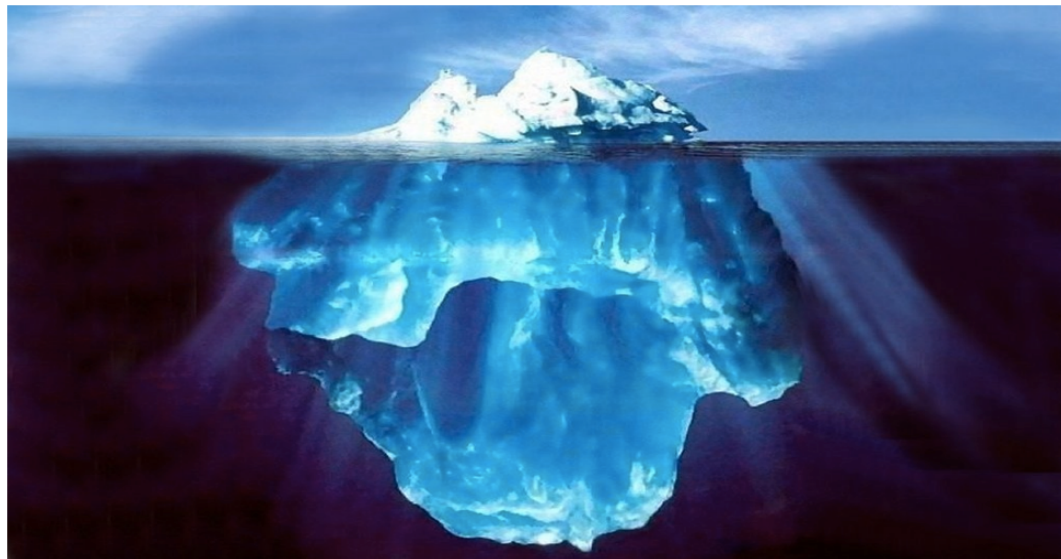


ML4NLP

Introduction to Structures



Dan Goldwasser
Purdue University

dgoldwas@purdue.edu

Structure Prediction

Language is highly structured.

We read words in the context of other words.

We make long range inferences, in order to decode language meaning.

e.g., “John at lunch, he later went to sleep”

How can we account for structural dependencies when learning and making predictions?

Spoiler Alert – *this is where we are headed*

$$\text{Solve: } \min_{w, \xi} \frac{1}{2} \|w\|^2 + \frac{C}{N} \sum_{n=1}^N \xi^n$$

subject to, for $i = 1, \dots, n$,

$$\langle w, \phi(x^n, y^n) \rangle - \langle w, \phi(x^n, y) \rangle \geq \Delta(y^n, y) - \xi^n \quad \text{for all } y \in \mathcal{Y} \setminus \{y^n\}.$$

Instead we can have an unconstrained version -

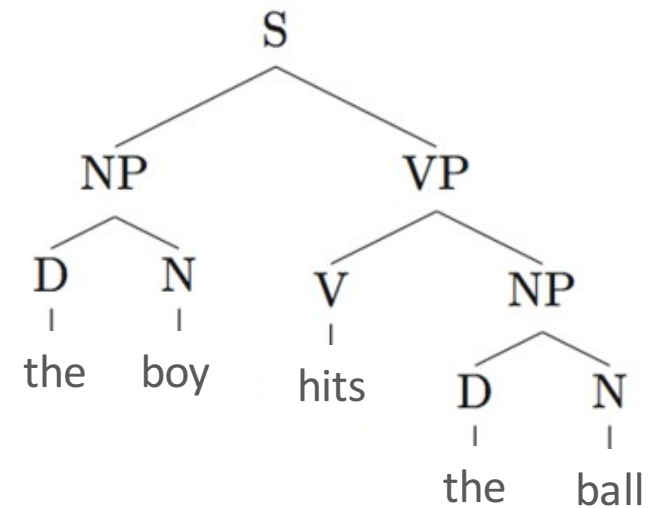
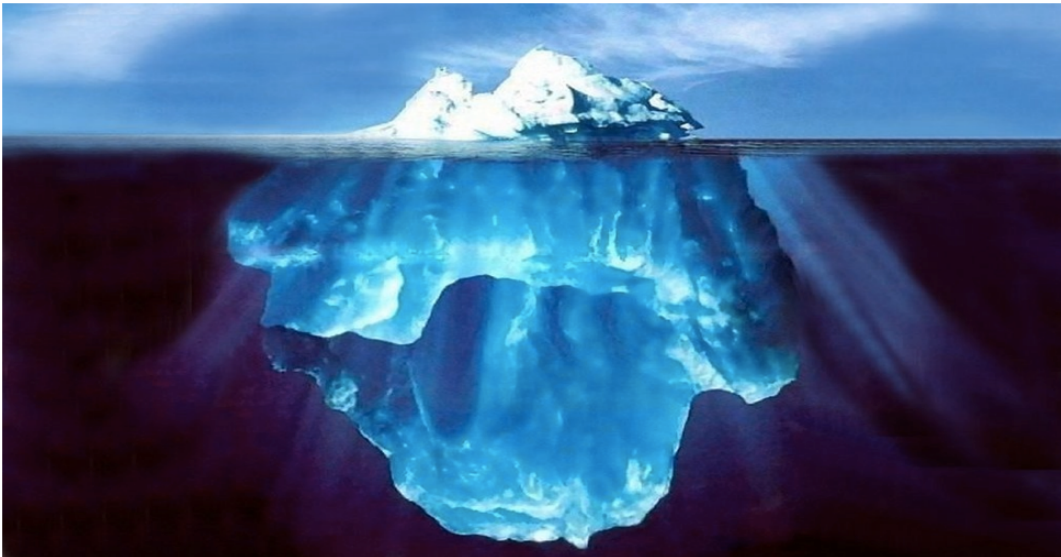
$$\text{Solve: } \min_{w, \xi} \frac{1}{2} \|w\|^2 + \frac{C}{N} \sum_{n=1}^N l(\mathbf{w}; (\mathbf{x}^n, y^n))$$

