Part-of-Speech Tagging: HMM & structured perceptron

MARINE CARPUAT
marine@cs.umd.edu
Last time...

• What are parts of speech (POS)?
  – Equivalence classes or categories of words
  – Open class vs. closed class
  – Nouns, Verbs, Adjectives, Adverbs (English)

• What is POS tagging?
  – Assigning POS tags to words in context
  – Penn Treebank

• How to POS tag text automatically?
  – Multiclass classification vs. sequence labeling
Today

• 2 approaches to POS tagging
  – Hidden Markov Models
  – Structured Perceptron
Hidden Markov Models

• Common approach to sequence labeling

• A finite state machine with probabilistic transitions

• Markov Assumption
  – next state only depends on the current state and independent of previous history
HMM: Formal Specification

- **Q**: a finite set of $N$ states
  - $Q = \{q_0, q_1, q_2, q_3, \ldots\}$
- **$N \times N$ Transition probability matrix** $A = [a_{ij}]$
  - $a_{ij} = P(q_j|q_i), \sum a_{ij} = 1 \ \forall i$
- **Sequence of observations** $O = o_1, o_2, \ldots o_T$
  - Each drawn from a given set of symbols (vocabulary $V$)
- **$N \times |V|$ Emission probability matrix**, $B = [b_{it}]$
  - $b_{it} = b_i(o_t) = P(o_t|q_i), \sum b_{it} = 1 \ \forall i$
- **Start and end states**
  - An explicit start state $q_0$ or alternatively, a prior distribution over start states: $\{\pi_1, \pi_2, \pi_3, \ldots\}, \sum \pi_i = 1$
  - The set of final states: $q_F$

Markov Assumption
Stock Market HMM

States? ✓
Transitions? ✓
Vocabulary? ✓
Emissions? ✓
Priors? ✓

\[
\begin{align*}
\text{BEAR} & \quad 0.3 & \quad 0.1 & \quad 0.2 & \quad 0.5 \\
\text{BULL} & \quad 0.2 & \quad 0.6 & \quad 0.2 & \quad 0.4 \\
\text{STATIC} & \quad 0.1 & \quad 0.5 & \quad 0.3 & \quad 0.2 \\
\end{align*}
\]

- \( P(\uparrow|\text{Bear}) = 0.1 \)
- \( P(\downarrow|\text{Bear}) = 0.6 \)
- \( P(\leftrightarrow|\text{Bear}) = 0.3 \)
- \( P(\uparrow|\text{Bull}) = 0.7 \)
- \( P(\downarrow|\text{Bull}) = 0.1 \)
- \( P(\leftrightarrow|\text{Bull}) = 0.2 \)
- \( P(\uparrow|\text{Static}) = 0.3 \)
- \( P(\downarrow|\text{Static}) = 0.3 \)
- \( P(\leftrightarrow|\text{Static}) = 0.4 \)

\[ V = \{\uparrow, \downarrow, \leftrightarrow\} \]
HMMs: Three Problems

- **Likelihood:** Given an HMM $\lambda = (A, B, \Pi)$, and a sequence of observed events $O$, find $P(O|\lambda)$

- **Decoding:** Given an HMM $\lambda = (A, B, \Pi)$, and an observation sequence $O$, find the most likely (hidden) state sequence

- **Learning:** Given a set of observation sequences and the set of states $Q$ in $\lambda$, compute the parameters $A$ and $B$
HMM Problem #1: Likelihood
Computing Likelihood

Assuming $\lambda_{stock}$ models the stock market, how likely are we to observe the sequence of outputs?
Computing Likelihood

- First try:
  - Sum over all possible ways in which we could generate $O$ from $\lambda$

$$P(O|\lambda) = \sum_Q P(O,Q|\lambda) = \sum_Q P(O|Q, \lambda)P(Q|\lambda)$$

$$= \sum_{q_1, q_2 \ldots q_T} \pi_{q_1} b_{q_1}(o_1) a_{q_1q_2} \ldots a_{q_{T-1}q_T} b_{q_T}(o_T)$$

