Predicting Structures: Conditional Models and Local Classifiers

CS 6355: Structured Prediction

THE UNIVERSITY OF UTAH

Outline

- Sequence models
- Hidden Markov models
  - Inference with HMM
  - Learning
- Conditional Models and Local Classifiers
- Global models
  - Conditional Random Fields
  - Structured Perceptron for sequences

Today’s Agenda

- Conditional models for predicting sequences

- Maximum Entropy Markov Models

Today’s Agenda

- Conditional models for predicting sequences

- Maximum Entropy Markov Models

HMM redux

- The independence assumption

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots, y_n) = P(y_1) \prod_{i=1}^{n-1} P(\, y_{i+1} \mid y_i \,) \prod_{i=1}^{n} P(\, x_i \mid y_i \,)$$

HMM redux

- The independence assumption

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots, y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1} \mid y_i) \prod_{i=1}^{n} P(x_i \mid y_i)$$

- Training via maximum likelihood

$$\max_{\pi, A, B} P(D \mid \pi, A, B) = \max_{\pi, A, B} \prod_{i} P(\mathbf{x}_i, \mathbf{y}_i \mid \pi, A, B)$$

We are optimizing joint likelihood of the input and the output for training

HMM redux

- The independence assumption

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots, y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1} \mid y_i) \prod_{i=1}^{n} P(x_i \mid y_i)$$

- Training via maximum likelihood

$$\max_{\pi,A,B} P(D \mid \pi, A, B) = \max_{\pi,A,B} \prod_i P(\mathbf{x}_i, \mathbf{y}_i \mid \pi, A, B)$$

We are optimizing joint likelihood of the input and the output for training

HMM redux

Probability of input given the prediction!

- The independence assumption

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots, y_n) = \underline{P(y_1) \prod_{i=1}^{n-1} P(y_{i+1} \mid y_i) \boxed{\prod_{i=1}^{n} P(x_i \mid y_i)}}$$

- Training via maximum likelihood

$$\max_{\pi, A, B} P(D \mid \pi, A, B) = \max_{\pi, A, B} \prod_i P(\mathbf{x}_i, \mathbf{y}_i \mid \pi, A, B)$$

We are optimizing joint likelihood of the input and the output for training

HMM redux

Probability of input given the prediction!

- The independence assumption

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots, y_n) = \underline{P(y_1) \prod_{i=1}^{n-1} P(y_{i+1} \mid y_i) \boxed{\prod_{i=1}^{n} P(x_i \mid y_i)}}$$

- Training via maximum likelihood

$$\max_{\pi,A,B} P(D \mid \pi, A, B) = \max_{\pi,A,B} \prod_i P(\mathbf{x}_i, \mathbf{y}_i \mid \pi, A, B)$$

We are optimizing joint likelihood of the input and the output for training

At prediction time, we only care about the probability of output given the input:

$$P(y_1, y_2, \cdots, y_n \mid x_1, x_2, \cdots, x_n)$$

Why not directly optimize this *conditional likelihood* instead?

Modeling next-state directly

- Instead of modeling the joint distribution $P(\mathbf{x}, \mathbf{y})$, focus on $P(\mathbf{y} \mid \mathbf{x})$ only
  - Which is what we care about eventually anyway

    (At least in this context)

- For sequences, different formulations
  - Maximum Entropy Markov Model [McCallum, et al 2000]

  (other names: discriminative/conditional Markov model, projection-based Markov model…)

Generative vs Discriminative models

- Generative models
  - learn P(x, y)
  - Characterize how the data is generated (both inputs and outputs)
  - Eg: Naïve Bayes, Hidden Markov Model

- Discriminative models
  - learn P(y | x)
  - Directly characterizes the decision boundary only
  - Eg: Logistic Regression, Conditional models (several names)

Generative vs Discriminative models

- Generative models
  - learn P(x, y)
  - Characterize how the data is generated (both inputs and outputs)
  - Eg: Naïve Bayes, Hidden Markov Model

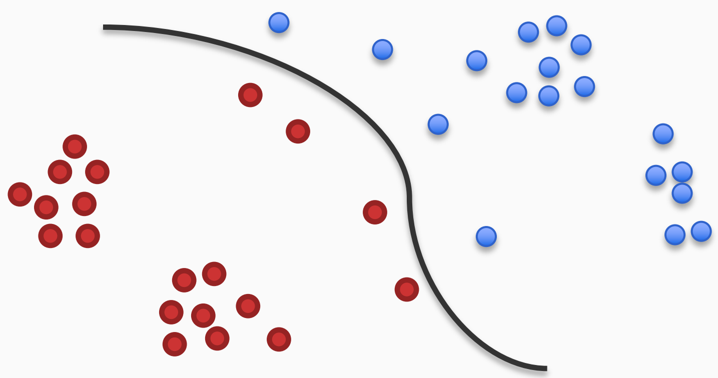
> A generative model tries to characterize the distribution of the inputs, a discriminative model doesn’t care

- Discriminative models
  - learn P(y | x)
  - Directly characterizes the decision boundary only
  - Eg: Logistic Regression, Conditional models (several names)

Generative vs Discriminative models

- Generative models
  - learn P(x, y)
  - Characterize how the data is generated (both inputs and outputs)
  - Eg: Naïve Bayes, Hidden Markov Model