$$l(\mathbf{w}; (\mathbf{x}^n, y^n)) = \max_{y' \in Y} \Delta(y, y') - \langle \mathbf{w}, \phi(\mathbf{x}^n, y^n) \rangle + \langle \mathbf{w}, \phi(\mathbf{x}^n, y') \rangle$$

NL Structures: *the 10,000 feet View*

- ***Tip of the iceberg***

Natural language, expressed in words, is just a surface representation underlying a complex structure



A core concept is ***Inference***: generalizing the notion of ***classification***

Structure Prediction

Let's look at a couple of examples...

- Part of speech Tagging
- Named Entity Recognition
- Parsing
- Information extraction
- Co-reference Resolution

How would you categorize the structural dependencies for each task?

Can you define NER as a multiclass classification problem?

Named entity recognition (NER)

Identify mentions of named entity in text

People (PER), places (LOC) and organizations (ORG)

Begin_p

Barak Obama visited Mount Sinai hospital in New York

PER

ORG

LOC

Conceptually two considerations:

- Entity **boundary** (Begin, Inside ,Outside)
- Entity **type** (PER,ORG,LOC)

Our Labels:

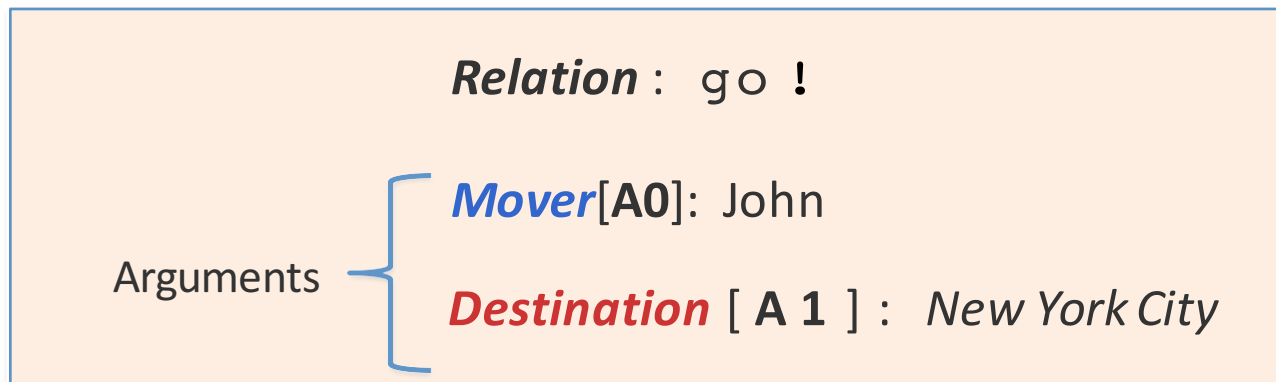
Begin_p,Begin_L,Begin_o
Inside_p,Inside_L,Inside_o
Outside

Given a sentence, are consecutive predictions independent of each other?

NL Structures: *Examples*

- Semantic role labeling
- Predict the arguments of verbs
 - “who did what to whom”
 - Large human annotated corpus of semantic relations (*see first lecture for more details on this task*)

John was going to New York City



Assume you already have a trained model

Prediction Verb Arguments

Input: Text with pre-processing (word spans within the sentence are treated as argument candidates)

Prediction:
Several possible decisions for each candidate (argument by type, or not-an-argument)

Each decision corresponds to a binary variable, **exactly one variable is true for each candidate.**

Output: *the set of variables that are “on” constitute the output structure*

1. **Identify** candidate arguments for verb using parse tree
 - Filtered using a binary classifier
2. **Classify** argument candidates
 - Multiclass classifier (one of multiple labels per candidate)
3. **Inference**
 - Using probability estimates from argument classifier
 - Must respect structural and linguistic constraints
 - Eg: No overlapping arguments

Structured output is...

