WestlakeNLP

Natural Language Processing

Yue Zhang
Westlake University

西湖大學
WESTLAKE UNIVERSITY

Chapter 8

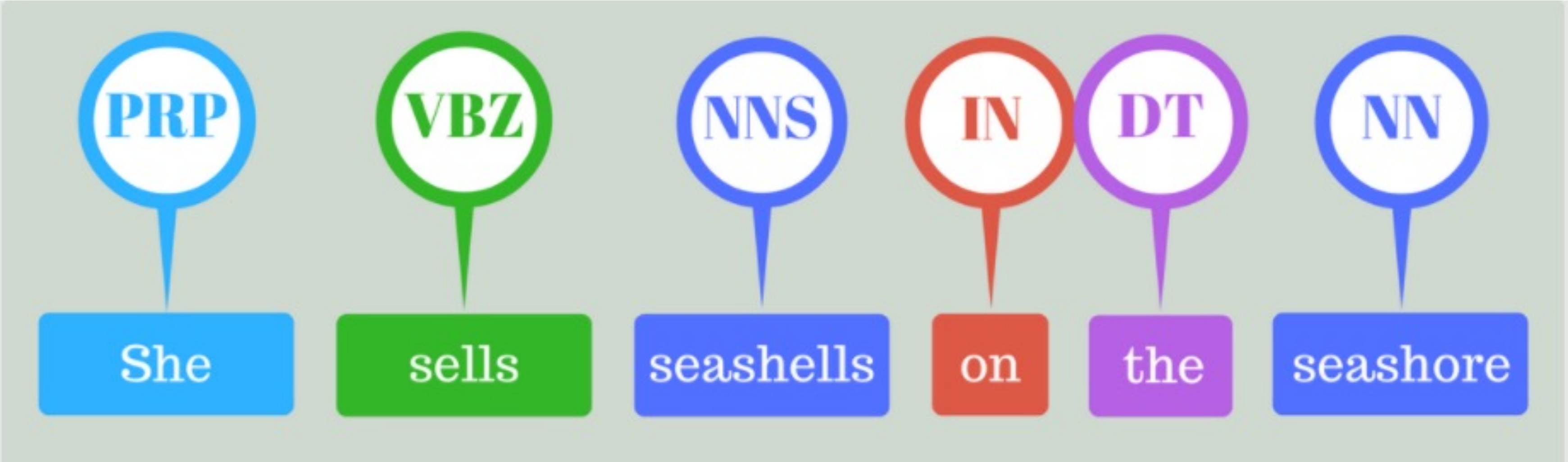
Discriminative Sequence Labeling

Contents

Contents

A sequence labeling task: POS Tagging

- Input: a word sequence $W_{i:n}$
- Output: the most probable tag sequence $\hat{T}_{1:n}$



Discriminative Model

- Generative Model
  - Estimate joint probabilities $P(T_{1:n}, W_{1:n})$
  - Examples: HMM, Naive Bayes
- Discriminative Model
  - Calculates $P(T_{1:n}|W_{1:n})$ directly
  - Examples: Perceptron, SVM, log-linear models
  - Advantage: rich features
  - Disadvantage: cannot enumerate $T_{1:n}$ values.

Contents

Local trained discriminative model

- Simply $P(T_{1:n}|W_{1:n})$ before parameterization using features.
- Factorize the sequence level probability by **chain rule**:

$$P(T_{1:n}|W_{1:n}) = \prod_{i=1}^{n} P(t_i|T_{1:i-1}, W_{1:n}) \approx \prod_{i=1}^{n} P(t_i|T_{i-k:i-1}, W_{1:n})$$

- Model target $P(t_i|T_{i-k:i-1}, W_{1:n})$

Local trained discriminative model

- A log-linear model (**maximum entropy Markov model** (MEMM)):

$$P(t_i = t | T_{i-k:i-1}, W_{1:n}) = \frac{\exp(\vec{\theta} \cdot \vec{\phi}(t_i = t, T_{i-k:i-1}, W_{1:n}))}{\sum_{t' \in L} \exp\left(\vec{\theta} \cdot \vec{\phi}(t_i = t', T_{i-k:i-1}, W_{1:n})\right)}$$

