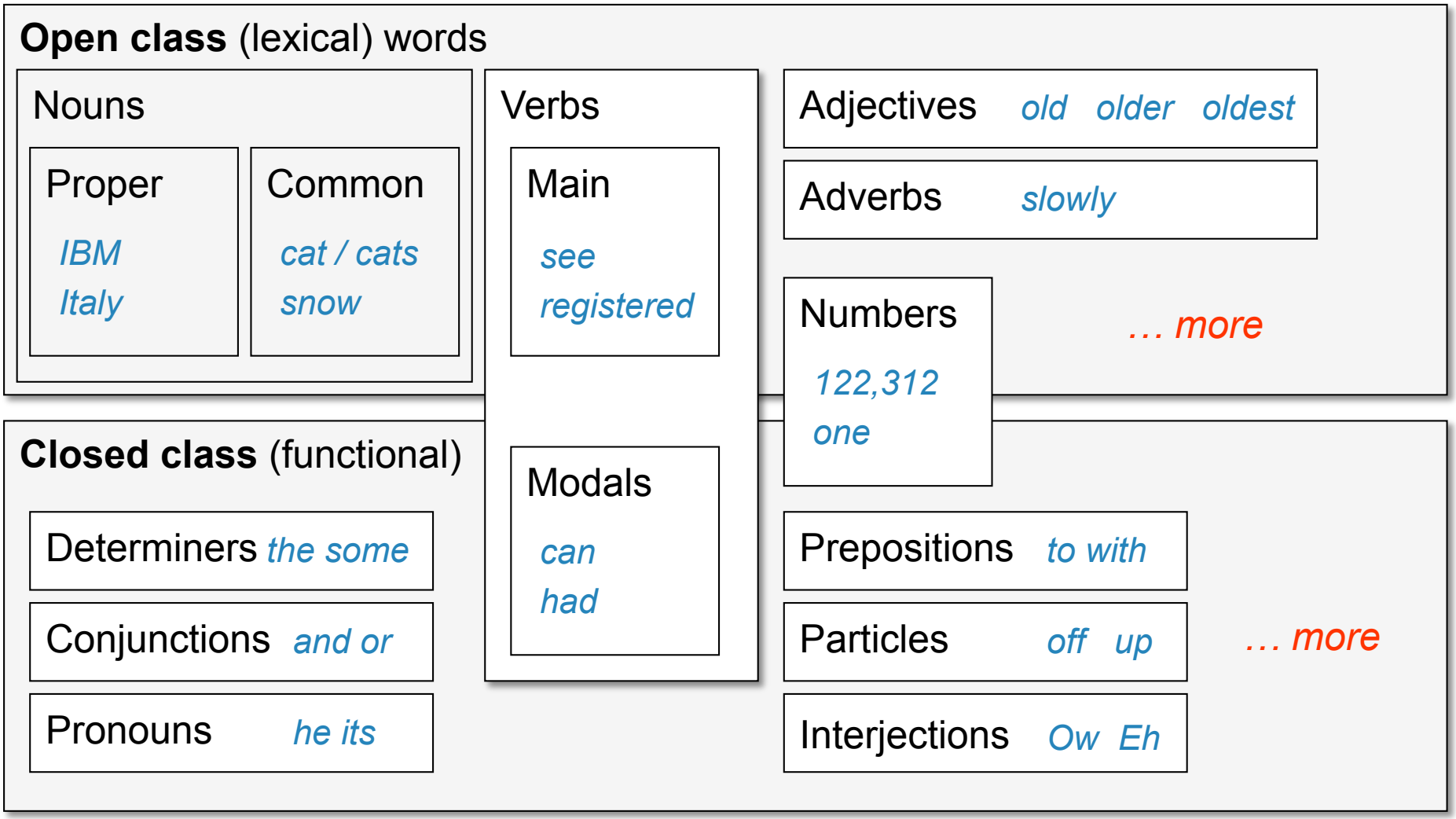
Content

1. Sequence labeling problems
   - Part-of-speech tagging
   - Named entity recognition
   - Other sequence labeling tasks

Parts of Speech

- Perhaps starting with Aristotle in the West (384–322 BCE) the idea of having parts of speech
  - lexical categories, word classes, “tags”, POS
- Dionysius Thrax of Alexandria (c. 100 BCE): 8 parts of speech
  - Still with us! But his 8 aren’t exactly the ones we are taught today
    - *Thrax*: noun, verb, article, adverb, preposition, conjunction, participle, pronoun
    - *School grammar*: noun, verb, adjective, adverb, preposition, conjunction, pronoun, interjection

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)



Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

Open vs. Closed classes

- Open vs. Closed classes
  - Closed:
    - determiners: ***a, an, the***
    - pronouns: ***she, he, I***
    - prepositions: ***on, under, over, near, by, …***
    - Why "closed"?
  - Open:
    - Nouns, Verbs, Adjectives, Adverbs.

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

POS Tagging

- Words often have more than one POS: *back*
  - The *back* door = JJ
  - On my *back* = NN
  - Win the voters *back* = RB
  - Promised to *back* the bill = VB
- The POS tagging problem is to determine the POS tag for a particular instance of a word.

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

POS Tagging

- Input: Plays well with others
- Ambiguity: NNS/VBZ UH/JJ/NN/RB IN NNS
- Output: Plays/VBZ well/RB with/IN others/NNS
- Uses:
  - MT: reordering of adjectives and nouns (say from Spanish to English)
  - Text-to-speech (how do we pronounce “lead”?)
  - Can write regexps like (Det) Adj* N+ over the output for phrases, etc.
  - Input to a syntactic parser

Penn Treebank POS tags

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

The Penn TreeBank Tagset

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*			

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

Penn Treebank tags

The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.

There/EX are/VBP 70/CD children/NNS **there/RB**

Preliminary/JJ findings/NNS were/VBD **reported/VBN** in/IN today/NN **'s/POS** New/NNP England/NNP Journal/NNP of/IN Medicine/NNP ./.

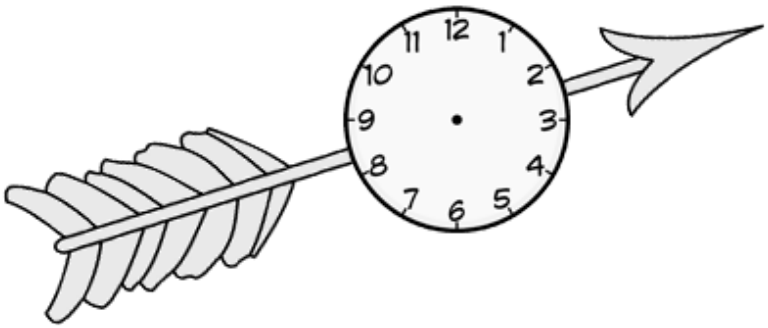
8

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

POS tagging is important for NLU

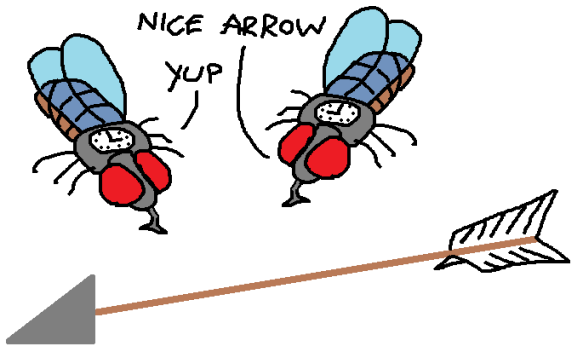
An example: Time flies like an arrow.

When POS('flies') = VB (verb):



http://myenglishguide.com/joke-time-flies-like-an-arrow-fruit-flies-like-a-banana/

When POS('flies') = NN (noun):



https://www.reddit.com/r/funny/comments/1b35wi/time_flies_like_an_arrow_fruit_flies_like_a_banana/

POS tagging performance

- How many tags are correct? (Tag accuracy)
  - About 97% currently
  - But baseline is already 90%
    - Baseline is performance of stupidest possible method
      - Tag every word with its most frequent tag
      - Tag unknown words as nouns
  - Partly easy because
    - Many words are unambiguous
    - You get points for them (*the*, *a*, etc.) and for punctuation marks!

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

Deciding on the correct part of speech can be difficult even for people

- Mrs/NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG
- All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN
- Chateau/NNP Petrus/NNP costs/VBZ around/RB 250/CD

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

How difficult is POS tagging?

- About 11% of the word types in the Brown corpus are ambiguous with regard to part of speech
- But they tend to be very common words. E.g., *that*
  - I know *that* he is honest = IN
  - Yes, *that* play was nice = DT
  - You can't go *that* far = RB
- 40% of the word tokens are ambiguous

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

Sources of information

- What are the main sources of information for POS tagging?
  - Knowledge of neighboring words
    - Bill saw that man yesterday
    - NNP NN DT NN NN
    - VB VB(D) IN VB NN
  - Knowledge of word probabilities
    - *man* is rarely used as a verb….
- The latter proves the most useful, but the former also helps

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

More and Better Features → Feature-based tagger

- Can do surprisingly well just looking at a word by itself:
  - Word: the: the → DT
  - Lowercased word: Importantly: importantly → RB
  - Prefixes: unfathomable: un- → JJ
  - Suffixes: Importantly: -ly → RB
  - Capitalization: Meridian: CAP → NNP
  - Word shapes: 35-year: d-x → JJ
- Then build a classifier to predict tag
  - Maxent P(t|w): 93.7% overall / 82.6% unknown

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

Overview: POS Tagging Accuracies

- Rough accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - Maxent P(t|w): 93.7% / 82.6%
  - TnT (HMM++): 96.2% / 86.0%
  - MEMM tagger: 96.9% / 86.9%
  - Bidirectional dependencies: 97.2% / 90.0%
  - Upper bound: ~98% (human agreement)

Most errors on unknown words

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

POS tagging as a sequence classification task

- We are given a sentence (an "observation" or "sequence of observations")
  - Secretariat is expected to race tomorrow
  - She promised to back the bill
- What is the best sequence of tags which corresponds to this sequence of observations?
- Probabilistic view:
  - Consider all possible sequences of tags
  - Out of this universe of sequences, choose the tag sequence which is most probable given the observation sequence of n words w1…wn.

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

Content

1. Sequence labeling problems
   - Part-of-speech tagging
   - Named entity recognition
   - Other sequence labeling tasks

Christopher Manning

Named Entity Recognition (NER)

- A very important sub-task: find and classify names in text, for example:

  - The decision by the independent MP Andrew Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.

Christopher Manning, Information Extraction and Named Entity Recognition (slides)

Christopher Manning

Named Entity Recognition (NER)

- A very important sub-task: find and classify names in text, for example:
  - The decision by the independent MP Andrew Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.

Christopher Manning, Information Extraction and Named Entity Recognition (slides)

Christopher Manning

Named Entity Recognition (NER)

- A very important sub-task: find and classify names in text, for example:
  - The decision by the independent MP Andrew Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.

Person
Date
Location
Organi-
zation

Christopher Manning, Information Extraction and Named Entity Recognition (slides)

Named Entity Recognition and Classification

<PER>**Prof. Jerry Hobbs**</PER> taught CS544 during <DATE>**February 2010**</DATE>.
<PER>**Jerry Hobbs**</PER> killed his daughter in <LOC>**Ohio**</LOC>.
<ORG>**Hobbs corporation**</ORG> bought <ORG>**FbK**</ORG>.

- Identify mentions in text and classify them into a predefined set of categories of interest:
  - Person Names: **Prof. Jerry Hobbs**, **Jerry Hobbs**
  - Organizations: **Hobbs corporation**, **FbK**
  - Locations: **Ohio**
  - Date and time expressions: **February 2010**
  - E-mail: **mkg@gmail.com**
  - Web address: **www.usc.edu**
  - Names of drugs: **paracetamol**
  - Names of ships: **Queen Marry**
  - Bibliographic references:
  - ...

Zornitsa Kozareva, USC/ISI CS544: Named Entity Recognition and Classification (slides)

Christopher Manning

Named Entity Recognition (NER)

- The uses:
  - Named entities can be indexed, linked off, etc.
  - Sentiment can be attributed to companies or products
  - A lot of IE relations are associations between named entities
  - For question answering, answers are often named entities.
- Concretely:
  - Many web pages tag various entities, with links to bio or topic pages, etc.
    - Reuters' OpenCalais, Evri, AlchemyAPI, Yahoo's Term Extraction, …
  - Apple/Google/Microsoft/… smart recognizers for document content

Christopher Manning, Information Extraction and Named Entity Recognition (slides)

Christopher Manning

The Named Entity Recognition Task

Task: Predict entities in a text

Foreign	ORG
Ministry	ORG
spokesman	O
Shen	PER
Guofang	PER
told	O
Reuters	ORG
:	:

} Standard evaluation is per entity, *not* per token

Christopher Manning, Information Extraction and Named Entity Recognition (slides)

Christopher Manning

Three standard approaches to NER (and IE)

1. Hand-written regular expressions
   - Perhaps stacked
2. Using classifiers
   - Generative: Naïve Bayes
   - Discriminative: Maxent models
3. Sequence models
   - HMMs
   - CMMs/MEMMs
   - CRFs

Christopher Manning, Information Extraction and Named Entity Recognition (slides)

Rule Based NER (1)

- **Create regular expressions to extract:**
  - Telephone number
  - E-mail
  - Capitalized names

Zornitsa Kozareva, USC/ISI CS544: Named Entity Recognition and Classification (slides)

Rule Based NER (1)

- Regular expressions provide a flexible way to match strings of text, such as particular characters, words, or patterns of characters

Suppose you are looking for a word that:

1. starts with a capital letter “P”
2. is the first word on a line
3. the second letter is a lower case letter
4. is exactly three letters long
5. the third letter is a vowel

the regular expression would be “^P[a-z][aeiou]” where

^ - indicates the beginning of the string
[a-z] – any letter in range a to z
[aeiou] – any vowel

Zornitsa Kozareva, USC/ISI CS544: Named Entity Recognition and Classification (slides)

Perl RegEx

- \w (word char) any alpha-numeric
- \d (digit char) any digit
- \s (space char) any whitespace
- . (wildcard) anything
- \b word bounday
- ^ beginning of string
- $ end of string
- ? For 0 or 1 occurrences
- + for 1 or more occurrences
- specific range of number of occurrences: {min,max}.
  - A{1,5} One to five A's.
  - A{5,} Five or more A's
  - A{5} Exactly five A's

Zornitsa Kozareva, USC/ISI CS544: Named Entity Recognition and Classification (slides)

Rule Based NER (1)

- **Create regular expressions to extract:**
  - Telephone number
  - E-mail
  - Capitalized names

blocks of digits separated by hyphens

RegEx = (\d+\-)+\d+

Zornitsa Kozareva, USC/ISI CS544: Named Entity Recognition and Classification (slides)

Rule Based NER (1)

- **Create regular expressions to extract:**
  - Telephone number
  - E-mail
  - Capitalized names

blocks of digits separated by hyphens

RegEx = (\d+\-)+\d+

- matches valid phone numbers like 900-865-1125 and 725-1234
- incorrectly extracts social security numbers 123-45-6789
- fails to identify numbers like 800.865.1125 and (800)865-CARE

Improved RegEx = (\d{3}[-.\ ()]){1,2}[\dA-Z]{4}

Zornitsa Kozareva, USC/ISI CS544: Named Entity Recognition and Classification (slides)

Rule Based NER (2)

- **Create rules to extract locations**
  - Capitalized word + {city, center, river} indicates location

    Ex. *New York city*

    Hudson river

  - Capitalized word + {street, boulevard, avenue} indicates location

    Ex. *Fifth avenue*

Zornitsa Kozareva, USC/ISI CS544: Named Entity Recognition and Classification (slides)

Rule Based NER (3)

- **Use context patterns**
  - [*PERSON*] earned [*MONEY*]
    Ex. *Frank earned $20*
  - [*PERSON*] joined [*ORGANIZATION*]
    Ex. *Sam joined IBM*
  - [*PERSON*],[*JOBTITLE*]
    Ex. *Mary, the teacher*

Zornitsa Kozareva, USC/ISI CS544: Named Entity Recognition and Classification (slides)

Rule Based NER (3)

- **Use context patterns**

  still not so simple:

  – [*PERSON*|*ORGANIZATION*|*ANIMAL*] fly to [*LOCATION*|*PERSON*|*EVENT*]

  Ex. *Jerry flew to Japan*

  Sarah flies to the party

  Delta flies to Europe

  bird flies to trees

  bee flies to the wood

Zornitsa Kozareva, USC/ISI CS544: Named Entity Recognition and Classification (slides)

Why simple things would not work?

- Capitalization is a strong indicator for capturing proper names, but it can be tricky:
  - first word of a sentence is capitalized
  - sometimes titles in web pages are all capitalized
  - nested named entities contain non-capital words

    University of Southern California is Organization
  - all nouns in German are capitalized

Zornitsa Kozareva, USC/ISI CS544: Named Entity Recognition and Classification (slides)

Why simple things would not work?

- We already have discussed that currently no gazetteer contains all existing proper names.
- New proper names constantly emerge

  movie titles

  books

  singers

  restaurants

  etc.

Zornitsa Kozareva, USC/ISI CS544: Named Entity Recognition and Classification (slides)

Christopher Manning

The ML sequence model approach to NER

Training

1. Collect a set of representative training documents
2. Label each token for its entity class or other (O)
3. Design feature extractors appropriate to the text and classes
4. Train a sequence classifier to predict the labels from the data

Testing

1. Receive a set of testing documents
2. Run sequence model inference to label each token
3. Appropriately output the recognized entities

Christopher Manning, Information Extraction and Named Entity Recognition (slides)

Christopher Manning

Encoding classes for sequence labeling

	IO encoding	IOB encoding
Fred	PER	B-PER
showed	O	O
Sue	PER	B-PER
Mengqiu	PER	B-PER
Huang	PER	I-PER
‘s	O	O
new	O	O
painting	O	O

Christopher Manning, Information Extraction and Named Entity Recognition (slides)

Content

1. Sequence labeling problems
   - Part-of-speech tagging
   - Named entity recognition
   - Other sequence labeling tasks

Many NLP tasks are sequence labeling tasks

Input: a sequence of tokens/words:

```
Pierre Vinken , 61 years old , will join IBM 's board
as a nonexecutive director Nov. 29 .
```

Output: a sequence of **labeled** tokens/words:

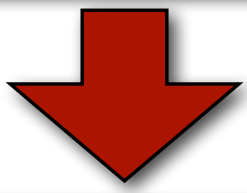
POS-tagging:
```
Pierre_NNP Vinken_NNP ,_, 61_CD years_NNS
old_JJ ,_, will_MD join_VB IBM_NNP 's_POS board_NN
as_IN a_DT nonexecutive_JJ director_NN Nov._NNP
29_CD ._.
```

Named Entity Recognition:
```
Pierre_B-PERS Vinken_I-PERS ,_O
61_O years_O old_O ,_O will_O join_O IBM_B-ORG 's_O
board_O as_O a_O nonexecutive_O director_O Nov._B-DATE
29_I-DATE ._O
```

Julia Hockenmaier, UIUC CS447: Natural Language Processing (slides)

POS tagging

```
Pierre Vinken , 61 years old , will join IBM 's board
as a nonexecutive director Nov. 29 .
```



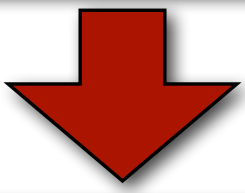
```
Pierre_NNP Vinken_NNP ,_, 61_CD years_NNS old_JJ ,_,
will_MD join_VB IBM_NNP 's_POS board_NN as_IN a_DT
nonexecutive_JJ director_NN Nov._NNP 29_CD ._.
```

Task: assign POS tags to words

Julia Hockenmaier, UIUC CS447: Natural Language Processing (slides)

Noun phrase (NP) chunking

```
Pierre Vinken , 61 years old , will join IBM 's board
as a nonexecutive director Nov. 29 .
```



```
[NP Pierre Vinken] , [NP 61 years] old , will join
[NP IBM] 's [NP board] as [NP a nonexecutive director]
[NP Nov. 2] .
```

Task: identify all non-recursive NP chunks

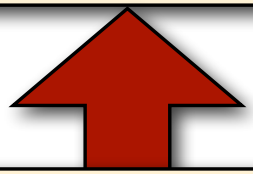
Julia Hockenmaier, UIUC CS447: Natural Language Processing (slides)

The BIO encoding

We define three new tags:

- **B-NP**: beginning of a noun phrase chunk
- **I-NP**: inside of a noun phrase chunk
- **O**: outside of a noun phrase chunk

```
[NP Pierre Vinken] , [NP 61 years] old , will join
[NP IBM] 's [NP board] as [NP a nonexecutive director]
[NP Nov. 2] .
```

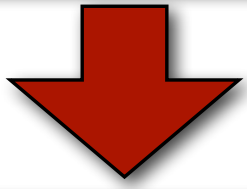


```
Pierre_B-NP Vinken_I-NP ,_O 61_B-NP years_I-NP
old_O ,_O will_O join_O IBM_B-NP 's_O board_B-NP as_O
a_B-NP nonexecutive_I-NP director_I-NP Nov._B-NP
29_I-NP ._O
```

Julia Hockenmaier, UIUC CS447: Natural Language Processing (slides)

Shallow parsing

```
Pierre Vinken , 61 years old , will join IBM 's board
as a nonexecutive director Nov. 29 .
```



```
[NP Pierre Vinken] , [NP 61 years] old , [VP will join]
[NP IBM] 's [NP board] [PP as] [NP a nonexecutive
director] [NP Nov. 2] .
```

Task: identify all non-recursive NP, verb ("VP") and preposition ("PP") chunks

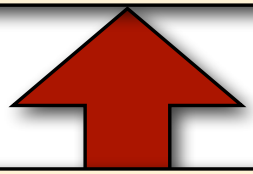
Julia Hockenmaier, UIUC CS447: Natural Language Processing (slides)

The BIO encoding for shallow parsing

We define several new tags:

- **B-NP B-VP B-PP**: beginning of an NP, "VP", "PP" chunk
- **I-NP I-VP I-PP**: inside of an NP, "VP", "PP" chunk
- **O**: outside of any chunk

```
[NP Pierre Vinken] , [NP 61 years] old , [VP will join]
[NP IBM] 's [NP board] [PP as] [NP a nonexecutive
director] [NP Nov. 2] .
```

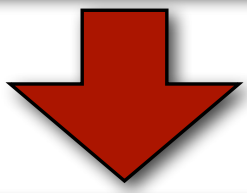


```
Pierre_B-NP Vinken_I-NP ,_O 61_B-NP years_I-NP
old_O ,_O will_B-VP join_I-VP IBM_B-NP 's_O board_B-NP
as_B-PP a_B-NP nonexecutive_I-NP director_I-NP Nov._B-
NP 29_I-NP ._O
```

Julia Hockenmaier, UIUC CS447: Natural Language Processing (slides)

Named Entity Recognition

```
Pierre Vinken , 61 years old , will join IBM 's board
as a nonexecutive director Nov. 29 .
```



```
[PERS Pierre Vinken] , 61 years old , will join
[ORG IBM] 's board as a nonexecutive director
[DATE Nov. 2] .
```

Task: identify all mentions of named entities (people, organizations, locations, dates)

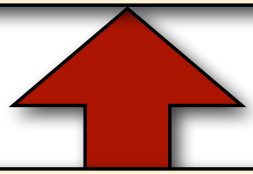
Julia Hockenmaier, UIUC CS447: Natural Language Processing (slides)

The BIO encoding for NER

We define many new tags:

- **B-PERS**, **B-DATE, ...:** beginning of a mention of a person/date...
- **I-PERS**, **I-DATE, ...:** inside of a mention of a person/date...
- **O**: outside of any mention of a named entity

```
[PERS Pierre Vinken] , 61 years old , will join
[ORG IBM] 's board as a nonexecutive director
[DATE Nov. 2] .
```



```
Pierre_B-PERS Vinken_I-PERS ,_O 61_O years_O old_O ,_O
will_O join_O IBM_B-ORG 's_O board_O as_O a_O
nonexecutive_O director_O Nov._B-DATE 29_I-DATE ._O
```

Julia Hockenmaier, UIUC CS447: Natural Language Processing (slides)

Content

1 Sequence labeling problems

2 Word window classification

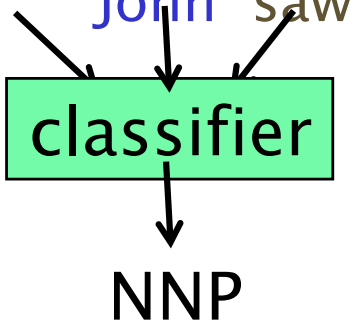
3 Hidden Markov models (HMMs)

4 Graphical models for sequence labeling

Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

John saw the saw and decided to take it to the table.



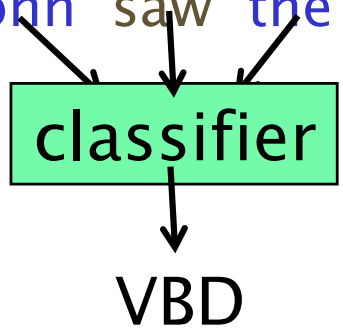
Slide from Ray Mooney

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

John saw the saw and decided to take it to the table.

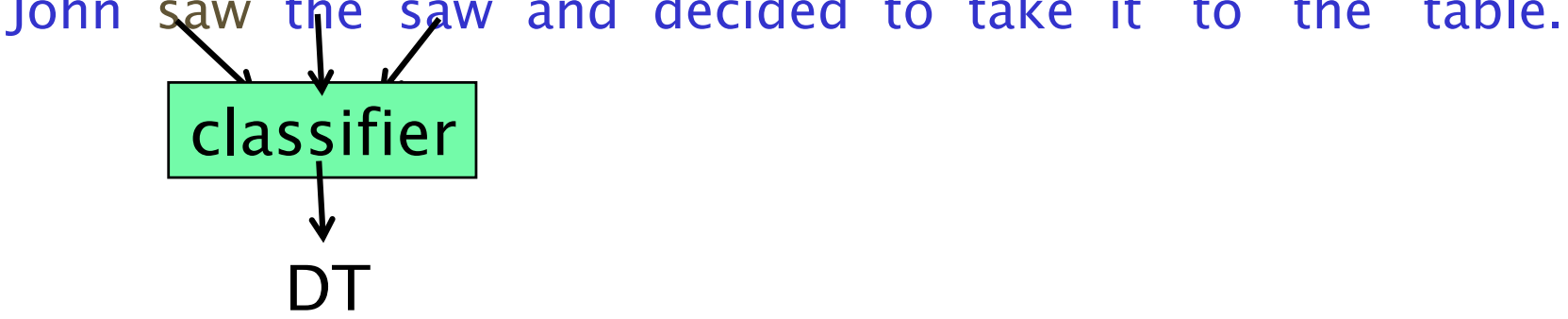


Slide from Ray Mooney

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



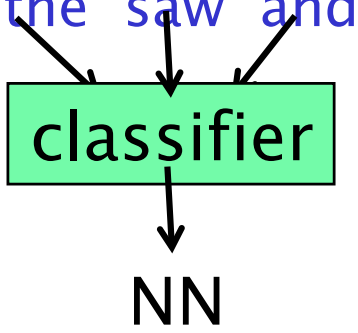
Slide from Ray Mooney

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

John saw the saw and decided to take it to the table.



Slide from Ray Mooney

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

5. Word-Window classification

- **Idea**: classify a word in its context window of neighboring words.

- For example, **Named Entity Classification** of a word in context:
  - Person, Location, Organization, None

- A simple way to classify a word in context might be to average the word vectors in a window and to classify the average vector
  - Problem: that would lose position information

26

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

Window classification: Softmax

- Train softmax classifier to classify a center word by taking concatenation of word vectors surrounding it in a window

- Example: Classify "Paris" in the context of this sentence with window length 2:

$$X_{window} = [\, x_{museums} \quad x_{in} \quad x_{Paris} \quad x_{are} \quad x_{amazing} \,]^T$$

- Resulting vector $x_{window} = x \in R^{5d}$, a column vector!

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

Simplest window classifier: Softmax

- With $x = x_{window}$ we can use the same softmax classifier as before

predicted model output probability

$$\hat{y}_y = p(y|x) = \frac{\exp(W_y.x)}{\sum_{c=1}^{C} \exp(W_c.x)}$$

- With cross entropy error as before:

same

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} -\log \left(\frac{e^{f_{y_i}}}{\sum_{c=1}^{C} e^{f_c}} \right)$$

- How do you update the word vectors?

  Short answer: Just take derivatives like last week and optimize

28

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

Slightly more complex: Multilayer Perceptron

- Introduce an additional layer in our softmax classifier with a non-linearity.
- MLPs are fundamental building blocks of more complex neural systems!
- Assume we want to classify whether the center word is a Location
- Similar to word2vec, we will go over all positions in a corpus. But this time, it will be supervised s.t. positions that are true NER Locations should assign high probability to that class, and others should assign low probability.