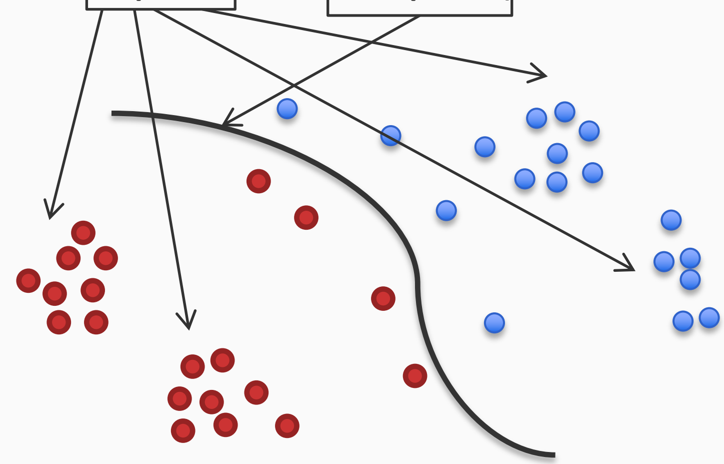
A generative model tries to characterize the distribution of the inputs, a discriminative model doesn't care



- Discriminative models
  - learn P(y | x)
  - Directly characterizes the decision boundary only
  - Eg: Logistic Regression, Conditional models (several names)

Generative vs Discriminative models

- Generative models
  - learn P(x, y)
  - Characterize how the data is generated (both inputs and outputs)
  - Eg: Naïve Bayes, Hidden Markov Model



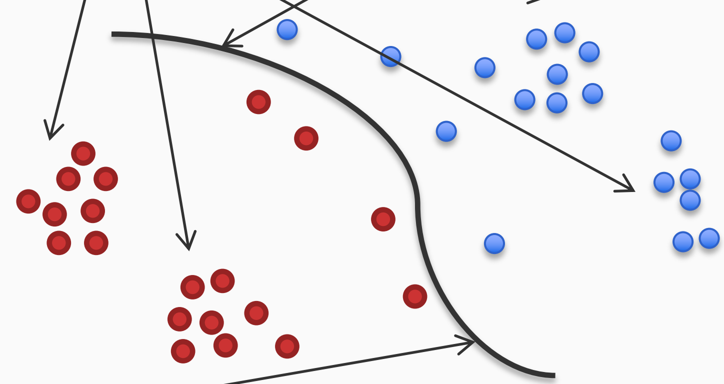
A generative model tries to characterize the distribution of the inputs, a discriminative model doesn't care

- Discriminative models
  - learn P(y | x)
  - Directly characterizes the decision boundary only
  - Eg: Logistic Regression, Conditional models (several names)

Generative vs Discriminative models

- Generative models
  - learn P(x, y)
  - Characterize how the data is generated (both inputs and outputs)
  - Eg: Naïve Bayes, Hidden Markov Model

A generative model tries to characterize the distribution of the inputs, a discriminative model doesn't care



- Discriminative models
  - learn P(y | x)
  - Directly characterizes the decision boundary only
  - Eg: Logistic Regression, Conditional models (several names)

HMM redux

Probability of input given the prediction!

- The independence assumption

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots, y_n) = \underline{P(y_1) \prod_{i=1}^{n-1} P(y_{i+1} \mid y_i) \prod_{i=1}^{n} P(x_i \mid y_i)}$$

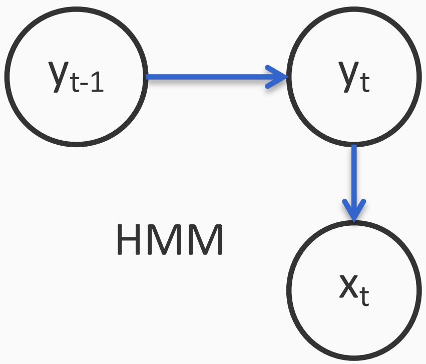
- Training via maximum likelihood

$$\max_{\pi,A,B} P(D \mid \pi, A, B) = \max_{\pi,A,B} \prod_i P(\mathbf{x}_i, \mathbf{y}_i \mid \pi, A, B)$$

We are optimizing joint likelihood of the input and the output for training

At prediction time, we only care about the probability of output given the input. Why not directly optimize this *conditional likelihood* instead?

Let's revisit the independence assumptions

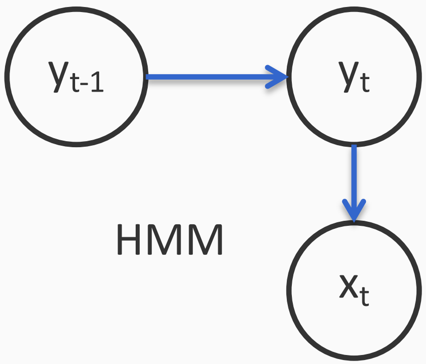


$$P(\, y_i \mid y_{i-1}, \text{anything else} \,) = P(\, y_i \mid y_{i-1} \,)$$

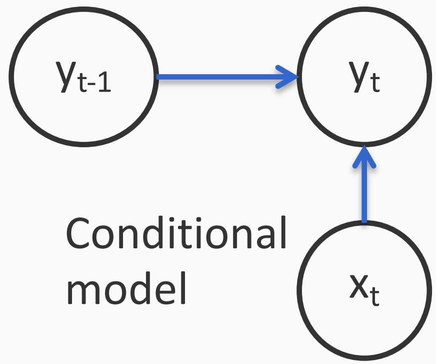
$$P(\, x_i \mid y_i, \text{anything else}\,) = P(\, x_i \mid y_i \,)$$

Another independence assumption

$$P(y_i \mid y_{i-1}, y_{i-2}, \cdots, x_i, x_{i-1}, \cdots) = P(y_i \mid y_{i-1}, x_i)$$



