Sequential Data Modeling - The Structured Perceptron

Graham Neubig
Nara Institute of Science and Technology (NAIST)

Prediction Problems

Given x, predict y

Prediction Problems

Given x,	predict y	
A book review Oh, man I love this book! This book is so boring...	Is it positive? yes no	Binary Prediction (2 choices)
A tweet On the way to the park! 公園に行くなう！	Its language English Japanese	Multi-class Prediction (several choices)
A sentence I read a book	Its parts-of-speech N VBD DET NN I read a book	Structured Prediction (millions of choices)

Sequential prediction is a subset

Simple Prediction:
The Perceptron Model

Example we will use:

- Given an introductory sentence from Wikipedia
- Predict whether the article is about a person

Given		Predict
Gonso was a Sanron sect priest (754-827) in the late Nara and early Heian periods.	→	Yes!
Shichikuzan Chigogataki Fudomyoo is a historical site located at Magura, Maizuru City, Kyoto Prefecture.	→	No!

- This is binary classification (of course!)

How do We Predict?

Gonso was a Sanron sect priest (754 – 827) in the late Nara and early Heian periods .

Shichikuzan Chigogataki Fudomyoo is a historical site located at Magura , Maizuru City , Kyoto Prefecture .

How do We Predict?

Contains “priest” → probably person!

Contains “(<#>-<#>)” → probably person!

Gonso was a Sanron sect priest (754 – 827) in the late Nara and early Heian periods .

Contains “site” → probably not person!

Shichikuzan Chigogataki Fudomyoo is a historical site located at Magura , Maizuru City , Kyoto Prefecture .

Contains “Kyoto Prefecture” → probably not person!

Combining Pieces of Information

- Each element that helps us predict is a *feature*

contains “priest” contains “(<#>-<#>)”
contains “site” contains “Kyoto Prefecture”

- Each feature has a weight, *positive* if it indicates “yes”, and *negative* if it indicates “no”

$w_{\text{contains “priest”}} = 2$ $w_{\text{contains “(<\#>-<\#>)”}} = 1$
$w_{\text{contains “site”}} = -3$ $w_{\text{contains “Kyoto Prefecture”}} = -1$

- For a new example, sum the weights

Kuya (903-972) was a priest born in Kyoto Prefecture.

$2 + -1 + 1 = 2$

- If the sum is at least 0: “yes”, otherwise: “no”

Let me Say that in Math!

$$
\begin{aligned}
y &= \operatorname{sign}(\mathbf{w} \cdot \boldsymbol{\varphi}(x)) \\
&= \operatorname{sign}\left(\sum_{i=1}^{I} w_i \cdot \varphi_i(x)\right)
\end{aligned}
$$

- **x**: the input
- **φ(x)**: vector of feature functions $\{\varphi_1(x), \varphi_2(x), \ldots, \varphi_I(x)\}$
- **w**: the weight vector $\{w_1, w_2, \ldots, w_I\}$
- **y**: the prediction, +1 if "yes", -1 if "no"
  - (sign(v) is +1 if v >= 0, -1 otherwise)

Example Feature Functions: Unigram Features

- Equal to "number of times a particular word appears"

x = A site , located in Maizuru , Kyoto

$\varphi_{\text{unigram “A”}}(x) = 1$ $\varphi_{\text{unigram “site”}}(x) = 1$ $\varphi_{\text{unigram “,”}}(x) = 2$

$\varphi_{\text{unigram “located”}}(x) = 1$ $\varphi_{\text{unigram “in”}}(x) = 1$

$\varphi_{\text{unigram “Maizuru”}}(x) = 1$ $\varphi_{\text{unigram “Kyoto”}}(x) = 1$

$\left.\begin{array}{l} \varphi_{\text{unigram “the”}}(x) = 0 \quad \varphi_{\text{unigram “temple”}}(x) = 0 \\ \ldots \end{array}\right\}$ The rest are all 0

- For convenience, we use feature names ($\varphi_{\text{unigram “A”}}$) instead of feature indexes (φ_1)