29

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

Neural Network Feed-forward Computation

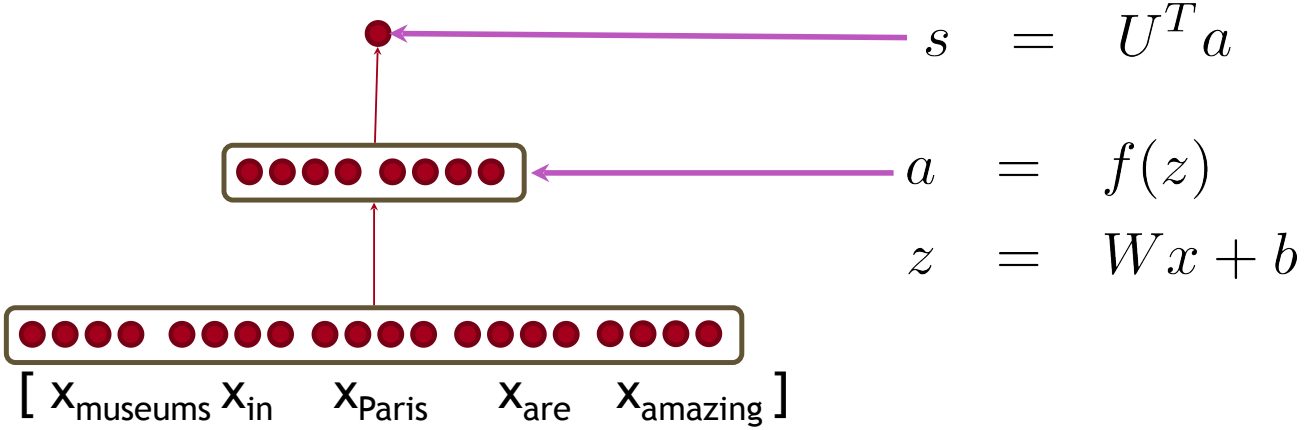
$$score(x) = U^T a \in \mathbb{R}$$

We compute a window's score with a 3-layer neural net:

- *s* = *score*("museums in Paris are amazing")

$$s = U^T f(Wx + b)$$

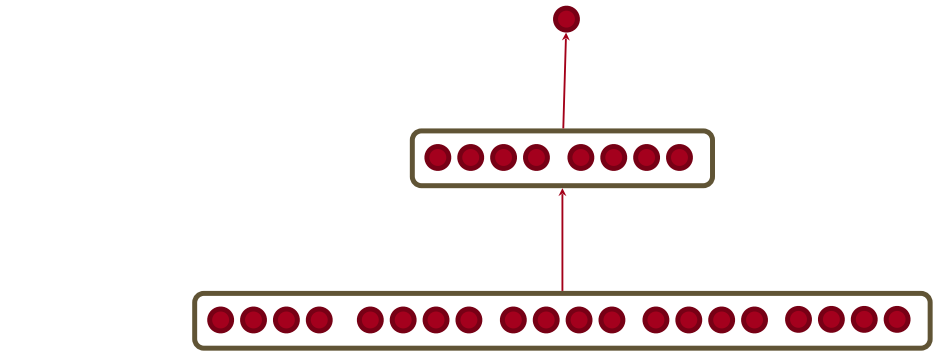
$$x \in \mathbb{R}^{20\times 1}, W \in \mathbb{R}^{8x20}, U \in \mathbb{R}^{8x2}$$



Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

Main intuition for extra layer

The middle layer learns non-linear interactions between the input word vectors.



Example: only if "*museums*" is first vector should it matter that "*in*" is in the second position

31

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

Content

1 Sequence labeling problems

2 Word window classification

3 Hidden Markov models (HMMs)

4 Graphical models for sequence labeling

Weakness of word classification

- A weakness of the word classification method for sequence labeling is that it is not capable to make use of the dependencies between POS tags in prediction.
- An extreme example:

  It is true for all that that that that that that that refers to is not the same that that that that refers to.

It	is	true	for	all	that	that	that	that	that	that	that	refers	to	is	not	the	same	that	that	that	that	refers	to	.
					pron.	conj.	det.	noun	rel.pron.	det.	noun							noun	rel.pron.	det.	noun			
							(adj.)	"that"	which	(adj.)	"that"							"that"	which	(adj.)	"that"			

  - The word windows for some of the occurrences of “that” in this sentence are the same, if the window size is not greater than 2.
  - The word window classification method will not be able to distinguish those “that”.
  - The POS tags of the previous words may be helpful!

Sequence Labeling as Classification Using Outputs as Inputs

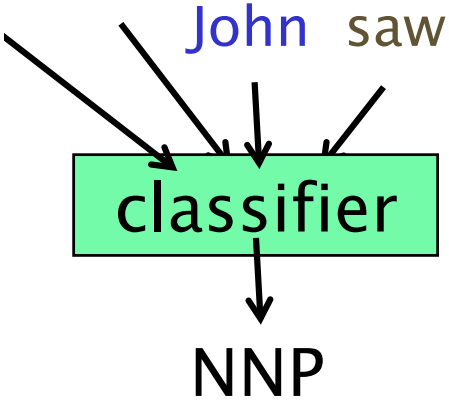
- Better input features are usually the categories of the surrounding tokens, but these are not available yet.
- Can use category of either the preceding or succeeding tokens by going forward or back and using previous output.

Slide from Ray Mooney

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

Forward Classification

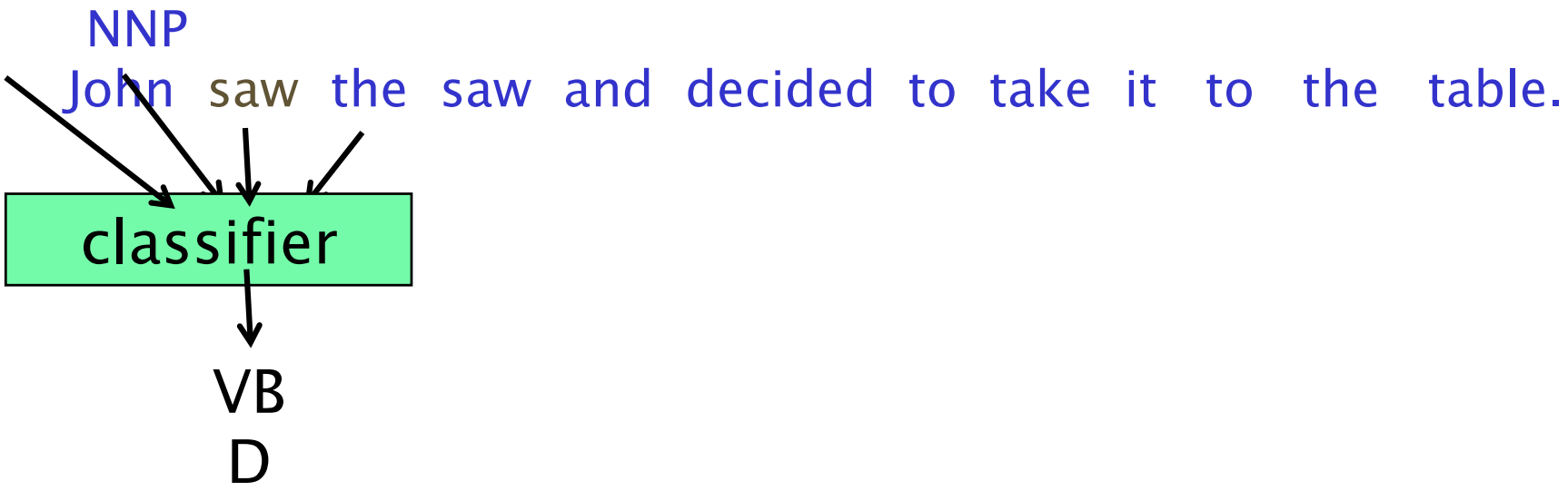
John saw the saw and decided to take it to the table.



Slide from Ray Mooney

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

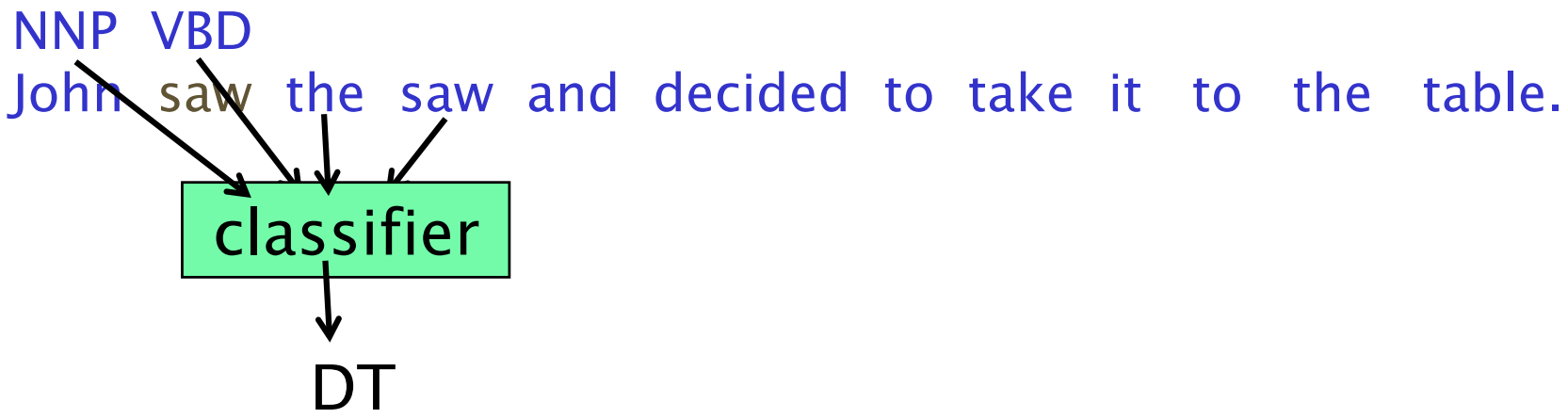
Forward Classification



Slide from Ray Mooney

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

Forward Classification

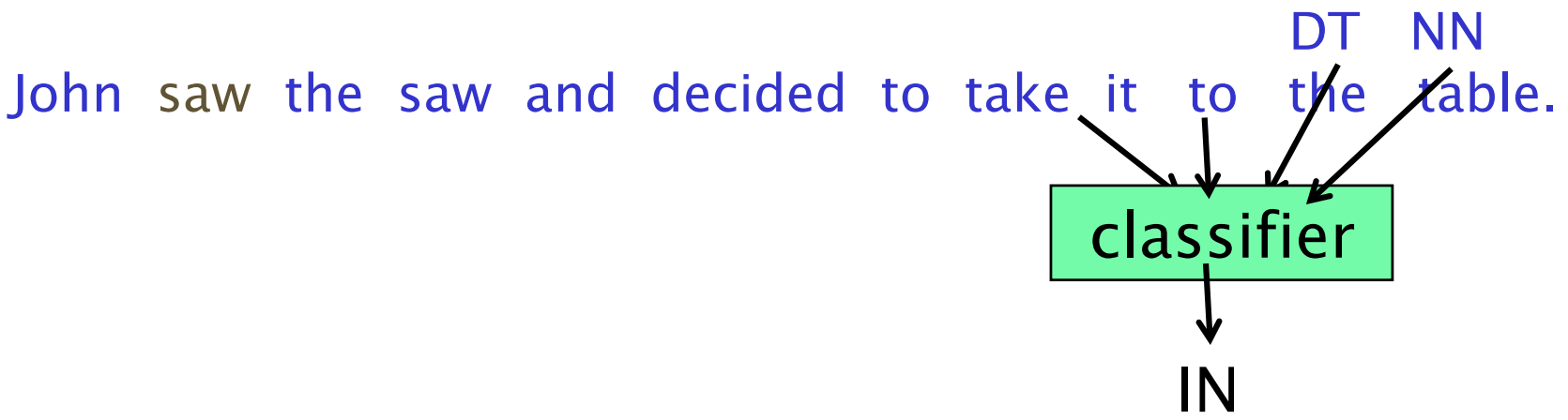


Slide from Ray Mooney

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

Backward Classification

- Disambiguating “to” in this case would be even easier backward.



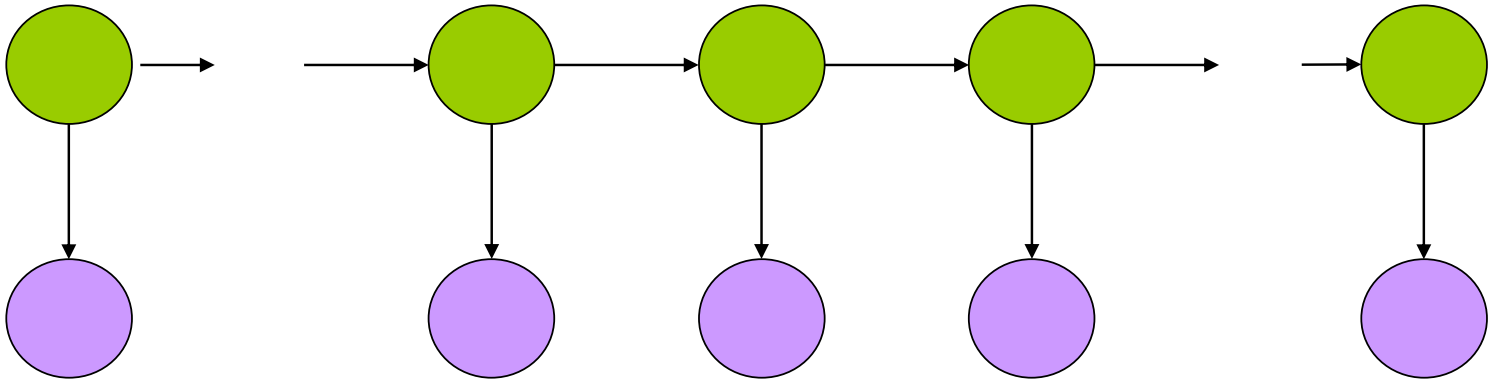
Slide from Ray Mooney

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

Content

3 Hidden Markov models (HMMs)
- Model definition
- Inference in an HMM
- Decoding
- Forward Procedure
- Backward Procedure
- Decoding Solution
- Viterbi Algorithm
- Parameter Estimation
- HMM Applications

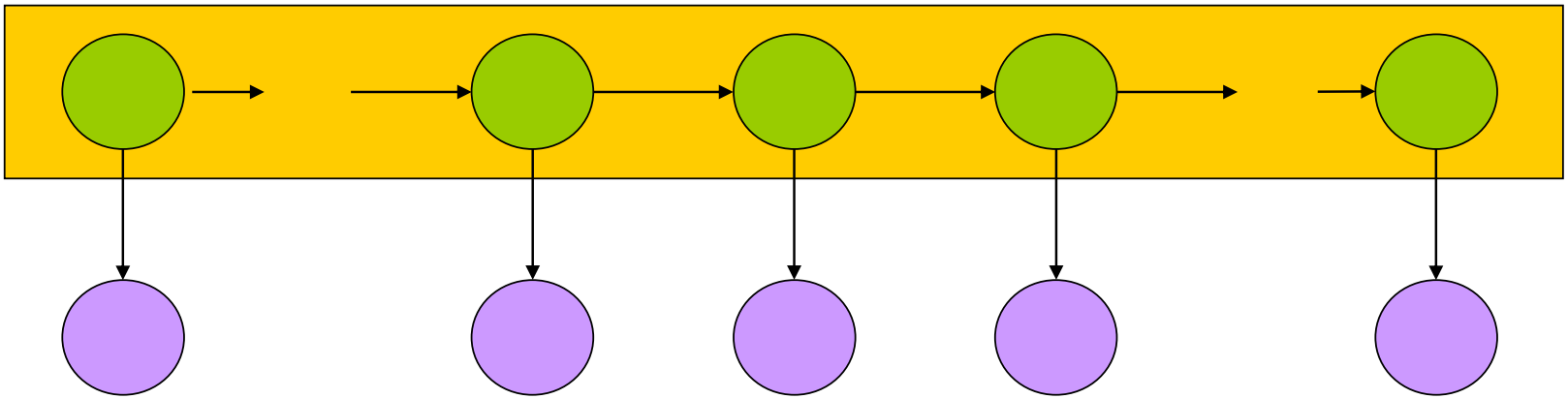
What is an HMM?



- Graphical Model
- Circles indicate states
- Arrows indicate probabilistic dependencies between states

David Meir Blei, Hidden Markov Models, 1999 (Slides)

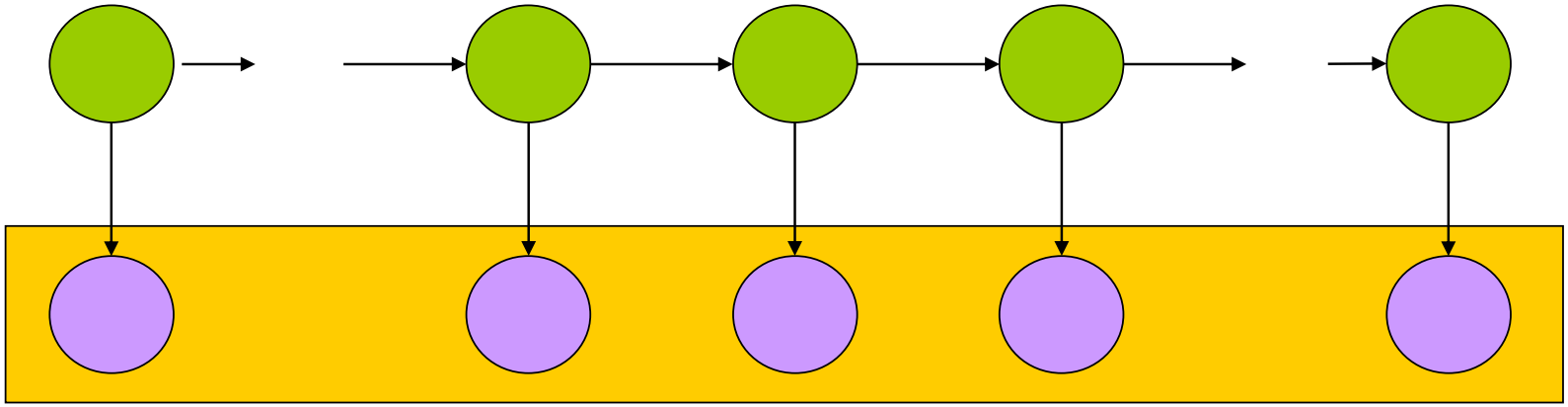
What is an HMM?



- Green circles are ***hidden states***
- Dependent only on the previous state
- "The past is independent of the future given the present."

David Meir Blei, Hidden Markov Models, 1999 (Slides)

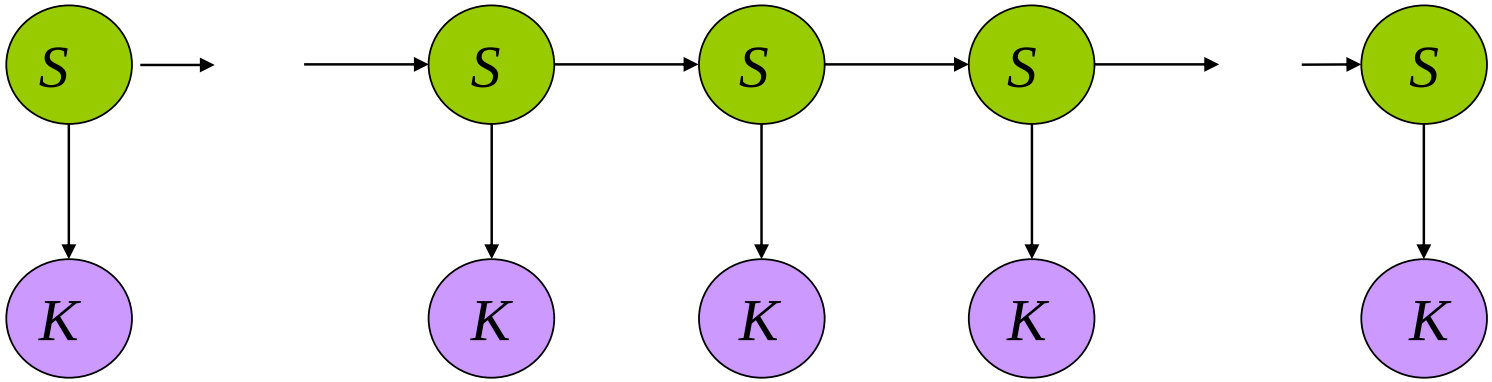
What is an HMM?



- Purple nodes are ***observed states***
- Dependent only on their corresponding hidden state

David Meir Blei, Hidden Markov Models, 1999 (Slides)

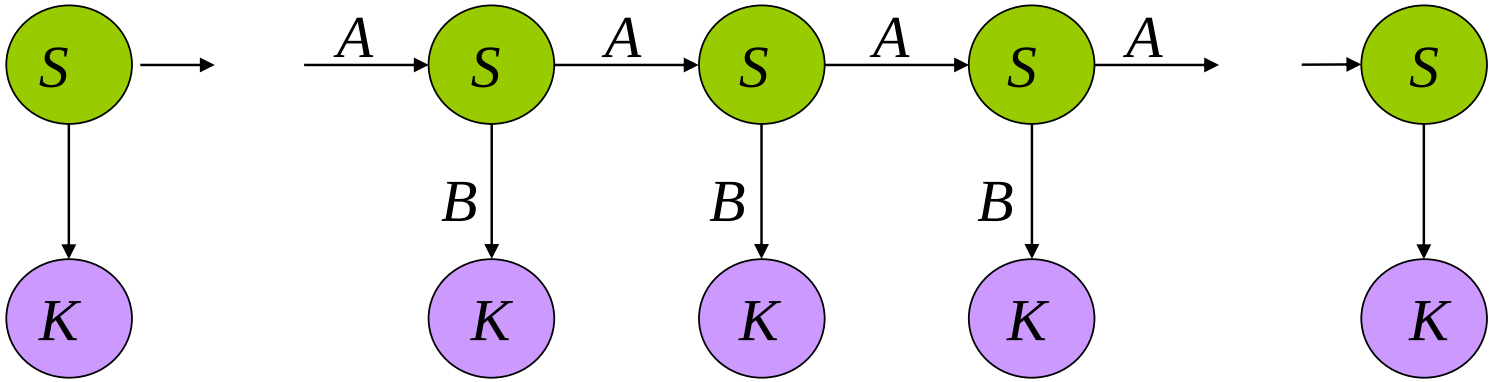
HMM Formalism



- $\{S, K, \Pi, A, B\}$
- $S : \{s_1 \ldots s_N\}$ are the values for the hidden states
- $K : \{k_1 \ldots k_M\}$ are the values for the observations

David Meir Blei, Hidden Markov Models, 1999 (Slides)

HMM Formalism

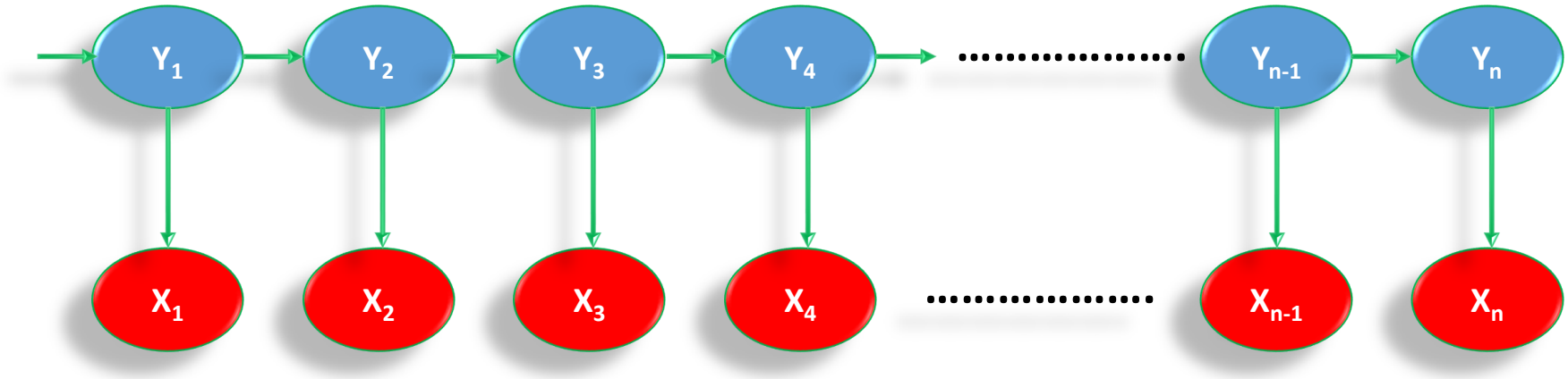


- $\{S, K, \Pi, A, B\}$
- $\Pi = \{\pi_\iota\}$ are the initial state probabilities
- $A = \{a_{ij}\}$ are the state transition probabilities
- $B = \{b_{ik}\}$ are the observation state probabilities

David Meir Blei, Hidden Markov Models, 1999 (Slides)

Trellis Diagram

- An HMM can be graphically depicted by Trellis diagram



Anantharaman Narayana Iyer, Natural Language Processing, Unit 2 – Tagging Problems and HMM (Slides)

Another example: Coin tossing

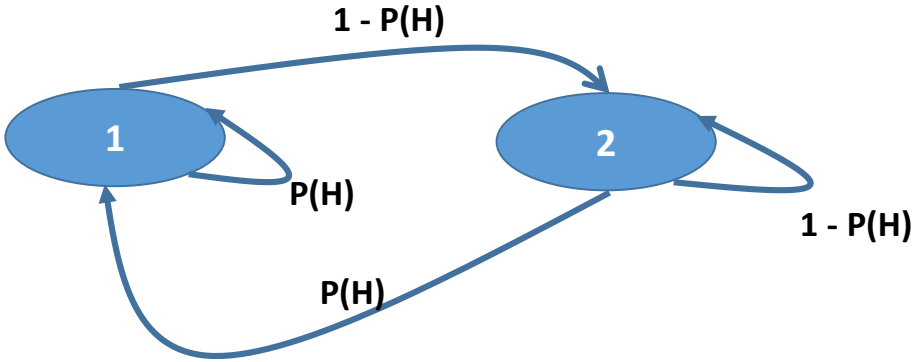


Fig (a)

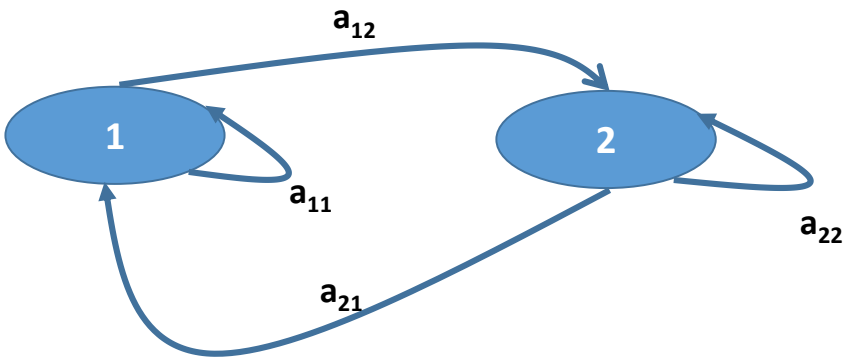


Fig (b)

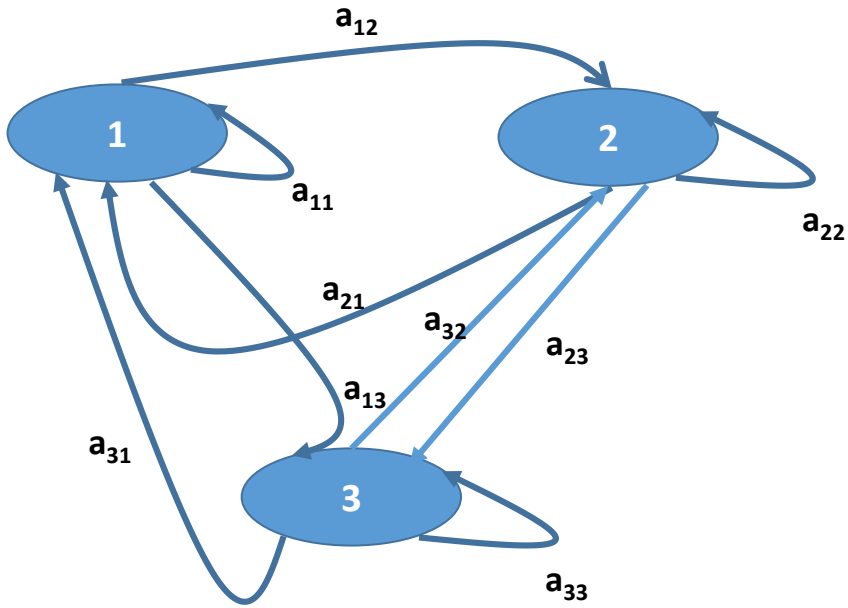


Fig (c)