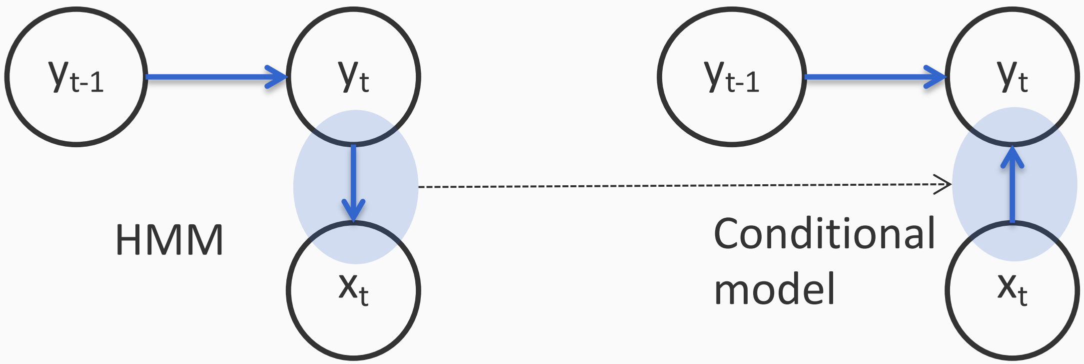
HMM



Conditional model

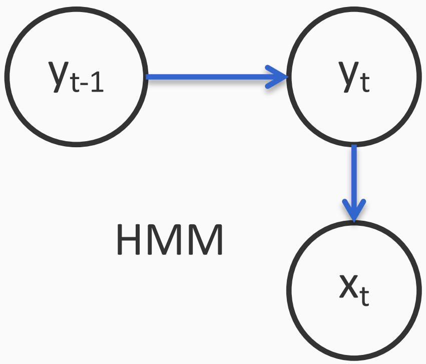
Another independence assumption

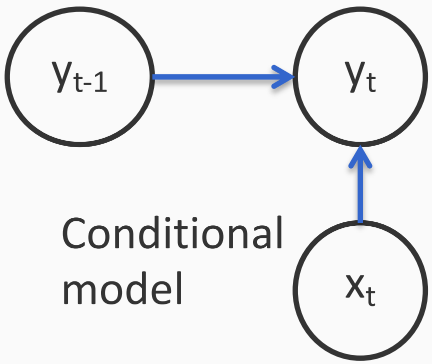
$$P(y_i \mid y_{i-1}, y_{i-2}, \cdots, x_i, x_{i-1}, \cdots) = P(y_i \mid y_{i-1}, x_i)$$



Another independence assumption

$$P(y_i \mid y_{i-1}, y_{i-2}, \cdots, x_i, x_{i-1}, \cdots) = P(y_i \mid y_{i-1}, x_i)$$



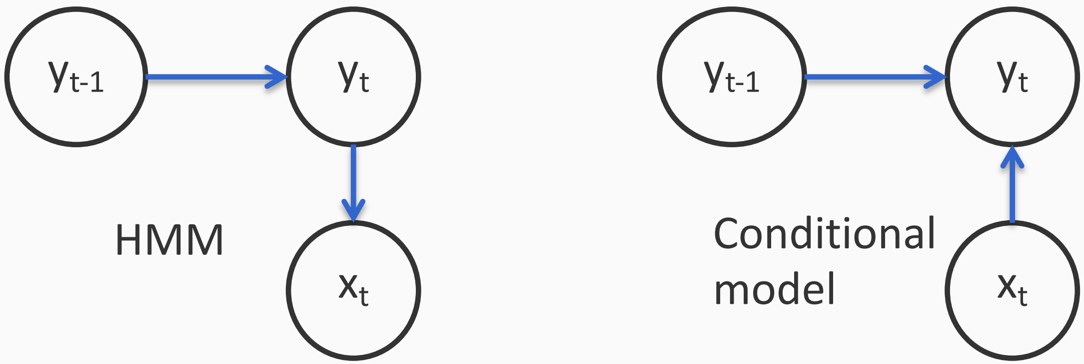


This assumption lets us write the conditional probability of the entire output sequence $\mathbf{y}$ as

$$P(\mathbf{y} \mid \mathbf{x}) = \prod_i P(y_i \mid y_{i-1}, x_i)$$

Another independence assumption

$$P(y_i \mid y_{i-1}, y_{i-2}, \cdots, x_i, x_{i-1}, \cdots) = P(y_i \mid y_{i-1}, x_i)$$



This assumption lets us write the conditional probability of the entire output sequence $\mathbf{y}$ as

$$P(\mathbf{y} \mid \mathbf{x}) = \prod_i \boxed{P(y_i \mid y_{i-1}, x_i)}$$

We need to learn this function

Modeling $P(y_i \mid y_{i-1}, x_i)$

This is a multiclass classifier whose input is the pair y_{i-1}, x_i, and the label is the state y_i

Different approaches possible

1. Train a *maximum entropy* classifier (i.e., a multiclass logistic regression classifier)

2. Or, ignore the fact that we are predicting a probability, we only care about maximizing some *score.* Train any multiclass classifier, using say the perceptron algorithm

Modeling $P(y_i \mid y_{i-1}, x_i)$

This is a multiclass classifier whose input is the pair y_{i-1}, x_i, and the label is the state y_i

Different approaches possible

1. Train a *maximum entropy* classifier (i.e., a multiclass logistic regression classifier)

2. Or, ignore the fact that we are predicting a probability, we only care about maximizing some *score.* Train any multiclass classifier, using say the perceptron algorithm

In any case, we can use any features from y_{i-1} and x_i. This was not possible with HMMs.