Calculating the Weighted Sum

x = A site , located in Maizuru , Kyoto

$$
\begin{array}{llcllcl}
\varphi_{\text{unigram "A"}}(x) & = 1 & & w_{\text{unigram "a"}} & = 0 & & 0 \; + \\
\varphi_{\text{unigram "site"}}(x) & = 1 & & w_{\text{unigram "site"}} & = -3 & & -3 \; + \\
\varphi_{\text{unigram "located"}}(x) & = 1 & & w_{\text{unigram "located"}} & = 0 & & 0 \; + \\
\varphi_{\text{unigram "Maizuru"}}(x) & = 1 & & w_{\text{unigram "Maizuru"}} & = 0 & & 0 \; + \\
\varphi_{\text{unigram ","}}(x) & = 2 & * & w_{\text{unigram ","}} & = 0 & = & 0 \; + \\
\varphi_{\text{unigram "in"}}(x) & = 1 & & w_{\text{unigram "in"}} & = 0 & & 0 \; + \\
\varphi_{\text{unigram "Kyoto"}}(x) & = 1 & & w_{\text{unigram "Kyoto"}} & = 0 & & 0 \; + \\
\varphi_{\text{unigram "priest"}}(x) & = 0 & & w_{\text{unigram "priest"}} & = 2 & & 0 \; + \\
\varphi_{\text{unigram "black"}}(x) & = 0 & & w_{\text{unigram "black"}} & = 0 & & 0 \; + \\
 & & \ldots & & & & \ldots \\
 & & & & & & = \\
 & & & & & & -3 \rightarrow \text{No!}
\end{array}
$$

Learning Weights Using the Perceptron Algorithm

Learning Weights

- Manually creating weights is hard
  - Many many possible useful features
  - Changing weights changes results in unexpected ways
- Instead, we can learn from labeled data

y	x
1	FUJIWARA no Chikamori (year of birth and death unknown) was a samurai and poet who lived at the end of the Heian period .
1	Ryonen (1646 - October 29 , 1711) was a Buddhist nun of the Obaku Sect who lived from the early Edo period to the mid-Edo period .
-1	A moat settlement is a village surrounded by a moat .
-1	Fushimi Momoyama Athletic Park is located in Momoyama-cho , Kyoto City , Kyoto Prefecture .

Online Learning

```
create map w
for I iterations
    for each labeled pair x, y in the data
        phi = CREATE_FEATURES(x)
        y' = PREDICT_ONE(w, phi)
        if y' != y
            UPDATE_WEIGHTS(w, phi, y)
```

- In other words
  - Try to classify each training example
  - Every time we make a mistake, update the weights
- Many different online learning algorithms
  - The most simple is the perceptron

Perceptron Weight Update

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y\,\varphi(x)$$

- In other words:
  - If y=1, increase the weights for features in φ(x)
    - Features for positive examples get a higher weight
  - If y=-1, decrease the weights for features in φ(x)
    - Features for negative examples get a lower weight

→ Every time we update, our predictions get better!

Example: Initial Update

- Initialize **w=0**

x = A site , located in Maizuru , Kyoto y = -1

$$\boldsymbol{w}\cdot\varphi(x)=0 \qquad y'=\text{sign}(\boldsymbol{w}\cdot\varphi(x))=1$$

$$\downarrow$$

$$y' \neq y$$

$$\downarrow$$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y\varphi(x)$$

$$\downarrow$$

$w_{\text{unigram “Maizuru”}}$ = -1
$w_{\text{unigram “,”}}$ = -2
$w_{\text{unigram “in”}}$ = -1
$w_{\text{unigram “Kyoto”}}$ = -1
$w_{\text{unigram “A”}}$ = -1
$w_{\text{unigram “site”}}$ = -1
$w_{\text{unigram “located”}}$ = -1

Example: Second Update

x = Shoken , monk born in Kyoto   y = 1

-2   -1   -1

$$\boldsymbol{w}\cdot\varphi(x)=-4 \qquad y'=\text{sign}(\boldsymbol{w}\cdot\varphi(x))=-1$$

$$y' \neq y$$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y\,\varphi(x)$$

$w_{\text{unigram “Maizuru”}} = -1$

$w_{\text{unigram “,”}} = -1$

$w_{\text{unigram “in”}} = 0$

$w_{\text{unigram “Kyoto”}} = 0$

$w_{\text{unigram “A”}} = -1$

$w_{\text{unigram “site”}} = -1$

$w_{\text{unigram “located”}} = -1$

$w_{\text{unigram “Shoken”}} = 1$

$w_{\text{unigram “monk”}} = 1$

$w_{\text{unigram “born”}} = 1$

Review:
The HMM Model

Part of Speech (POS) Tagging

- Given a sentence X, predict its part of speech sequence Y

Natural language processing (NLP) is a field of computer science

JJ NN NN -LRB- NN -RRB- VBZ DT NN IN NN NN

- A type of "structured" prediction, from two weeks ago
- How can we do this? Any ideas?