Intuition: transform data to structured information (DB)

- A **graph**, possibly labeled and/or directed
 - Possibly from a restricted family, such as chains, trees, etc.
- A collection of **inter-dependent decisions**
 - Eg: The sequence of decisions used to construct the output
- The result of a combinatorial optimization problem
 - $\operatorname{argmax}_y \operatorname{score}(x, y)$



We have seen something similar before in the context of multiclass

There is a countable number of graphs

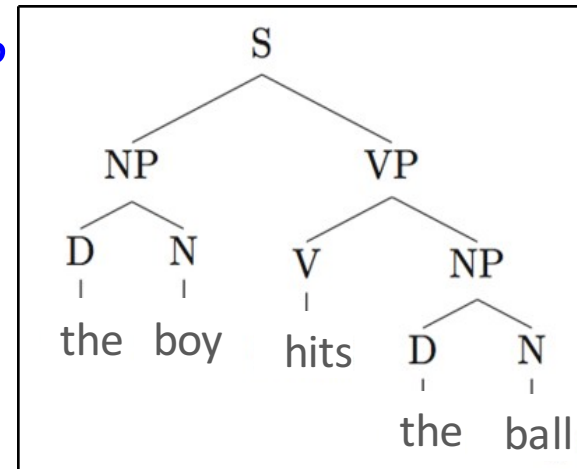
Question: **Why can't we treat each output as a label and train/predict as multiclass?**

Take 1: *reduce to multiclass*

- Our view of multiclass classification:
 - A set of labels: *politics, sports, finance*
- Multiclass classification:
 - Define feature functions: $\varphi(\mathbf{x}, y)$
 - Prediction: $\operatorname{argmax}_y w_y \varphi(\mathbf{x}, y)$
 - Training: find W s.t. $\operatorname{argmax}_y w_y \varphi()$ will return correct label
- **Can we define the structure prediction this way?**
 - What are the classes for:

“The boy hits the ball” ?

- S_0 : $\langle D-N-NP-V-D-N-NP-VP-St_i \rangle$
- S_1 : $\langle N-N-NP-V-D-N-NP-VP-St_i \rangle$
- S_2 : $\langle N-D-NP-V-D-N-NP-VP-St_i \rangle$
- ...



From Multiclass to Structures

Option 1: treat each structure as a different class

- Well, that didn't work...
- **Option 2: ?**
 - *What are the potential problems there?*

What are the other points along this line?

*Independent
decisions*

*Treat every structure as
an output class*

Take 2: *decompose the output!*

We cannot directly reduce structured prediction to multiclass classification.

- Enumerating all possible structures is infeasible
- Maintaining the parameters of all possible structures separately

Instead, we can decompose the output object into parts.

Basic idea:

- *Instead of scoring structures, we can score the parts!*
 - Use an **inference** algorithm to construct the structure from the parts (optimal score for entire set of decisions)
- Question:** How do decisions influence one another?

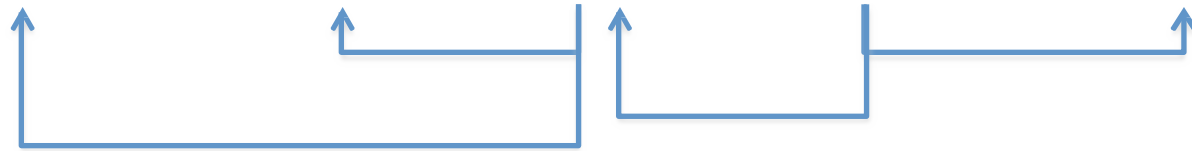
Example: *Dependency parsing*

- **A dependency parse can be viewed as a graph**
 - **Nodes:** words
 - **Edges:** syntactic relations between words
- **Parsing: We need to produce a graph**
 - *Nodes are given, predict the edges and their labels*
 - *We cannot enumerate all possible graphs for the argmax*
- **Solution:** Construct the graph by predicting edge-parts
 - Each part has a score
 - The total score for the graph is the sum of scores of each part
 - The parts should agree with each other in the final output

Output

Decomposition

Colorless green ideas sleep furiously



How can you score each one of these trees?

Colorless green ideas sleep furiously



~~Colorless green ideas sleep furiously~~

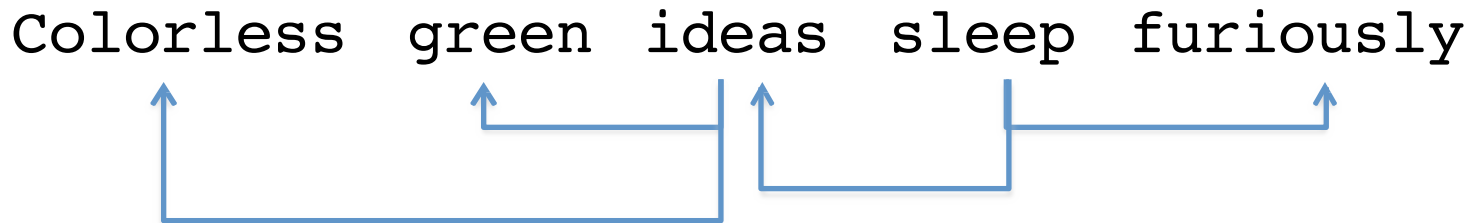


Output

Option 1

Decomposition

- For every two words, decide if there is a directed edge based on a local scoring function

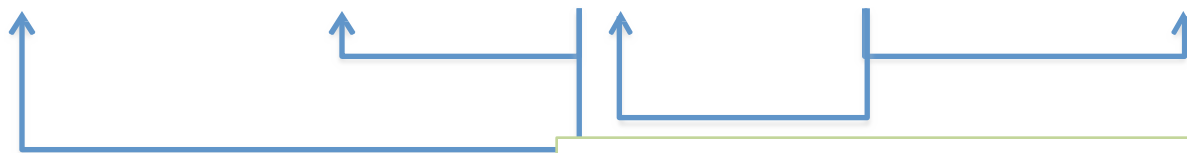


Decomposing the output: Example

Option 2

- Can you consider multiple edges when constructing the graph? (e.g., pairs of edges)

Colorless green ideas sleep furiously



	English	
	Accuracy	Complete
1st-order-projective	90.7	36.7
2nd-order-projective	91.5	42.1

Table 1: Dependency parsing results for English.