  - $\vec{\phi}$ denotes the feature vector
  - L denotes the set of all possible labels
  - $\vec{\theta}$ denotes the parameter vector

An example of feature template

- Input: The man went to the park .
- Feature vector for labelling the word "park“ with "NN":

$\vec{\phi}(t_6 = NN, t_5, W_{1:n})$=<0,0,…,0,1,,0,…,0,1(park | NN),0(park | VV),…,0,1,0,…,0,1,0>

ID	Feature Template	Feature
1	$t_{i-1}t_i$	DT|NN
2	t_i	NN
3	$w_i t_i$	park|NN
4	$w_{i-1}t_i$, $w_{i+1}t_i$, $w_{i-2}t_i$, $w_{i+2}t_i$	the|NN, .|NN, to|NN, </S>|NN
5	PREFIX(w_i) t_i, SUFFIX(w_i) t_i	"p"|NN, "k"|NN
6	HYPHEN(w_i) t_i, CASE(w_i) t_i	0|NN, 0|NN

- Note overlapping feature $w_i, f(w_i)$.

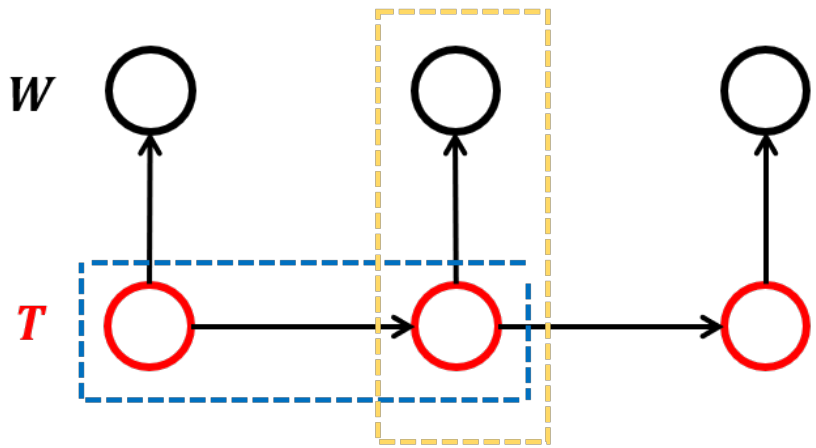
Training

- Parameterization

$$P(t_i = t | T_{i-k:i-1}, W_{1:n}) = \frac{\exp(\vec{\theta} \cdot \vec{\phi}(t_i = t, T_{i-k:i-1}, W_{1:n}))}{\sum_{t' \in L} \exp\left(\vec{\theta} \cdot \vec{\phi}(t_i = t', T_{i-k:i-1}, W_{1:n})\right)}$$

- Objective
  - maximum log-likelihood of individual $(t_i, T_{i-k:i-1}, W_{1:n})$ pairs
- Optimization method
  - Stochastic gradient descent (SGD).

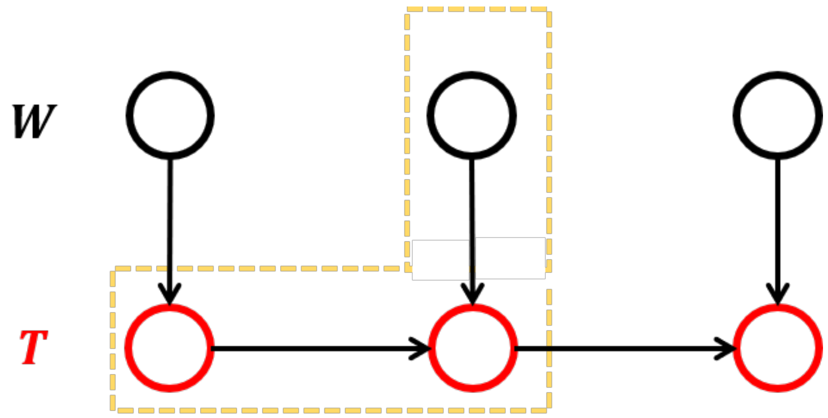
Contrast with HMM



$P(t_i|t_{i-1})P(w_i|t_i)$

HMM

Direct parameterization



$P(t_i|w_i, t_{i-1})$

MEMM

Feature parameterization

Contents

Decoding objective

WestlakeNLP

- Find the highest score:

$$\hat{T}_{1:n} = argmax_{T_{1:n}} P(T_{1:n}|W_{1:n})$$

$$= argmax_{T_{1:n}} \prod_{i=1}^{n} P(t_i|T_{i-k:i-1}, W_{1:n})$$

- find $\hat{T}_{1:n}$ from L^n candidate sequences
- With same Markov assumption, one can leverage the same **dynamic programming** principle as HMM decoding

Decoding algorithm

- Viterbi Algorithm
  - Markov assumption (e.g., k=1)

    $$P(T_{1:i}|W_{1:n}) = P(T_{1:i-1}|W_{1:n}) \cdot P(t_i|t_{i-1}, W_{1:n})$$

  - Dynamic programming
    - Denote $T_{1:i}(t_i = t)$ as a tag sequence with the last tag being t .
      $\hat{T}_{1:i}(t_i = t)$ as the highest scored sequence among $T_{1:i}(t_i = t)$.
    - Then $\hat{T}_{1:i}(t_i = t, t_{i-1} = t')$ must contain $\hat{T}_{1:i}(t_{i-1} = t')$
    - Thus $P\left(\hat{T}_{1:i}(t_i = t)|W_{1:n}\right)$

      $$= argmax_{t' \in L} P\left(\hat{T}_{1:i}(t_i = t, t_{i-1} = t')|W_{1:n}\right)$$

      $$= argmax_{t' \in L} \mathrm{P}\left(\hat{T}_{1:\mathrm{i}-1}(\mathrm{t}_{\mathrm{i}-1} = \mathrm{t}')\middle|W_{1:n}\right) \cdot P(t_i|t_{i-1}, W_{1:n})$$

- Time complexity is $O(nL^2)$

Decoding algorithm

WestlakeNLP

$tb \rightarrow P(\hat{T}_{1:i}(t_i = t)|W_{1:n})$

$t\backslash i$	1	2	3	4	5	6	7
NN	$\hat{T}_{1:1}(t_1 = NN)$	$\hat{T}_{1:2}(t_2 = NN)$	…	$\hat{T}_{1:i-1}(t_{i-1} = NN)$ ①	$\hat{T}_{1:i}(t_i = NN)$	…	$\hat{T}_{1:n}(t_n = NN)$
VV	$\hat{T}_{1:1}(t_1 = VV)$	$\hat{T}_{1:2}(t_2 = VV)$	…	$\hat{T}_{1:i-1}(t_{i-1} = \text{VV})$ ②	$\hat{T}_{1:i}(t_i = \text{VV})$	…	$\hat{T}_{1:n}(t_n = \text{VV})$
⋮	⋮	⋮	⋮	⋮ Ⓛ	⋮	⋮	⋮
AD	$\hat{T}_{1:1}(t_1 = AD)$	$\hat{T}_{1:2}(t_2 = \text{AD})$	…	$\hat{T}_{1:i-1}(t_{i-1} = AD)$	$\hat{T}_{1:i}(t_i = AD)$	…	$\hat{T}_{1:n}(t_n = \text{AD})$

$$P\left(\hat{T}_{1:i}(t_i = t)\middle|W_{1:n}\right) = argmax_{t' \in L} P\left(\hat{T}_{1:i-1}(t_{i-1} = t')\middle|W_{1:n}\right) \cdot P(t_i|t_{i-1}, W_{1:n})$$