Probabilistic Model for Tagging

- “Find the most probable tag sequence, given the sentence”

Natural language processing (NLP) is a field of computer science

JJ NN NN LRB NN RRB VBZ DT NN IN NN NN

$$\underset{Y}{\operatorname{argmax}}\, P(Y|X)$$

- Any ideas?

Generative Sequence Model

- First decompose probability using Bayes' law

$$\underset{Y}{\operatorname{argmax}}\, P(Y|X) = \underset{Y}{\operatorname{argmax}} \frac{P(X|Y)\,P(Y)}{P(X)}$$

$$= \underset{Y}{\operatorname{argmax}}\, P(X|Y)\,P(Y)$$

Model of word/POS interactions
"natural" is probably a JJ

Model of POS/POS interactions
NN comes after DET

- Also sometimes called the "noisy-channel model"

Hidden Markov Models (HMMs) for POS Tagging

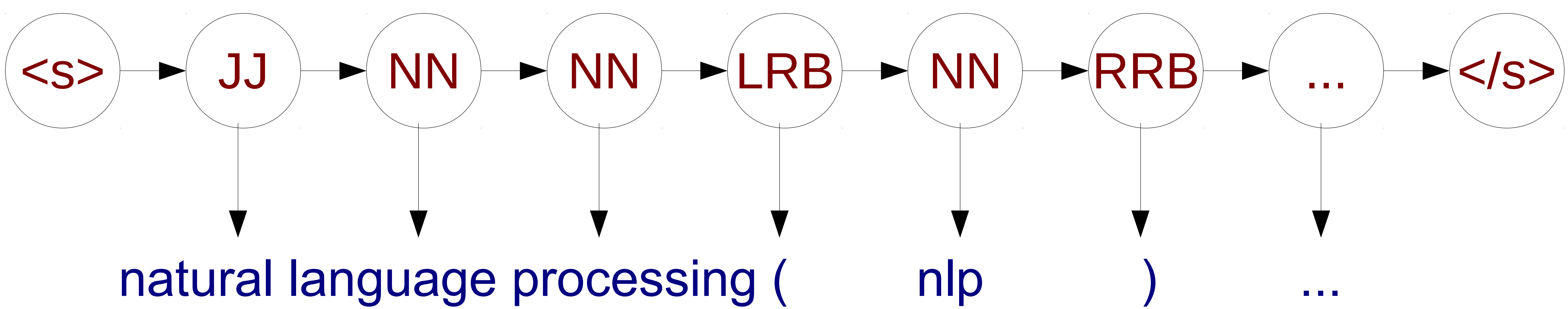
- POS→POS **transition** probabilities
  - Like a bigram model!

$$P(Y) \approx \prod_{i=1}^{I+1} P_T(y_i \mid y_{i-1})$$

- POS→Word **emission** probabilities

$$P(X \mid Y) \approx \prod_{1}^{I} P_E(x_i \mid y_i)$$

$P_T(JJ|<s>) * P_T(NN|JJ) * P_T(NN|NN)$ …



$P_E(natural|JJ) * P_E(language|NN) * P_E(processing|NN)$ …

Learning Markov Models (with tags)

- Count the number of occurrences in the corpus and

natural language processing (nlp) is …

c(JJ→natural)++ c(NN→language)++ …

<s> → JJ → NN NN LRB NN RRB VB … </s>

c(<s> JJ)++ c(JJ NN)++ …

- Divide by context to get probability

$$P_T(LRB|NN) = c(NN\ LRB)/c(NN) = 1/3$$

$$P_E(language|NN) = c(NN \rightarrow language)/c(NN) = 1/3$$

Remember: 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
  - This is easy, almost the same as word segmentation

Forward Step: Part 1

- First, calculate transition from <S> and emission of the first word for every POS

I

0:<S>

1:NN best_score[“1 NN”] = -log P_T(NN|<S>) + -log P_E(I | NN)

1:JJ best_score[“1 JJ”] = -log P_T(JJ|<S>) + -log P_E(I | JJ)

1:VB best_score[“1 VB”] = -log P_T(VB|<S>) + -log P_E(I | VB)