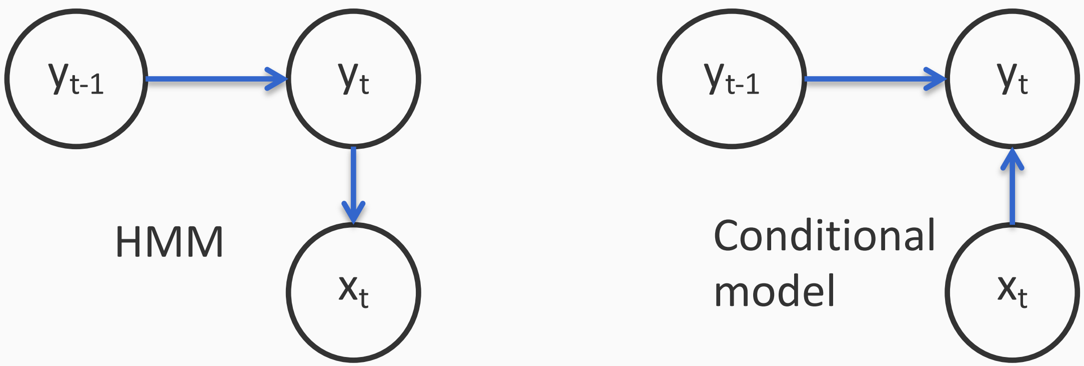
Today’s Agenda

- Conditional models for predicting sequences

- Maximum Entropy Markov Models

The next-state model

$$P(y_i \mid y_{i-1}, y_{i-2}, \cdots, x_i, x_{i-1}, \cdots) = P(y_i \mid y_{i-1}, x_i)$$



This assumption lets us write the conditional probability of the output as

$$P(\mathbf{y}|\mathbf{x}) = \prod_i P(y_i|y_{i-1}, x_i)$$

We need to learn this function

Modeling $\cancel{P(y_i \mid y_{i-1}, x_i)}$ $P(y_i \mid y_{i-1}, \mathbf{x})$

Different approaches possible

1. Train a *maximum entropy* classifier (i.e., a multiclass logistic regression classifier)

2. Or, ignore the fact that we are predicting a probability, we only care about maximizing some *score.* Train any multiclass classifier, using say the perceptron algorithm

Modeling $\cancel{P(y_i \mid y_{i-1}, x_i)}$ $P(y_i \mid y_{i-1}, \mathbf{x})$

Different approaches possible

1. Train a *maximum entropy* classifier (i.e., a multiclass logistic regression classifier)

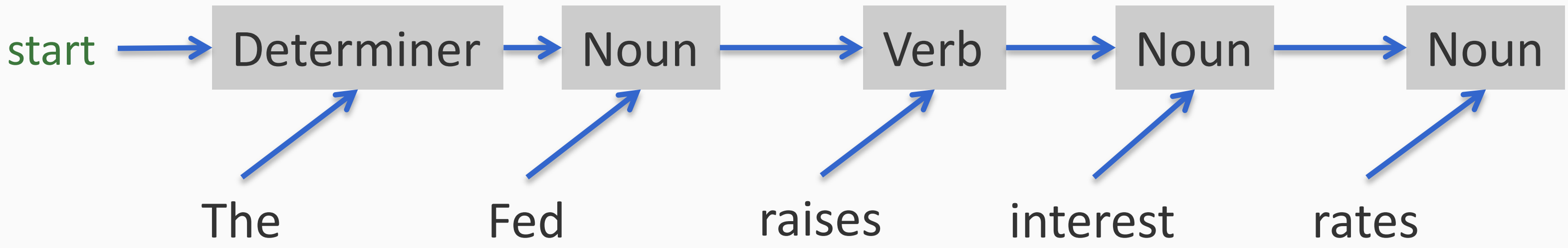
2. Or, ignore the fact that we are predicting a probability, we only care about maximizing some *score.* Train any multiclass classifier, using say the perceptron algorithm

For both cases*:*

- Use rich features that depend on input and previous state
- We can increase the dependency to arbitrary neighboring x_i's
  - Eg. Neighboring words influence this words POS tag

Maximum Entropy Markov Model

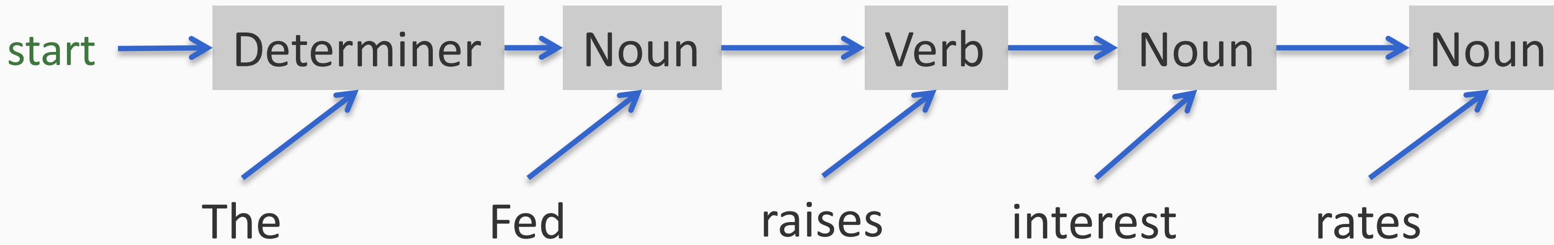
Goal: Compute P(**y** | **x**) $$P(y_i|y_{i-1}, \mathbf{x}) \propto \exp\left(\mathbf{w}^T \phi(\mathbf{x}, i, y_i, y_{i-1})\right)$$



The prediction task: Using the entire input and the current label, predict the next label