Learning Structured Predictors

- The first decision we had to take is how to decompose the output structure.
- Given the decomposition, we can learn a predictor using:
 - **Local Training regime:**
 - Learning **local models**, no inference during training
 - **Global/joint training**
 - Inference based training aims to optimize the global scoring during training
- Similar to the discussion for the multiclass case
- **Inference complexity** is an important consideration in choice of modeling and training

Let's look at an example -

Sequence models!

Simple but very useful in NLP (..and in general)

Sequence Tagging

- **Sequence: transition between states**
 - Text is a sequence of words or (even characters)
 - General formulation, typically used to capture temporal transitions
- We want to define a probabilistic model, representing the probability distribution of state sequences.
 - Let x_1, x_2, \dots, x_n be a state sequence with n tokens, we want to define $P(x_1, x_2, \dots, x_n)$
 - Notable example – **language models**

Language Modeling with N-grams

A language model assigns a probability score to sentences

$$P(w_1 \dots w_i) = P(w_1)P(w_2|w_1)P(w_3|w_1w_2)\dots P(w_i|w_1 \dots w_{i-1})$$

Independence Assumption: An N-gram language models assume each word depends on the previous $n-1$ words

$$P_{n\text{gram}}(w_1 \dots w_i) := P(w_1)P(w_2|w_1)\dots P(\underbrace{w_i}_{\text{nth word}} \mid \underbrace{w_{i-n-1} \dots w_{i-1}}_{\text{prev. } n-1 \text{ words}})$$

Unigram model $P(w_1)P(w_2)\dots P(w_i)$

Bigram model $P(w_1)P(w_2|w_1)\dots P(w_i|w_{i-1})$

Trigram model $P(w_1)P(w_2|w_1)\dots P(w_i|w_{i-2} w_{i-1})$

And now – **Hidden** Markov Model

- **So far every thing was observed:**
 - States follow a Markov chain
 - *Each state is an **observation***
- **Hidden Markov Model:**
 - States follow a Markov chain
 - **States are not observed**
 - *Each state stochastically emits an observation*

Key Question--

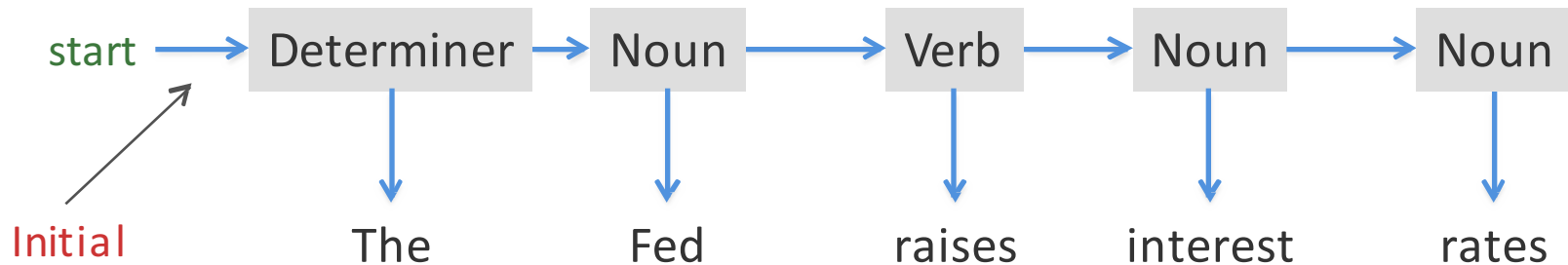
Can we recover the state sequence based on observations?

Simple example: **Sequence labeling**

The Fed raises interest

- The word “Fed” can be a noun or a verb
- **Local decision**: more likely to appear as a verb
- **Context**: can help identify noun occurrences
 - Can you think of one?
 - Words following a **determiner** tend to be **nouns**
- **Assumption**:
 - Contextual considerations tend to be “**localized**”

PoS tagging example:



$$\arg \max_y p(x_1, \dots, x_n | y_1, \dots, y_n) p(y_1, \dots, y_n)$$

Based on our independence assumptions:

$$\arg \max_y \prod_{i=1}^n p(x_i | y_i) \prod_{i=1}^n p(y_i | y_{i-1})$$

Decoding (prediction)

Given an observation sequence and an HMM, we need to find the **optimal state sequence**:

$$\arg \max_y p(x_1, \dots, x_n | y_1, \dots, y_n) p(y_1, \dots, y_n)$$

- How can we find it?
 - *Combinatorial optimization problem*

Wrong Way: Enumerating all assignments

NNP John NNP ate VBD lunch NNP at NNP the NNP watering NNP hole NNP cafe ti.tititi1

DET John DET ate VBD lunch NNP at NNP the NNP watering NNP hole NNP cafe ti.tititi
2

VBD John VBD ate VBD lunch NNP at NNP the NNP watering NNP hole NNP cafe ti.tititi
2

...