- *bp* stores the argmax, $|L| \times n$

Viterbi algorithm for first-order MEMM

WestlakeNLP

Input: $s = W_{1:n}$, first-order POS tagging model with feature vector $\vec{\phi}(t_i, t_{i-1}, W_{1:n})$ and feature weight vector $\vec{\theta}$;

Variables: tb, bp;

Initialization: $tb\,[t][i] \leftarrow -\infty$; $bp[t][i] \leftarrow$ NULL for $t \in L \cup \{\langle \mathrm{B} \rangle\}, i \in [0, \ldots, n]$,

$tb[\langle B \rangle][0] \leftarrow 0$;

for $i \in [1, \ldots, n]$ **do**

  for $t \in L$ **do**

    for $t' \in L \cup \{\langle \mathrm{B} \rangle\}$ **do**

      $score \leftarrow \log P(t_i | t_{i-1} = t', W_{1:n})$

      if $tb[t'][i-1] + score > tb[t][i]$ **then**

        $tb[t][i] \leftarrow tb[t'][i-1] + score$;

        $bp[t][i] \leftarrow t'$;

$y_n \leftarrow \arg\max_t tb[t][n]$;

for $i \in [n, \ldots, 2]$ **do**

  $y_{i-1} \leftarrow bp[y_i][i]$;

Output: $y_1 \ldots y_n$;

Contents

The label bias problem

- MEMM is locally trained (or normalized)
  - factorized by

$$P(t_i|T_{i-k:i-1}, W_{1:n}) = \frac{\exp(\vec{\theta}\cdot\vec{\phi}(t_i{=}t, T_{i-k:i-1}, W_{1:n}))}{\sum_{t'\in L}\exp\left(\vec{\theta}\cdot\vec{\phi}(t_i{=}t', T_{i-k:i-1}, W_{1:n})\right)}$$

  - only consider individual label contexts
- However, the global probability $P(T_{1:n}|W_{1:n})$ is calculated during testing
  - lead to incorrect estimations of label sequence probabilities
- **Label Bias** –when one specific label prefers a certain label as its successor, then the output sequence tends to have the label pair.

Example of label bias

WestlakeNLP

- Suppose that our label set contains only four labels:

  L = {<B>, l_1, l_2, l_3},

ID	Tag Sequence	ID	Tag Sequence
d_1	$l_3 l_3 l_3$	d_4	$l_1 l_3 l_1$
d_2	$l_1 l_1 l_2$	d_5	$l_1 l_3 l_1$
d_3	$l_1 l_1$	d_6	$l_1 l_2 l_2$

- Calculate the probabilities of d_4 and d_6.

Example of label bias

WestlakeNLP

ID	Tag Sequence	ID	Tag Sequence
d_1	$l_3 l_3 l_3$	d_4	$l_1 l_3 l_1$
d_2	$l_1 l_1 l_2$	d_5	$l_1 l_3 l_1$
d_3	$l_1 l_1$	d_6	$l_1 l_2 l_2$

Item	Probability	Item	Probability
$P(l_1\|\langle B\rangle)$	$\frac{5}{6}$	$P(l_1\|l_2)$	0
$P(l_2\|\langle B\rangle)$	0	$P(l_2\|l_2)$	1
$P(l_3\|\langle B\rangle)$	$\frac{1}{6}$	$P(l_3\|l_2)$	0
$P(l_1\|l_1)$	$\frac{1}{3}$	$P(l_1\|l_3)$	$\frac{1}{2}$
$P(l_2\|l_1)$	$\frac{1}{3}$	$P(l_2\|l_3)$	0
$P(l_3\|l_1)$	$\frac{1}{3}$	$P(l_3\|l_3)$	$\frac{1}{2}$

- Direct MLE

$$P(d_4) = P(l_1 l_3 l_1) = \frac{2}{6} \quad P(d_6) = P(l_1 l_2 l_2) = \frac{1}{6}$$