Maximum Entropy Markov Model

Goal: Compute P(**y** | **x**) $$P(y_i|y_{i-1},\mathbf{x}) \propto \exp\left(\mathbf{w}^T\phi(\mathbf{x},i,y_i,y_{i-1})\right)$$



word

Caps

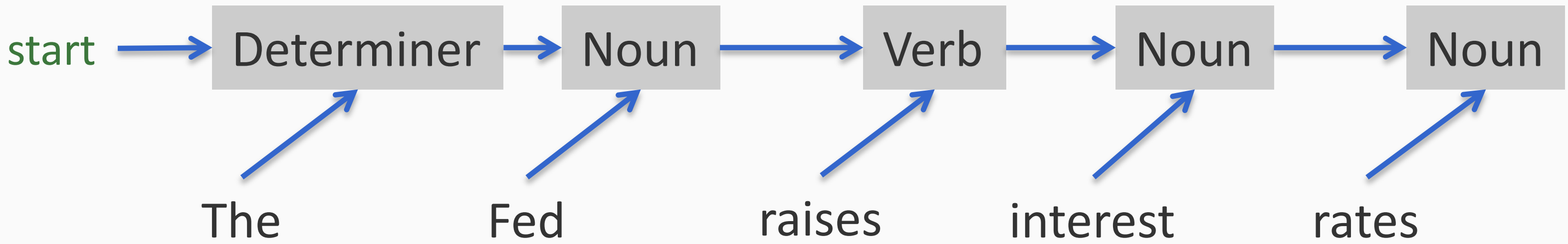
-es

Previous

To model the probability, first, we need to define features for the current classification problem

Maximum Entropy Markov Model

Goal: Compute P(**y** | **x**) $\quad P(y_i | y_{i-1}, \mathbf{x}) \propto \exp\left(\mathbf{w}^T \phi(\mathbf{x}, i, y_i, y_{i-1})\right)$



$$\begin{array}{ll} \textit{word} & \\ \textit{Caps} & \\ \textit{-es?} & \\ \textit{Previous} & \end{array} \begin{bmatrix} \text{The} \\ \text{Y} \\ \text{N} \\ \text{start} \end{bmatrix}$$

$\phi(\mathbf{x}, 0, \text{start})$

Maximum Entropy Markov Model

Goal: Compute P(**y** | **x**)

$$P(y_i | y_{i-1}, \mathbf{x}) \propto \exp\left(\mathbf{w}^T \phi(\mathbf{x}, i, y_i, y_{i-1})\right)$$

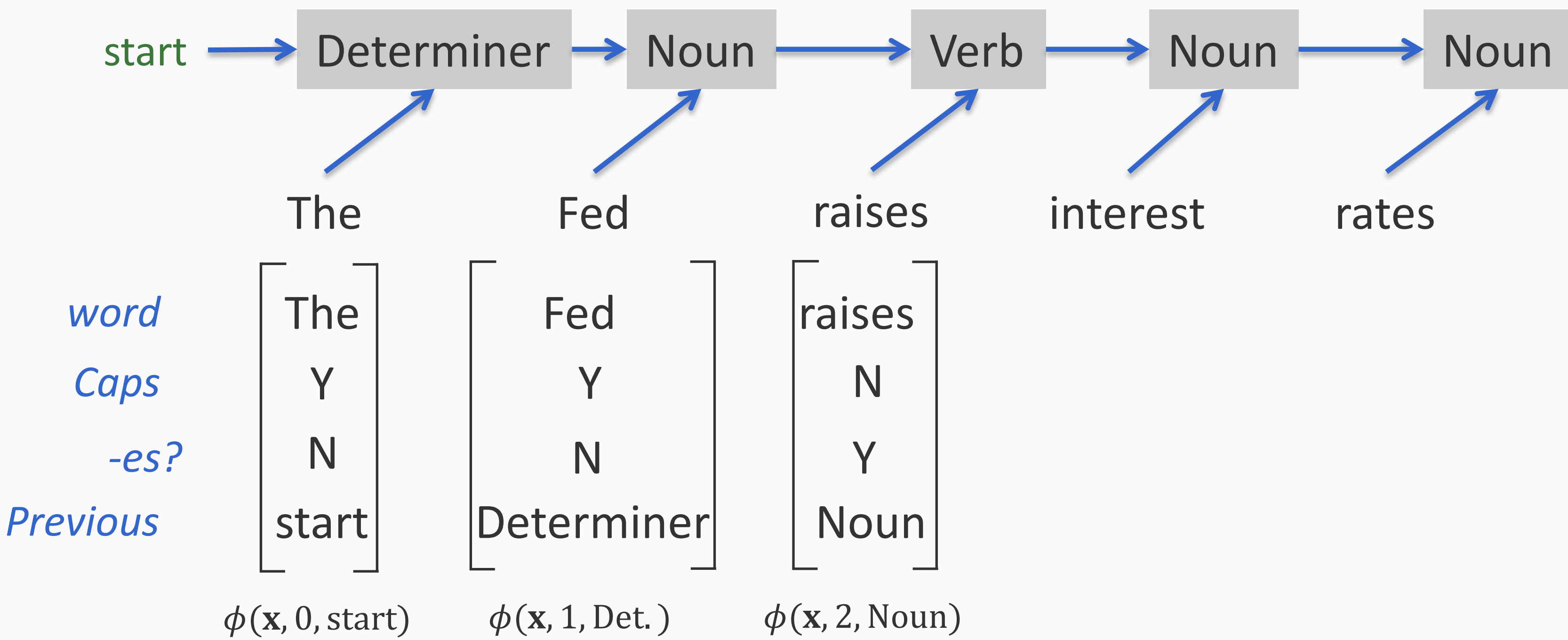
start → Determiner → Noun → Verb → Noun → Noun