Takes $O(N^T)$ time to compute!
Forward Algorithm

- Use an $N \times T$ trellis or chart $[\alpha_{tj}]$
- Forward probabilities: $\alpha_{tj}$ or $\alpha_t(j)$
  
  $= P($being in state $j$ after seeing $t$ observations$)$
  
  $= P(o_1, o_2, \ldots, o_t, q_t=j)$

- Each cell $= \sum$ extensions of all paths from other cells
  
  $\alpha_t(j) = \sum_i \alpha_{t-1}(i) a_{ij} b_j(o_t)$
  
  $- \alpha_{t-1}(i)$: forward path probability until $(t-1)$
  
  $- a_{ij}$: transition probability of going from state $i$ to $j$
  
  $- b_j(o_t)$: probability of emitting symbol $o_t$ in state $j$

- $P(O|\lambda) = \sum_i \alpha_T(i)$
Forward Algorithm: Formal Definition

• Initialization
  \[ \alpha_1(j) = \pi_j b_j(o_1), \quad 1 \leq j \leq N \]

• Recursion
  \[ \alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t); \quad 1 \leq j \leq N, \quad 2 \leq t \leq T \]

• Termination
  \[ P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \]
Forward Algorithm

\[ O = \uparrow \downarrow \uparrow \]

\[ \text{find } P(O|\lambda_{stock}) \]
Forward Algorithm

states

Static

Bear

Bull

↑
t=1

↓
t=2

↑
t=3

time
Forward Algorithm: Initialization

\[ \alpha_1(j) = \pi_j b_j(o_1), 1 \leq j \leq N \]

States:

- **Static** \( \alpha_1(\text{Static}) \):
  \[ 0.3 \times 0.3 = 0.09 \]

- **Bear** \( \alpha_1(\text{Bear}) \):
  \[ 0.5 \times 0.1 = 0.05 \]

- **Bull** \( \alpha_1(\text{Bull}) \):
  \[ 0.2 \times 0.7 = 0.14 \]

Time:

- **t=1**: Up
- **t=2**: Down
- **t=3**: Up
Forward Algorithm: Recursion

\[ \alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i)a_{ij}b_j(o_t); \; 1 \leq j \leq N, \; 2 \leq t \leq T \]

States:
- **Static**
  - 0.3x0.3 = 0.09
- **Bear**
  - 0.5x0.1 = 0.05
- **Bull**
  - 0.2x0.7 = 0.14

\[ \alpha_1(\text{Bull}) \times a_{\text{BullBull}} \times b_{\text{Bull}}(\downarrow) = 0.14 \times 0.6 \times 0.1 = 0.0084 \]

Time:
- t=1
- t=2
- t=3

... and so on
Forward Algorithm: Recursion

Work through the rest of these numbers...

Static

- 0.3 × 0.3 = 0.09
- ?
- ?

Bear

- 0.5 × 0.1 = 0.05
- ?
- ?

Bull

- 0.2 × 0.7 = 0.14
- 0.0145
- ?

↑

<table>
<thead>
<tr>
<th>time</th>
<th>t=1</th>
<th>t=2</th>
<th>t=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bear</td>
<td>0.14</td>
<td>0.0145</td>
<td>?</td>
</tr>
<tr>
<td>Bull</td>
<td>0.09</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

What’s the asymptotic complexity of this algorithm?
HMM Problem #2: Decoding
Decoding

Given $\lambda_{stock}$ as our model and $O$ as our observations, what are the most likely states the market went through to produce $O$?
Decoding

• “Decoding” because states are hidden

• First try:
  – Compute $P(O)$ for all possible state sequences, then choose sequence with highest probability
Viterbi Algorithm

• “Decoding” = computing most likely state sequence
  – Another dynamic programming algorithm
  – Efficient: polynomial vs. exponential (brute force)

