NLP Programming Tutorial 11 - The Structured Perceptron

Graham Neubig
Nara Institute of Science and Technology (NAIST)

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)

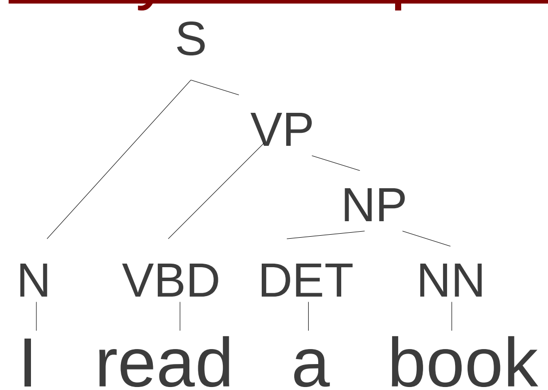
A tweet
On the way to the park!
公園に行くなう!

Its language
English
Japanese

Multi-class Prediction
(several choices)

A sentence
I read a book

Its syntactic parse

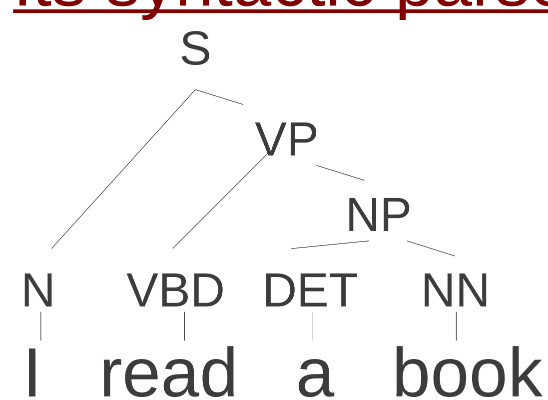


Structured Prediction
(millions of choices)

Prediction Problems

Given x, predict y

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 syntactic parse	Most NLP Problems! Structured Prediction (millions of choices)



So Far, We Have Learned

Classifiers	Generative Models
Perceptron, SVM, Neural Net	HMM POS Tagging CFG Parsing
Lots of features	Conditional probabilities
Binary prediction	Structured prediction

Structured Perceptron

Classifiers	Generative Models
Perceptron, SVM, Neural Net	HMM POS Tagging CFG Parsing
Lots of features	Conditional probabilities
Binary prediction	Structured prediction

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

Uses of Structured Perceptron (or Variants)

- **POS Tagging with HMMs**
  Collins “Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms” ACL02
- **Parsing**
  Huang+ “Forest Reranking: Discriminative Parsing with Non-Local Features” ACL08
- **Machine Translation**
  Liang+ “An End-to-End Discriminative Approach to Machine Translation” ACL06
  (Neubig+ “Inducing a Discriminative Parser for Machine Translation Reordering, EMNLP12”, Plug :))
- **Discriminative Language Models**
  Roark+ “Discriminative Language Modeling with Conditional Random Fields and the Perceptron Algorithm” ACL04

Example: 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

Hidden Markov Models (HMMs) for POS Tagging

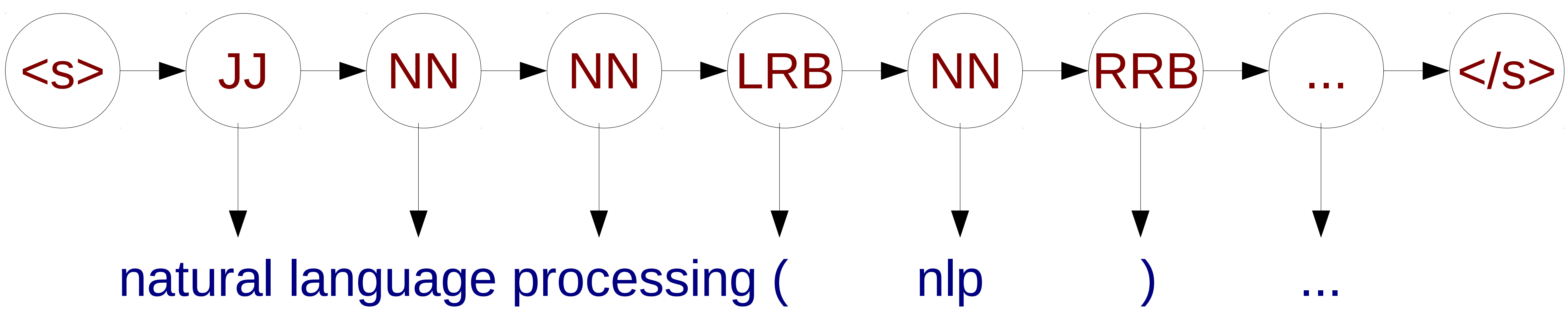
- POS→POS **transition** probabilities
  - Like a bigram model!