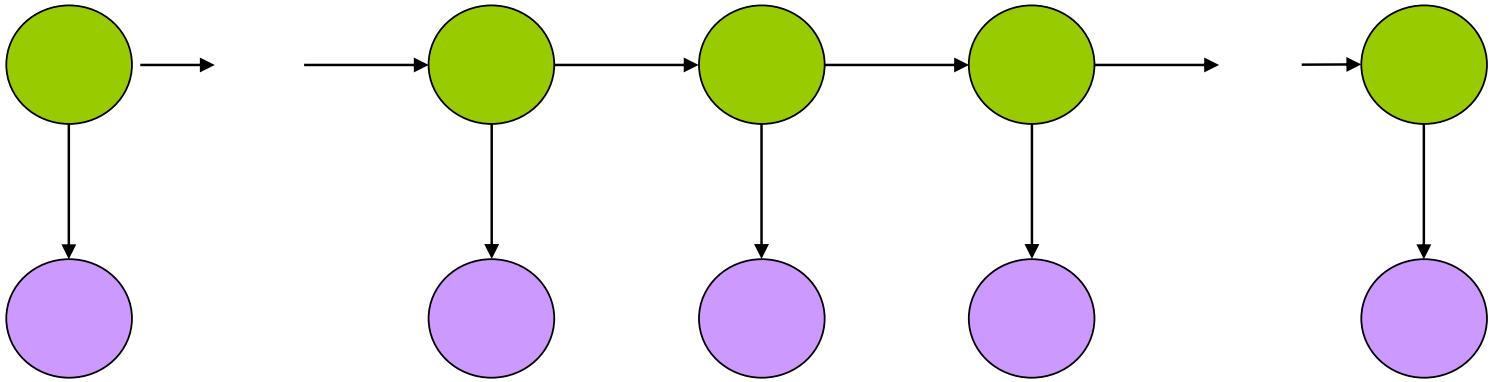
	State 1	State 2	State 3
P(H)	0.5	0.75	0.25
P(T)	0.5	0.25	0.75

Anantharaman Narayana Iyer, Natural Language Processing, Unit 2 – Tagging Problems and HMM (Slides)

Content

3 Hidden Markov models (HMMs)
- Model definition
- Inference in an HMM
- Decoding
- Forward Procedure
- Backward Procedure
- Decoding Solution
- Viterbi Algorithm
- Parameter Estimation
- HMM Applications

Inference in an HMM



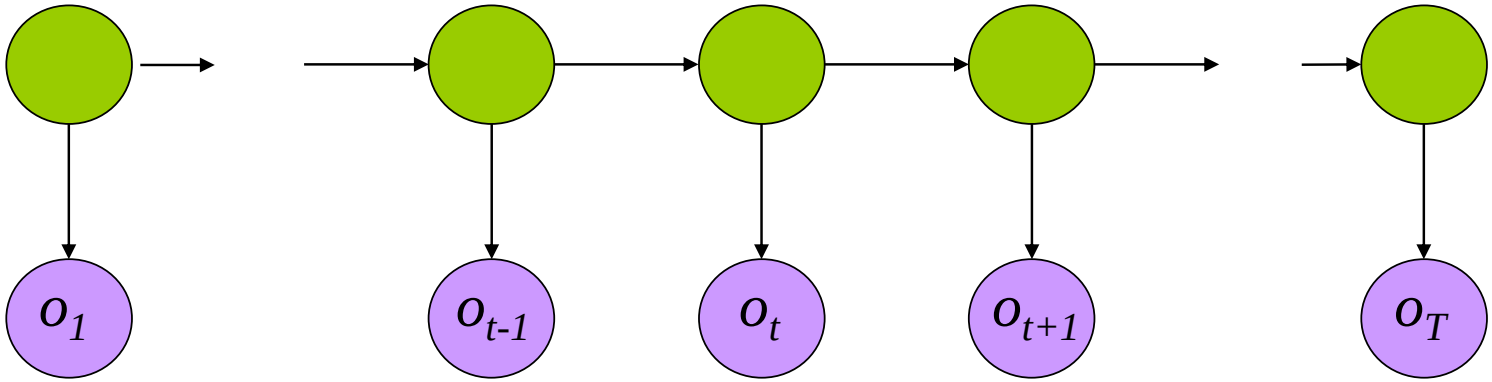
- Compute the probability of a given observation sequence
- Given an observation sequence, compute the most likely hidden state sequence
- Given an observation sequence and set of possible models, which model most closely fits the data?

David Meir Blei, Hidden Markov Models, 1999 (Slides)

Content

3 Hidden Markov models (HMMs)
- Model definition
- Inference in an HMM
- Decoding
- Forward Procedure
- Backward Procedure
- Decoding Solution
- Viterbi Algorithm
- Parameter Estimation
- HMM Applications

Decoding



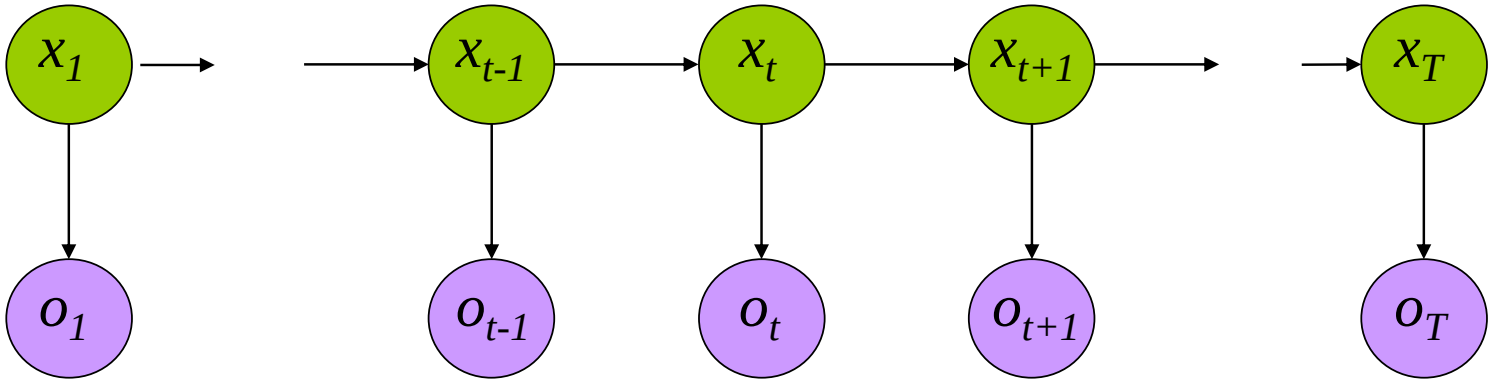
Given an observation sequence and a model, compute the probability of the observation sequence

$$O = (o_1 \ldots o_T),\ \mu = (A, B, \Pi)$$

Compute $P(O \mid \mu)$

David Meir Blei, Hidden Markov Models, 1999 (Slides)

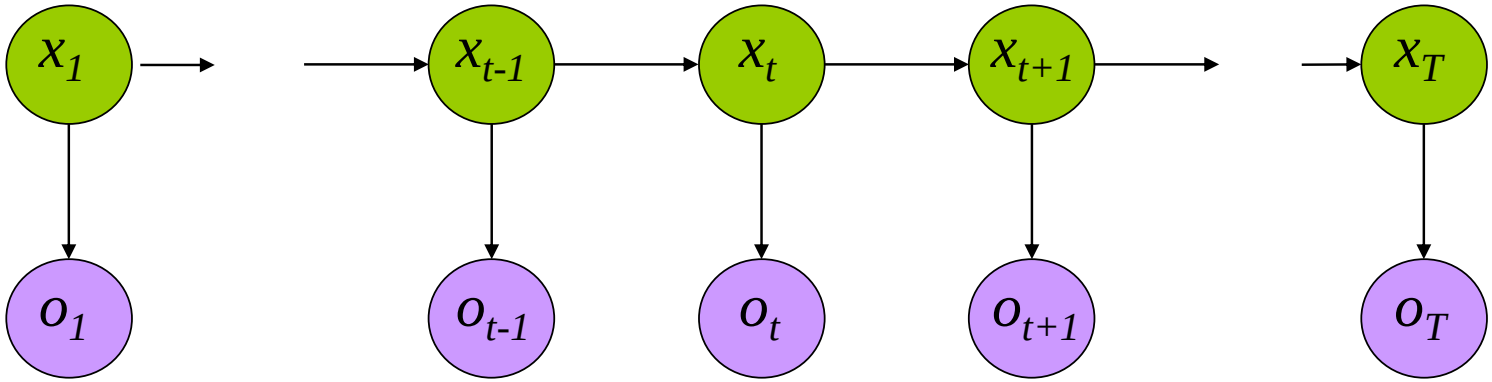
Decoding



$$P(O \mid X, \mu) = b_{x_1 o_1} b_{x_2 o_2} \ldots b_{x_T o_T}$$

David Meir Blei, Hidden Markov Models, 1999 (Slides)

Decoding

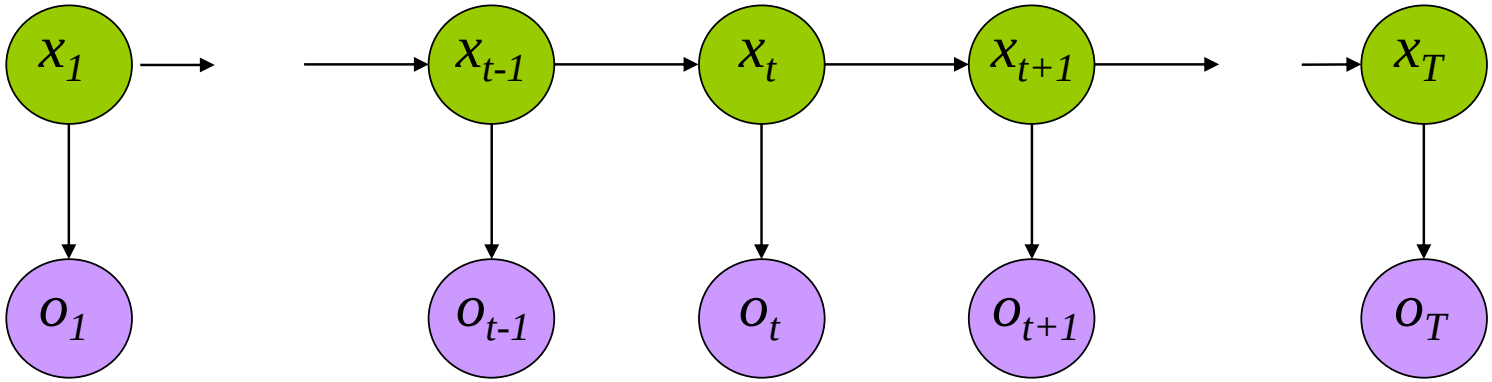


$$P(O \mid X, \mu) = b_{x_1 o_1} b_{x_2 o_2} \ldots b_{x_T o_T}$$

$$P(X \mid \mu) = \pi_{x_1} a_{x_1 x_2} a_{x_2 x_3} \ldots a_{x_{T-1} x_T}$$

David Meir Blei, Hidden Markov Models, 1999 (Slides)

Decoding



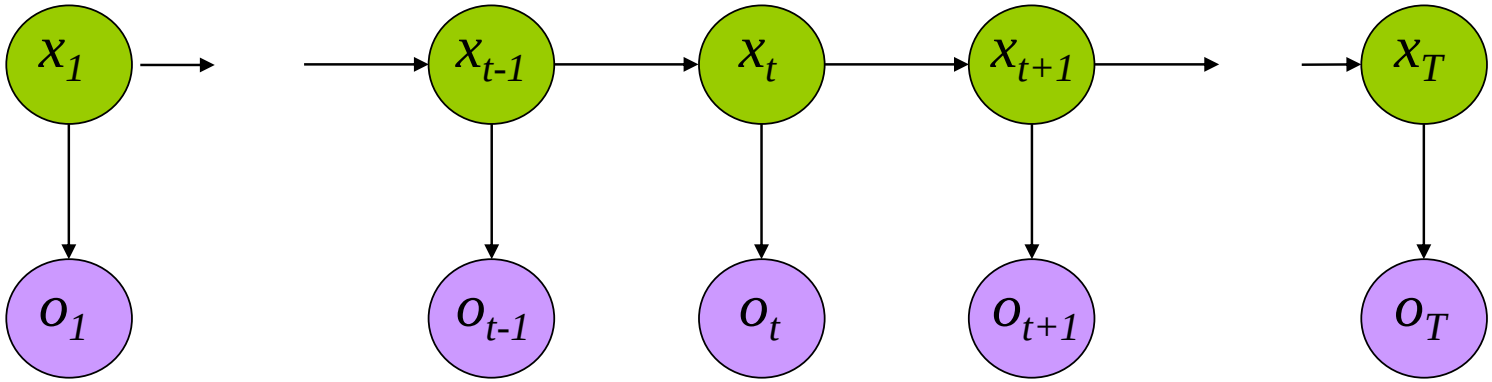
$$P(O \mid X, \mu) = b_{x_1 o_1} b_{x_2 o_2} \ldots b_{x_T o_T}$$

$$P(X \mid \mu) = \pi_{x_1} a_{x_1 x_2} a_{x_2 x_3} \ldots a_{x_{T-1} x_T}$$

$$\boxed{P(O, X \mid \mu) = P(O \mid X, \mu) P(X \mid \mu)}$$

David Meir Blei, Hidden Markov Models, 1999 (Slides)

Decoding



$$P(O \mid X, \mu) = b_{x_1 o_1} b_{x_2 o_2} \ldots b_{x_T o_T}$$

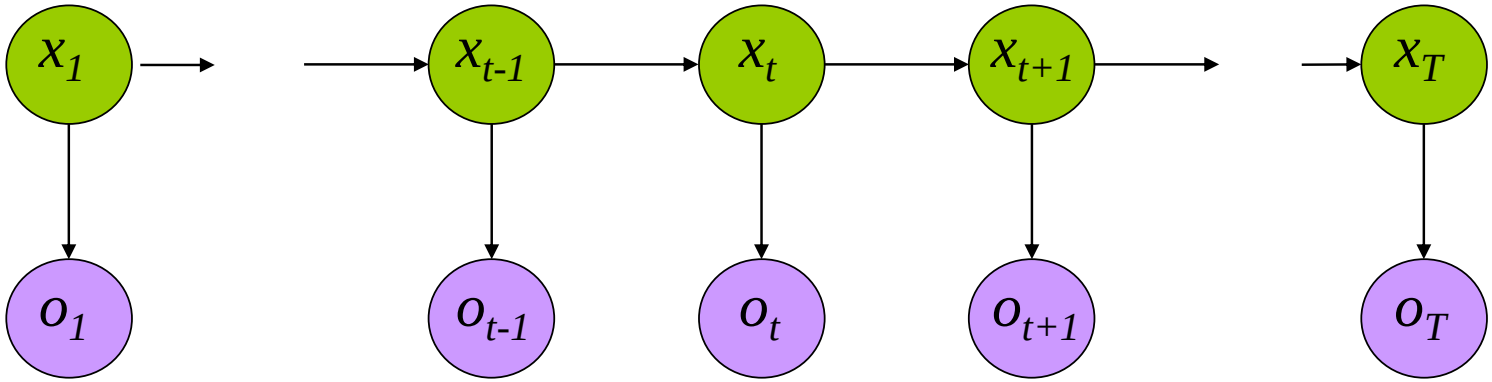
$$P(X \mid \mu) = \pi_{x_1} a_{x_1 x_2} a_{x_2 x_3} \ldots a_{x_{T-1} x_T}$$

$$P(O, X \mid \mu) = P(O \mid X, \mu) P(X \mid \mu)$$

$$P(O \mid \mu) = \sum_X P(O \mid X, \mu) P(X \mid \mu)$$

David Meir Blei, Hidden Markov Models, 1999 (Slides)

Decoding



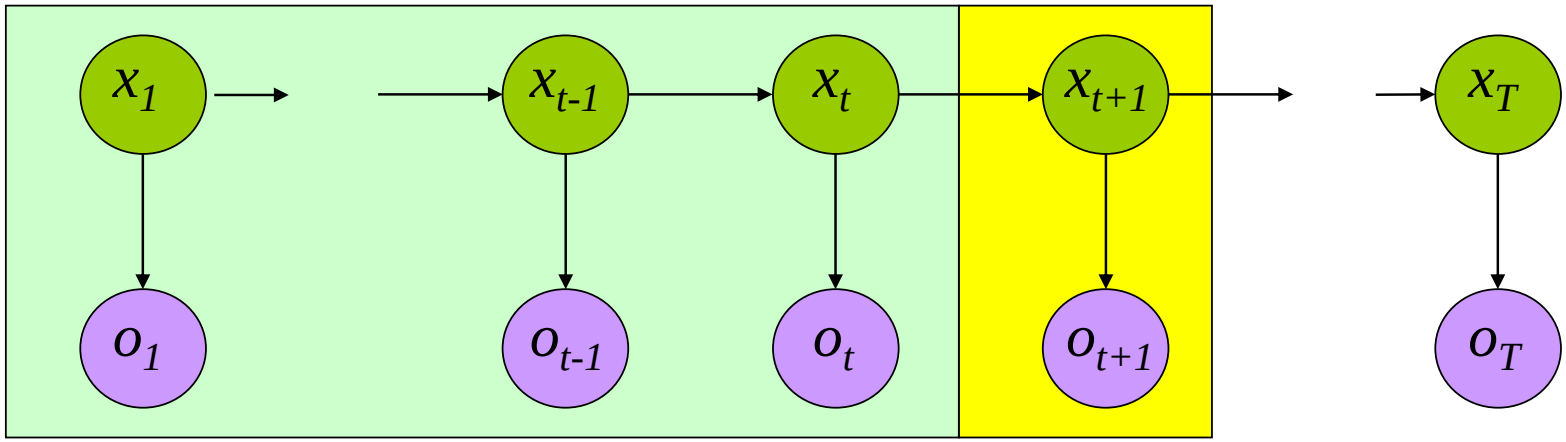
$$P(O \mid \mu) = \sum_{\{x_1 \ldots x_T\}} \pi_{x_1} b_{x_1 o_1} \prod_{t=1}^{T-1} a_{x_t x_{t+1}} b_{x_{t+1} o_{t+1}}$$

David Meir Blei, Hidden Markov Models, 1999 (Slides)

Content

3 Hidden Markov models (HMMs)
- Model definition
- Inference in an HMM
- Decoding
- Forward Procedure
- Backward Procedure
- Decoding Solution
- Viterbi Algorithm
- Parameter Estimation
- HMM Applications

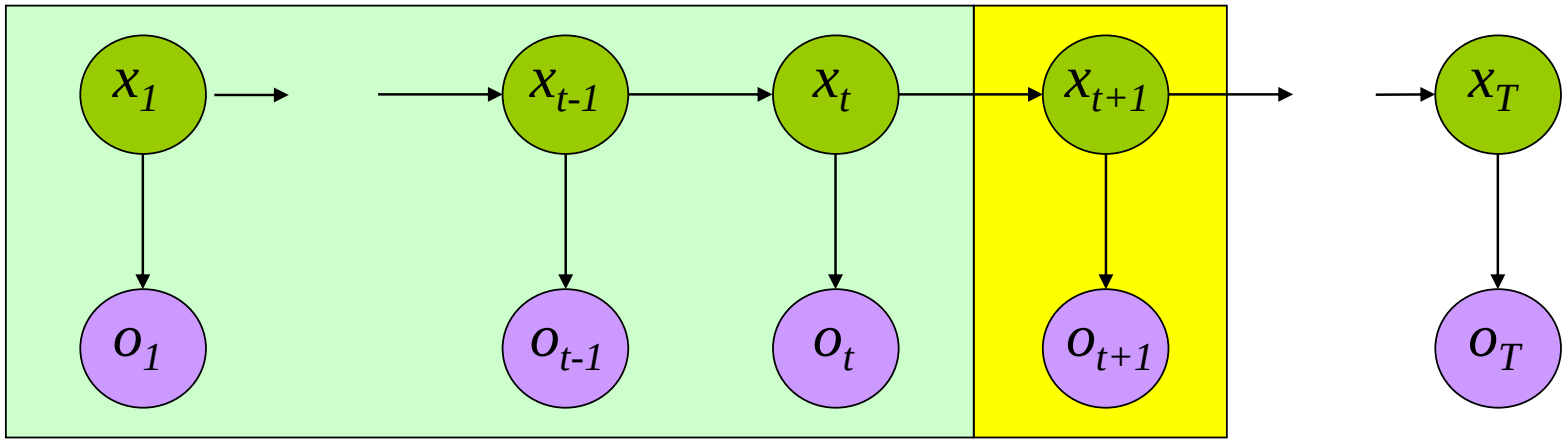
Forward Procedure



- Special structure gives us an efficient solution using *dynamic programming*.
- **Intuition**: Probability of the first t observations is the same for all possible t+1 length state sequences.
- **Define:** $\alpha_i(t) = P(o_1 \ldots o_t, x_t = i \mid \mu)$

David Meir Blei, Hidden Markov Models, 1999 (Slides)

Forward Procedure



$$\alpha_j(t+1)$$

$$= P(o_1 \ldots o_{t+1}, x_{t+1} = j)$$

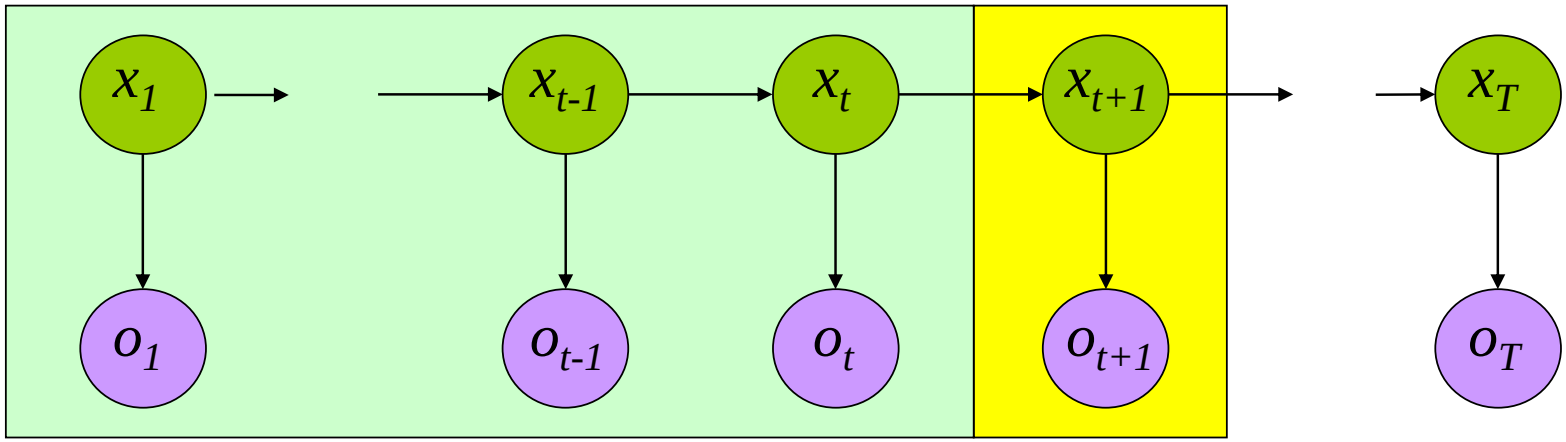
$$= P(o_1 \ldots o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t \mid x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

David Meir Blei, Hidden Markov Models, 1999 (Slides)

Forward Procedure



$$\alpha_j(t+1)$$

$$= P(o_1 \ldots o_{t+1}, x_{t+1} = j)$$

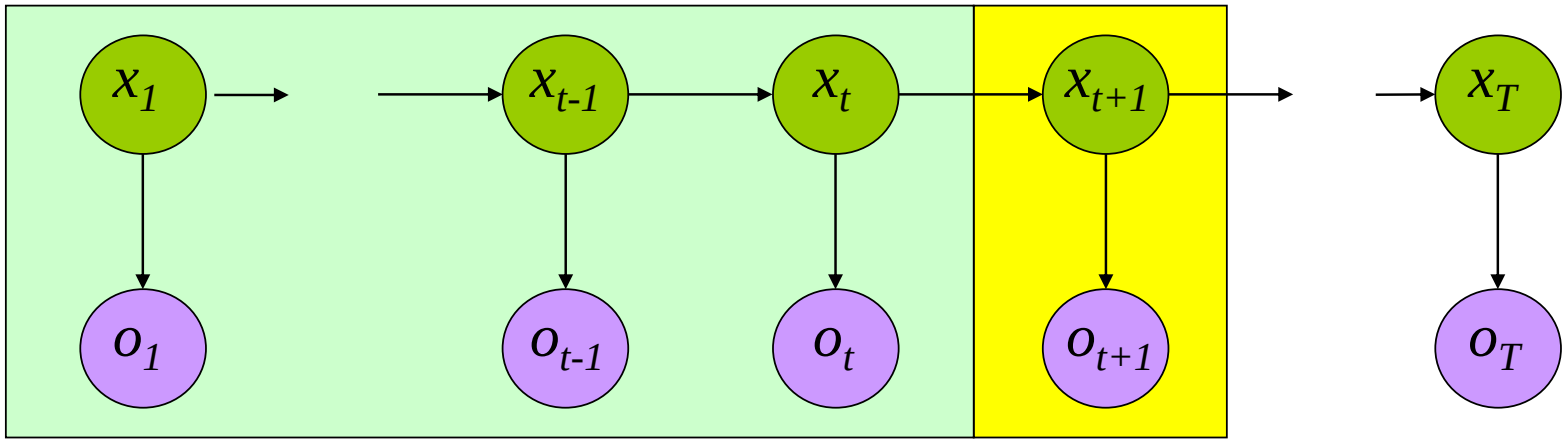
$$= P(o_1 \ldots o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t \mid x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

David Meir Blei, Hidden Markov Models, 1999 (Slides)

Forward Procedure



$$\alpha_j(t+1)$$

$$= P(o_1 \ldots o_{t+1}, x_{t+1} = j)$$

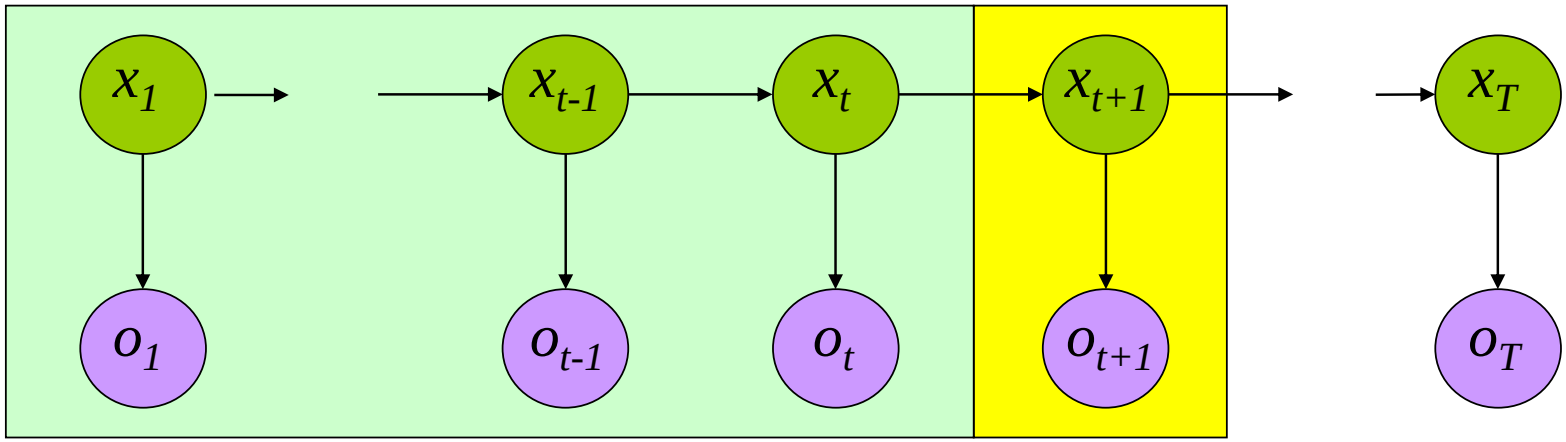
$$= P(o_1 \ldots o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t \mid x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

David Meir Blei, Hidden Markov Models, 1999 (Slides)