• Same idea as the forward algorithm
  – Store intermediate computation results in a trellis
  – Build new cells from existing cells
Viterbi Algorithm

- Use an $N \times T$ trellis [$v_{tj}$]
  - Just like in forward algorithm

- $v_{tj}$ or $v_{t}(j)$
  - $P$ (in state $j$ after seeing $t$ observations and passing through the most likely state sequence so far)
  - $v_{t} = P(q_1, q_2, \ldots, q_{t-1}, q_{t=j}, o_1, o_2, \ldots, o_t)$

- Each cell = extension of most likely path from other cells
  - $v_{t}(j) = \max_i v_{t-1}(i) \cdot a_{ij} \cdot b_j(o_t)$
    - $v_{t-1}(i)$: Viterbi probability until $(t-1)$
    - $a_{ij}$: transition probability of going from state $i$ to $j$
    - $b_j(o_t)$: probability of emitting symbol $o_t$ in state $j$

- $P = \max_i v_T(i)$
Viterbi vs. Forward

• **Maximization instead of summation** over previous paths

• **This algorithm is still missing something!**
  – In forward algorithm, we only care about the probabilities
  – What’s different here?

• **We need to store the most likely path** (transition):
  – Use “backpointers” to keep track of most likely transition
  – At the end, follow the chain of backpointers to recover the most likely state sequence
Viterbi Algorithm: Formal Definition

• Initialization
  \[ v_1(j) = \pi_i b_i(o_1); \quad 1 \leq i \leq N \]
  \[ BT_1(i) = 0 \]

• Recursion
  \[ v_t(j) = \max_{i=1}^{N} [v_{t-1}(i)a_{ij}] b_j(o_t); \quad 1 \leq i \leq N, 2 \leq t \leq T \]
  \[ BT_1(i) = \arg \max_{i=1}^{N} [v_{t-1}(i)a_{ij}] \]

• Termination
  \[ P^* = \max_{1=1}^{N} v_T(j) \]
  \[ q_T^* = \arg \max_{1=1}^{N} v_T(j) \]
Viterbi Algorithm

$O = \uparrow \downarrow \uparrow$

find most likely state sequence given $\lambda_{stock}$
Viterbi Algorithm

Static

Bear

Bull

↑
t=1

downarrow
t=2

up

t=3

time
Viterbi Algorithm: Initialization

\[ v_1(j) = \pi_i b_i(o_1); 1 \leq i \leq N \]
\[ BT_1(i) = 0 \]

**States**

- **Static** \( \alpha_1(Static) \)
  - Probability: \( 0.3 \times 0.3 = 0.09 \)

- **Bear** \( \alpha_1(Bear) \)
  - Probability: \( 0.5 \times 0.1 = 0.05 \)

- **Bull** \( \alpha_1(Bull) \)
  - Probability: \( 0.2 \times 0.7 = 0.14 \)

**Time Evolution**

- \( t=1 \)
- \( t=2 \)
- \( t=3 \)
Viterbi Algorithm: Recursion

\[ v_t(j) = \max_{i=1}^{N} [v_{t-1}(i) a_{ij}] b_j(o_t); \quad 1 \leq i \leq N, \, 2 \leq t \leq T \]

\[ BT_1(i) = \arg \max_{i=1}^{N} [v_{t-1}(i) a_{ij}] \]

**States**

- **Bear**
  - 0.5 \times 0.1 = 0.05
  - 0.05 \times 0.5 \times 0.1 = 0.0025

- **Bull**
  - 0.2 \times 0.7 = 0.14
  - 0.14 \times 0.6 \times 0.1 = 0.0084

**Static**

- 0.3 \times 0.3 = 0.09

**Time**

- \( t = 1 \)
- \( t = 2 \)
- \( t = 3 \)
Viterbi Algorithm: Recursion

\[ v_t(j) = \max_{i=1}^{N} [v_{t-1}(i) a_{ij}] b_j(o_t); \quad 1 \leq i \leq N, \quad 2 \leq t \leq T \]

\[ BT_1(i) = \arg \max_{i=1}^{N} [v_{t-1}(i) a_{ij}] \]

... and so on

0.3 \times 0.3 = 0.09

0.5 \times 0.1 = 0.05

0.2 \times 0.7 = 0.14

0.0084

Bull

Bear

Static

store backpointer

↑

t=1

↓

t=2

↑

t=3

time
Viterbi Algorithm: Recursion

Work through the rest of the algorithm...