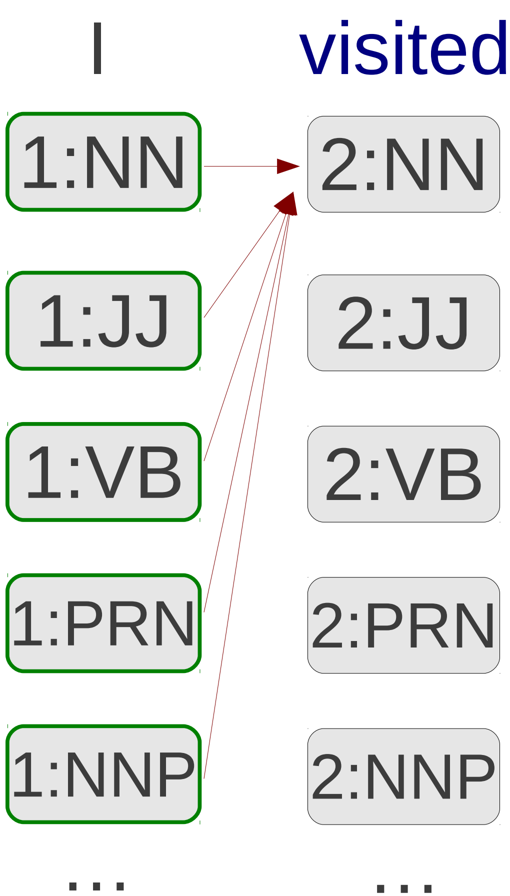
1:PRN best_score[“1 PRN”] = -log P_T(PRN|<S>) + -log P_E(I | PRN)

1:NNP best_score[“1 NNP”] = -log P_T(NNP|<S>) + -log P_E(I | NNP)

…

Forward Step: Middle Parts

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



best_score[“2 NN”] = min(
best_score[“1 NN”] + -log P_T(NN|NN) + -log P_E(visited | NN),
best_score[“1 JJ”] + -log P_T(NN|JJ) + -log P_E(visited | NN),
best_score[“1 VB”] + -log P_T(NN|VB) + -log P_E(visited | NN),
best_score[“1 PRN”] + -log P_T(NN|PRN) + -log P_E(visited | NN),
best_score[“1 NNP”] + -log P_T(NN|NNP) + -log P_E(visited | NN),
...
)

best_score[“2 JJ”] = min(
best_score[“1 NN”] + -log P_T(JJ|NN) + -log P_E(visited | JJ),
best_score[“1 JJ”] + -log P_T(JJ|JJ) + -log P_E(visited | JJ),
best_score[“1 VB”] + -log P_T(JJ|VB) + -log P_E(visited | JJ),
...

The Structured Perceptron

So Far, We Have Learned

Classifiers	Generative Models
Perceptron	HMM
Lots of features	Conditional probabilities
Binary prediction	Structured prediction

Structured Perceptron

Classifiers	Generative Models
Perceptron	HMM
Lots of features	Conditional probabilities
Binary prediction	Structured prediction

Structured perceptron →
Classification with lots of features
over structured models!

Why are Features Good?

- Can easily try many different ideas
  - Are capital letters usually nouns?
  - Are words that end with -ed usually verbs? -ing?

Restructuring HMM With Features

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

Restructuring HMM With Features

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

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

Restructuring HMM With Features

Normal HMM: $P(X,Y)=\prod_1^I P_E(x_i|y_i)\prod_{i=1}^{I+1} P_T(y_i|y_{i-1})$

Log Likelihood: $\log P(X,Y)=\sum_1^I \log P_E(x_i|y_i)\sum_{i=1}^{I+1} \log P_T(y_i|y_{i-1})$

Score $S(X,Y)=\sum_1^I w_{E,y_i,x_i}\sum_{i=1}^{I+1} w_{T,y_{i-1},y_i}$

Restructuring HMM With Features

Normal HMM:

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

Log Likelihood:

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

Score

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

When: $w_{E,y_i,x_i}=\log P_E(x_i|y_i)$ $\quad w_{T,y_{i-1},y_i}=\log P_T(y_i|y_{i-1})$

$$\log P(X,Y) = S(X,Y)$$

Example

$$\varphi\left(\begin{array}{cccc} & \text{I} & \text{visited} & \text{Nara} \\ & \uparrow & \uparrow & \uparrow \\ \rightarrow & \text{PRN} \rightarrow & \text{VBD} \rightarrow & \text{NNP} \rightarrow \end{array}\right) =$$