Forward Procedure



$$\alpha_j(t+1)$$

$$= P(o_1 \ldots o_{t+1}, x_{t+1} = j)$$

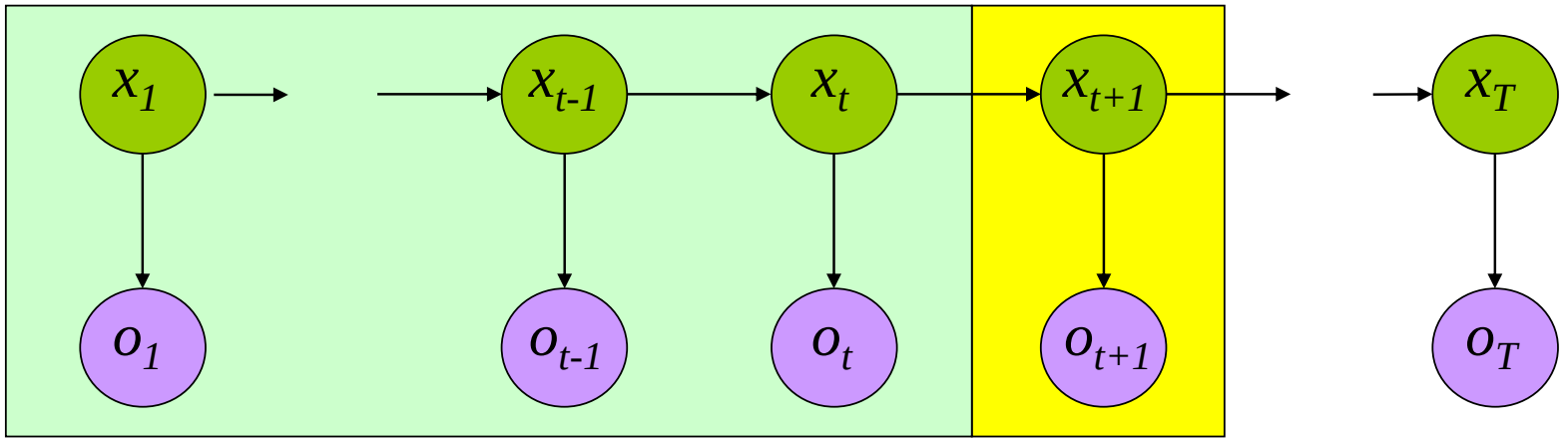
$$= P(o_1 \ldots o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t \mid x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

David Meir Blei, Hidden Markov Models, 1999 (Slides)

Forward Procedure



$$= \sum_{i=1...N} P(o_1 ... o_t, x_t = i, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

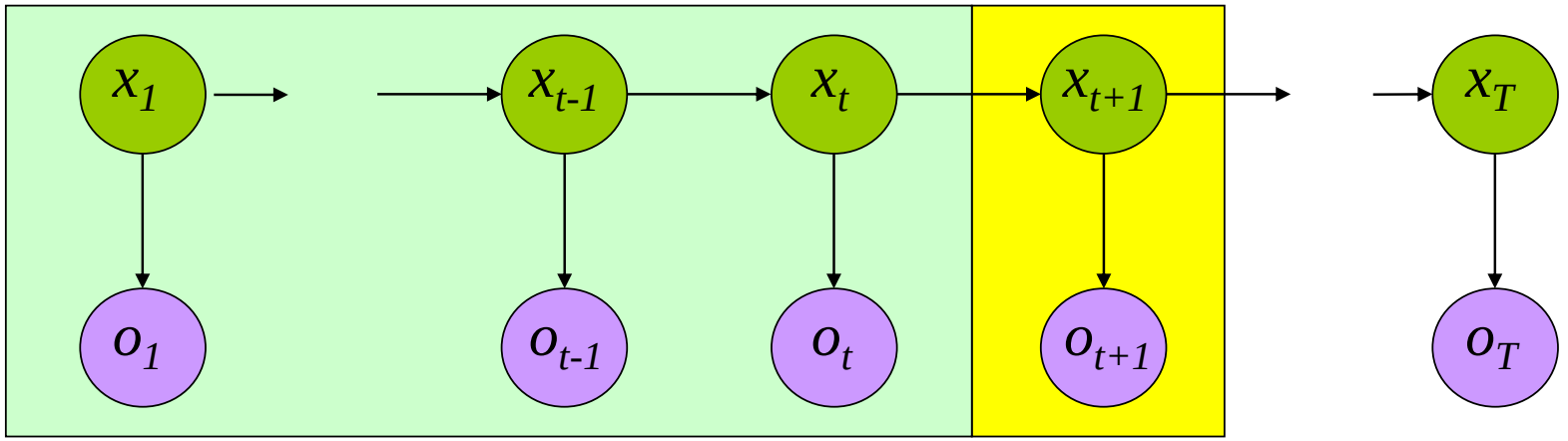
$$= \sum_{i=1...N} P(o_1 ... o_t, x_{t+1} = j \mid x_t = i) P(x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1...N} P(o_1 ... o_t, x_t = i) P(x_{t+1} = j \mid x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1...N} \alpha_i(t) a_{ij} b_{j o_{t+1}}$$

David Meir Blei, Hidden Markov Models, 1999 (Slides)

Forward Procedure



$$= \sum_{i=1...N} P(o_1 ... o_t, x_t = i, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

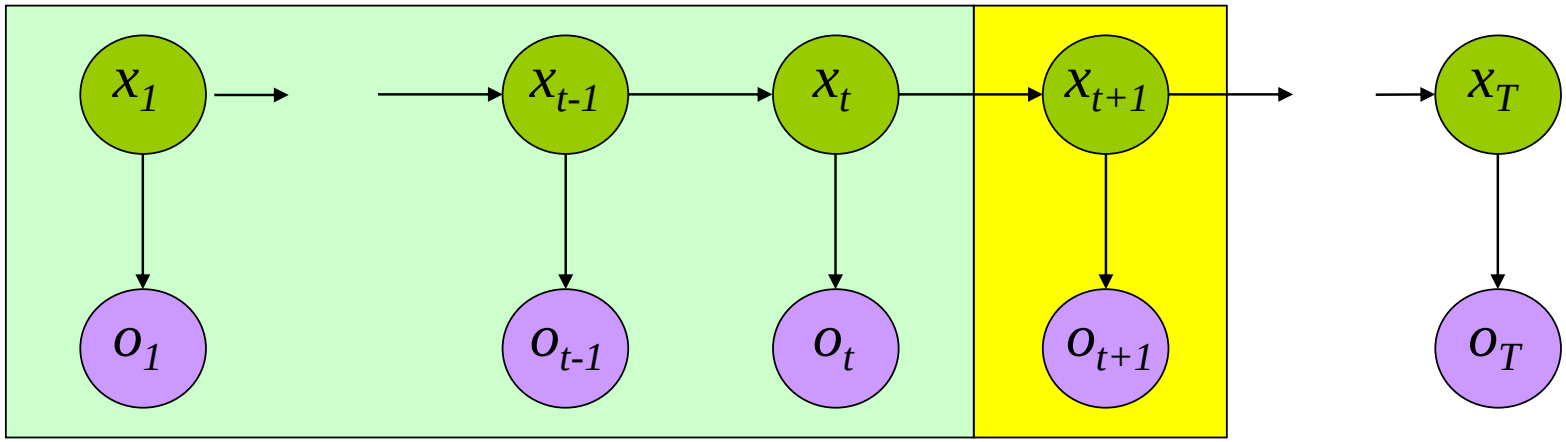
$$= \sum_{i=1...N} P(o_1 ... o_t, x_{t+1} = j \mid x_t = i) P(x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1...N} P(o_1 ... o_t, x_t = i) P(x_{t+1} = j \mid x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1...N} \alpha_i(t) a_{ij} b_{j o_{t+1}}$$

David Meir Blei, Hidden Markov Models, 1999 (Slides)

Forward Procedure



$$= \sum_{i=1...N} P(o_1 ... o_t, x_t = i, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

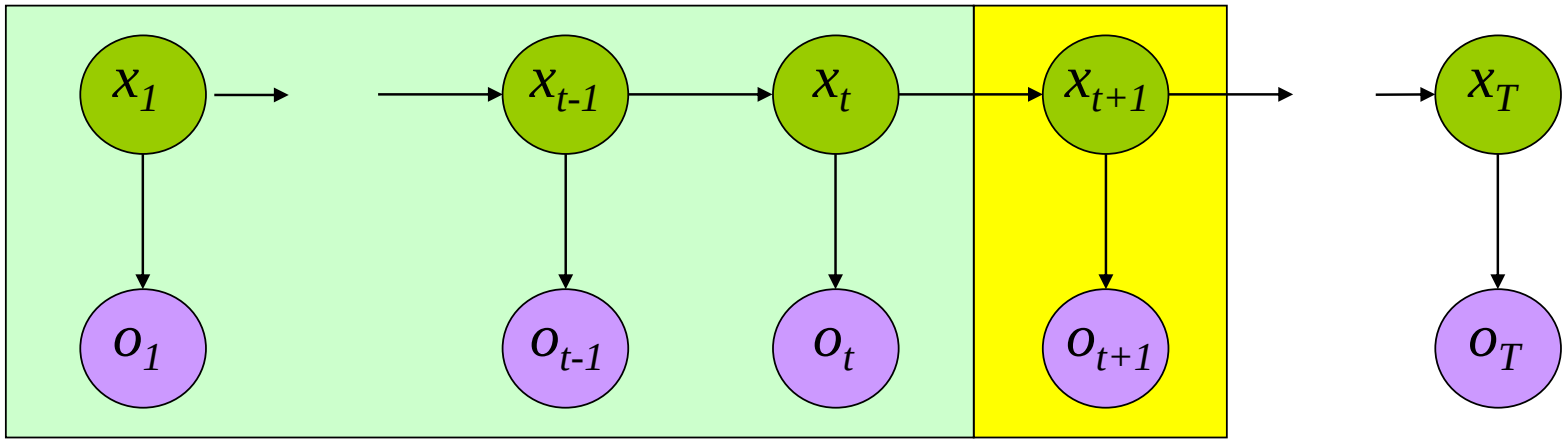
$$= \sum_{i=1...N} P(o_1 ... o_t, x_{t+1} = j \mid x_t = i) P(x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1...N} P(o_1 ... o_t, x_t = i) P(x_{t+1} = j \mid x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1...N} \alpha_i(t) a_{ij} b_{j o_{t+1}}$$

David Meir Blei, Hidden Markov Models, 1999 (Slides)

Forward Procedure



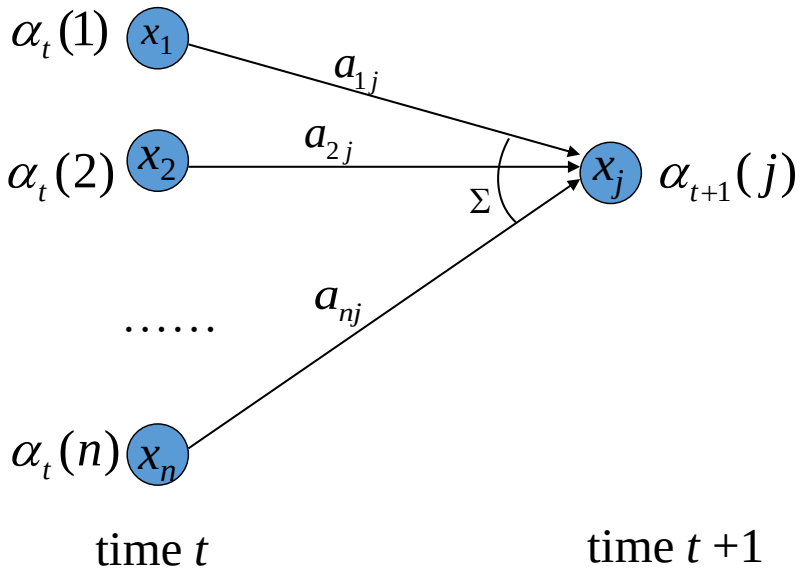
$$= \sum_{i=1...N} P(o_1 ... o_t, x_t = i, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

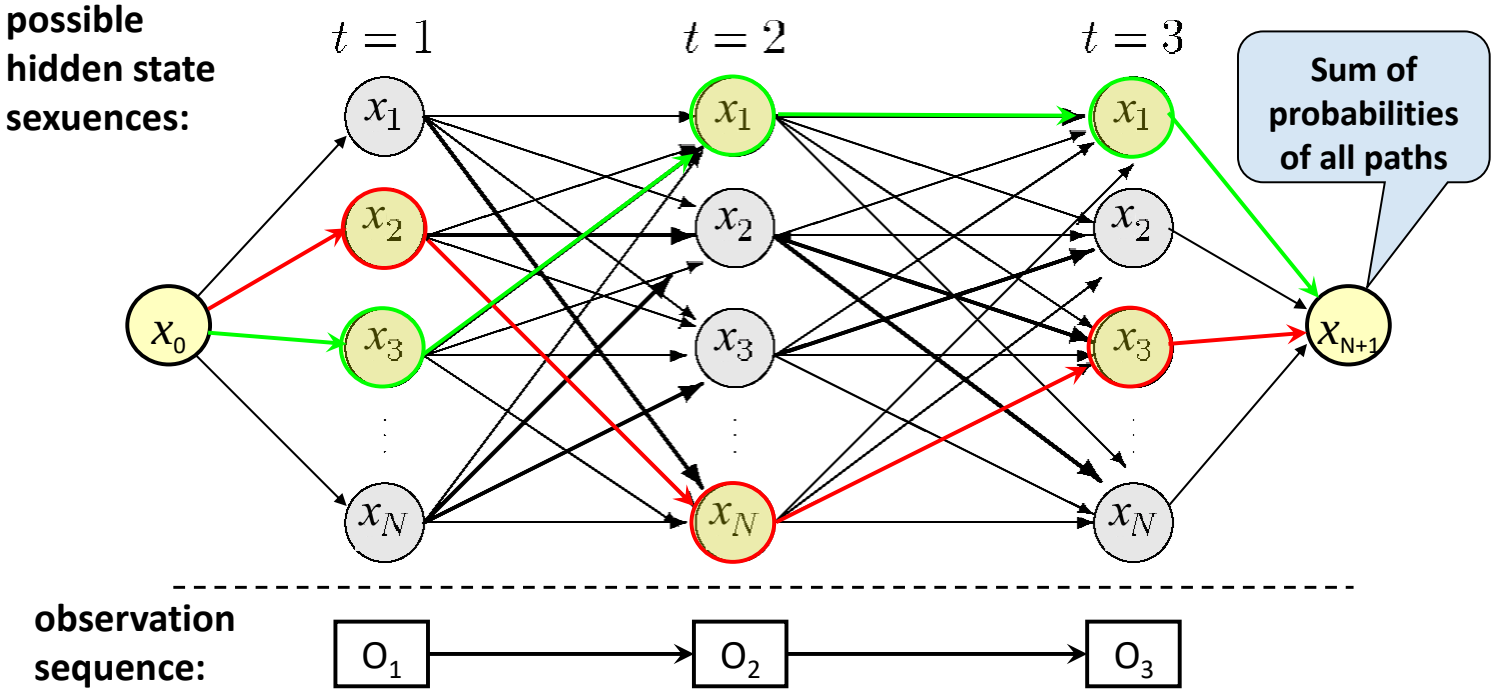
$$= \sum_{i=1...N} P(o_1 ... o_t, x_{t+1} = j \mid x_t = i) P(x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1...N} P(o_1 ... o_t, x_t = i) P(x_{t+1} = j \mid x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1...N} \alpha_i(t) a_{ij} b_{j o_{t+1}}$$

David Meir Blei, Hidden Markov Models, 1999 (Slides)


$\alpha_t(1)$
x_1
a_{1j}
$\alpha_t(2)$
x_2
a_{2j}
x_j
$\alpha_{t+1}(j)$
Σ
.....
a_{nj}
$\alpha_t(n)$
x_n
time t
time t +1



# of possible hidden state sequences: N^T

Time complexity of Forward algorithm: O(N2T)

Forward Algorithm Summary

1. $\alpha_1(i) = \pi_i b_i(O_1),\ 1 \le i \le N$

2. for $t = 1, 2, \ldots, T-1,\ 1 \le j \le N$
$\alpha_{t+1}(j) = [\Sigma_{i=1 \text{ to } N}\ \alpha_t(i) * a_{ij}] * b_j(O_{t+1})$

3. Finally we have:
$P(O|\ \lambda) = \Sigma_{i=1 \text{ to } N}\ \alpha_T(i)$

Multiplications

$= N$

$= (N+1)N(T-1)$

$= 0$

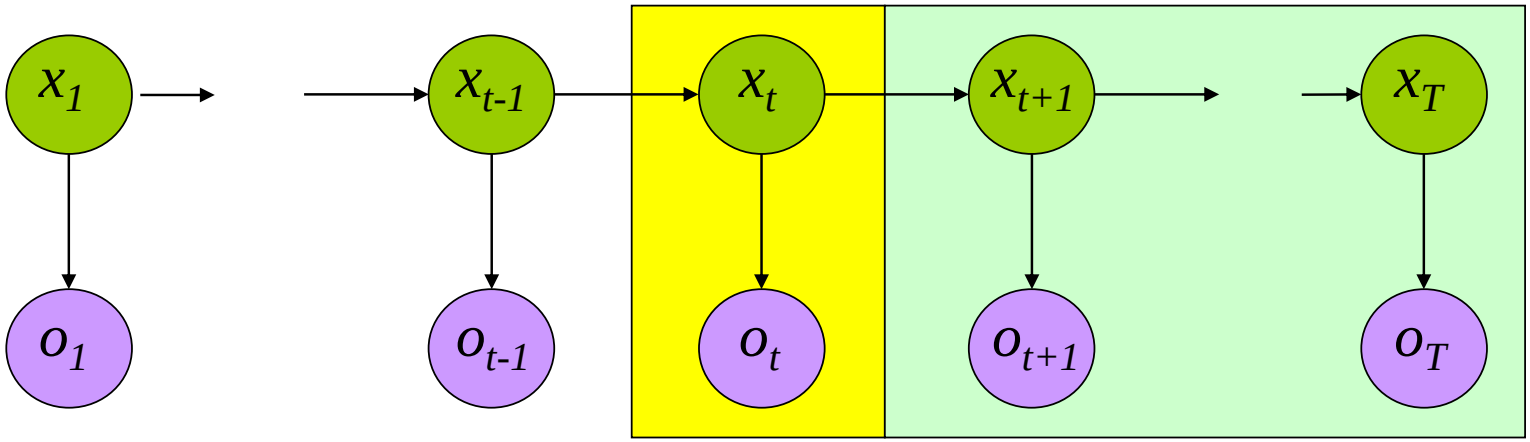
Total:
$N+(N+1)N(T-1)$

Anantharaman Narayana Iyer, Natural Language Processing, Unit 2 – Tagging Problems and HMM (Slides)

Content

3 Hidden Markov models (HMMs)
- Model definition
- Inference in an HMM
- Decoding
- Forward Procedure
- Backward Procedure
- Decoding Solution
- Viterbi Algorithm
- Parameter Estimation
- HMM Applications

Backward Procedure



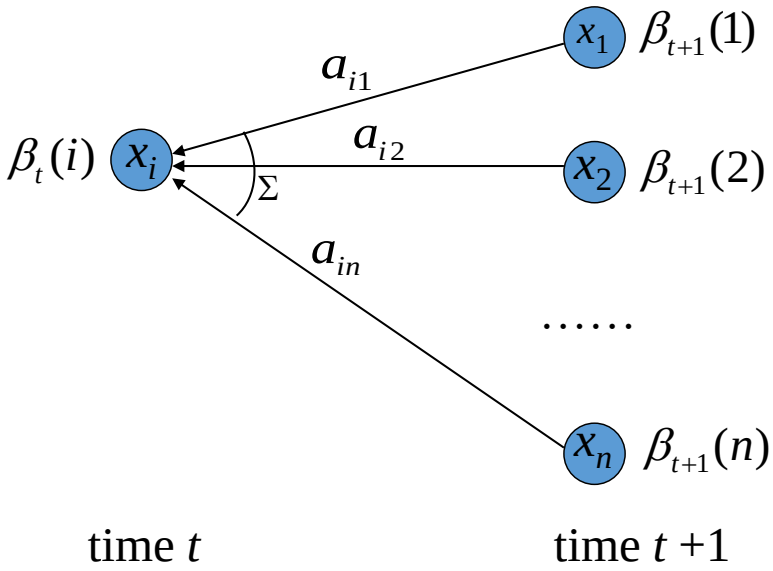
$$\beta_i(T+1) = 1$$

$$\beta_i(t) = P(o_t \ldots o_T \mid x_t = i)$$

$$\beta_i(t) = \sum_{j=1\ldots N} a_{ij} b_{io_t} \beta_j(t+1)$$

Probability of the rest of the states given the first state

David Meir Blei, Hidden Markov Models, 1999 (Slides)


x_1
$\beta_{t+1}(1)$
a_{i1}
$\beta_t(i)$
x_i
a_{i2}
x_2
$\beta_{t+1}(2)$
Σ
a_{in}
.....
x_n
$\beta_{t+1}(n)$
time t
time t +1

Backward Algorithm: Summary

1. $\beta_T(i) = 1, 1 \leq i \leq N$

Multiplications

= 0

2. for t = T-1, T-2, ..., 1, $1 \leq i \leq N$

$\beta_t(i) = \Sigma_{j=1 \text{ to } N} a_{ij} * b_j(O_{t+1}) * \beta_{t+1}(j)$

= (2N)N(T-1)

3. Finally we have:

$P(O|\lambda) = \Sigma_{i=1 \text{ to } N} \pi_i * b_i(O_1) * \beta_1(i)$

= 2N

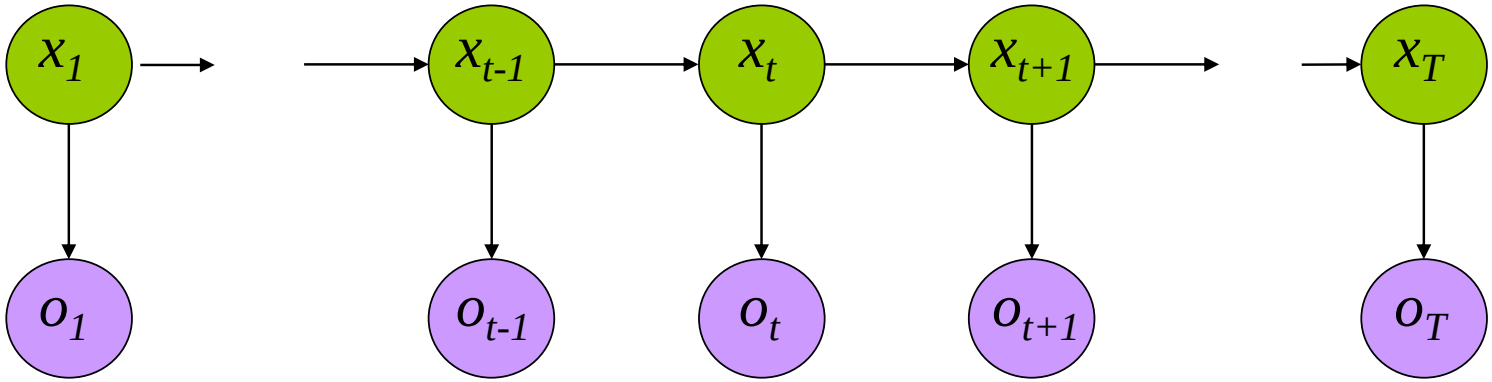
Total:
2N+(2N)N(T-1)

Anantharaman Narayana Iyer, Natural Language Processing, Unit 2 – Tagging Problems and HMM (Slides)

Content

3 Hidden Markov models (HMMs)
- Model definition
- Inference in an HMM
- Decoding
- Forward Procedure
- Backward Procedure
- Decoding Solution
- Viterbi Algorithm
- Parameter Estimation
- HMM Applications

Decoding Solution



$$P(O \mid \mu) = \sum_{i=1}^{N} \alpha_i(T)$$ **Forward Procedure**

$$P(O \mid \mu) = \sum_{i=1}^{N} \pi_i \beta_i(1)$$ **Backward Procedure**

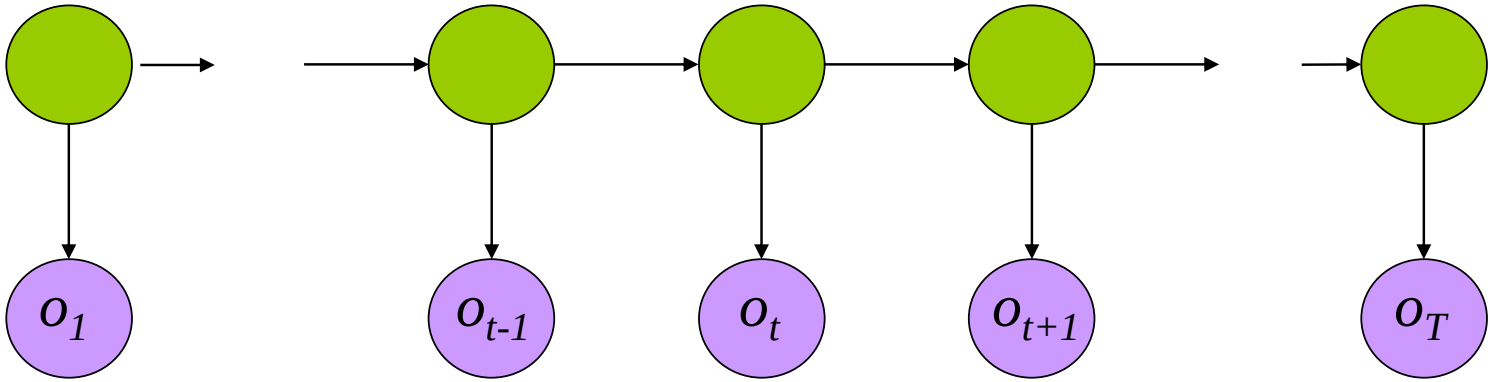
$$P(O \mid \mu) = \sum_{i=1}^{N} \alpha_i(t)\beta_i(t)$$ **Combination**

David Meir Blei, Hidden Markov Models, 1999 (Slides)

Content

3 Hidden Markov models (HMMs)
- Model definition
- Inference in an HMM
- Decoding
- Forward Procedure
- Backward Procedure
- Decoding Solution
- Viterbi Algorithm
- Parameter Estimation
- HMM Applications

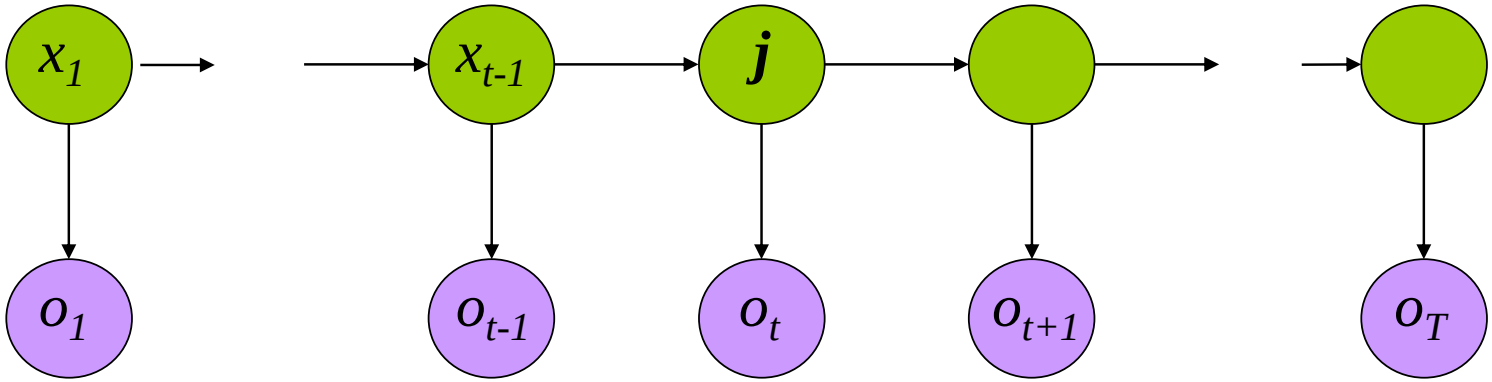
Best State Sequence



- Find the state sequence that best explains the observations
- **Viterbi** algorithm
- $\arg\max_X P(X \mid O)$

David Meir Blei, Hidden Markov Models, 1999 (Slides)