- Static
  - $0.3 \times 0.3 = 0.09$
  - $0.5 \times 0.1 = 0.05$
- Bear
  - $0.2 \times 0.7 = 0.14$
  - $0.0084$
- Bull
  - $0.0084$

$t=1$

$t=2$

$t=3$
POS Tagging with HMMs
HMM for POS tagging: intuition

Credit: Jordan Boyd Graber
HMM for POS tagging: intuition
HMMs: Three Problems

• **Likelihood:** Given an HMM $\lambda = (A, B, \pi)$, and a sequence of observed events $O$, find $P(O|\lambda)$

• **Decoding:** Given an HMM $\lambda = (A, B, \pi)$, and an observation sequence $O$, find the most likely (hidden) state sequence

• **Learning:** Given a set of observation sequences and the set of states $Q$ in $\lambda$, compute the parameters $A$ and $B$
HMM Problem #3: Learning
Learning HMMs for POS tagging is a supervised task

• A POS tagged corpus tells us the hidden states!

• We can compute Maximum Likelihood Estimates (MLEs) for the various parameters
  – MLE = fancy way of saying “count and divide”

• These parameter estimates maximize the likelihood of the data being generated by the model
Supervised Training

• Transition Probabilities

  – Any $P(t_i \mid t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$, from the tagged data

  – Example: for $P(\text{NN} \mid \text{VB})$
    • count how many times a noun follows a verb
    • divide by the total number of times you see a verb
Supervised Training

• Emission Probabilities

  – Any $P(w_i \mid t_i) = \frac{C(w_i, t_i)}{C(t_i)}$, from the tagged data

  – For $P(\text{bank}|\text{NN})$
    • count how many times bank is tagged as a noun
    • divide by how many times anything is tagged as a noun
Supervised Training

• Priors
  – Any $P(q_1 = t_i) = \pi_i = C(t_i)/N$, from the tagged data
  – For $\pi_{NN}$, count the number of times NN occurs and divide by the total number of tags (states)
HMMs: Three Problems

- **Likelihood**: Given an HMM $\lambda = (A, B, \Pi)$, and a sequence of observed events $O$, find $P(O|\lambda)$

- **Decoding**: Given an HMM $\lambda = (A, B, \Pi)$, and an observation sequence $O$, find the most likely (hidden) state sequence

- **Learning**: Given a set of observation sequences and the set of states $Q$ in $\lambda$, compute the parameters $A$ and $B$
## Prediction Problems

- **Given** $x$, predict $y$

<table>
<thead>
<tr>
<th>A book review</th>
<th>Is it positive?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oh, man I love this book!</td>
<td>yes</td>
</tr>
<tr>
<td>This book is so boring...</td>
<td>no</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A tweet</th>
<th>Its language</th>
</tr>
</thead>
<tbody>
<tr>
<td>On the way to the park!</td>
<td>English</td>
</tr>
<tr>
<td>公園に行くなう！</td>
<td>Japanese</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A sentence</th>
<th>Its syntactic parse</th>
</tr>
</thead>
</table>
| I read a book | S 
| | VP  
| | DET  
| | NP  
| | I  
| | VBD  
| | a  
| | NN  

- **Binary Prediction** (2 choices)
- **Multi-class Prediction** (several choices)
- **Structured Prediction** (millions of choices)
Approaches to POS tagging

**Classifiers**
- Multiclass classification problem
- Logistic Regression
- Model context using lots of features

**Generative Models**
- Structured prediction problem (Sequence labeling)
- Hidden Markov Models
- Models transitions between states/POS

Structured perceptron → Classification with lots of features over structured models!
Let’s restructure HMMs with features...