$\varphi_{\text{T,<S>,PRN}}(X,Y_1) = 1$ $\varphi_{\text{T,PRN,VBD}}(X,Y_1) = 1$ $\varphi_{\text{T,VBD,NNP}}(X,Y_1) = 1$ $\varphi_{\text{T,NNP,</S>}}(X,Y_1) = 1$

$\varphi_{\text{E,PRN,"I"}}(X,Y_1) = 1$ $\varphi_{\text{E,VBD,"visited"}}(X,Y_1) = 1$ $\varphi_{\text{E,NNP,"Nara"}}(X,Y_1) = 1$

$\varphi_{\text{CAPS,PRN}}(X,Y_1) = 1$ $\varphi_{\text{CAPS,NNP}}(X,Y_1) = 1$ $\varphi_{\text{SUF,VBD,"...ed"}}(X,Y_1) = 1$

$$\varphi\left(\begin{array}{cccc} & \text{I} & \text{visited} & \text{Nara} \\ & \uparrow & \uparrow & \uparrow \\ \rightarrow & \text{NNP} \rightarrow & \text{VBD} \rightarrow & \text{NNP} \rightarrow \end{array}\right) =$$

$\varphi_{\text{T,<S>,NNP}}(X,Y_1) = 1$ $\varphi_{\text{T,NNP,VBD}}(X,Y_1) = 1$ $\varphi_{\text{T,VBD,NNP}}(X,Y_1) = 1$ $\varphi_{\text{T,NNP,</S>}}(X,Y_1) = 1$

$\varphi_{\text{E,NNP,"I"}}(X,Y_1) = 1$ $\varphi_{\text{E,VBD,"visited"}}(X,Y_1) = 1$ $\varphi_{\text{E,NNP,"Nara"}}(X,Y_1) = 1$

$\varphi_{\text{CAPS,NNP}}(X,Y_1) = 2$ $\varphi_{\text{SUF,VBD,"...ed"}}(X,Y_1) = 1$

Finding the Best Solution

- We must find the POS sequence that satisfies:

$$\hat{Y} = \operatorname{argmax}_Y \sum_i w_i \varphi_i(X, Y)$$

HMM Viterbi with Features

- Same as probabilities, use feature weights

I

0:<S>

1:NN best_score["1 NN"] = $w_{T,<S>,NN} + w_{E,NN,I}$

1:JJ best_score["1 JJ"] = $w_{T,<S>,JJ} + w_{E,JJ,I}$

1:VB best_score["1 VB"] = $w_{T,<S>,VB} + w_{E,VB,I}$

1:PRN best_score["1 PRN"] = $w_{T,<S>,PRN} + w_{E,PRN,I}$

1:NNP best_score["1 NNP"] = $w_{T,<S>,NNP} + w_{E,NNP,I}$

…

HMM Viterbi with Features

- Can add additional features

I

0:<S> → 1:NN best_score["1 NN"] = $w_{T,<S>,NN} + w_{E,NN,I} + w_{CAPS,NN}$

0:<S> → 1:JJ best_score["1 JJ"] = $w_{T,<S>,JJ} + w_{E,JJ,I} + w_{CAPS,JJ}$

0:<S> → 1:VB best_score["1 VB"] = $w_{T,<S>,VB} + w_{E,VB,I} + w_{CAPS,VB}$

0:<S> → 1:PRN best_score["1 PRN"] = $w_{T,<S>,PRN} + w_{E,PRN,I} + w_{CAPS,PRN}$

0:<S> → 1:NNP best_score["1 NNP"] = $w_{T,<S>,NNP} + w_{E,NNP,I} + w_{CAPS,NNP}$

…

Learning In the Structured Perceptron

- Remember the perceptron algorithm
- If there is a mistake:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y\,\boldsymbol{\varphi}(x)$$

- Update weights to:
  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:

φ(I visited Nara)
→ PRN → VBD → NNP →

- Negative example, incorrect feature vector:

φ(I visited Nara)
→ NNP → VBD → NNP →

Choosing an Incorrect Feature Vector

- There are too many incorrect feature vectors!