- Estimation by local model

$$P(d_4) = P(l_1 l_3 l_1) = P(l_1|\langle B\rangle)P(l_3|l_1)P(l_1|l_3) = \frac{5}{6} \times \frac{1}{3} \times \frac{1}{2} = \frac{5}{36}$$

$$P(d_6) = P(l_1 l_2 l_2) = P(l_1|\langle B\rangle)P(l_2|l_1)P(l_2|l_2) = \frac{5}{6} \times \frac{1}{3} \times 1 = \frac{5}{18}$$

Example of label bias

ID	Tag Sequence	ID	Tag Sequence
d_1	$l_3 l_3 l_3$	d_4	$l_1 l_3 l_1$
d_2	$l_1 l_1 l_2$	d_5	$l_1 l_3 l_1$
d_3	$l_1 l_1$	d_6	$l_1 l_2 l_2$

Item	Probability	Item	Probability
$P(l_1 \| \langle B \rangle)$	$\frac{5}{6}$	$P(l_1 \| l_2)$	0
$P(l_2 \| \langle B \rangle)$	0	$P(l_2 \| l_2)$	1
$P(l_3 \| \langle B \rangle)$	$\frac{1}{6}$	$P(l_3 \| l_2)$	0
$P(l_1 \| l_1)$	$\frac{1}{3}$	$P(l_1 \| l_3)$	$\frac{1}{2}$
$P(l_2 \| l_1)$	$\frac{1}{3}$	$P(l_2 \| l_3)$	0
$P(l_3 \| l_1)$	$\frac{1}{3}$	$P(l_3 \| l_3)$	$\frac{1}{2}$

- Where is the problem?
  - l_2 is only followed by l_2. But l_3 is also followed by l_3.

    Although $count(l_2 \rightarrow l_2) = 1$, $count(l_3 \rightarrow l_1) = 2$,

    $P(l_2 \rightarrow l_2) = 1$ is overestimated due to local normalization

Example of label bias

ID	Tag Sequence	ID	Tag Sequence
d_1	$l_3 l_3 l_3$	d_4	$l_1 l_3 l_1$
d_2	$l_1 l_1 l_2$	d_5	$l_1 l_3 l_1$
d_3	$l_1 l_1$	d_6	$l_1 l_2 l_2$

Item	Probability	Item	Probability
$P(l_1 \mid \langle B \rangle)$	$\frac{5}{6}$	$P(l_1 \mid l_2)$	0
$P(l_2 \mid \langle B \rangle)$	0	$P(l_2 \mid l_2)$	1
$P(l_3 \mid \langle B \rangle)$	$\frac{1}{6}$	$P(l_3 \mid l_2)$	0
$P(l_1 \mid l_1)$	$\frac{1}{3}$	$P(l_1 \mid l_3)$	$\frac{1}{2}$
$P(l_2 \mid l_1)$	$\frac{1}{3}$	$P(l_2 \mid l_3)$	0
$P(l_3 \mid l_1)$	$\frac{1}{3}$	$P(l_3 \mid l_3)$	$\frac{1}{2}$

- Solution
  - take full sequences of labels as single units
  - calculating statistics (e.g., counting features) over full sequences of inputs and outputs before doing model normalization

Contents

Conditional random fields

- A form of log-linear models for sequence labelling
- The same features as MEMM
- MEMM is locally normalized
  - $P(t_i|T_{i-k:i-1}, W_{1:n}) = softmax\ (label\ score)$
- Globally normalized into a sequence distribution
  - $P(T_{1:n}|W_{1:n}) = softmax\ (label\ sequence\ score)$
  - Global score $\vec{\theta} \cdot \vec{\phi}(T_{1:n}, W_{1:n})$
  - Global feature vector

Global feature vectors

- Define $\vec{\phi}(T_{1:n}, W_{1:n})$ by aggregating $\vec{\phi}(t_i, T_{i-k:i-1}, W_{1:n})$ over the input sequence $1 \leq i \leq n$:

$$\vec{\phi}(T_{1:n}, W_{1:n}) = \sum_{i=1}^{n} \vec{\phi}(t_i, T_{i-k:i-1}, W_{1:n})$$

- Taking the first-order Markov chain (i.e. k=1):

$$\vec{\phi}(T_{1:n}, W_{1:n}) = \sum_{i=1}^{n} \vec{\phi}(t_i, T_{i-k:i-1}, W_{1:n})$$

Example of global feature vectors

WestlakeNLP

- Input: The | DT man | NN went | VBD to | TO the | DT park | NN . | .