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

- POS→Word **emission** probabilities

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

$P_T(JJ|\text{<s>}) * P_T(NN|JJ) * P_T(NN|NN)$ …



$P_E(\text{natural}|JJ) * P_E(\text{language}|NN) * P_E(\text{processing}|NN)$ …

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

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

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

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

Example

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

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

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

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

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

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

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

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

Finding the Best Solution

- We must find the POS sequence that satisfies:

$$\hat{Y} = \text{argmax}_Y \sum_i w_i \phi_i(X, Y)$$

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)

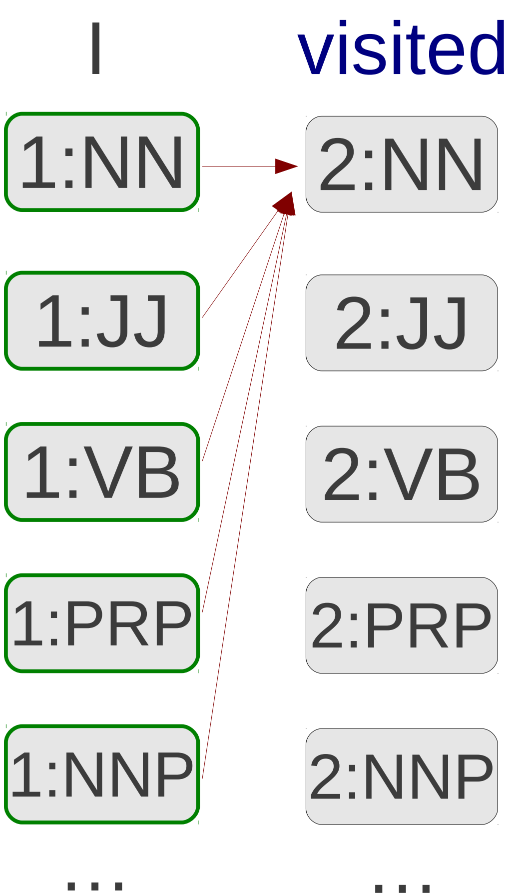
1:PRP best_score["1 PRP"] = -log P_T(PRP|<S>) + -log P_E(I | PRP)

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(language | NN),
 best_score["1 VB"] + -log P_T(NN|VB) + -log P_E(language | NN),
 best_score["1 PRP"] + -log P_T(NN|PRP) + -log P_E(language | NN),
 best_score["1 NNP"] + -log P_T(NN|NNP) + -log P_E(language | NN),
 ...
)

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

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:PRP best_score[“1 PRP”] = $w_{T,<S>,PRP} + w_{E,PRP,I}$

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"] = $w_{T,<S>,NN} + w_{E,NN,I} + w_{CAPS,NN}$

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

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

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

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:

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

- 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 / PRP VBD NNP)