The → Determiner; Fed → Noun; raises → Verb; interest → Noun; rates → Noun

	$\phi(\mathbf{x}, 0, \text{start})$	$\phi(\mathbf{x}, 1, \text{Det.})$
word	The	Fed
Caps	Y	Y
-es?	N	N
Previous	start	Determiner

Maximum Entropy Markov Model

Goal: Compute P(**y** | **x**) $$P(y_i|y_{i-1}, \mathbf{x}) \propto \exp\left(\mathbf{w}^T \phi(\mathbf{x}, i, y_i, y_{i-1})\right)$$



Maximum Entropy Markov Model

Goal: Compute P(**y** | **x**)

$$P(y_i|y_{i-1},\mathbf{x}) \propto \exp\left(\mathbf{w}^T\phi(\mathbf{x},i,y_i,y_{i-1})\right)$$

start → Determiner → Noun → Verb → Noun → Noun

The Fed raises interest rates

	$\phi(\mathbf{x},0,\text{start})$	$\phi(\mathbf{x},1,\text{Det.})$	$\phi(\mathbf{x},2,\text{Noun})$	$\phi(\mathbf{x},3,\text{Verb})$
word	The	Fed	raises	interest
Caps	Y	Y	N	N
-es?	N	N	Y	N
Previous	start	Determiner	Noun	Verb

Maximum Entropy Markov Model

Goal: Compute P(**y** | **x**)

$$P(y_i | y_{i-1}, \mathbf{x}) \propto \exp\left(\mathbf{w}^T \phi(\mathbf{x}, i, y_i, y_{i-1})\right)$$

start → Determiner → Noun → Verb → Noun → Noun

The Fed raises interest rates

word	The	Fed	raises	interest	rates
Caps	Y	Y	N	N	N
-es?	N	N	Y	N	N
Previous	start	Determiner	Noun	Verb	Noun
	$\phi(\mathbf{x}, 0, \text{start})$	$\phi(\mathbf{x}, 1, \text{Det.})$	$\phi(\mathbf{x}, 2, \text{Noun})$	$\phi(\mathbf{x}, 3, \text{Verb})$	$\phi(\mathbf{x}, 4, \text{Noun})$

Maximum Entropy Markov Model

Goal: Compute P(**y** | **x**)

$$P(y_i | y_{i-1}, \mathbf{x}) \propto \exp\left(\mathbf{w}^T \phi(\mathbf{x}, i, y_i, y_{i-1})\right)$$

start → Determiner → Noun → Verb → Noun → Noun

	The	Fed	raises	interest	rates
word	The	Fed	raises	interest	rates
Caps	Y	Y	N	N	N
-es?	N	N	Y	N	N
Previous	start	Determiner	Noun	Verb	Noun
	$\phi(\mathbf{x}, 0, \text{start})$	$\phi(\mathbf{x}, 1, \text{Det.})$	$\phi(\mathbf{x}, 2, \text{Noun})$	$\phi(\mathbf{x}, 3, \text{Verb})$	$\phi(\mathbf{x}, 4, \text{Noun})$

Can get very creative here

Maximum Entropy Markov Model

Goal: Compute P(**y** | **x**)

$$P(y_i | y_{i-1}, \mathbf{x}) \propto \exp\left(\mathbf{w}^T \phi(\mathbf{x}, i, y_i, y_{i-1})\right)$$

start → Determiner → Noun → Verb → Noun → Noun

The ↗ Determiner, Fed ↗ Noun, raises ↗ Verb, interest ↗ Noun, rates ↗ Noun

	$\phi(\mathbf{x}, 0, \text{start})$	$\phi(\mathbf{x}, 1, \text{Det.})$	$\phi(\mathbf{x}, 2, \text{Noun})$	$\phi(\mathbf{x}, 3, \text{Verb})$	$\phi(\mathbf{x}, 4, \text{Noun})$
word	The	Fed	raises	interest	rates
Caps	Y	Y	N	N	N
-es?	N	N	Y	N	N
Previous	start	Determiner	Noun	Verb	Noun

Can get very creative here

Compare to HMM: Only depends on the word and the previous tag

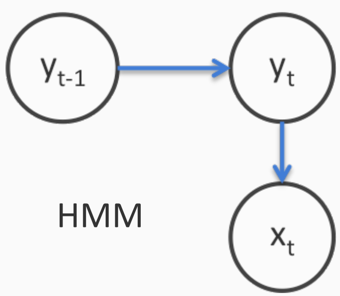
start → Determiner → Noun → Verb → Noun → Noun

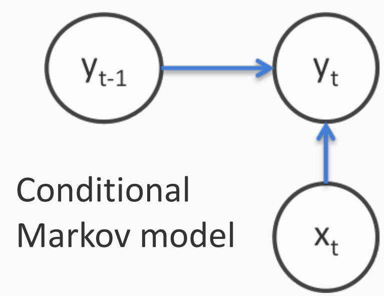
Determiner ↓ The, Noun ↓ Fed, Verb ↓ raises, Noun ↓ interest, Noun ↓ rates

Questions?