Viterbi Algorithm

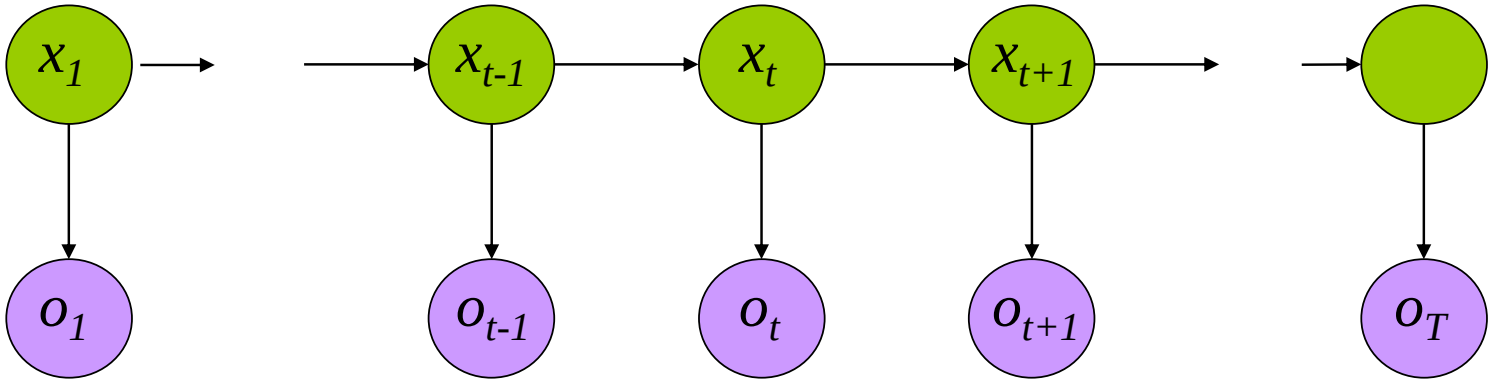


$$\delta_j(t) = \max_{x_1 \ldots x_{t-1}} P(x_1 \ldots x_{t-1}, o_1 \ldots o_{t-1}, x_t = j, o_t)$$

The state sequence which maximizes the probability of seeing the observations to time t-1, landing in state j, and seeing the observation at time t

David Meir Blei, Hidden Markov Models, 1999 (Slides)

Viterbi Algorithm



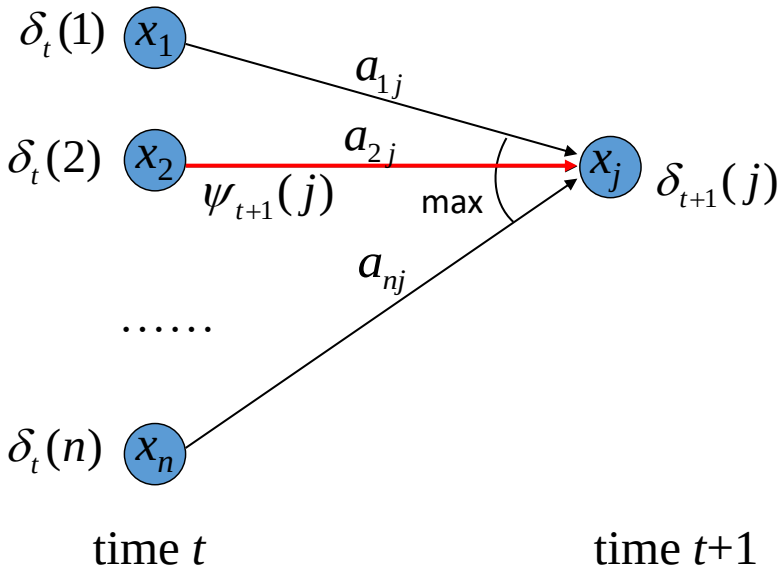
$$\delta_j(t) = \max_{x_1 \ldots x_{t-1}} P(x_1 \ldots x_{t-1}, o_1 \ldots o_{t-1}, x_t = j, o_t)$$

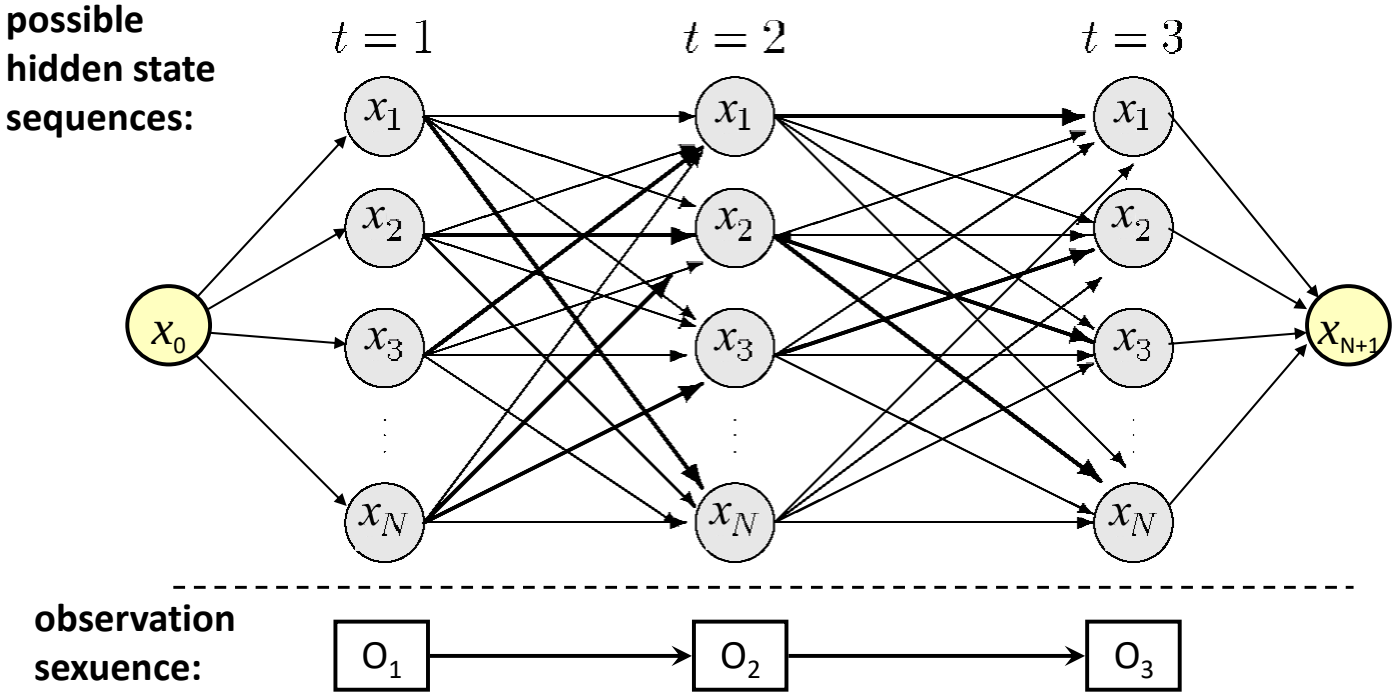
$$\delta_j(t+1) = \max_i \delta_i(t) a_{ij} b_{jo_{t+1}}$$

$$\psi_j(t+1) = \arg\max_i \delta_i(t) a_{ij} b_{jo_{t+1}}$$

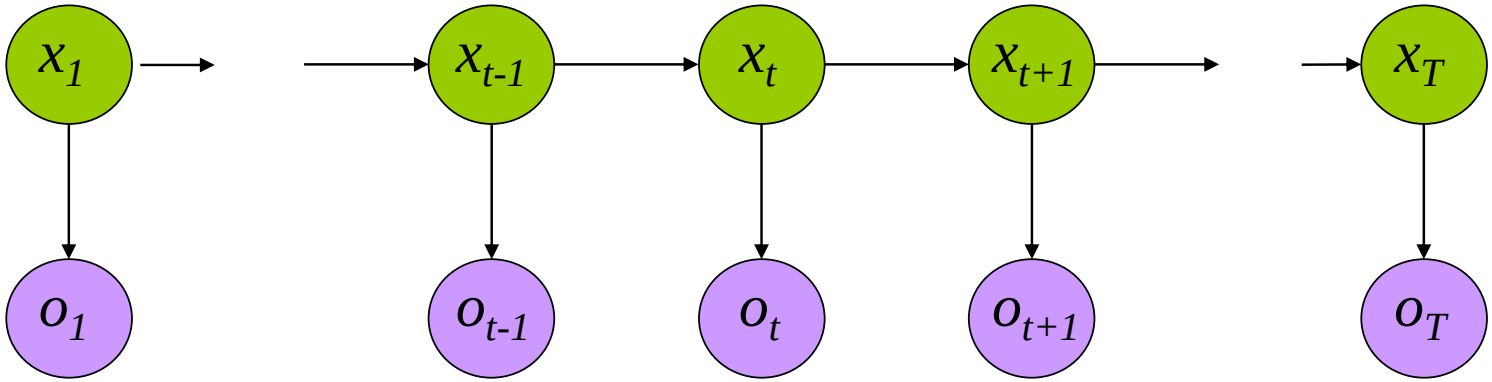
Recursive Computation

David Meir Blei, Hidden Markov Models, 1999 (Slides)


$\delta_t(1)$
x_1
a_{1j}
$\delta_t(2)$
x_2
a_{2j}
x_j
$\delta_{t+1}(j)$
$\psi_{t+1}(j)$
max
a_{nj}
.....
$\delta_t(n)$
x_n
time t
time t+1


possible
hidden state
sequences:
$t = 1$
$t = 2$
$t = 3$
x_1
x_2
x_3
x_N
x_0
x_{N+1}
observation
sexuence:
O_1
O_2
O_3

Viterbi Algorithm



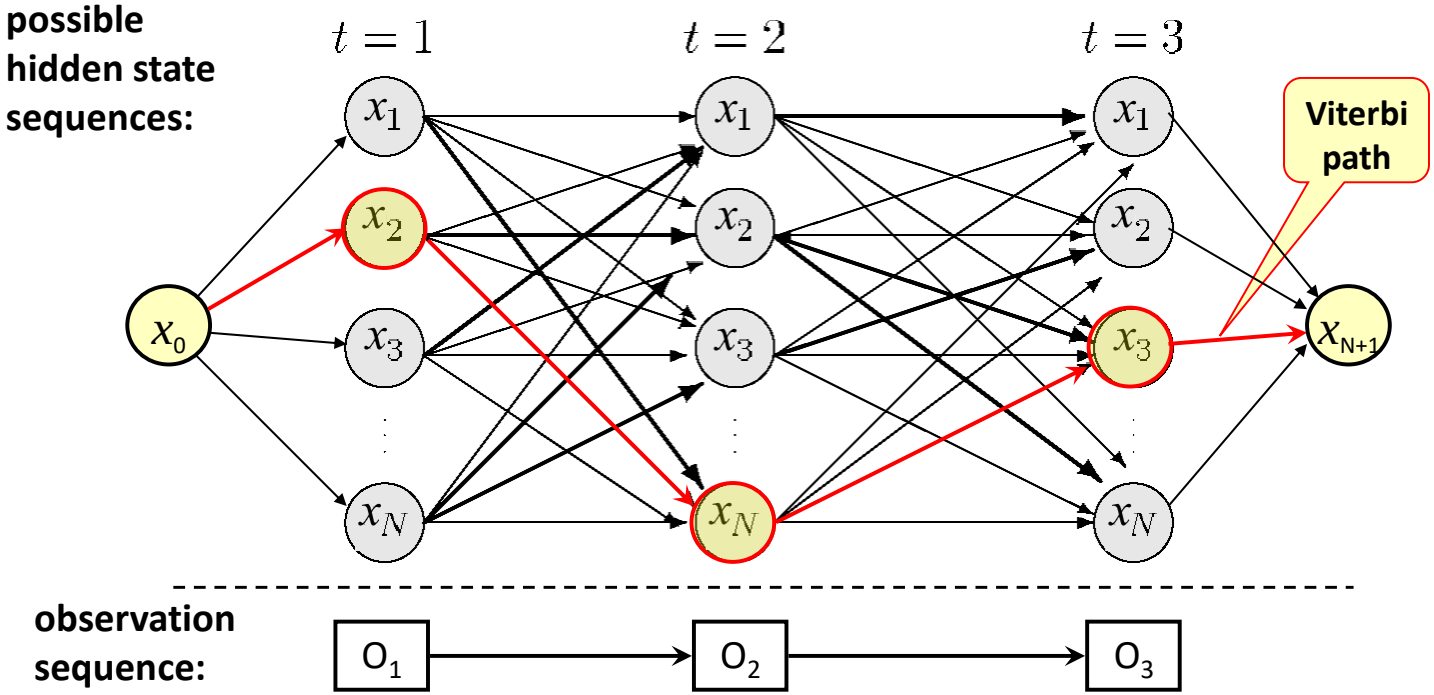
$$\hat{X}_T = \arg\max_i \delta_i(T)$$

$$\hat{X}_t = \psi_{\hat{X}_{t+1}}(t+1)$$

$$P(\hat{X}) = \arg\max_i \delta_i(T)$$

Compute the most likely state sequence by working backwards

David Meir Blei, Hidden Markov Models, 1999 (Slides)



# of possible hidden state sequences: N^T

Time complexity of Viterbi algorithm: O(N2T)

Viterbi Algorithm

- Finding the best single sequence means computing $\text{argmax}_Q P(Q|O,\lambda)$, equivalent to $\text{argmax}_Q P(Q,O|\lambda)$
- The Viterbi algorithm (dynamic programming) defines $\delta_j(t)$, i.e., the highest probability of a single path of length t which accounts for the observations and ends in state S_j

$$\delta_j(t) = \max_{q_1,q_2,\ldots,q_{t-1}} P(q_1 q_2 \ldots q_t = j, O_1 O_2 \ldots O_t|\lambda)$$

- By induction

$$\begin{aligned} \delta_j(1) &= \pi_j b_{jO_1} & 1 \le j \le N \\ \delta_j(t+1) &= \left(\max_i \delta_i(t) a_{ij}\right) b_{jO_{t+1}} & 1 \le t \le T-1 \end{aligned}$$

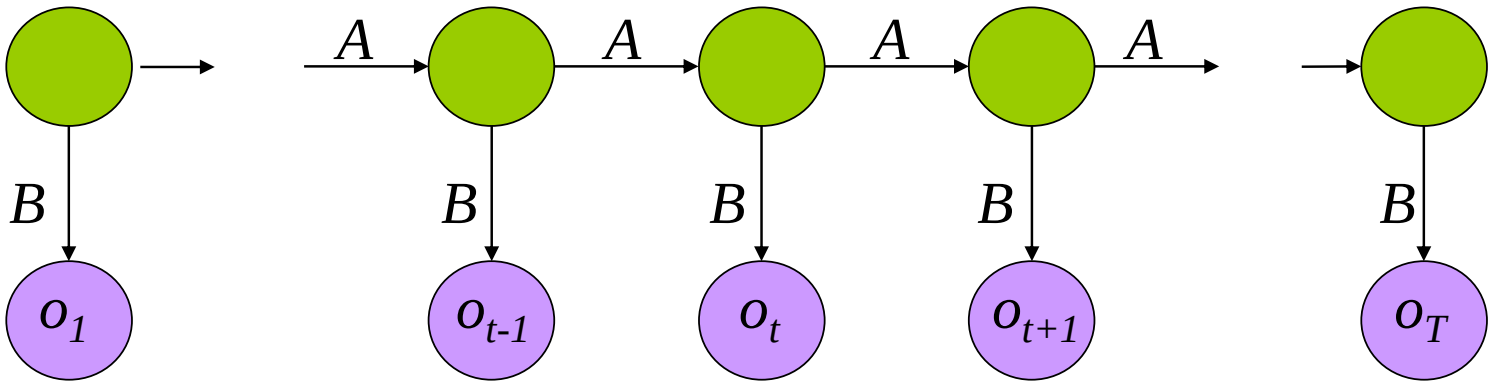
- With backtracking (keeping the maximizing argument for each t and j) we find the optimal solution

Anantharaman Narayana Iyer, Natural Language Processing, Unit 2 – Tagging Problems and HMM (Slides)

Content

3 Hidden Markov models (HMMs)
- Model definition
- Inference in an HMM
- Decoding
- Forward Procedure
- Backward Procedure
- Decoding Solution
- Viterbi Algorithm
- Parameter Estimation
- HMM Applications

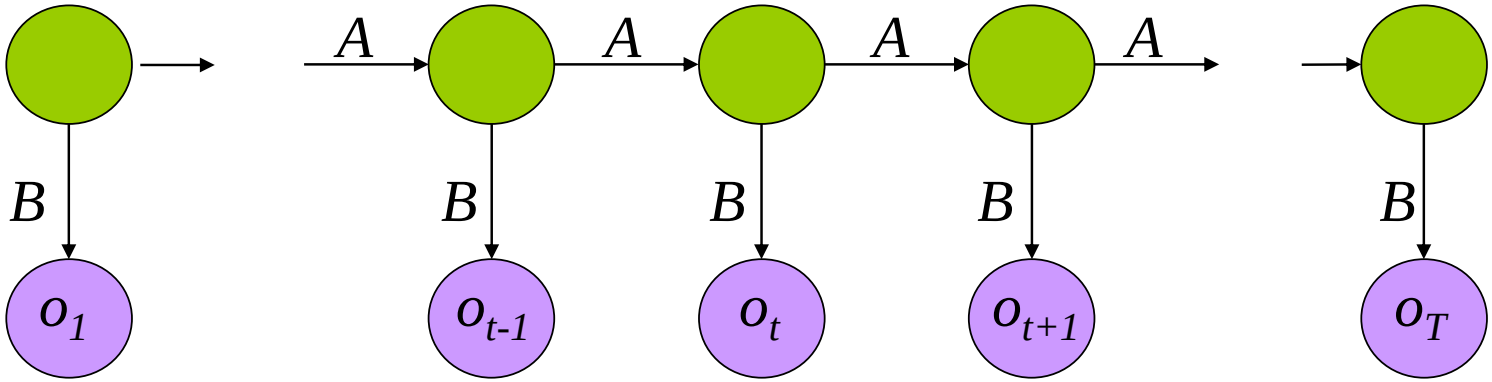
Parameter Estimation



- Given an observation sequence, find the model that is most likely to produce that sequence.
- No analytic method
- Given a model and observation sequence, update the model parameters to better fit the observations.

David Meir Blei, Hidden Markov Models, 1999 (Slides)

Parameter Estimation



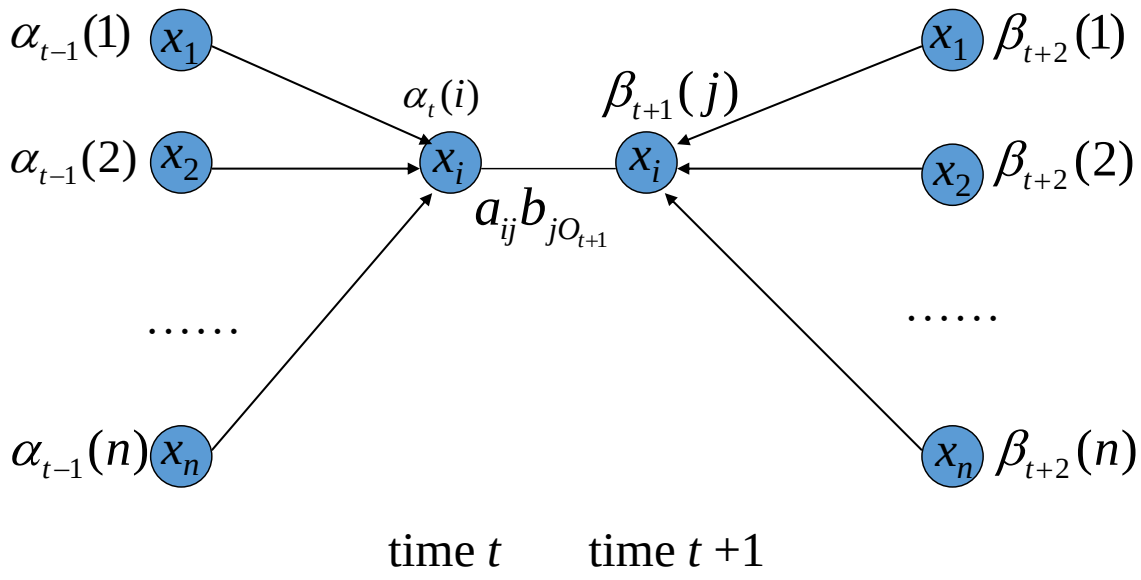
$$p_t(i,j) = \frac{\alpha_i(t) a_{ij} b_{jo_{t+1}} \beta_j(t+1)}{\sum_{m=1\ldots N} \alpha_m(t)\beta_m(t)}$$

Probability of traversing an arc

$$\gamma_i(t) = \sum_{j=1\ldots N} p_t(i,j)$$

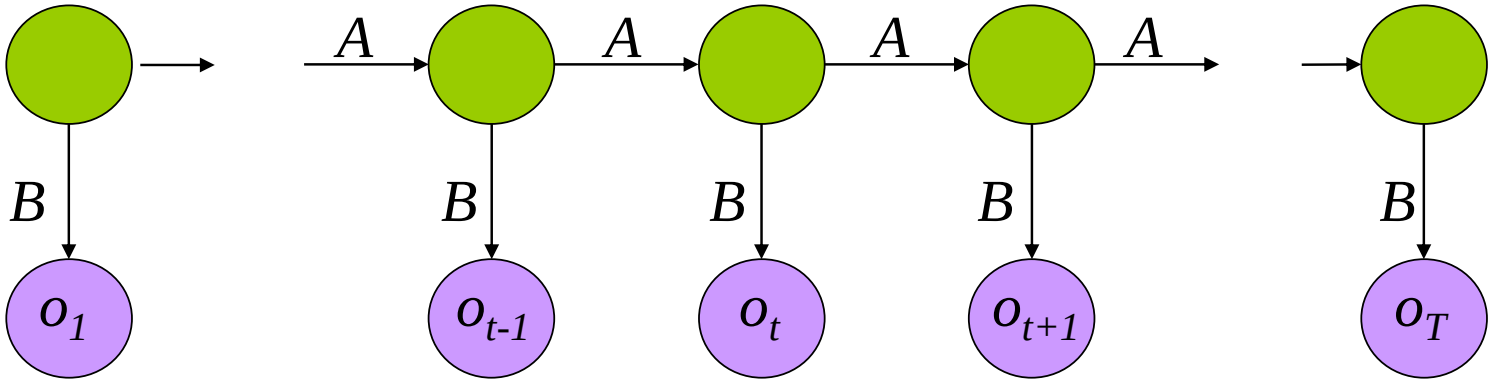
Probability of being in state i

David Meir Blei, Hidden Markov Models, 1999 (Slides)



forward variable and backward variable are used in $\gamma_t(i,j)$

Parameter Estimation



$$\hat{\pi}_i = \gamma_i(1)$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} p_t(i,j)}{\sum_{t=1}^{T} \gamma_i(t)}$$

$$\hat{b}_{ik} = \frac{\sum_{\{t:o_t=k\}} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_i(t)}$$

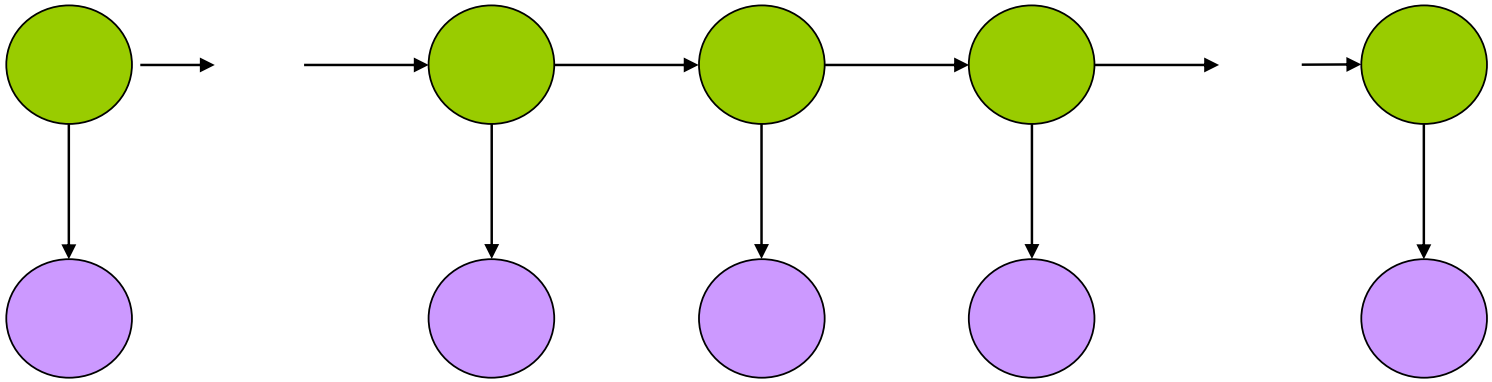
Now we can compute the new estimates of the model parameters.

David Meir Blei, Hidden Markov Models, 1999 (Slides)

Content

3 Hidden Markov models (HMMs)
- Model definition
- Inference in an HMM
- Decoding
- Forward Procedure
- Backward Procedure
- Decoding Solution
- Viterbi Algorithm
- Parameter Estimation
- HMM Applications

HMM Applications



- Generating parameters for n-gram models
- Tagging speech
- Speech recognition

David Meir Blei, Hidden Markov Models, 1999 (Slides)

Content

1 Sequence labeling problems

2 Word window classification

3 Hidden Markov models (HMMs)

4 Graphical models for sequence labeling

Content

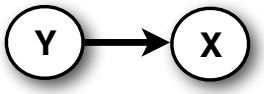
4 Graphical models for sequence labeling
- Maximum entropy Markov models (MEMMs)
- Conditional random fields (CRFs)

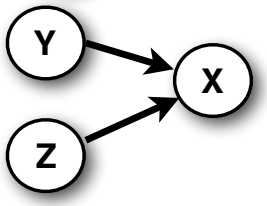
Directed graphical models

Graphical models are a **notation for probability models**.
In a **directed** graphical model, **each node** represents a distribution over a random variable:

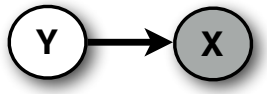
– $P(X) =$ 

Arrows represent dependencies (they define what other random variables the current node is conditioned on)

– $P(Y)\ P(X \mid Y) =$ 

– $P(Y)\ P(Z)\ P(X \mid Y, Z) =$ 

Shaded nodes represent observed variables.
White nodes represent hidden variables

– $P(Y)\ P(X \mid Y)$ with Y hidden and X observed = 

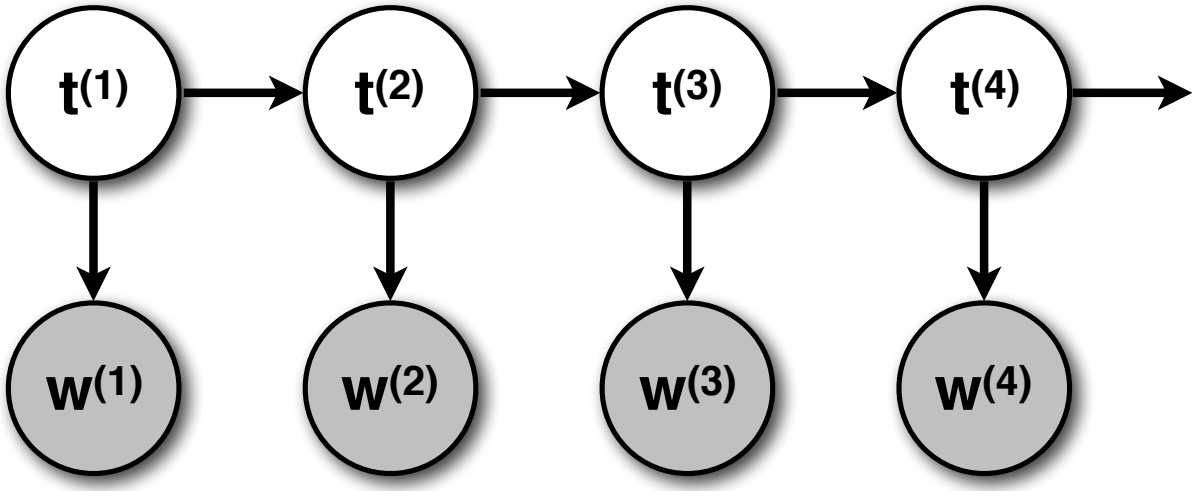
Julia Hockenmaier, UIUC CS447: Natural Language Processing (slides)

HMMs as graphical models

HMMs are **generative** models of the observed input string **w**

They 'generate' **w** with $P(\mathbf{w},\mathbf{t}) = \prod_i P(t^{(i)}|\, t^{(i-1)}) P(w^{(i)}|\, t^{(i)})$

When we use an HMM to tag, we observe **w**, and need to find **t**



Julia Hockenmaier, UIUC CS447: Natural Language Processing (slides)

Models for sequence labeling

Sequence labeling: Given an input sequence $\mathbf{w} = w^{(1)} \ldots w^{(n)}$, predict the best (most likely) label sequence $\mathbf{t} = t^{(1)} \ldots t^{(n)}$

$$\underset{\mathbf{t}}{\operatorname{argmax}} P(\mathbf{t}|\mathbf{w})$$

Generative models use Bayes Rule:

$$\begin{aligned}
\underset{\mathbf{t}}{\operatorname{argmax}} P(\mathbf{t}|\mathbf{w}) &= \underset{\mathbf{t}}{\operatorname{argmax}} \frac{P(\mathbf{t},\mathbf{w})}{P(\mathbf{w})} \\
&= \underset{\mathbf{t}}{\operatorname{argmax}} P(\mathbf{t},\mathbf{w}) \\
&= \underset{\mathbf{t}}{\operatorname{argmax}} P(\mathbf{t})P(\mathbf{w}|\mathbf{t})
\end{aligned}$$

Discriminative (conditional) models model $\mathrm{P}(\mathbf{t}\,|\mathbf{w})$ directly

Julia Hockenmaier, UIUC CS447: Natural Language Processing (slides)

Advantages of discriminative models

We're usually not really interested in $\mathrm{P}(\mathbf{w} \mid \mathbf{t})$.