NNP John VBD ate NN lunch IN at DT the NN watering NN hole NN cafe ti.ti1ti2

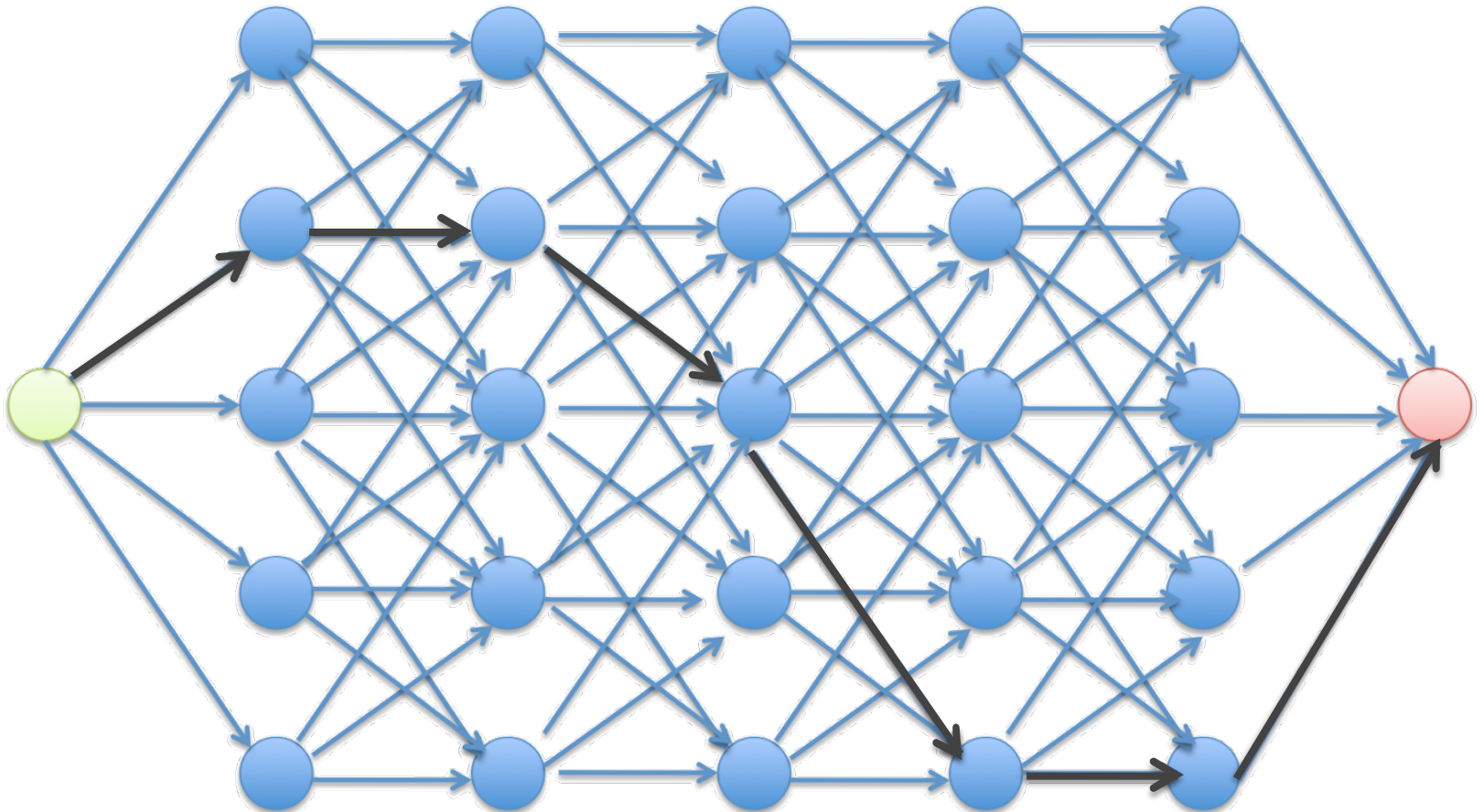
How many possible assignments are there?

Basic idea:
$$\arg \max_y \prod_{i=1}^n p(x_i | y_i) \prod_{i=1}^n p(y_i | y_{i-1})$$

Independence assumptions lead to an algorithmic solution!

Viterbi algorithm as best path

Goal: To find the highest scoring path in this trellis



Viterbi Algorithm

Definitions:

n : length of input, S_k : possible symbols at position k

Truncated version of the probability (defined over k long sequences, $k < n$)

$$r(y_1, \dots, y_k) = \prod_{i=1}^k p(y_i | y_{i-1}) \prod_{i=1}^k p(x_i | y_i)$$

DP table:

$$\pi(k, v) = \max_{(y_1, \dots, y_k; y_k = v)} r(y_1, \dots, y_k)$$

max probability tag sequence of size k ending with v

Recursive definition of DP table:

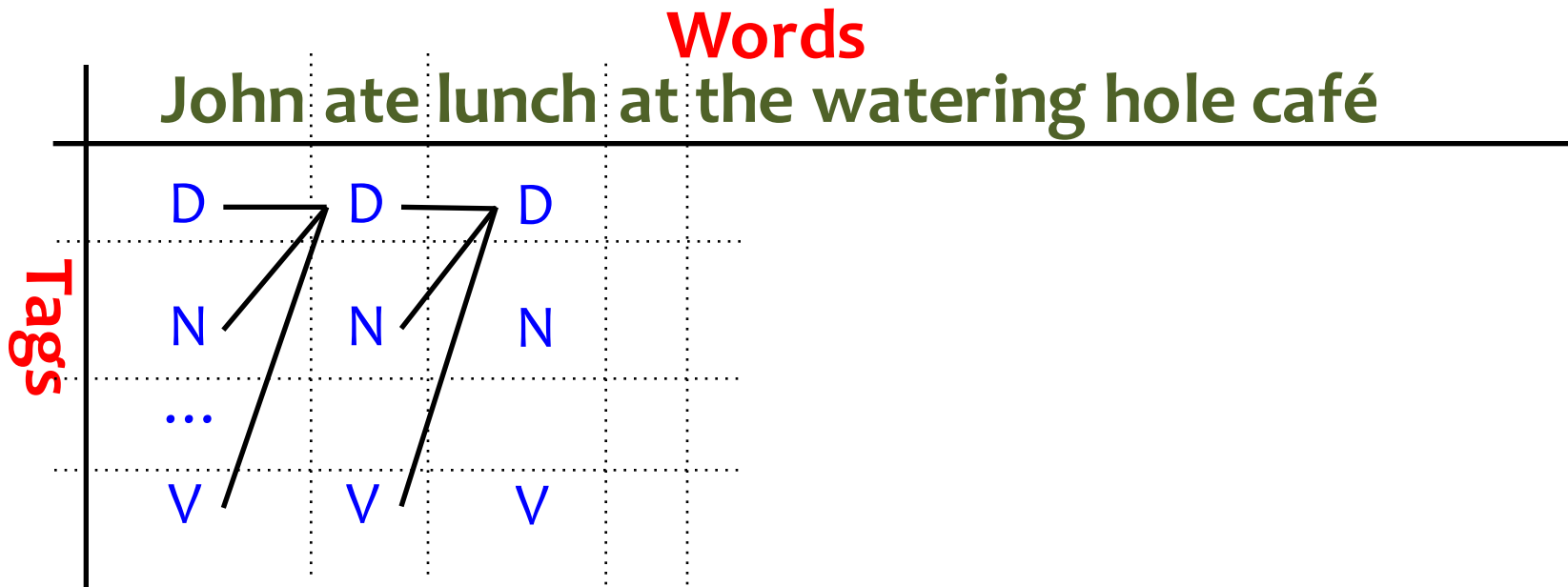
$$\pi(k, v) = \max_{u \in S_{k-1}} (\pi(k-1, u) \times p(v|u) \times p(x_k|v))$$

Viterbi: DP Table

$$\pi(k, v) = \max_{(y_1, \dots, y_k; y_k=v)} r(y_1, \dots, y_k)$$

max probability tag sequence of size k ending with v

$$\pi(k, v) = \max_{u \in S_{k-1}} (\pi(k-1, u) \times p(v|u) \times p(x_k|v))$$



$$\pi(3, D) = \max_{(y_1, \dots, y_3; y_3=D)} r(y_1, \dots, y_3)$$

The Viterbi Algorithm

Input: a sequence x_1, \dots, x_n ,
parameters: $p(s|u)$, $p(x|s) \forall s, u \in S$

Initialization: $\pi(0, \epsilon) = 1$
(Note: ϵ is just a start symbol)

For $k = 1..n$

For $v \in S_k$

$$\pi(k, v) = \max_{u \in S_{k-1}} (\pi(k-1, u) \times p(v|u) \times p(x_k|v))$$

Return $\max_{u \in S_n} (\pi(n, u) \times p(\sigma|u))$
(Note: σ is just an end symbol)

Note:

We augment the set of tags with *start* symbol and compute parameters for these symbols

What is the run time complexity of Viterbi?

What does this algorithm return?

We are interested in the optimal sequence!

Solution: small modification to the algorithm, maintain a list of backpointers

In practice: smoothing is required!

Parameter Estimation

$$p(y_i | y_{i-1}) p(x_i | y_i)$$

Two terms:

$$p(y_i | y_{i-1}) \quad (\text{transitions probabilities})$$

$$p(NN | DET) = \frac{\text{count}(NN, DET)}{\text{count}(DET)}$$

$$A_{s',s} = \frac{\text{count}(s \rightarrow s')}{\text{count}(s)}$$

$$p(x_i | y_i) \quad (\text{emission probabilities})$$

$$p(\text{"watering"} | NN) = \frac{\text{count}(\text{"watering"}, NN)}{\text{count}(NN)}$$

$$B_{s,x} = \frac{\text{count} \left(\begin{array}{c} s \\ \downarrow \\ x \end{array} \right)}{\text{count}(s)}$$