- Given a sentence X, predict its part of speech sequence Y
Let’s restructure HMMs with features...

- **POS→POS transition probabilities**
  \[
P(Y) \approx \prod_{i=1}^{I+1} P_T(y_i \mid y_{i-1})
\]

- **POS→Word emission probabilities**
  \[
P(X \mid Y) \approx \prod_{i=1}^{I} P_E(x_i \mid y_i)
\]

\[P_T(JJ\langle s\rangle) \ast P_T(NN\mid JJ) \ast P_T(NN\mid NN) \ldots\]

\[P_E(\text{natural} \mid JJ) \ast P_E(\text{language} \mid NN) \ast P_E(\text{processing} \mid NN) \ldots\]
Restructuring HMM With Features

Normal HMM: \[ P(X, Y) = \prod_{i=1}^{I} P_E(x_i \mid y_i) \prod_{i=1}^{I+1} P_T(y_i \mid y_{i-1}) \]
Restructuring HMM With Features

Normal HMM:

\[ P(X, Y) = \prod_{i=1}^{I} P_E(x_i \mid y_i) \prod_{i=1}^{I+1} P_T(y_i \mid y_{i-1}) \]

Log Likelihood:

\[ \log P(X, Y) = \sum_{i=1}^{I} \log P_E(x_i \mid y_i) \sum_{i=1}^{I+1} \log P_T(y_i \mid y_{i-1}) \]
Restructuring HMM With Features

Normal HMM:
\[ P(X, Y) = \prod_{i=1}^{I} P_E(x_i \mid y_i) \prod_{i=1}^{I+1} P_T(y_i \mid y_{i-1}) \]

Log Likelihood:
\[ \log P(X, Y) = \sum_{i=1}^{I} \log P_E(x_i \mid y_i) \sum_{i=1}^{I+1} \log P_T(y_i \mid y_{i-1}) \]

Score
\[ S(X, Y) = \sum_{i=1}^{I} w_{E,y_i,x_i} \sum_{i=1}^{I+1} w_{T,y_{i-1},y_i} \]
Restructuring HMM With Features

Normal HMM:

\[ P(X, Y) = \prod_{i=1}^{I} P_E(x_i \mid y_i) \prod_{i=1}^{I+1} P_T(y_i \mid y_{i-1}) \]

Log Likelihood:

\[ \log P(X, Y) = \sum_{i=1}^{I} \log P_E(x_i \mid y_i) \sum_{i=1}^{I+1} \log P_T(y_i \mid y_{i-1}) \]

Score

\[ S(X, Y) = \sum_{i=1}^{I} w_{E,y_i,x_i} \sum_{i=1}^{I+1} w_{T,y_{i-1},y_i} \]

When:

\[ w_{E,y_i,x_i} = \log P_E(x_i \mid y_i) \quad w_{T,y_{i-1},y_i} = \log P_T(y_i \mid y_{i-1}) \]

\[ \log P(X, Y) = S(X, Y) \]
Example

\[ \phi(\text{I} \xrightarrow{\text{PRP}} \text{visited} \xrightarrow{\text{VBD}} \text{Nara} \xrightarrow{\text{NNP}} ) = \]

\[ \phi_{T,\text{PRP,VBD}}(X,Y_1) = 1 \quad \phi_{T,\text{VBD,NNP}}(X,Y_1) = 1 \quad \phi_{T,\text{NNP,}/S>(X,Y_1) = 1 \]

\[ \phi_{E,\text{PRP,"I"}}(X,Y_1) = 1 \quad \phi_{E,\text{VBD,\"visited\"}}(X,Y_1) = 1 \quad \phi_{E,\text{NNP,\"Nara\"}}(X,Y_1) = 1 \]