– $\mathbf{w}$ is given. We don't need to predict it!

Why not model what we're actually interested in: $\mathrm{P}(\mathbf{t} \mid \mathbf{w})$

Modeling $\mathrm{P}(\mathbf{w} \mid \mathbf{t})$ well is quite difficult:

– Prefixes (capital letters) or suffixes are good predictors for certain classes of $\mathbf{t}$ (proper nouns, adverbs,…)

– Se we don't want to model words as atomic symbols, but in terms of features

– These features may also help us deal with unknown words

– But features may not be independent

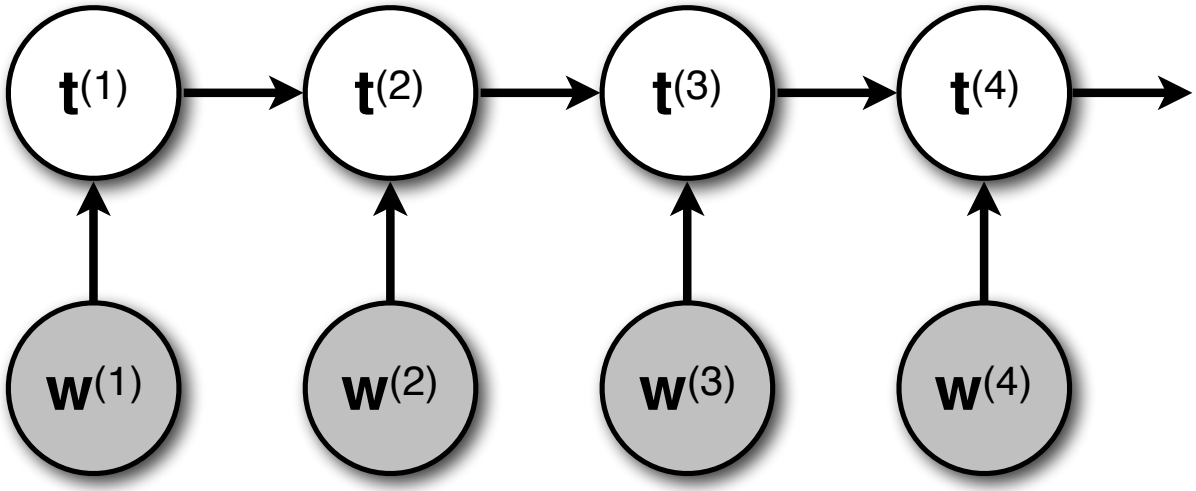
Modeling $\mathrm{P}(\mathbf{t} \mid \mathbf{w})$ with features should be easier:

– Now we can incorporate arbitrary features of the word, because we don't need to predict $\mathbf{w}$ anymore

Julia Hockenmaier, UIUC CS447: Natural Language Processing (slides)

Discriminative probability models

A discriminative or **conditional** model of the labels **t** given the observed input string **w** models $P(\mathbf{t} \mid \mathbf{w}) = \prod_i P(t^{(i)} \mid w^{(i)}, t^{(i-1)})$ directly.



Julia Hockenmaier, UIUC CS447: Natural Language Processing (slides)

Discriminative models

There are two main types of discriminative probability models:

- Maximum Entropy Markov Models (MEMMs)
- Conditional Random Fields (CRFs)

MEMMs and CRFs:

- are both based on logistic regression
- have the same graphical model
- require the Viterbi algorithm for tagging
- differ in that MEMMs consist of independently learned distributions, while CRFs are trained to maximize the probability of the entire sequence

Julia Hockenmaier, UIUC CS447: Natural Language Processing (slides)

Probabilistic classification

Classification:

Predict a class (label) c for an input $\mathbf{x}$

There are only a (small) finite number of possible class labels

Probabilistic classification:

– Model the probability $P(c \mid \mathbf{x})$

$P(c|\mathbf{x})$ is a probability if $0 \leq P(c_i \mid \mathbf{x}) \leq 1$, and $\sum_i P(c_i \mid \mathbf{x}) = 1$

– Return the class $c^* = \text{argmax}_i P(c_i \mid \mathbf{x})$ that has the highest probability

One standard way to model $P(c \mid \mathbf{x})$ is logistic regression (used by MEMMs and CRFs)

Julia Hockenmaier, UIUC CS447: Natural Language Processing (slides)

Using features

Think of feature functions as useful questions you can ask about the input $\mathbf{x}$:

- **Binary feature functions**:
  $f_{\text{first-letter-capitalized}}(\textbf{Urbana}) = 1$
  $f_{\text{first-letter-capitalized}}(\textbf{computer}) = 0$

- **Integer (or real-valued) features**:
  $f_{\text{number-of-vowels}}(\textbf{Urbana}) = 3$

Which specific feature functions are useful will depend on your task (and your training data).

Julia Hockenmaier, UIUC CS447: Natural Language Processing (slides)

From features to probabilities

We associate a real-valued weight w_{ic} with each feature function $f_i(\mathbf{x})$ and output class c

Note that the feature function $f_i(\mathbf{x})$ does not have to depend on c as long as the weight does (note the double index w_{ic})

This gives us a real-valued score for predicting class c for input $\mathbf{x}$: $\text{score}(\mathbf{x},c) = \sum_i w_{ic} f_i(\mathbf{x})$

This score could be negative, so we exponentiate it:
$\text{score}(\mathbf{x},c) = \exp(\sum_i w_{ic} f_i(\mathbf{x}))$
To get a probability distribution over all classes c, we renormalize these scores:

$$P(c \mid \mathbf{x}) = \text{score}(\mathbf{x},c) / \sum_j \text{score}(\mathbf{x},c_j)$$
$$= \exp(\sum_i w_{ic} f_i(\mathbf{x})) / \sum_j \exp(\sum_i w_{ij} f_i(\mathbf{x}))$$

Julia Hockenmaier, UIUC CS447: Natural Language Processing (slides)

Learning: finding $\boldsymbol{w}$

Learning = finding weights $\boldsymbol{w}$
We use conditional maximum likelihood estimation
(and standard convex optimization algorithms)
to find/learn $\boldsymbol{w}$

(for more details, attend CS446 and CS546)

The conditional MLE training objective:

Find the **w** that assigns highest probability to all observed outputs c_i given the inputs $\mathbf{x}_i$

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} \prod_i P(c_i|\mathbf{x}_i, \mathbf{w})$$

Julia Hockenmaier, UIUC CS447: Natural Language Processing (slides)

Terminology

Models that are of the form

$$P(c \mid \mathbf{x}) = \text{score}(\mathbf{x},c) / \sum_j \text{score}(\mathbf{x},c_j)$$
$$= \exp(\sum_i w_{ic} f_i(\mathbf{x})) / \sum_j \exp(\sum_i w_{ij} f_i(\mathbf{x}))$$

are also called loglinear models, Maximum Entropy (MaxEnt) models, or multinomial logistic regression models

CS446 and CS546 should give you more details about these.

The normalizing term $\sum_j \exp(\sum_i w_{ij} f_i(\mathbf{x}))$ is also called the partition function and is often abbreviated as Z

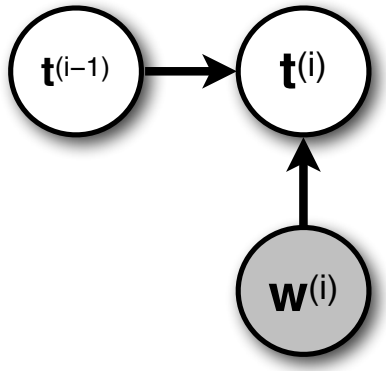
Julia Hockenmaier, UIUC CS447: Natural Language Processing (slides)

Terminology

- All these terms are the same:
  - Softmax
  - Softmax regression
  - Multinomial (or multiclass) logistic regression
  - Maximum Entropy (MaxEnt) model (classifier)
  - Multinomial logit
  - Monditional maximum entropy model

Maximum Entropy Markov Models

MEMMs use a MaxEnt classifier for each $P(t^{(i)} | w^{(i)}, t^{(i-1)})$:



Since we use w to refer to words, let's use λ_{jk} as the weight for the feature function $f_j(t^{(i-1)}, w^{(i)})$ when predicting tag t_k:

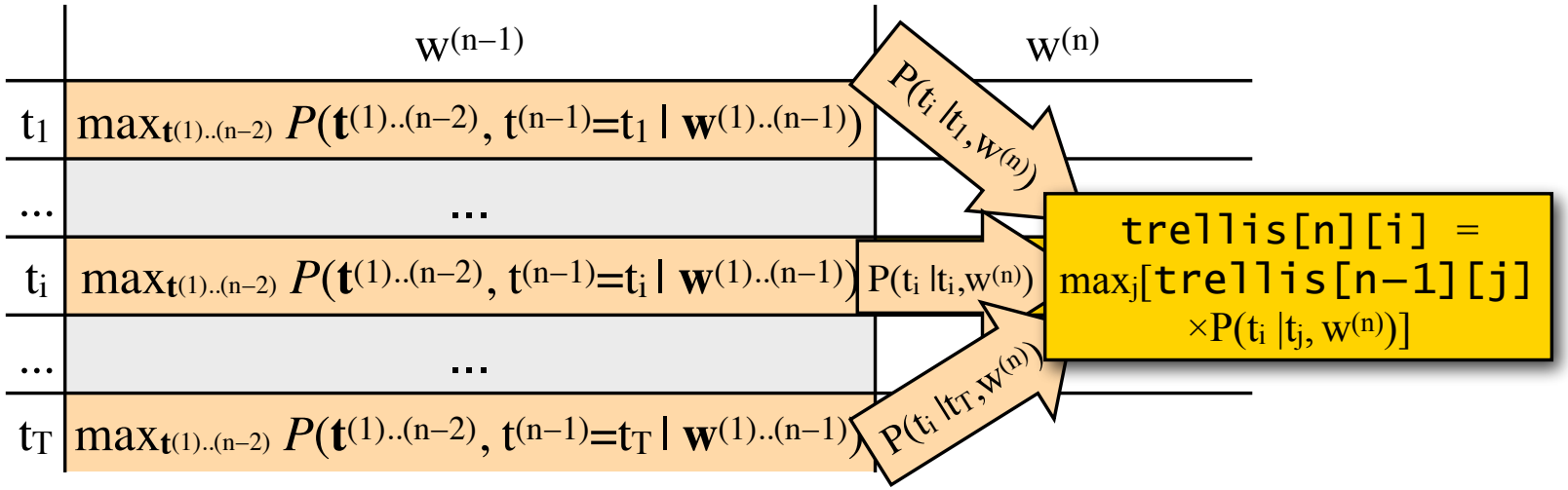
$$P(t^{(i)} = t_k \mid t^{(i-1)}, w^{(i)}) = \frac{\exp(\sum_j \lambda_{jk} f_j(t^{(i-1)}, w^{(i)})}{\sum_l \exp(\sum_j \lambda_{jl} f_j(t^{(i-1)}, w^{(i)})}$$

Julia Hockenmaier, UIUC CS447: Natural Language Processing (slides)

Viterbi for MEMMs

`trellis[n][i]` stores the probability of the most likely (Viterbi) tag sequence $\mathbf{t}^{(1)\ldots(n)}$ that ends in tag t_i for the prefix $w^{(1)}\ldots w^{(n)}$
Remember that we do not generate $\mathbf{w}$ in MEMMs. So:

$$\texttt{trellis[n][i]} = \max_{\mathbf{t}^{(1)..(n-1)}}[\, P(\mathbf{t}^{(1)\ldots(n-1)}, t^{(n)}{=}t_i \mid \mathbf{w}^{(1)\ldots(n)})\,]$$
$$= \max_j\,[\,\texttt{trellis[n-1][j]} \times P(t_i \mid t_j, w^{(n)})\,]$$
$$= \max_j\,[\max_{\mathbf{t}^{(1)..(n-2)}}[P(\mathbf{t}^{(1)..(n-2)}, t^{(n-1)}{=}t_j \mid \mathbf{w}^{(1)..(n-1)})] \times P(t_i \mid t_j, w^{(n)})]$$



Julia Hockenmaier, UIUC CS447: Natural Language Processing (slides)

The Maximum Entropy Markov Model (MEMM)

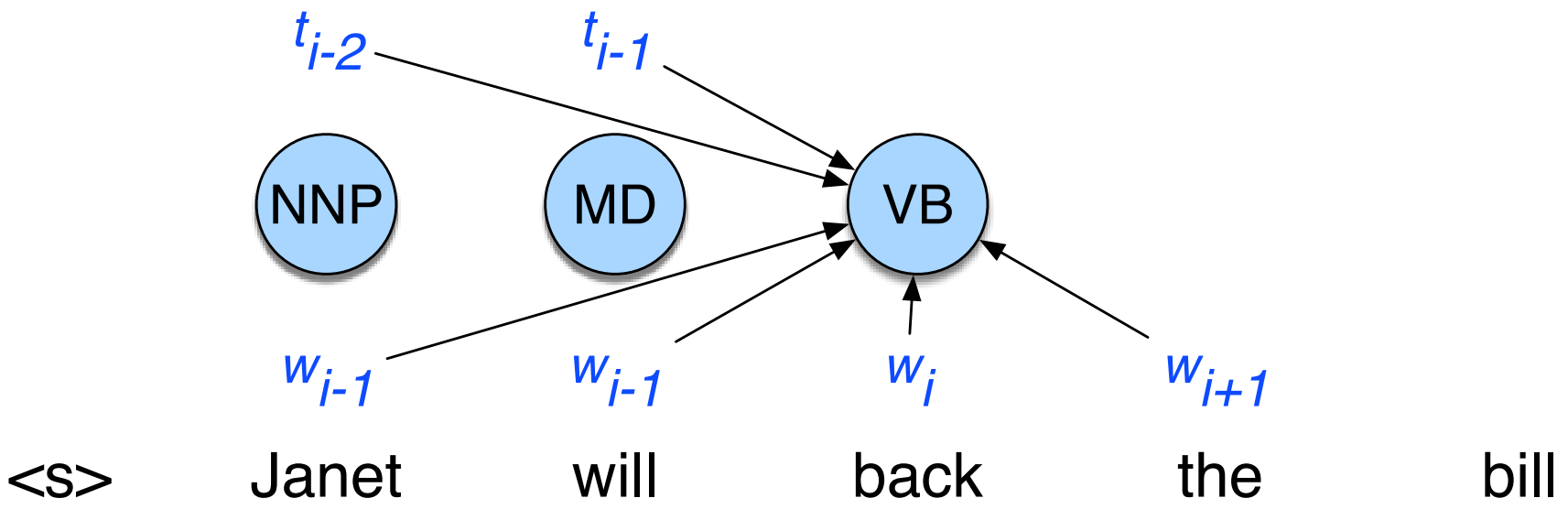
- A sequence version of the logistic regression (also called maximum entropy) classifier.
- Find the best series of tags:

$$\begin{aligned}\hat{T} &= \operatorname*{argmax}_T P(T|W) \\ &= \operatorname*{argmax}_T \prod_i P(t_i|w_i, t_{i-1})\end{aligned}$$

48

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

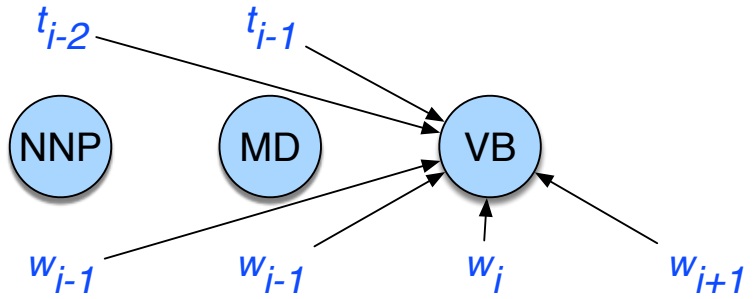
The Maximum Entropy Markov Model (MEMM)



49

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

Features for the classifier at each tag



<s> Janet will back the bill

$$t_i = \text{VB and } w_{i-2} = \text{Janet}$$
$$t_i = \text{VB and } w_{i-1} = \text{will}$$
$$t_i = \text{VB and } w_i = \text{back}$$
$$t_i = \text{VB and } w_{i+1} = \text{the}$$
$$t_i = \text{VB and } w_{i+2} = \text{bill}$$
$$t_i = \text{VB and } t_{i-1} = \text{MD}$$
$$t_i = \text{VB and } t_{i-1} = \text{MD and } t_{i-2} = \text{NNP}$$
$$t_i = \text{VB and } w_i = \text{back and } w_{i+1} = \text{the}$$

50

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

More features

w_i contains a particular prefix (from all prefixes of length ≤ 4)
w_i contains a particular suffix (from all suffixes of length ≤ 4)
w_i contains a number
w_i contains an upper-case letter
w_i contains a hyphen
w_i is all upper case
w_i's word shape
w_i's short word shape
w_i is upper case and has a digit and a dash (like *CFC-12*)
w_i is upper case and followed within 3 words by Co., Inc., etc.

51

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

MEMM computes the best tag sequence

$$
\begin{aligned}
\hat{T} &= \operatorname*{argmax}_T P(T|W) \\
&= \operatorname*{argmax}_T \prod_i P(t_i | w_{i-l}^{i+l}, t_{i-k}^{i-1}) \\
&= \operatorname*{argmax}_T \prod_i \frac{\exp\left(\sum_i w_i f_i(t_i, w_{i-l}^{i+l}, t_{i-k}^{i-1})\right)}{\sum_{t' \in \text{tagset}} \exp\left(\sum_i w_i f_i(t', w_{i-l}^{i+l}, t_{i-k}^{i-1})\right)}
\end{aligned}
$$

52

Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

MEMM Decoding

- Simplest algorithm:

function GREEDY MEMM DECODING(words W, model P) **returns** tag sequence T

for $i = 1$ **to** *length(W)*

$$\hat{t}_i = \underset{t' \in T}{\text{argmax}}\ P(t' \mid w_{i-l}^{i+l}, t_{i-k}^{i-1})$$

- What we use in practice: The **Viterbi** algorithm

ısed

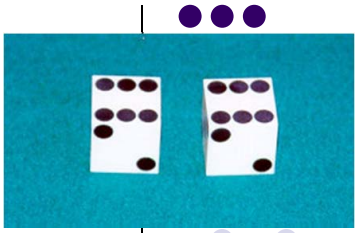
53

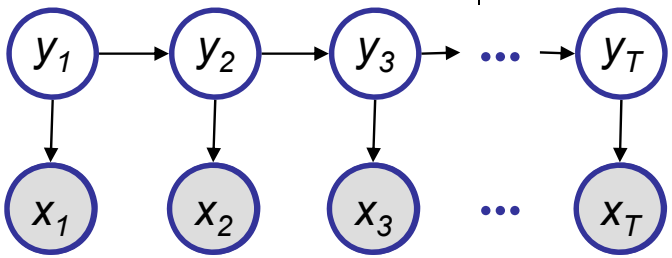
Daniel Jurafsky and James H. Martin, Part-of-speech tagging (slides)

Content

4 Graphical models for sequence labeling
- Maximum entropy Markov models (MEMMs)
- Conditional random fields (CRFs)

Hidden Markov Model revisit





- Transition probabilities between any two states

$$p(y_t^j = 1 \mid y_{t-1}^i = 1) = a_{i,j},$$

or $$p(y_t \mid y_{t-1}^i = 1) \sim \text{Multinomial}(a_{i,1}, a_{i,2}, \ldots, a_{i,M}), \forall i \in \mathbb{I}.$$

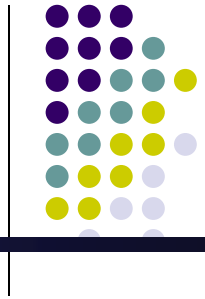
- Start probabilities

$$p(y_1) \sim \text{Multinomial}(\pi_1, \pi_2, \ldots, \pi_M).$$

- Emission probabilities associated with each state

$$p(x_t \mid y_t^i = 1) \sim \text{Multinomial}(b_{i,1}, b_{i,2}, \ldots, b_{i,K}), \forall i \in \mathbb{I}.$$

or in general: $$p(x_t \mid y_t^i = 1) \sim \text{f}(\cdot \mid \theta_i), \forall i \in \mathbb{I}.$$



Inference (review)

- Forward algorithm

$$\alpha_t^k \overset{\text{def}}{=} \mu_{t-1\to t}(k) = P(x_1,\ldots,x_{t-1},x_t,y_t^k=1)$$

$$\alpha_t^k = p(x_t \mid y_t^k=1)\sum_i \alpha_{t-1}^i a_{i,k}$$

- Backward algorithm

$$\beta_t^k = \sum_i a_{k,i}\, p(x_{t+1} \mid y_{t+1}^i = 1)\beta_{t+1}^i$$

$$\beta_t^k \overset{\text{def}}{=} \mu_{t-1\leftarrow t}(k) = P(x_{t+1},\ldots,x_T \mid y_t^k=1)$$

$$\gamma_t^i \overset{\text{def}}{=} p(y_t^i=1 \mid x_{1:T}) \propto \alpha_t^i\beta_t^i = \sum_j \xi_t^{i,j}$$

$$\xi_t^{i,j} \overset{\text{def}}{=} p(y_t^i=1, y_{t+1}^j=1, x_{1:T})$$

$$\propto \mu_{t-1\to t}(y_t^i=1)\mu_{t\leftarrow t+1}(y_{t+1}^j=1)\,p(x_{t+1}\mid y_{t+1})\,p(y_{t+1}\mid y_t)$$

$$\xi_t^{i,j} = \alpha_t^i\beta_{t+1}^j a_{i,j}\, p(x_{t+1}\mid y_{t+1}^i=1)$$

The matrix-vector form:

$$B_t(i) \overset{\text{def}}{=} p(x_t \mid y_t^i = 1)$$

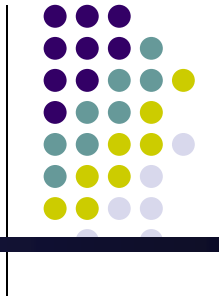
$$A(i,j) \overset{\text{def}}{=} p(y_{t+1}^j = 1 \mid y_t^i = 1)$$

$$\alpha_t = \left(A^T\alpha_{t-1}\right).*B_t$$

$$\beta_t = A\left(\beta_{t+1}.*B_{t+1}\right)$$

$$\xi_t = \left(\alpha_t\left(\beta_{t+1}.*B_{t+1}\right)^T\right).*A$$

$$\gamma_t = \alpha_t .*\beta_t$$



Learning HMM

- **Supervised learning**: estimation when the "right answer" is known
  - **Examples:**

    GIVEN: a genomic region $x = x_1 \quad x_{1,000,000}$ where we have good (experimental) annotations of the CpG islands

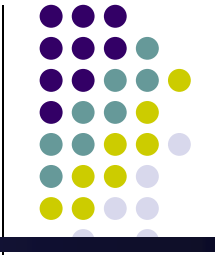
    GIVEN: the casino player allows us to observe him one evening, as he changes dice and produces 10,000 rolls

- **Unsupervised learning**: estimation when the "right answer" is unknown
  - **Examples:**

    GIVEN: the porcupine genome; we don't know how frequent are the CpG islands there, neither do we know their composition

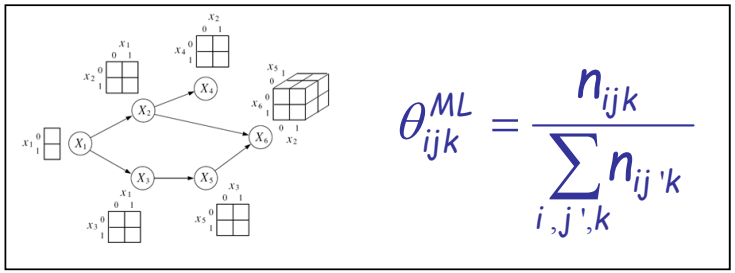
    GIVEN: 10,000 rolls of the casino player, but we don't see when he changes dice

- **QUESTION:** Update the parameters θ of the model to maximize $P(x|\theta)$ -- Maximal likelihood (ML) estimation



Learning HMM: two scenarios

- Supervised learning: if only we knew the true state path then ML parameter estimation would be trivial
  - E.g., recall that for complete observed tabular BN:

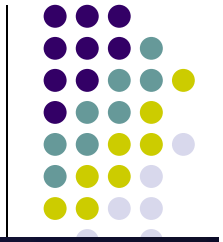


$$\theta_{ijk}^{ML} = \frac{n_{ijk}}{\sum_{i,j',k} n_{ij'k}}$$

$$a_{ij}^{ML} = \frac{\#(i \to j)}{\#(i \to \bullet)} = \frac{\sum_n \sum_{t=2}^{T} y_{n,t-1}^{i} y_{n,t}^{j}}{\sum_n \sum_{t=2}^{T} y_{n,t-1}^{i}}$$

$$b_{ik}^{ML} = \frac{\#(i \to k)}{\#(i \to \bullet)} = \frac{\sum_n \sum_{t=1}^{T} y_{n,t}^{i} x_{n,t}^{k}}{\sum_n \sum_{t=1}^{T} y_{n,t}^{i}}$$

  - What if y is continuous? We can treat $\{(x_{n,t}, y_{n,t}) : t = 1:T, n = 1:N\}$ as $N \times T$ observations of, e.g., a GLIM, and apply learning rules for GLIM
- Unsupervised learning: when the true state path is unknown, we can fill in the missing values using inference recursions.
  - The Baum Welch algorithm (i.e., EM)
    - Guaranteed to increase the log likelihood of the model after each iteration
    - Converges to local optimum, depending on initial conditions



The Baum Welch algorithm

- The complete log likelihood

$$\ell_c(\boldsymbol{\theta};\mathbf{x},\mathbf{y}) = \log p(\mathbf{x},\mathbf{y}) = \log \prod_n \left(p(y_{n,1}) \prod_{t=2}^{T} p(y_{n,t} \mid y_{n,t-1}) \prod_{t=1}^{T} p(x_{n,t} \mid x_{n,t}) \right)$$

- The expected complete log likelihood

$$\left\langle \ell_c(\boldsymbol{\theta};\mathbf{x},\mathbf{y}) \right\rangle = \sum_n \left(\left\langle y_{n,1}^i \right\rangle_{p(y_{n,1}|\mathbf{x}_n)} \log \pi_i \right) + \sum_n \sum_{t=2}^{T} \left(\left\langle y_{n,t-1}^i y_{n,t}^j \right\rangle_{p(y_{n,t-1},y_{n,t}|\mathbf{x}_n)} \log a_{i,j} \right) + \sum_n \sum_{t=1}^{T} \left(x_{n,t}^k \left\langle y_{n,t}^i \right\rangle_{p(y_{n,t}|\mathbf{x}_n)} \log b_{i,k} \right)$$

- EM
  - The E step

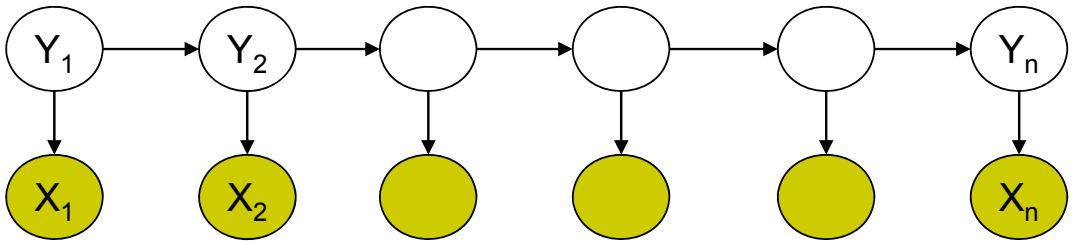
$$\gamma_{n,t}^i = \left\langle y_{n,t}^i \right\rangle = p(y_{n,t}^i = 1 \mid \mathbf{x}_n)$$

$$\xi_{n,t}^{i,j} = \left\langle y_{n,t-1}^i y_{n,t}^j \right\rangle = p(y_{n,t-1}^i = 1, y_{n,t}^j = 1 \mid \mathbf{x}_n)$$

  - The M step ("symbolically" identical to MLE)