Initial state probability

$$\pi_s = \frac{\text{count}(\text{start} \rightarrow s)}{n}$$

Generative vs. Discriminative

Hmm: *Model for the joint probability of (x,y)*

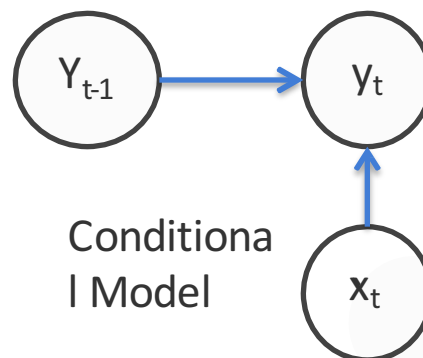
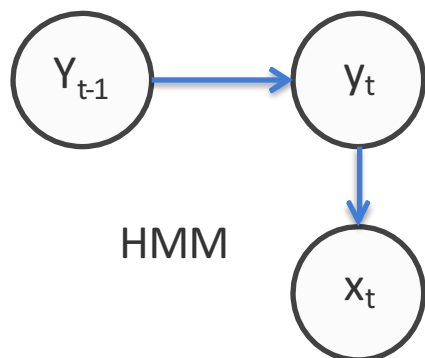
$$P(x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^n P(x_i|y_i)$$

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

- Instead of modeling the joint distribution $P(\mathbf{x}, \mathbf{y})$ only focus on $P(\mathbf{y}|\mathbf{x})$
 - Where have we seen it before?
 - **How can we extend this model to sequences?**
 - Maximum Entropy Markov Model [McCallum, et al 2000]

Conditional Models

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



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 learn this function

Learning Conditional Models

- **Advantages:**
 - Use rich features that depend on input and previous state
 - We can extract rich features from the entire input sequence
- **Learning algorithms:**
 - *Probabilistic*: Logistic regressions (=Max Entropy)
 - We can also use any multiclass classifier
 - Perceptron, SVM,..

Log-Linear Models for Multiclass Classification

Consider multiclass classification

- Input: \mathbf{x} , output: y (multiclass $1, \dots, k$)
- Feature representation: $\Phi(\mathbf{x}, y)$
 - *Joint feature function (input + output)*
- Conditional probability:

$$P(y|\mathbf{x}, \mathbf{w}) = \frac{e^{\mathbf{w}^T \phi(\mathbf{x}, y)}}{\sum_{y'} e^{\mathbf{w}^T \phi(\mathbf{x}, y)}}$$

- ***A generalization of logistic regression to multiclass***

Training is straightforward

Training: maximum likelihood

$$\max_{\mathbf{w}} \sum_i \log P(\mathbf{y}_i | \mathbf{x}_i, \mathbf{w})$$

For logistic Regression:

$$P(\mathbf{y} | \mathbf{x}, \mathbf{w}) = \frac{e^{\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y})}}{\sum_{y'} e^{\mathbf{w}^T \phi(\mathbf{x}, y')}}$$

Regularized version:

$$\max_{\mathbf{w}} -\frac{\lambda}{2} \mathbf{w}^T \mathbf{w} + \sum_i \log P(\mathbf{y}_i | \mathbf{x}_i, \mathbf{w})$$

Gradient Based optimization

- **Gradient based methods**

- using gradient of $L(\mathbf{w}) = \sum_i \log P(\mathbf{y}_i | \mathbf{x}_i, \mathbf{w})$


- Simple approach

1. Initialize $\mathbf{w} =$

- ti 2. For $t =$

- 1, 2, ... Update $\mathbf{w} = \mathbf{w} + a_t r \nabla L(\mathbf{w})$

3. Return \mathbf{w}



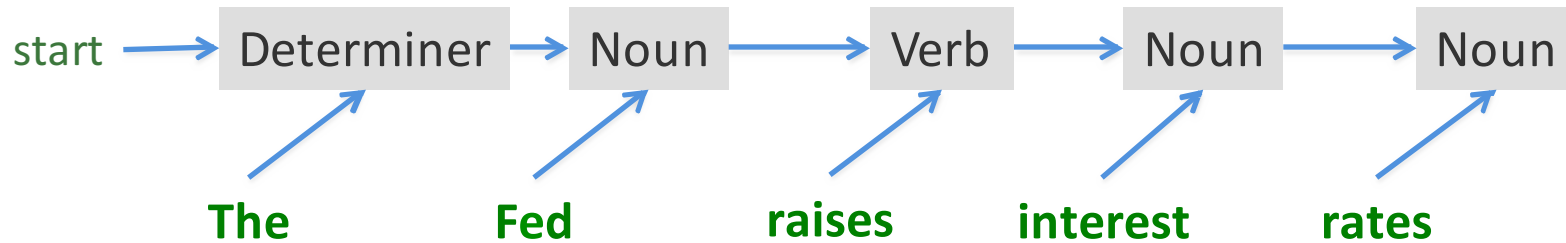
A vector, whose j^{th} element is the derivative of L with \mathbf{w}_j .

$$\frac{\partial}{\partial \mathbf{w}_j} L(\mathbf{w}) = \sum_i \left(\phi_j(\mathbf{x}_i, \mathbf{y}_i) - \sum_y P(\mathbf{y} | \mathbf{x}_i, \mathbf{w}) \phi_j(\mathbf{x}_i, \mathbf{y}) \right)$$

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

- **Different approaches possible**
 1. Train a *maximum entropy (log-linear)* classifier
 2. *Or, ignore the fact that we are predicting a probability,* we only care about maximizing some *score*. Train any 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