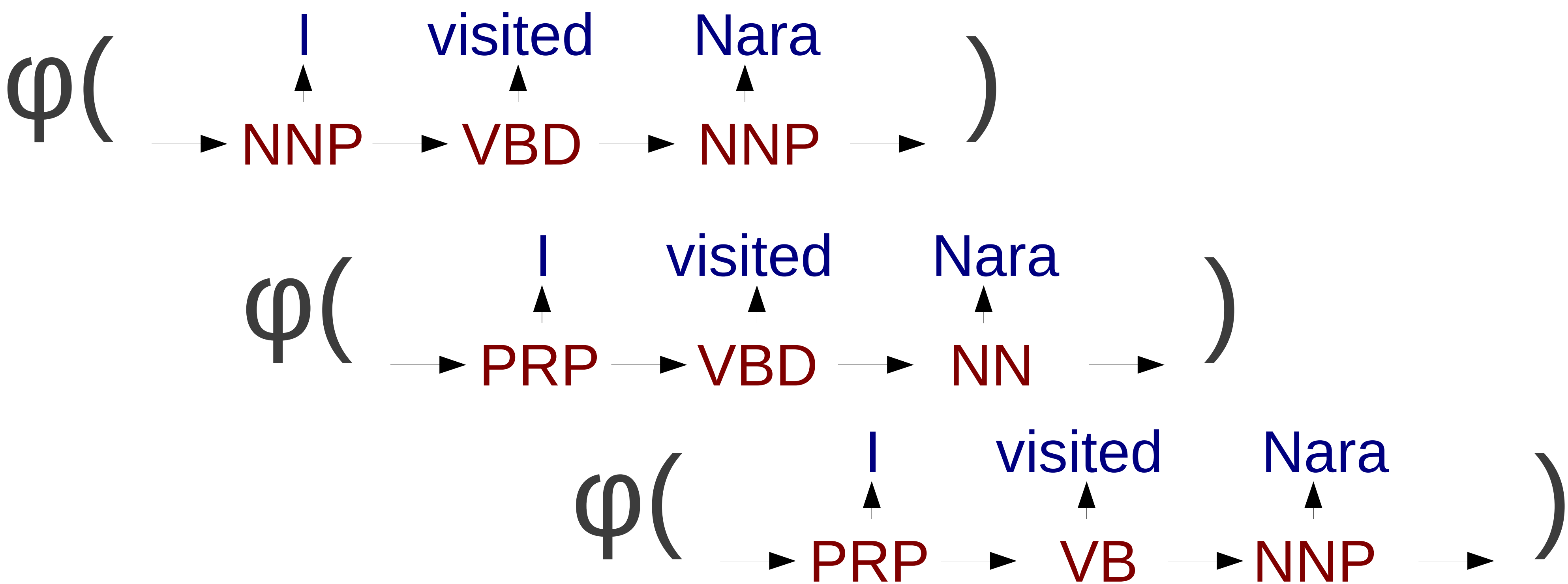
$$\varphi\left(\begin{array}{ccc} \text{I} & \text{visited} & \text{Nara} \\ \uparrow & \uparrow & \uparrow \\ \to \text{PRP} \to & \text{VBD} \to & \text{NNP} \to \end{array}\right)$$

- Negative example, incorrect feature vector:

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

Choosing an Incorrect Feature Vector

- There are too many incorrect feature vectors!



- Which do we use?

Choosing an Incorrect Feature Vector

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

$$\hat{Y} = \text{argmax}_Y \sum_i w_i \phi_i(X, Y)$$

- Our update rule becomes:

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

  - (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_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

Creating HMM Features

- Make “create features” for each transition, emission

`CREATE_TRANS(`NNP,VBD`)` → φ[“T,NNP,VBD”] = 1

`CREATE_EMIT(`NNP,Nara`)` → φ[“E,NNP,Nara”] = 1
φ[“CAPS,NNP”] = 1

Creating HMM Features

- The CREATE_FEATURES function does for all words

```
CREATE_FEATURES(X, Y):
    create map phi
    for i in 0 .. |Y|:
        if i == 0: first_tag = "<s>"
        else:      first_tag = Y[i-1]
        if i == |Y|: next_tag = "</s>"
        else:        next_tag = Y[i]
        phi += CREATE_TRANS(first_tag, next_tag)
    for i in 0 .. |Y|-1:
        phi += CREATE_EMIT(Y[i], X[i])
    return phi
```

Viterbi Algorithm Forward Step

split *line* **into** *words*
I = length*(words)*
make maps *best_score, best_edge*
best_score[“0 <s>”] = 0   # Start with <s>
best_edge[“0 <s>”] = NULL
for *i* **in** 0 … *I*-1:
    for each *prev* **in** keys of *possible_tags*
        for each *next* **in** keys of *possible_tags*
            if best_score[“*i prev*”] **and** transition[“prev next”] **exist**
                score = best_score[“i prev”] +
                    -log P_T(next|prev) + -log P_E(word[i]|next)
                    w*(CREATE_T(prev,next)+CREATE_E(next,word[i]))
                if *best_score*[“*i+1 next*”] **is** new **or** < *score*
                    best_score[“*i+1 next*”] = score
                    best_edge[“*i+1 next*”] = “*i prev*”
Finally, do the same for </s>

Exercise

Exercise

- Write train-hmm-percep and test-hmm-percep
- Test the program
  - Input: `test/05-{train,test}-input.txt`
  - Answer: `test/05-{train,test}-answer.txt`
- Train an HMM model on `data/wiki-en-train.norm_pos` and run the program on `data/wiki-en-test.norm`
- Measure the accuracy of your tagging with `script/gradepos.pl data/wiki-en-test.pos my_answer.pos`
- Report the accuracy (compare to standard HMM)
- Challenge:
  create new features
  use training with margin or regularization

Thank You!