\[ \phi_{\text{CAPS,PRP}}(X,Y_1) = 1 \quad \phi_{\text{CAPS,NNP}}(X,Y_1) = 1 \]

\[ \phi_{\text{SUF,VBD,\"...ed\"}}(X,Y_1) = 1 \]

\[ \phi(\text{NNP} \xrightarrow{\text{PRP}} \text{visited} \xrightarrow{\text{VBD}} \text{NNP} \xrightarrow{\text{NNP}} ) = \]

\[ \phi_{T,\text{NNP,VBD}}(X,Y_2) = 1 \quad \phi_{T,\text{VBD,NNP}}(X,Y_2) = 1 \quad \phi_{T,\text{NNP,}/S>(X,Y_2) = 1 \]

\[ \phi_{E,\text{NNP,"I"}}(X,Y_2) = 1 \quad \phi_{E,\text{VBD,\"visited\"}}(X,Y_2) = 1 \quad \phi_{E,\text{NNP,\"Nara\"}}(X,Y_2) = 1 \]

\[ \phi_{\text{CAPS,NNP}}(X,Y_2) = 2 \]

\[ \phi_{\text{SUF,VBD,\"...ed\"}}(X,Y_2) = 1 \]
How to decode?

- We must find the POS sequence that satisfies:

\[ \hat{Y} = \arg\max_Y \sum_i w_i \phi_i(X, Y) \]

Solution: Viterbi algorithm
HMM Viterbi Algorithm

- **Forward step**, calculate the best path to a node
  - Find the path to each node with the lowest negative log probability

- **Backward step**, reproduce the path
First, calculate transition from <S> and emission of the first word for every POS

\[
\text{best_score}[^{1}\text{NN}] = -\log P_T(\text{NN}|<S>) + \text{log } P_E(I|\text{NN})
\]

\[
\text{best_score}[^{1}\text{JJ}] = -\log P_T(\text{JJ}|<S>) + \text{log } P_E(I|\text{JJ})
\]

\[
\text{best_score}[^{1}\text{VB}] = -\log P_T(\text{VB}|<S>) + \text{log } P_E(I|\text{VB})
\]

\[
\text{best_score}[^{1}\text{PRP}] = -\log P_T(\text{PRP}|<S>) + \text{log } P_E(I|\text{PRP})
\]

\[
\text{best_score}[^{1}\text{NNP}] = -\log P_T(\text{NNP}|<S>) + \text{log } P_E(I|\text{NNP})
\]
Forward Step: Middle Parts

- For middle words, calculate the minimum score for all possible previous POS tags

\[
\begin{align*}
\text{best_score}[\text{“2 NN”}] &= \min(
\text{best_score}[\text{“1 NN”}] + -\log P_T(\text{NN|NN}) + -\log P_E(\text{visited | NN}), \\
\text{best_score}[\text{“1 JJ”}] + -\log P_T(\text{NN|JJ}) + -\log P_E(\text{language | NN}), \\
\text{best_score}[\text{“1 VB”}] + -\log P_T(\text{NN|VB}) + -\log P_E(\text{language | NN}), \\
\text{best_score}[\text{“1 PRP”}] + -\log P_T(\text{NN|PRP}) + -\log P_E(\text{language | NN}), \\
\text{best_score}[\text{“1 NNP”}] + -\log P_T(\text{NN|NNP}) + -\log P_E(\text{language | NN}), \\
\cdots
\big)
\end{align*}
\]

\[
\begin{align*}
\text{best_score}[\text{“2 JJ”}] &= \min(
\text{best_score}[\text{“1 NN”}] + -\log P_T(\text{JJ|NN}) + -\log P_E(\text{language | JJ}), \\
\text{best_score}[\text{“1 JJ”}] + -\log P_T(\text{JJ|JJ}) + -\log P_E(\text{language | JJ}), \\
\text{best_score}[\text{“1 VB”}] + -\log P_T(\text{JJ|VB}) + -\log P_E(\text{language | JJ}), \\
\cdots
\big)
\end{align*}
\]
HMM Viterbi with Features