Using MEMM

- Training
  - Next-state predictor **locally** as maximum likelihood
    - Similar to any maximum entropy classifier
- Prediction/decoding
  - Modify the Viterbi algorithm for the new independence assumptions





$$\text{score}_i(s) = \max_{y_{i-1}} P(s|y_{i-1})P(x_i|s)\text{score}_{i-1}(y_{i-1})$$

$$\text{score}_i(s) = \max_{y_{i-1}} P(s|y_{i-1}, x_i)\text{score}_{i-1}(y_{i-1})$$

Generalization: Any multiclass classifier

- **Viterbi decoding**: we only need a score for each decision
  - So far, probabilistic classifiers

- In general, use any learning algorithm to build get a score for the label y_i given y_{i-1} and $\mathbf{x}$
  - Multiclass versions of perceptron, SVM
  - Just like MEMM, these allow arbitrary features to be defined

Exercise: Viterbi needs to be re-defined to work with sum of scores rather than the product of probabilities

Comparison to HMM

What we gain

1. Rich feature representation for inputs
    - Helps generalize better by thinking about properties of the input tokens rather than the entire tokens
    - Eg: If a word ends with –es, it might be a present tense verb (such as raises). Could be a feature; HMM cannot capture this

2. Discriminative predictor
    - Model P(**y** | **x**) rather than P(**y**, x)
    - *Joint* vs *conditional*

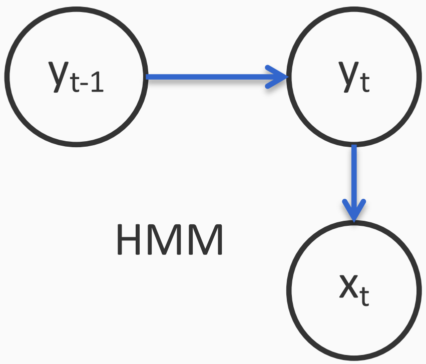
Questions?

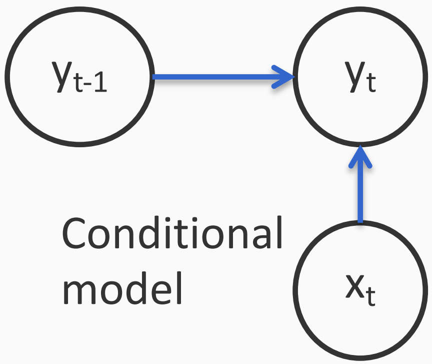
Outline

- Conditional models for predicting sequences
- Log-linear models for multiclass classification
- Maximum Entropy Markov Models
  - The Label Bias Problem

The next-state model for sequences

$$P(y_i|y_{i-1}, y_{i-2}, \cdots, x_i, x_{i-1}, \cdots) = P(y_i|y_{i-1}, x_i)$$





This assumption lets us write the conditional probability of the output as

$$P(\mathbf{y}|\mathbf{x}) = \prod_i P(y_i|y_{i-1}, x_i)$$

We need to train local multiclass classifiers that predicts the next state given the previous state and the input

…local classifiers → Label bias problem

Let's look at the independence assumption

$$P(y_i|y_{i-1}, y_{i-2}, \cdots, x_i, x_{i-1}, \cdots) = P(y_i|y_{i-1}, x_i)$$

"Next-state" classifiers are locally normalized

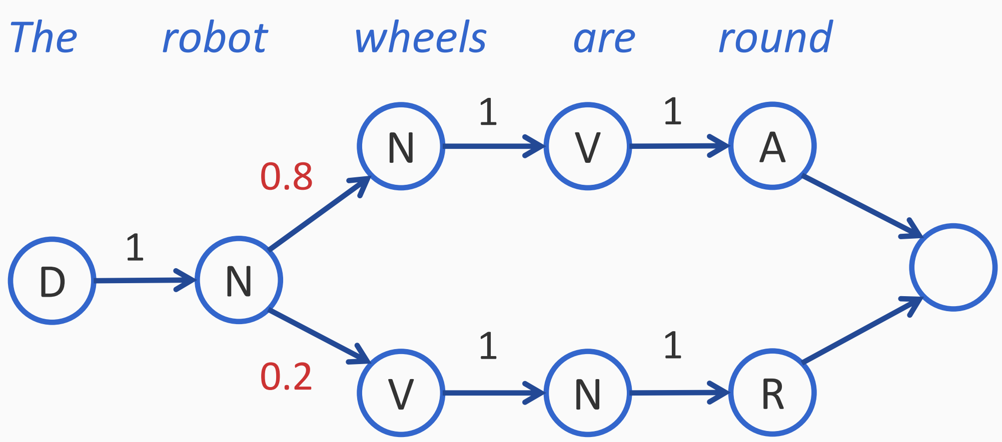
…local classifiers → Label bias problem

Let's look at the independence assumption

$$P(y_i|y_{i-1}, y_{i-2}, \cdots, x_i, x_{i-1}, \cdots) = P(y_i|y_{i-1}, x_i)$$

"Next-state" classifiers are locally normalized

Eg: Part-of-speech tagging the sentence



Suppose these are the only state transitions allowed

Example based on [Wallach 2002]

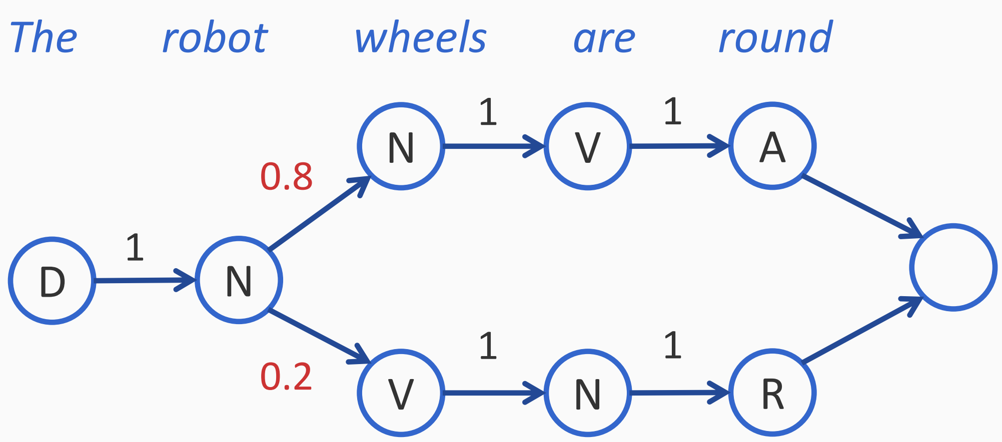
…local classifiers → Label bias problem

Let's look at the independence assumption

$$P(y_i | y_{i-1}, y_{i-2}, \cdots, x_i, x_{i-1}, \cdots) = P(y_i | y_{i-1}, x_i)$$