$$\varphi\left(\begin{array}{cccc} & \text{I} & \text{visited} & \text{Nara} \\ & \uparrow & \uparrow & \uparrow \\ \rightarrow & \text{NNP} \rightarrow & \text{VBD} \rightarrow & \text{NNP} \rightarrow \end{array}\right)$$

$$\varphi\left(\begin{array}{cccc} & \text{I} & \text{visited} & \text{Nara} \\ & \uparrow & \uparrow & \uparrow \\ \rightarrow & \text{PRN} \rightarrow & \text{VBD} \rightarrow & \text{NN} \rightarrow \end{array}\right)$$

$$\varphi\left(\begin{array}{cccc} & \text{I} & \text{visited} & \text{Nara} \\ & \uparrow & \uparrow & \uparrow \\ \rightarrow & \text{PRN} \rightarrow & \text{VB} \rightarrow & \text{NNP} \rightarrow \end{array}\right)$$

- Which do we use?

Choosing an Incorrect Feature Vector

- Answer: We update using the incorrect answer with the highest score:

$$\hat{Y} = \mathrm{argmax}_{Y} \sum_{i} w_{i} \varphi_{i}(X, Y)$$

- Our update rule becomes:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{\varphi}(X, Y') - \boldsymbol{\varphi}(X, \hat{Y})$$

  - (Y' is the correct answer)
- Note: If highest scoring answer is correct, no change

Example

$\varphi_{T,<S>,PRN}(X,Y_1) = 1$ $\varphi_{T,PRN,VBD}(X,Y_1) = 1$ $\varphi_{T,VBD,NNP}(X,Y_1) = 1$ $\varphi_{T,NNP,</S>}(X,Y_1) = 1$

$\varphi_{E,PRN,\text{"I"}}(X,Y_1) = 1$ $\varphi_{E,VBD,\text{"visited"}}(X,Y_1) = 1$ $\varphi_{E,NNP,\text{"Nara"}}(X,Y_1) = 1$

$\varphi_{CAPS,PRN}(X,Y_1) = 1$ $\varphi_{CAPS,NNP}(X,Y_1) = 1$ $\varphi_{SUF,VBD,\text{"...ed"}}(X,Y_1) = 1$

\-

$\varphi_{T,<S>,NNP}(X,Y_1) = 1$ $\varphi_{T,NNP,VBD}(X,Y_1) = 1$ $\varphi_{T,VBD,NNP}(X,Y_1) = 1$ $\varphi_{T,NNP,</S>}(X,Y_1) = 1$

$\varphi_{E,NNP,\text{"I"}}(X,Y_1) = 1$ $\varphi_{E,VBD,\text{"visited"}}(X,Y_1) = 1$ $\varphi_{E,NNP,\text{"Nara"}}(X,Y_1) = 1$

$\varphi_{CAPS,NNP}(X,Y_1) = 2$ $\varphi_{SUF,VBD,\text{"...ed"}}(X,Y_1) = 1$

=

$\varphi_{T,<S>,PRN}(X,Y_1) = 1$ $\varphi_{T,PRN,VBD}(X,Y_1) = 1$ $\varphi_{T,VBD,NNP}(X,Y_1) = 0$ $\varphi_{T,NNP,</S>}(X,Y_1) = 0$

$\varphi_{T,<S>,NNP}(X,Y_1) = -1$ $\varphi_{T,NNP,VBD}(X,Y_1) = -1$

$\varphi_{E,PRN,\text{"I"}}(X,Y_1) = 1$ $\varphi_{E,VBD,\text{"visited"}}(X,Y_1) = 0$ $\varphi_{E,NNP,\text{"Nara"}}(X,Y_1) = 0$

$\varphi_{E,NNP,\text{"I"}}(X,Y_1) = -1$

$\varphi_{CAPS,NNP}(X,Y_1) = -1$ $\varphi_{CAPS,PRN}(X,Y_1) = 1$ $\varphi_{SUF,VBD,\text{"...ed"}}(X,Y_1) = 0$

Structured Perceptron Algorithm

```
create map w
for I iterations
    for each labeled pair X, Y_prime in the data
        Y_hat = HMM_VITERBI(w, X)
        phi_prime = CREATE_FEATURES(X, Y_prime)
        phi_hat = CREATE_FEATURES(X, Y_hat)
        w += phi_prime - phi_hat
```

Conclusion

- The structured perceptron is a discriminative structured prediction model
  - HMM: generative structured prediction
  - Perceptron: discriminative binary prediction
- It can be used for many problems
  - Prediction of

Thank You!