MEMM: Max Entropy Markov Model



<i>word</i>	The	Fed	raises	interest	rates
<i>Caps</i>	Y	Y	N	N	N
<i>--es</i>	N	N	Y	N	N
<i>Previous</i>	start	Determiner	Noun	Verb	Noun
	$\Phi(x, 0, \text{start}, y_0)$	$\Phi(x, 1, y_0, y_1)$	$\Phi(x, 2, y_1, y_2)$	$\Phi(x, 3, y_2, y_3)$	$\Phi(x, 4, y_3, y_4)$

Using MEMM

- **Training**

- Train next-state predictor *locally* as maximum likelihood
 - Similar to any Log-linear model

In general, any algorithm can be used to score y_i given y_{i-1} and \mathbf{x}

- Pick your favorite multiclass classifier

- **Prediction/Decoding**

- *Modify the Viterbi algorithm for the new independence assumptions*

Question: *How would you modify Viterbi when using non-probabilistic classifiers?*

Label bias

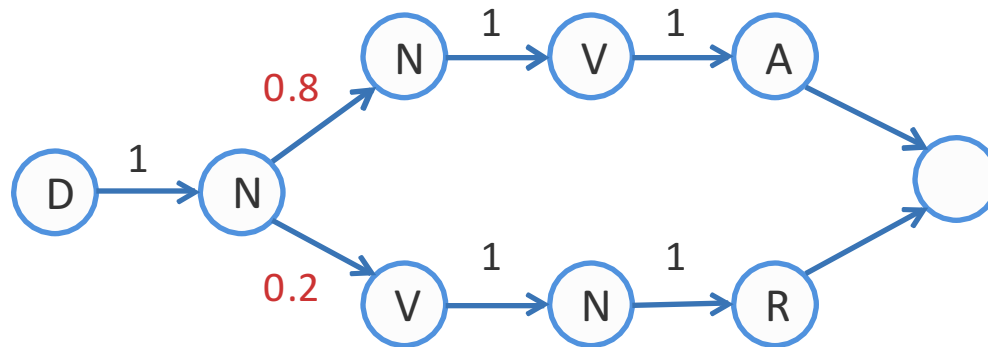
MEMM independence assumption

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

Learning local “next-state” classifiers

Example:

The robot wheels are round



Option 1:

$P(D | \text{The}) \cdot$
 $P(N | D, \text{robot}) \cdot$
 $P(N | N, \text{wheels}) \cdot$
 $P(V | N, \text{are}) \cdot$
 $P(A | V, \text{round})$

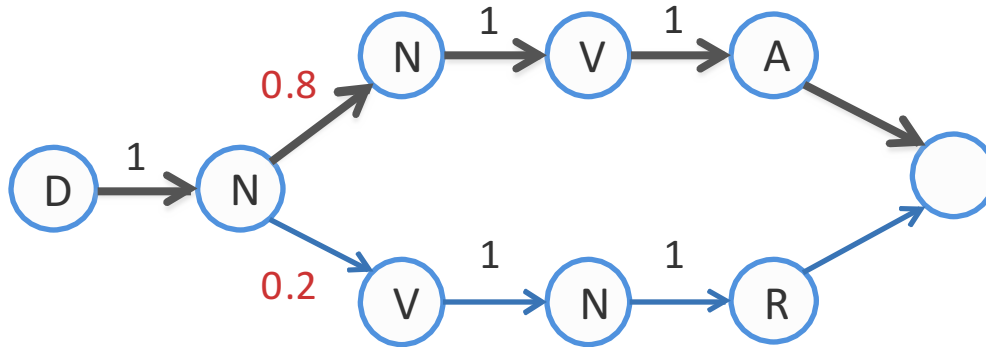
Option 2:

$P(D | \text{The}) \cdot$
 $P(N | D, \text{robot}) \cdot$
 $P(V | N, \text{wheels}) \cdot$
 $P(N | V, \text{are}) \cdot$
 $P(R | N, \text{round})$

Suppose these are the only state transition allowed

Label bias

The robot wheels are round



Option 1: $P(D \mid \text{The}) \cdot$
 $P(N \mid D, \text{robot}) \cdot$
 $P(N \mid N, \text{wheels}) \cdot$
 $P(V \mid N, \text{are})$ $\cdot P(V \mid N, \text{Fred}) \cdot$
 $P(A \mid V, \text{round})$

Option 2: $P(D \mid \text{The}) \cdot$
 $P(N \mid D, \text{robot}) \cdot$
 $P(V \mid N, \text{wheels}) \cdot$
 ~~$P(N \mid V, \text{are})$~~ $\cdot P(N \mid V, \text{Fred}) \cdot$
 $P(R \mid N, \text{round})$

The path scores are the same

-- regardless of the change in inputs!

Label Bias

- States with a single outgoing transition effectively ignore their input
 - States with fewer transitions will dominate the result
- Why?
 - **Each next-state classifier is normalized locally**
 - 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: POS tagging

Can you define NER as a MEMMM?

Named entity recognition (NER)

Identify mentions of named entity in text

People (PER), places (LOC) and organizations (ORG)

Begin_p

Barak Obama visited Mount Sinai hospital in New York

PER

ORG

LOC

Conceptually two considerations:

- Entity **boundary** (Begin, Inside ,Outside)
- Entity **type** (PER,ORG,LOC)

Our Labels:

Begin_p,Begin_L,Begin_o
Inside_p,Inside_L,Inside_o
Outside

Given a sentence, are consecutive predictions independent of each other?