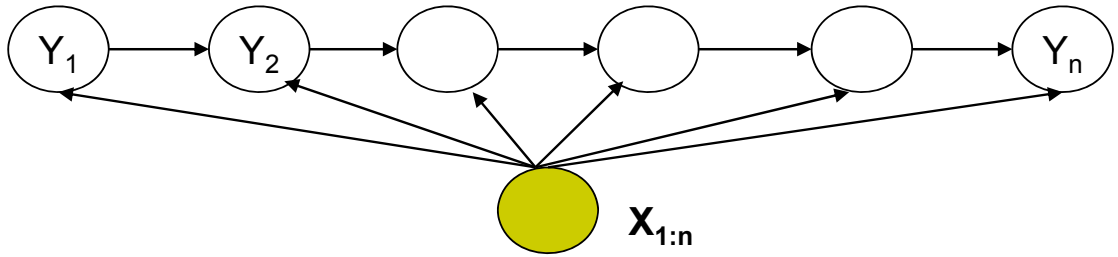
$$\pi_i^{ML} = \frac{\sum_n \gamma_{n,1}^i}{N} \qquad a_{ij}^{ML} = \frac{\sum_n \sum_{t=2}^{T} \xi_{n,t}^{i,j}}{\sum_n \sum_{t=1}^{T-1} \gamma_{n,t}^i} \qquad b_{ik}^{ML} = \frac{\sum_n \sum_{t=1}^{T} \gamma_{n,t}^i x_{n,t}^k}{\sum_n \sum_{t=1}^{T-1} \gamma_{n,t}^i}$$

Shortcomings of Hidden Markov Model (1): locality of features



- HMM models capture dependences between each state and only its corresponding observation
  - NLP example: In a sentence segmentation task, each segmental state may depend not just on a single word (and the adjacent segmental stages), but also on the (non-local) features of the whole line such as line length, indentation, amount of white space, etc.
- Mismatch between learning objective function and prediction objective function
  - HMM learns a joint distribution of states and observations P(**Y**, **X**), but in a prediction task, we need the conditional probability P(**Y**|**X**)

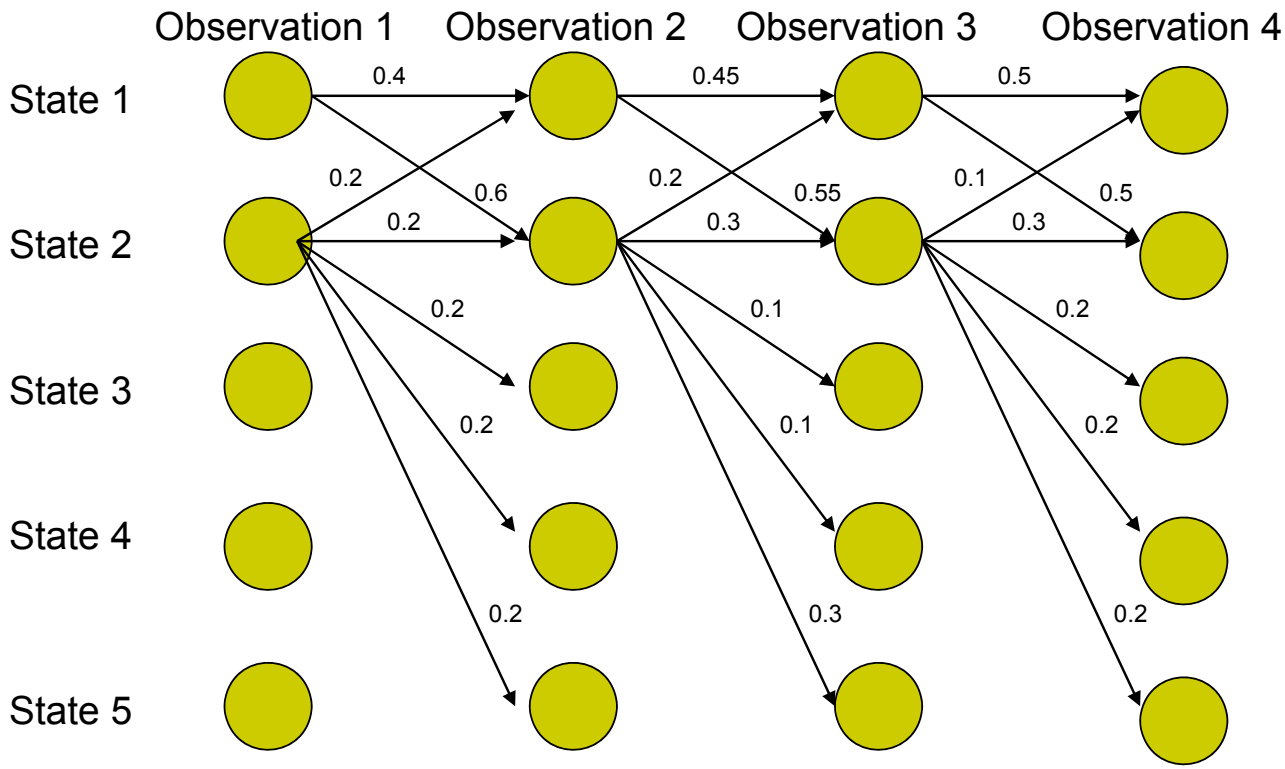
Solution:
Maximum Entropy Markov Model (MEMM)



$$P(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}) = \prod_{i=1}^{n} P(y_i|y_{i-1},\mathbf{x}_{1:n}) = \prod_{i=1}^{n} \frac{\exp(\mathbf{w}^T\mathbf{f}(y_i,y_{i-1},\mathbf{x}_{1:n}))}{Z(y_{i-1},\mathbf{x}_{1:n})}$$

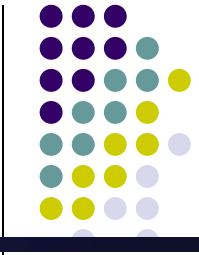
- Models dependence between each state and the full observation sequence explicitly
  - More expressive than HMMs
- Discriminative model
  - Completely ignores modeling P(**X**): saves modeling effort
  - Learning objective function consistent with predictive function: P(**Y**|**X**)

Then, shortcomings of MEMM (and HMM) (2): the Label bias problem

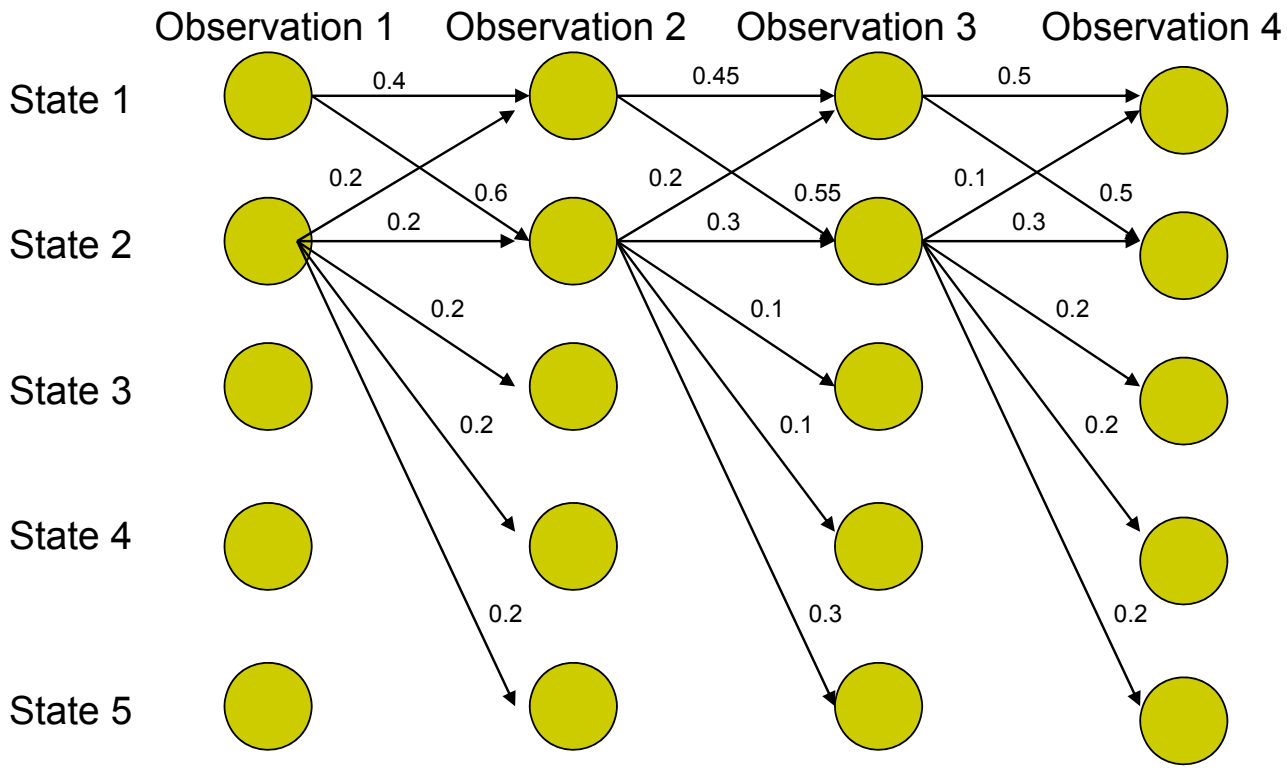


What the local transition probabilities say:

- State 1 almost always prefers to go to state 2
- State 2 almost always prefer to stay in state 2

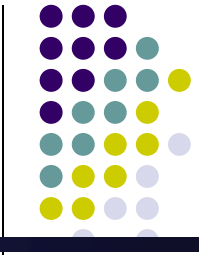


MEMM: the Label bias problem

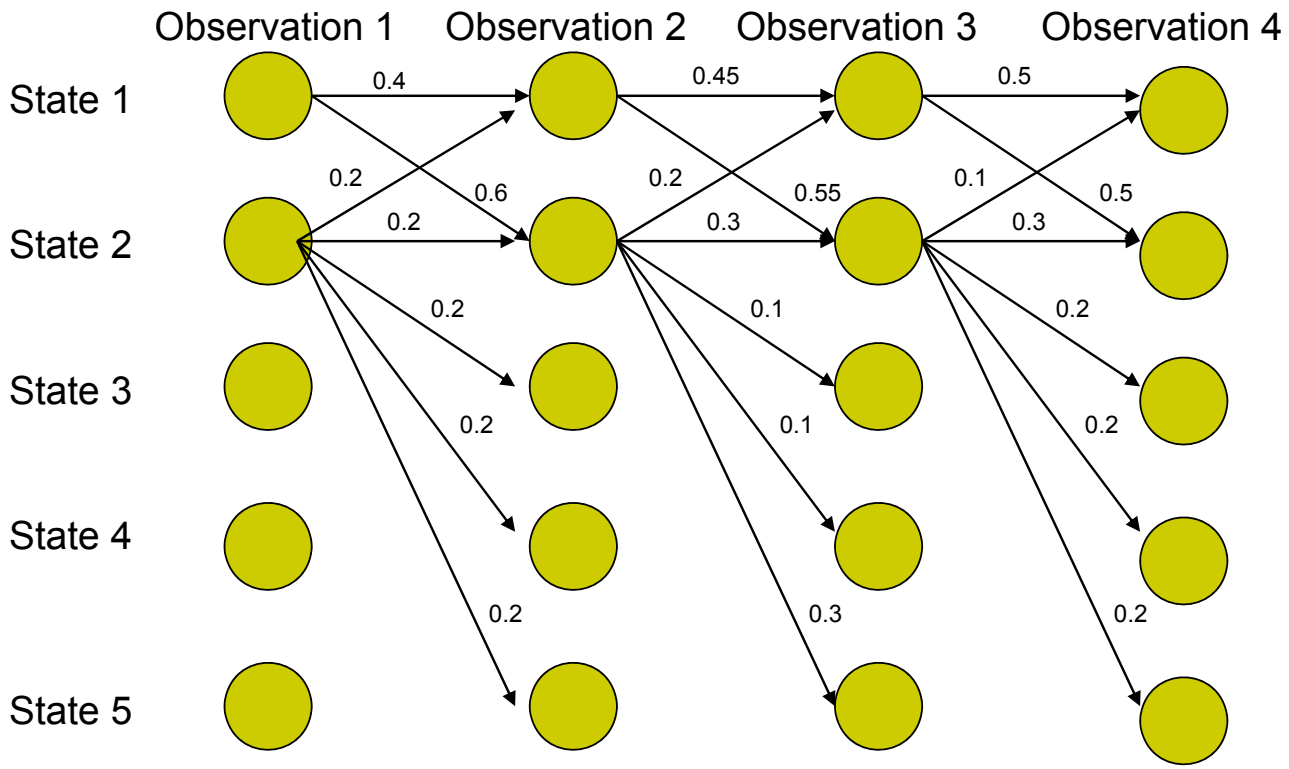


Probability of path 1-> 1-> 1-> 1:

- 0.4 x 0.45 x 0.5 = 0.09



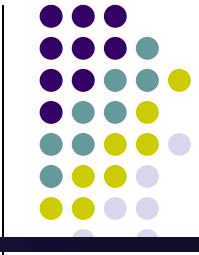
MEMM: the Label bias problem



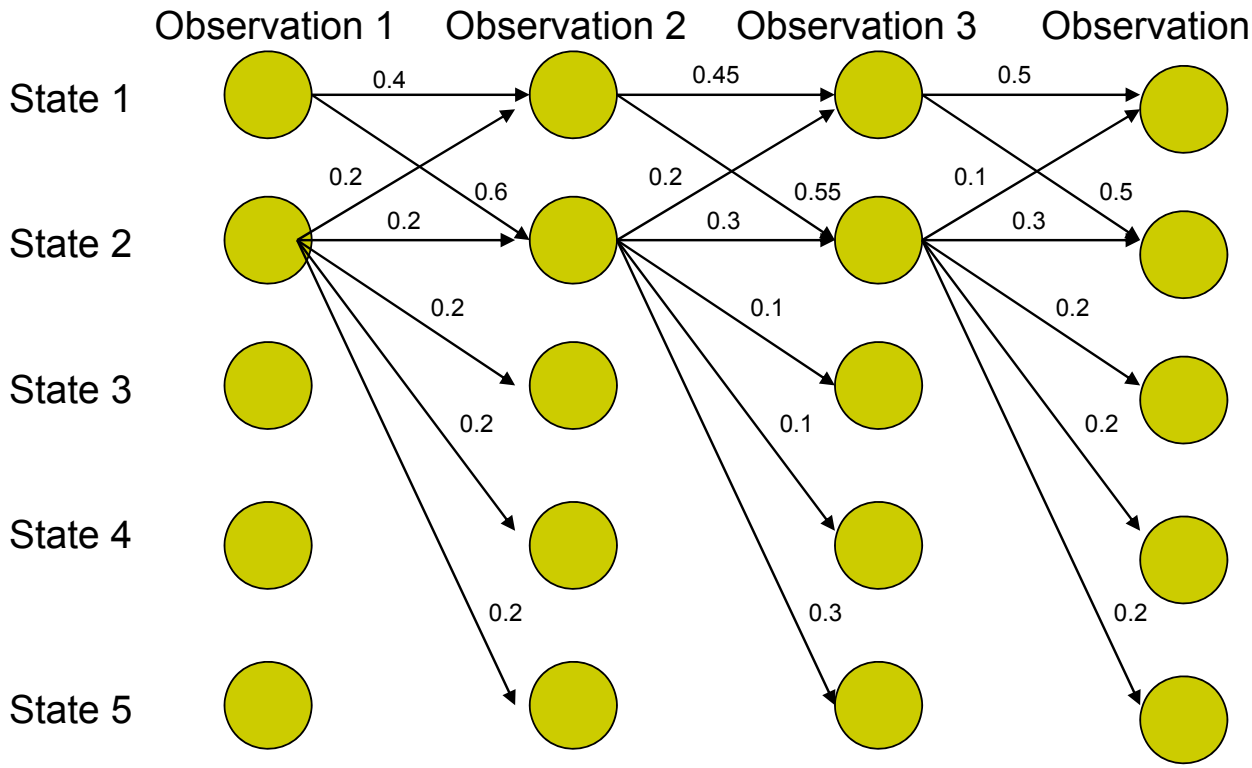
Probability of path 2->2->2->2 :

- 0.2 X 0.3 X 0.3 = 0.018

Other paths:
1-> 1-> 1-> 1: 0.09



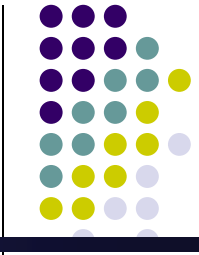
MEMM: the Label bias problem



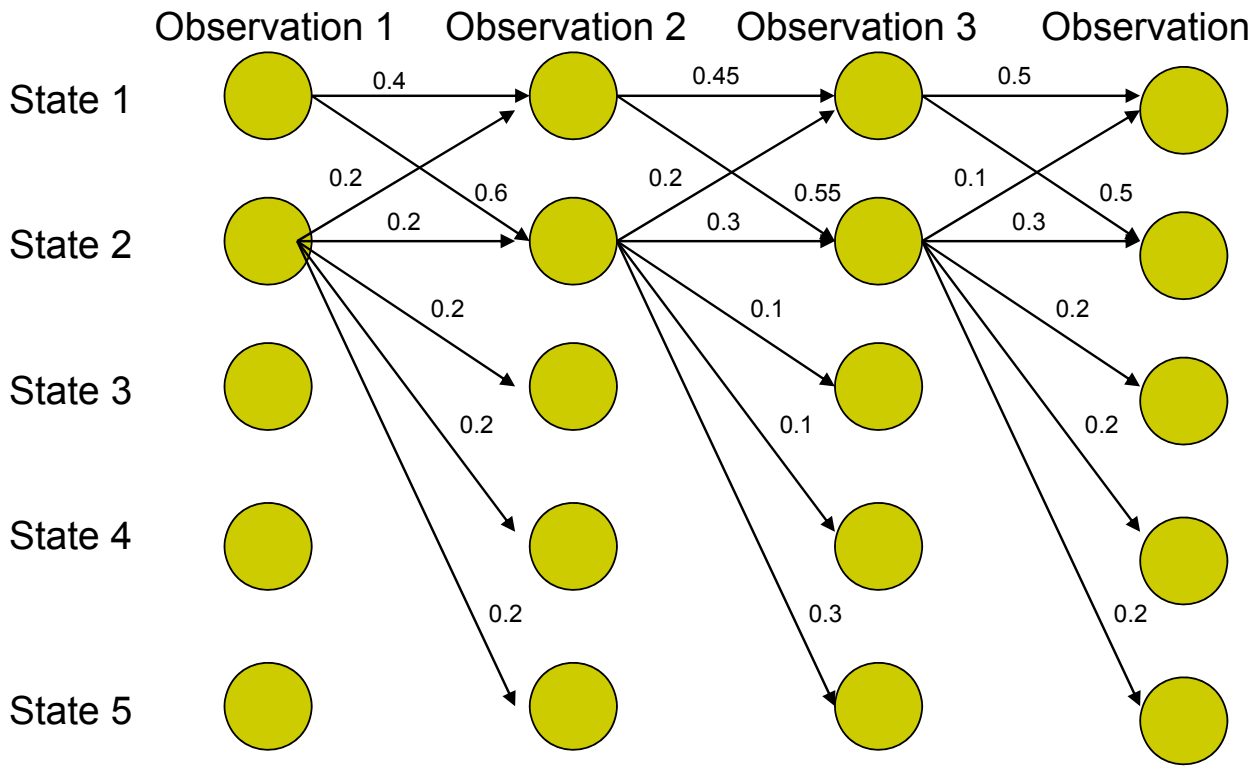
Probability of path 1->2->1->2:

- 0.6 X 0.2 X 0.5 = 0.06

Other paths:
1->1->1->1: 0.09
2->2->2->2: 0.018



MEMM: the Label bias problem

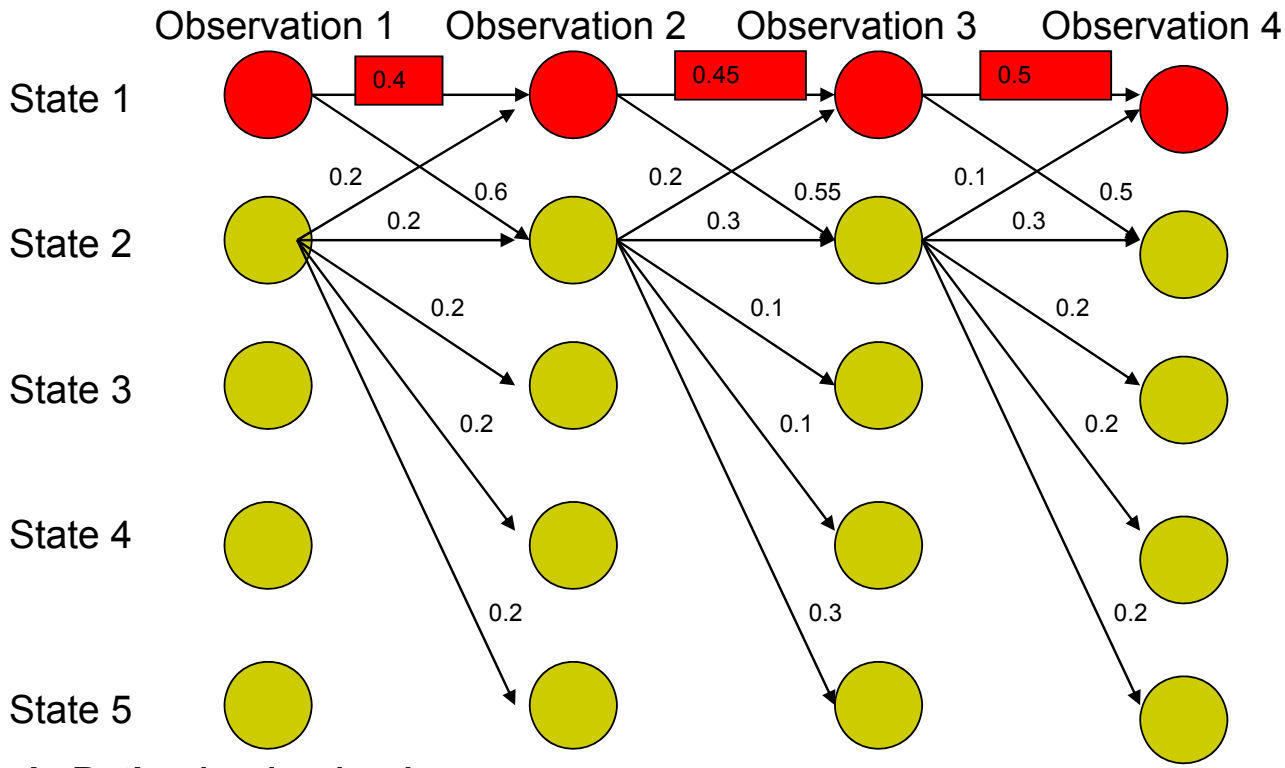


Probability of path 1->1->2->2:

- 0.4 X 0.55 X 0.3 = 0.066

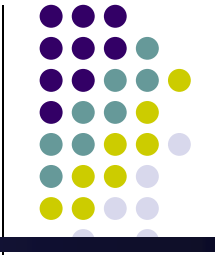
Other paths:
1->1->1->1: 0.09
2->2->2->2: 0.018
1->2->1->2: 0.06

MEMM: the Label bias problem

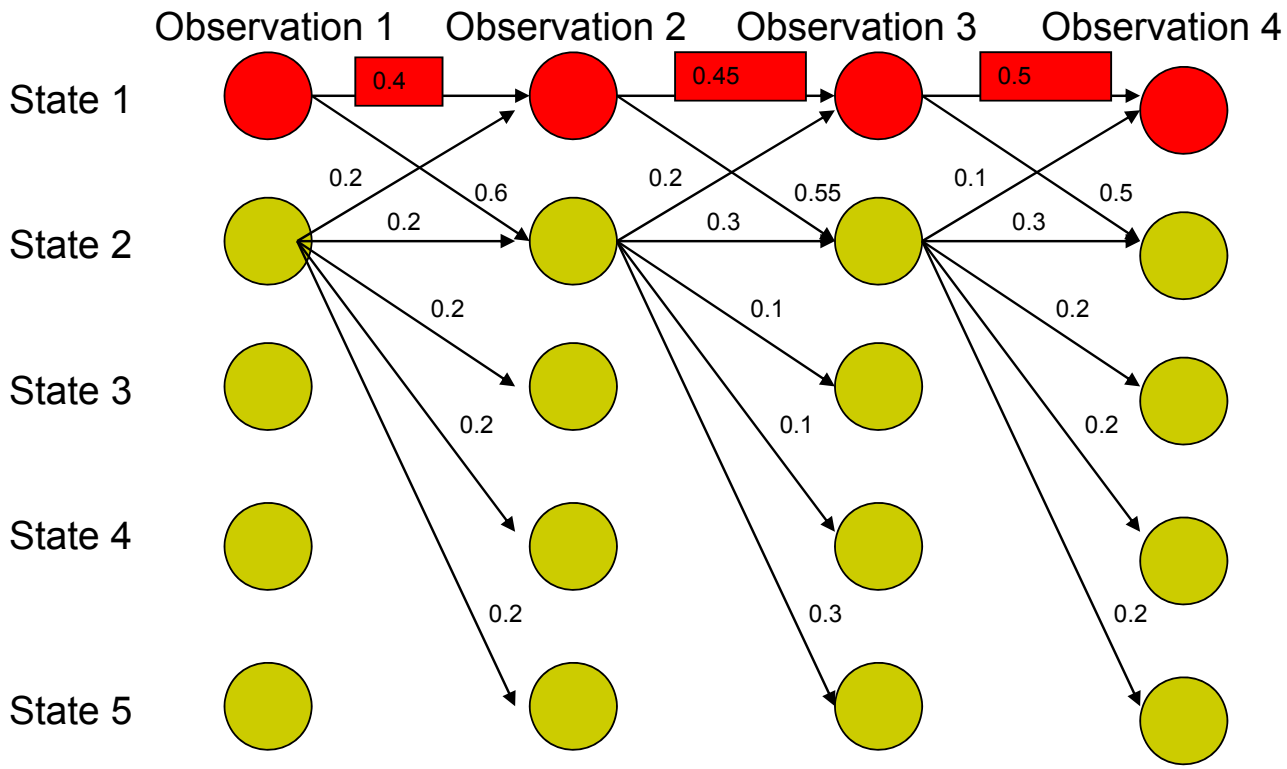


Most Likely Path: 1-> 1-> 1-> 1

- Although **locally** it seems state 1 wants to go to state 2 and state 2 wants to remain in state 2.
- **why?**



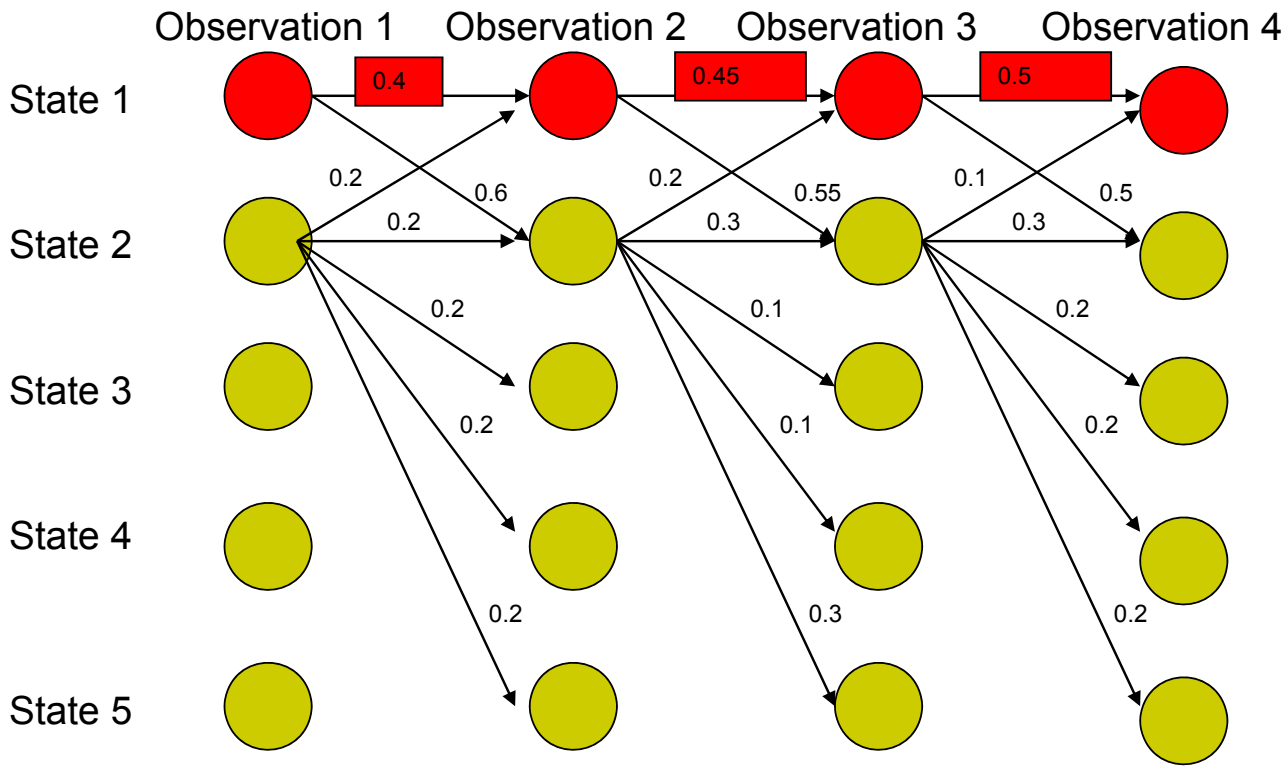
MEMM: the Label bias problem



Most Likely Path: 1-> 1-> 1-> 1

- State 1 has only two transitions but state 2 has 5:
  - Average transition probability from state 2 is lower