• Same as probabilities, use feature weights

\[
\text{best_score["1 NN"]} = w_{T,<S>,NN} + w_{E,NN,I} \\
\text{best_score["1 JJ"]} = w_{T,<S>,JJ} + w_{E,JJ,I} \\
\text{best_score["1 VB"]} = w_{T,<S>,VB} + w_{E,VB,I} \\
\text{best_score["1 PRP"]} = w_{T,<S>,PRP} + w_{E,PRP,I} \\
\text{best_score["1 NNP"]} = w_{T,<S>,NNP} + w_{E,NNP,I}
\]
HMM Viterbi with Features

- Can add additional features

\[
\begin{align*}
0:<S> & \quad \text{best_score["1 NN"]} = w_{T,<S>,NN} + w_{E,NN,I} + w_{CAPS,NN} \\
1:NN & \quad \text{best_score["1 JJ"]} = w_{T,<S>,JJ} + w_{E,JJ,I} + w_{CAPS,JJ} \\
1:VB & \quad \text{best_score["1 VB"]} = w_{T,<S>,VB} + w_{E,VB,I} + w_{CAPS,VB} \\
1:PRP & \quad \text{best_score["1 PRP"]} = w_{T,<S>,PRP} + w_{E,PRP,I} + w_{CAPS,PRP} \\
1:NNP & \quad \text{best_score["1 NNP"]} = w_{T,<S>,NNP} + w_{E,NNP,I} + w_{CAPS,NNP} \\
\end{align*}
\]
Learning in the Structured Perceptron

- Remember the perceptron algorithm
- If there is a mistake:
  \[ \mathbf{w} \leftarrow \mathbf{w} + y\phi(x) \]
  - increase score of positive examples
  - decrease score of negative examples
- What is positive/negative in structured perceptron?
Learning in the Structured Perceptron

- Positive example, correct feature vector:

\[ \phi \left( \begin{array}{c}
I \\
\text{visited} \\
\text{Nara}
\end{array} \right) \]

- Negative example, incorrect feature vector:

\[ \phi \left( \begin{array}{c}
\text{NNP} \\
\text{visited} \\
\text{NNP}
\end{array} \right) \]
Choosing an Incorrect Feature Vector

- There are many incorrect feature vectors!

\[ \varphi(\text{I visited Nara}) \]

\[ \varphi(\text{I visited Nara}) \]

\[ \varphi(\text{I visited Nara}) \]
Choosing an Incorrect Feature Vector

- Answer: We update using the incorrect answer with the highest score
  \[ \hat{Y} = \arg\max_Y \sum_i w_i \phi_i(X, Y) \]

- Our update rule becomes:
  \[ w \leftarrow w + \phi(X, Y') - \phi(X, \hat{Y}) \]
  - \( Y' \) is the correct answer
  - Note: If highest scoring answer is correct, no change
Structured Perceptron Algorithm

- create map $w$
  for $I$ iterations
  for each labeled pair $X, Y_{\text{prime}}$ in the data
    $Y_{\text{hat}} = \text{hmm_viterbi}(w, X)$
    $\phi_{\text{prime}} = \text{create_features}(X, Y_{\text{prime}})$
    $\phi_{\text{hat}} = \text{create_features}(X, Y_{\text{hat}})$
    $w += \phi_{\text{prime}} - \phi_{\text{hat}}$
Recap: POS tagging...

• A structured prediction task
  – Hidden Markov Models
    • Decoding with Viterbi algorithm
    • Supervised training: counts-based estimates
  – Structured Perceptron
    • Decoding with Viterbi algorithm
    • Supervised training: perceptron algorithm with structured weight updates

• First step toward modeling syntactic structure