"Next-state" classifiers are locally normalized

Eg: Part-of-speech tagging the sentence



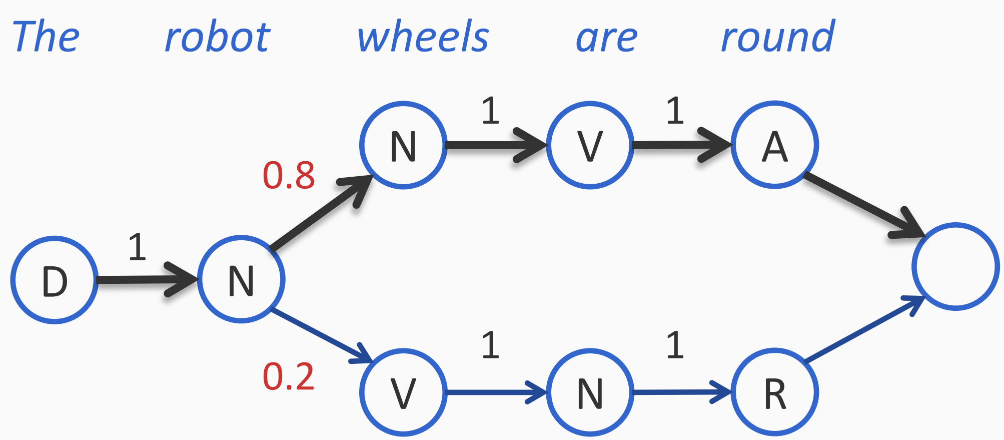
Option 1: P(D | The) ×
P(N | D, robot) ×
P(N | N, wheels) ×
P(V | N, are) ×
P(A | V, round)

Option 2: P(D | The) ×
P(N | D, robot) ×
P(V | N, wheels) ×
P(N | V, are) ×
P(R| N, round)

Suppose these are the only state transitions allowed

Example based on [Wallach 2002]

…local classifiers → Label bias problem



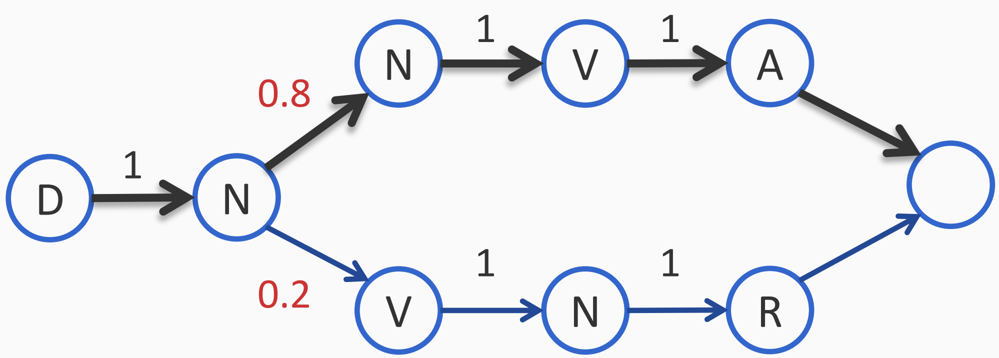
Suppose these are the only state transitions allowed

Option 1: P(D | The) ×
P(N | D, robot) ×
P(N | N, wheels) ×
P(V | N, are) ×
P(A | V, round)

Option 2: P(D | The) ×
P(N | D, robot) ×
P(V | N, wheels) ×
P(N | V, are) ×
P(R| N, round)

…local classifiers → Label bias problem

The *robot* *wheels* *are* *round*



Option 1: P(D | The) ×
P(N | D, robot) ×
P(N | N, wheels) ×
~~P(V | N, are)~~ × P(V | N, Fred) ×
P(A | V, round)

Option 2: P(D | The) ×
P(N | D, robot) ×
P(V | N, wheels) ×
~~P(N | V, are)~~ × P(N | V, Fred) ×
P(R| N, round)

Suppose these are the only state transitions allowed

The *robot* *wheels* *Fred* *round*

The path scores are the same

Even if the word Fred is never observed as a verb in the data, it will be predicted as one

The input *Fred* does not influence the output at all

Label Bias

- States with a single outgoing transition effectively ignore their input
  - States with lower-entropy next states are less influenced by observations
- Why?
  - Because each the next-state classifiers are locally normalized
  - If a state has fewer next states, each of those will get a higher probability mass
    - …and hence preferred
- Side note: Surprisingly doesn't affect some tasks
  - Eg: part-of-speech tagging

Summary: Local models for Sequences

- Conditional models
- Use rich features in the mode
- Possibly suffer from label bias problem

(Other “local” models may have their own version of the label bias problem too.)

Outline

- Sequence models
- Hidden Markov models
  - Inference with HMM
  - Learning
- Conditional Models and Local Classifiers
- Global models
  - Conditional Random Fields
  - Structured Perceptron for sequences

Outline

- Sequence models
- Hidden Markov models
  - Inference with HMM
  - Learning
- Conditional Models and Local Classifiers
- Global models
  - Conditional Random Fields
  - Structured Perceptron for sequences