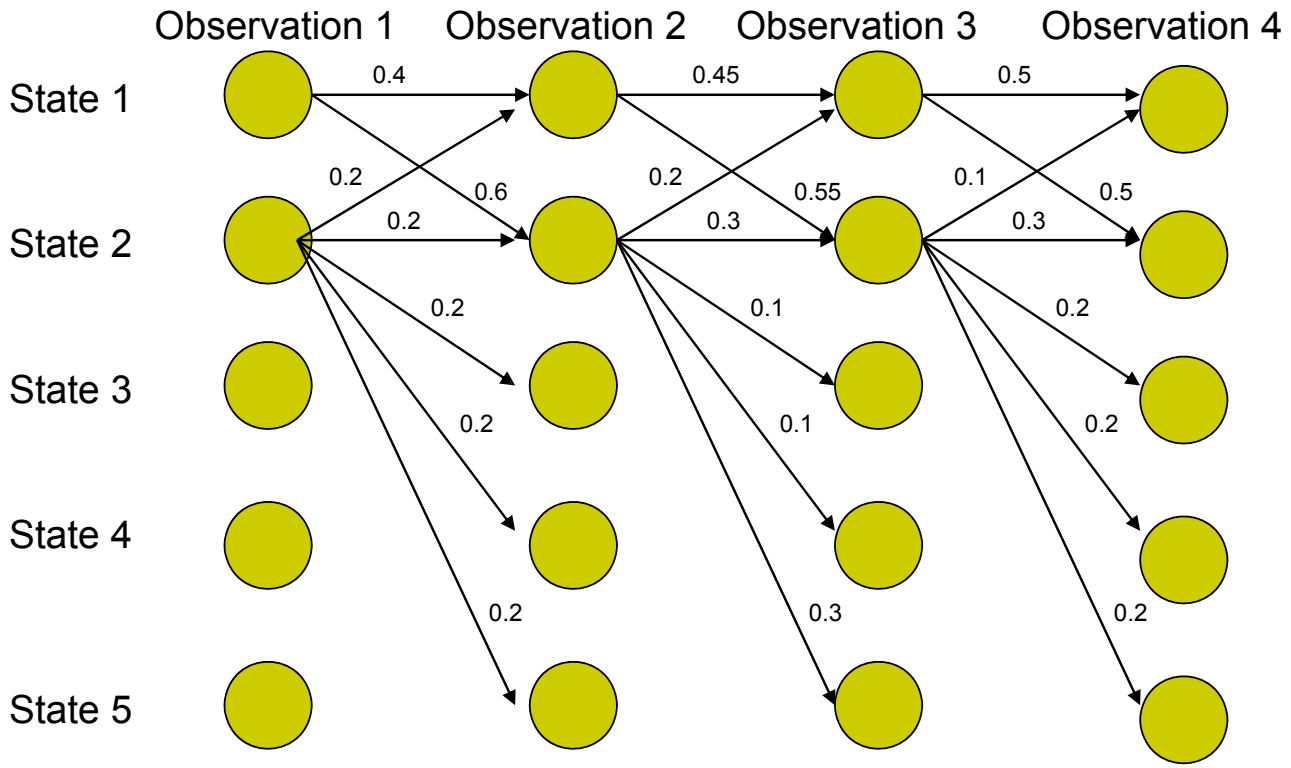
MEMM: the Label bias problem



Label bias problem in MEMM:

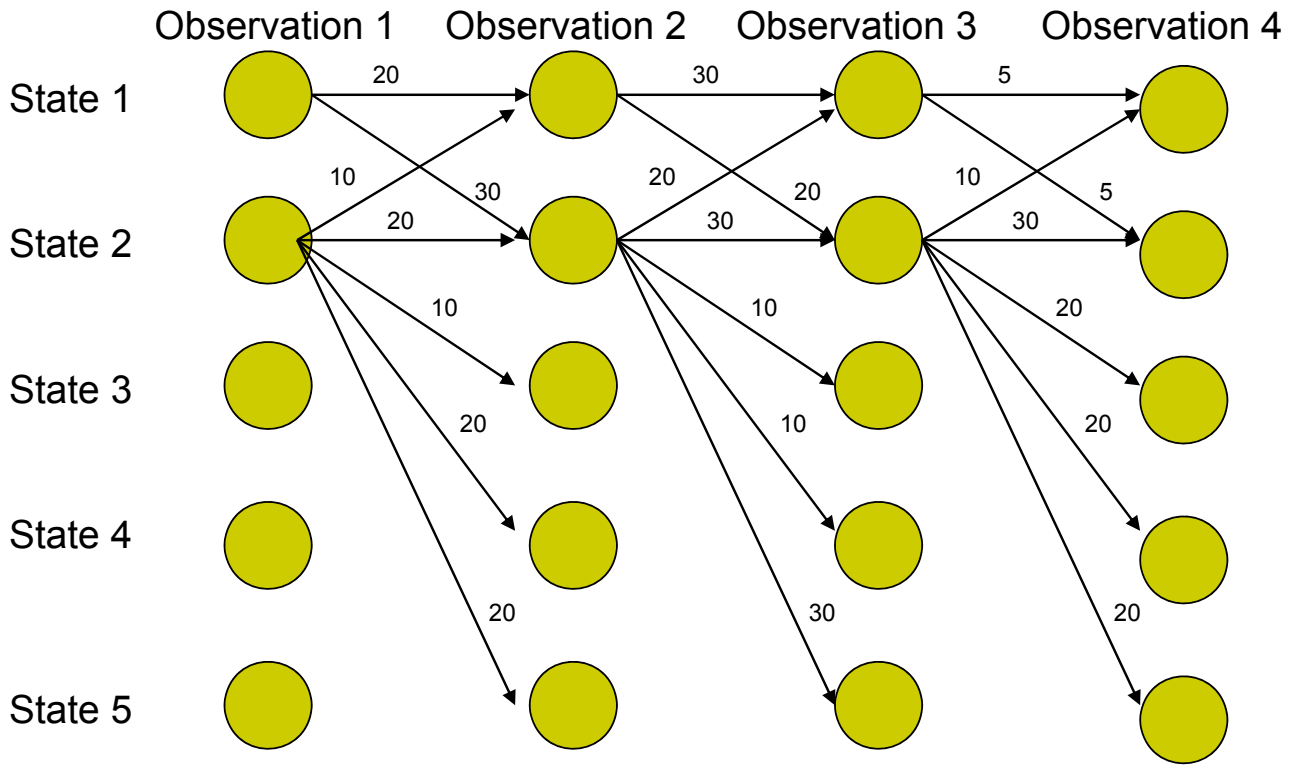
- Preference of states with lower number of transitions over others

Solution:
Do not normalize probabilities locally



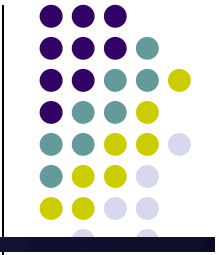
From local probabilities .

Solution: Do not normalize probabilities locally

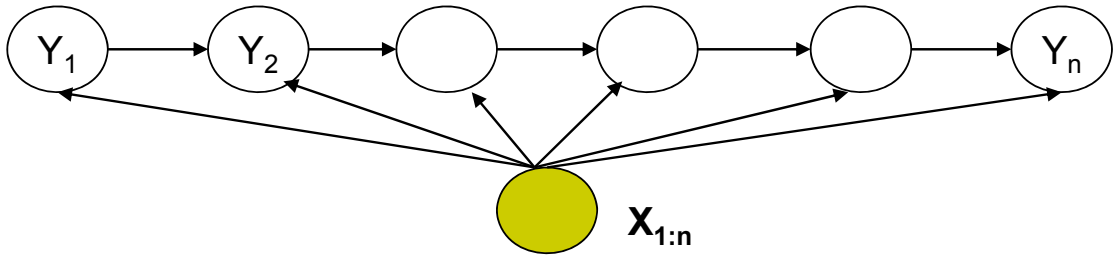


From local probabilities to local potentials

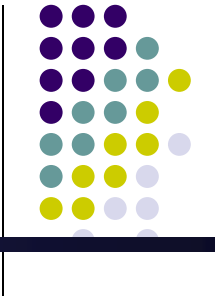
- States with lower transitions do not have an unfair advantage!



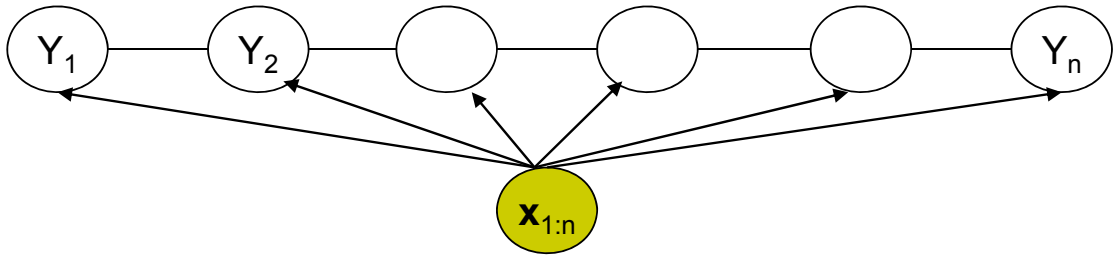
From MEMM .



$$P(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}) = \prod_{i=1}^{n} P(y_i|y_{i-1},\mathbf{x}_{1:n}) = \prod_{i=1}^{n} \frac{\exp(\mathbf{w}^T\mathbf{f}(y_i,y_{i-1},\mathbf{x}_{1:n}))}{Z(y_{i-1},\mathbf{x}_{1:n})}$$

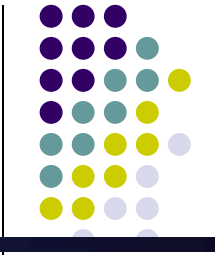


From MEMM to CRF



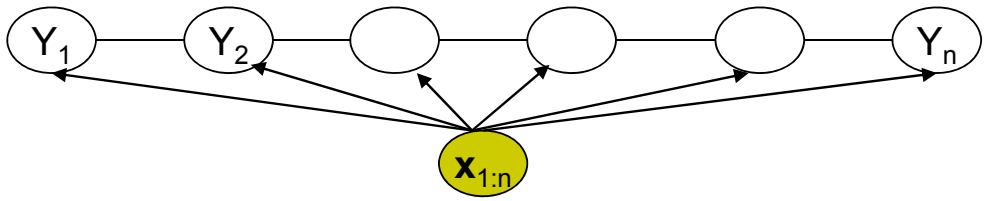
$$P(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}) = \frac{1}{Z(\mathbf{x}_{1:n})}\prod_{i=1}^{n}\phi(y_i, y_{i-1}, \mathbf{x}_{1:n}) = \frac{1}{Z(\mathbf{x}_{1:n}, \mathbf{w})}\prod_{i=1}^{n}\exp(\mathbf{w}^T\mathbf{f}(y_i, y_{i-1}, \mathbf{x}_{1:n}))$$

- CRF is a partially directed model
  - Discriminative model like MEMM
  - Usage of global normalizer Z(**x**) overcomes the label bias problem of MEMM
  - Models the dependence between each state and the entire observation sequence (like MEMM)



Conditional Random Fields

- General parametric form:



$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x},\lambda,\mu)} \exp\Big(\sum_{i=1}^{n}\Big(\sum_{k}\lambda_k f_k(y_i, y_{i-1}, \mathbf{x}) + \sum_{l}\mu_l g_l(y_i, \mathbf{x})\Big)\Big)$$

$$= \frac{1}{Z(\mathbf{x},\lambda,\mu)} \exp\Big(\sum_{i=1}^{n}\big(\lambda^T \mathbf{f}(y_i, y_{i-1}, \mathbf{x}) + \mu^T \mathbf{g}(y_i, \mathbf{x})\big)\Big)$$

$$\text{where } Z(\mathbf{x},\lambda,\mu) = \sum_{\mathbf{y}} \exp\Big(\sum_{i=1}^{n}\big(\lambda^T \mathbf{f}(y_i, y_{i-1}, \mathbf{x}) + \mu^T \mathbf{g}(y_i, \mathbf{x})\big)\Big)$$

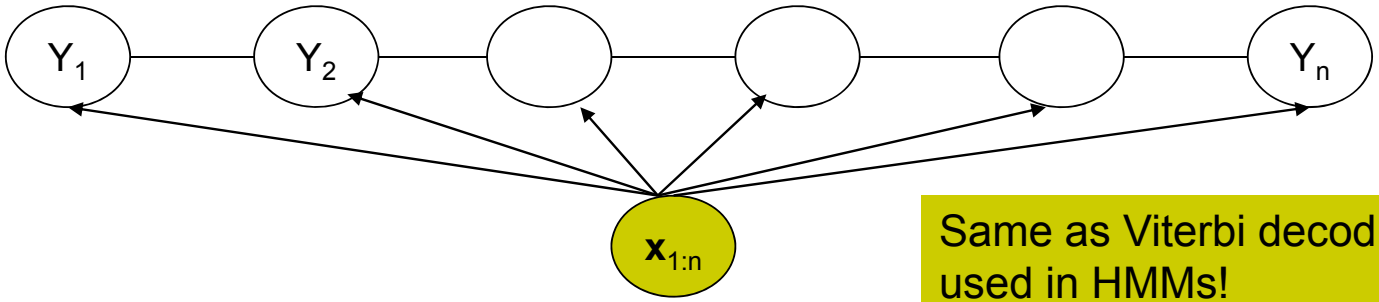
CRFs: Inference

- Given CRF parameters λ and μ, find the $\mathbf{y}^*$ that maximizes $P(\mathbf{y}|\mathbf{x})$

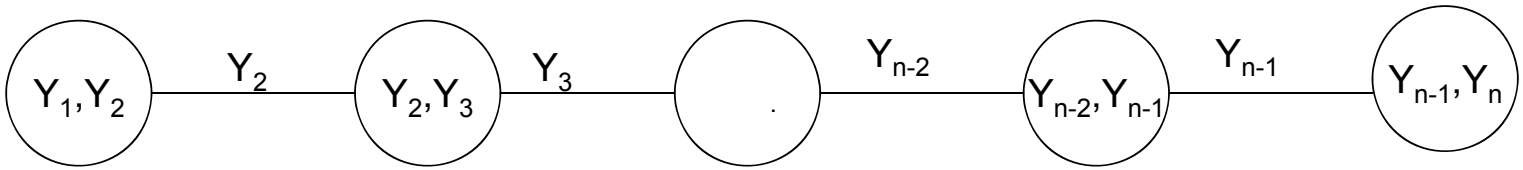
$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \exp\left(\sum_{i=1}^{n} (\lambda^T \mathbf{f}(y_i, y_{i-1}, \mathbf{x}) + \mu^T \mathbf{g}(y_i, \mathbf{x}))\right)$$

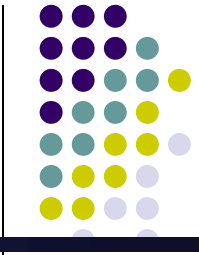
  - Can ignore $Z(\mathbf{x})$ because it is not a function of $\mathbf{y}$

- Run the max-product algorithm on the junction-tree of CRF:



Same as Viterbi decoding used in HMMs!





CRF learning

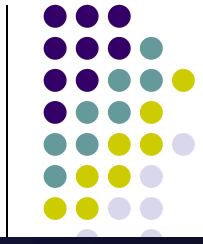
- Given $\{(\mathbf{x}_d, \mathbf{y}_d)\}_{d=1}^N$, find λ^*, μ^* such that

$$\begin{aligned}
\lambda*, \mu* &= \arg\max_{\lambda,\mu} L(\lambda, \mu) = \arg\max_{\lambda,\mu} \prod_{d=1}^{N} P(\mathbf{y}_d|\mathbf{x}_d, \lambda, \mu) \\
&= \arg\max_{\lambda,\mu} \prod_{d=1}^{N} \frac{1}{Z(\mathbf{x}_d, \lambda, \mu)} \exp(\sum_{i=1}^{n}(\lambda^T \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) + \mu^T \mathbf{g}(y_{d,i}, \mathbf{x}_d))) \\
&= \arg\max_{\lambda,\mu} \sum_{d=1}^{N}(\sum_{i=1}^{n}(\lambda^T \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) + \mu^T \mathbf{g}(y_{d,i}, \mathbf{x}_d)) - \log Z(\mathbf{x}_d, \lambda, \mu))
\end{aligned}$$

- Computing the gradient w.r.t λ:

Gradient of the log-partition function in an exponential family is the expectation of the sufficient statistics.

$$\nabla_\lambda L(\lambda, \mu) = \sum_{d=1}^{N}(\sum_{i=1}^{n} \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) - \sum_{\mathbf{y}}(P(\mathbf{y}|\mathbf{x_d}) \sum_{i=1}^{n} \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d)))$$



CRF learning

$$\nabla_\lambda L(\lambda,\mu) = \sum_{d=1}^{N}\left(\sum_{i=1}^{n}\mathbf{f}(y_{d,i},y_{d,i-1},\mathbf{x}_d) - \sum_{\mathbf{y}}\left(P(\mathbf{y}|\mathbf{x_d})\sum_{i=1}^{n}\mathbf{f}(y_i,y_{i-1},\mathbf{x}_d)\right)\right)$$

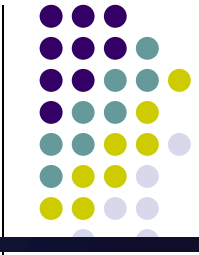
- Computing the model expectations:
  - Requires exponentially large number of summations: Is it intractable?

$$\sum_{\mathbf{y}}\left(P(\mathbf{y}|\mathbf{x}_d)\sum_{i=1}^{n}\mathbf{f}(y_i,y_{i-1},\mathbf{x}_d)\right) = \sum_{i=1}^{n}\left(\sum_{\mathbf{y}}\mathbf{f}(y_i,y_{i-1},\mathbf{x}_d)P(\mathbf{y}|\mathbf{x}_d)\right)$$

$$= \sum_{i=1}^{n}\sum_{y_i,y_{i-1}}\mathbf{f}(y_i,y_{i-1},\mathbf{x}_d)P(y_i,y_{i-1}|\mathbf{x}_d)$$

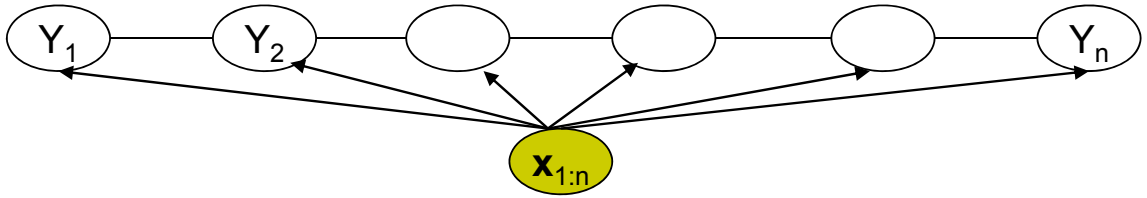
Expectation of **f** over the corresponding marginal probability of neighboring nodes!!

- Tractable!
  - Can compute marginals using the sum-product algorithm on the chain



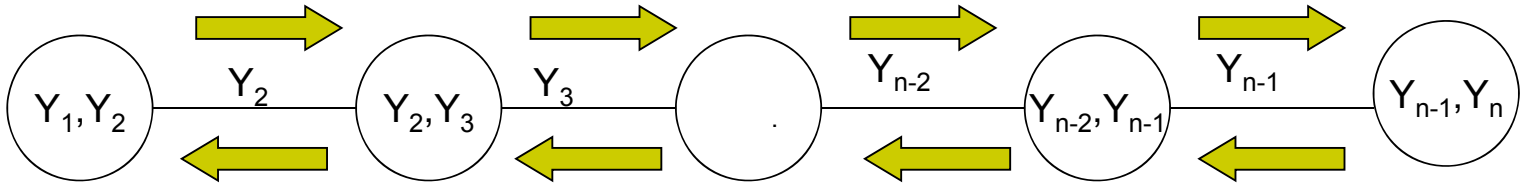
CRF learning

- Computing marginals using junction-tree calibration:



- Junction Tree Initialization:

$$\alpha^0(y_i, y_{i-1}) = \exp(\lambda^T \mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d) + \mu^T \mathbf{g}(y_i, \mathbf{x}_d))$$

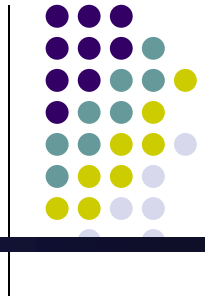


- After calibration:

$$P(y_i, y_{i-1}|\mathbf{x}_d) \propto \alpha(y_i, y_{i-1})$$

$$\Rightarrow P(y_i, y_{i-1}|\mathbf{x}_d) = \frac{\alpha(y_i, y_{i-1})}{\sum_{y_i, y_{i-1}} \alpha(y_i, y_{i-1})} = \alpha'(y_i, y_{i-1})$$

Also called forward-backward algorithm



CRF learning

- Computing feature expectations using calibrated potentials:

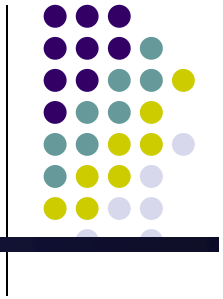
$$\sum_{y_i, y_{i-1}} \mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d) P(y_i, y_{i-1} | \mathbf{x}_d) = \sum_{y_i, y_{i-1}} \mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d) \alpha'(y_i, y_{i-1})$$

- Now we know how to compute $r_\lambda L(\lambda,\mu)$:

$$\begin{aligned}
\nabla_\lambda L(\lambda, \mu) &= \sum_{d=1}^{N} (\sum_{i=1}^{n} \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) - \sum_{\mathbf{y}} (P(\mathbf{y}|\mathbf{x_d}) \sum_{i=1}^{n} \mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d))) \\
&= \sum_{d=1}^{N} (\sum_{i=1}^{n} (\mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) - \sum_{y_i, y_{i-1}} \alpha'(y_i, y_{i-1}) \mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d)))
\end{aligned}$$

- Learning can now be done using gradient ascent:

$$\begin{aligned}
\lambda^{(t+1)} &= \lambda^{(t)} + \eta \nabla_\lambda L(\lambda^{(t)}, \mu^{(t)}) \\
\mu^{(t+1)} &= \mu^{(t)} + \eta \nabla_\mu L(\lambda^{(t)}, \mu^{(t)})
\end{aligned}$$



CRF learning

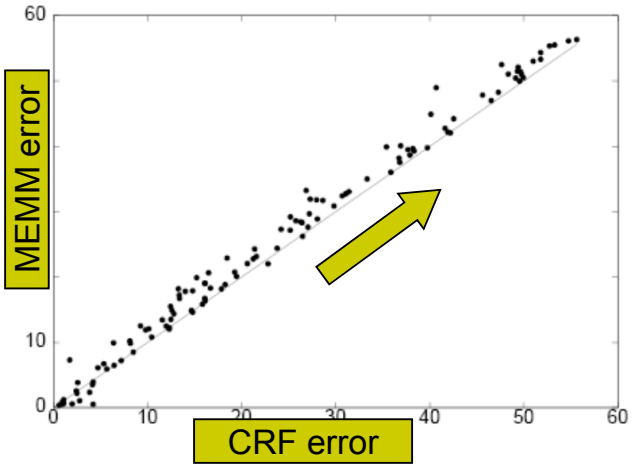
- In practice, we use a Gaussian Regularizer for the parameter vector to improve generalizability

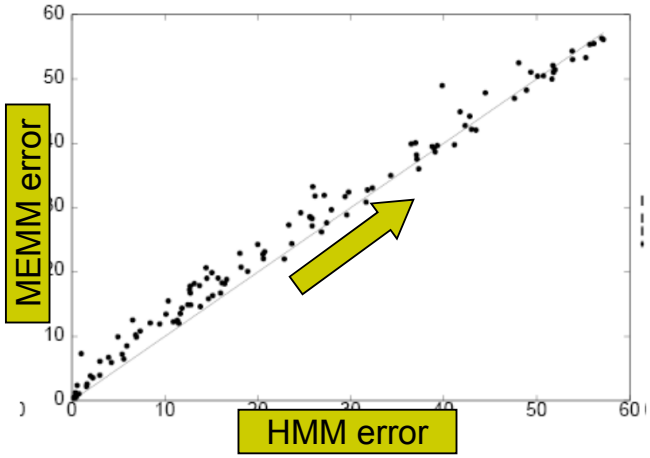
$$\lambda*, \mu* \quad = \quad \arg\max_{\lambda,\mu} \sum_{d=1}^{N} \log P(\mathbf{y}_d|\mathbf{x}_d, \lambda, \mu) - \frac{1}{2\sigma^2}(\lambda^T\lambda + \mu^T\mu)$$

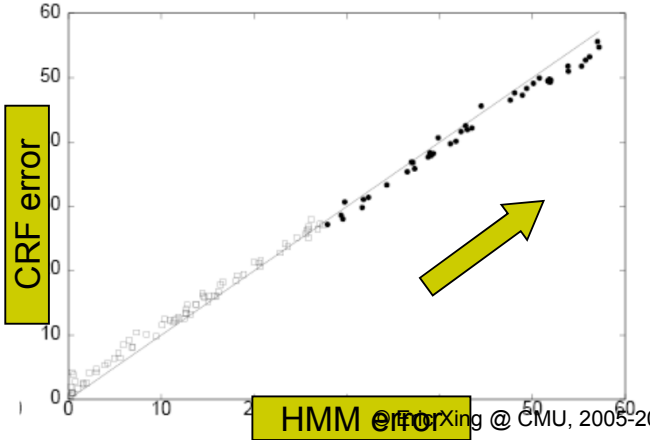
- In practice, gradient ascent has very slow convergence
  - Alternatives:
    - Conjugate Gradient method
    - Limited Memory Quasi-Newton Methods

CRFs: some empirical results

- Comparison of error rates on synthetic data

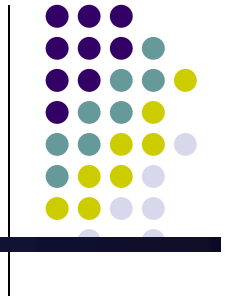






Data is increasingly higher order in the direction of arrow

CRFs achieve the lowest error rate for higher order data



CRFs: some empirical results

- Parts of Speech tagging

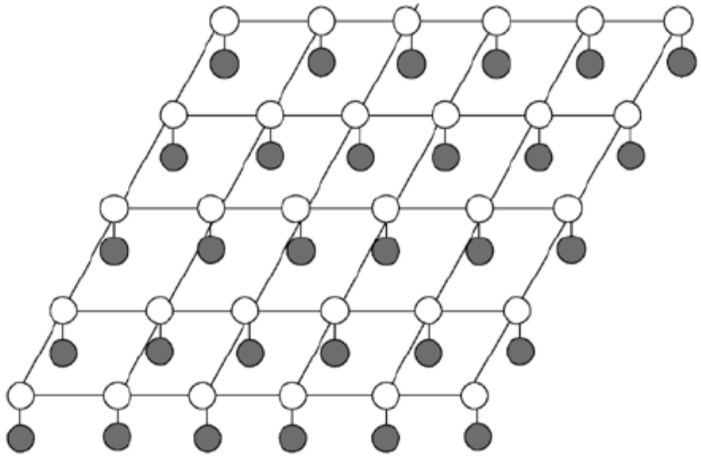
model	*error*	*oov error*
HMM	5.69%	45.99%
MEMM	6.37%	54.61%
CRF	5.55%	48.05%
MEMM$^+$	4.81%	26.99%
CRF$^+$	4.27%	23.76%

$^+$Using spelling features

- Using same set of features: HMM >=< CRF > MEMM
- Using additional overlapping features: CRF$^+$ > MEMM$^+$ >> HMM

Other CRFs

- So far we have discussed only 1-dimensional chain CRFs
  - Inference and learning: exact
- We could also have CRFs for arbitrary graph structure
  - E.g: Grid CRFs
  - Inference and learning no longer tractable
  - Approximate techniques used
    - MCMC Sampling
    - Variational Inference
    - Loopy Belief Propagation
  - We will discuss these techniques soon



Content

1. Sequence labeling problems
2. Word window classification
3. Hidden Markov models (HMMs)
4. Graphical models for sequence labeling