Feature Entry	Feature Vector
$\vec{\phi}(t_1, t_0, W_1^7)$	0, 0, ⋯ $f_{47}(t_i = DT) = 1$, 0, ⋯ $f_{201}(t_{i-1}t_i =< B > DT) = 1$, 0, ⋯ $f_{501}(w_i = the, t_i = DT) = 1$, 0, ⋯
$\vec{\phi}(t_2, t_1, W_1^7)$	0, 0, ⋯ $f_{59}(t_i = NN) = 1$, 0, ⋯ $f_{472}(t_{i-1}t_i = DTNN) = 1$, 0, ⋯ $f_{478}(w_i = man, t_i = NN) = 1$, 0, ⋯
⋯	⋯
$\vec{\phi}(t_6, t_5, W_1^7)$	0, 0, ⋯ $f_{59}(t_i = NN) = 1$, 0, ⋯ $f_{472}(t_{i-1}t_i = DTNN) = 1$, 0, ⋯ $f_{932}(w_i = park, t_i = NN) = 1$, 0, ⋯
$\vec{\phi}(t_7, t_6, W_1^7)$	0, 0, ⋯ $f_{80}(t_i =.) = 1$, 0, ⋯ $f_{516}(t_{i-1}t_i = NN.) = 1$, 0, ⋯ $f_{1063}(w_i =., t_i =.) = 1$, 0, ⋯
$\vec{\phi}(T_1^7, W_1^7)$	0, 0, ⋯$f_{47} = 1$, 0, ⋯ $f_{59} = 2$, ⋯ $f_{201} = 1$, ⋯ $f_{472} = 2$, ⋯ $f_{501} = 1$, ⋯, $f_{748} = 1$, ⋯, $f_{932} = 1$, ⋯, $f_{1063} = 1$

Conditional random fields

- log-linear models for sequence labelling
- take $P(T_{1:n}|W_{1:n})$ as a single unit

$$P(T_{1:n}|W_{1:n}) = \frac{\exp(\sum_{i=1}^{n}\vec{\phi}(t_i,T_{i-k:i-1},W_{1:n}))}{\sum_{\overline{T}_{1:n}}\exp(\sum_{i=1}^{n}\vec{\phi}(\overline{t}_i,\overline{T}_{i-k:i-1},W_{1:n}))} = \frac{\exp\left(\vec{\theta}\cdot\vec{\phi}(T_{1:n},W_{1:n})\right)}{\sum_{\overline{T}_{1:n}}\exp\left(\vec{\theta}\cdot\vec{\phi}(\overline{T}_{1:n},W_{1:n})\right)}$$

- Compared to MEMM

$$P(t_i = t|T_{i-k:i-1}, W_{1:n}) = \frac{exp\left(\vec{\theta}\cdot\vec{\phi}(t_i=t,T_{i-k:i-1},W_{1:n})\right)}{\sum_{t' \in L} exp\left(\vec{\theta}\cdot\vec{\phi}(t_i=t',T_{i-k:i-1},W_{1:n})\right)}$$

  - directly model the probability of candidate sequence $P(T_{1:n}|W_{1:n})$
  - use **global feature vector** $\vec{\phi}(T_{1:n}, W_{1:n})$

Contents

CRF decoding

- Objective
  - $argmax_{T_{1:n}} P(T_{1:n}|W_{1:n}) = argmax_{T_{1:n}} \frac{\exp\left(\vec{\theta}\cdot\vec{\phi}(T_{1:n},W_{1:n})\right)}{Z}$
    $= argmax_{T_{1:n}} \vec{\theta} \cdot \vec{\phi}(T_{1:n}, W_{1:n})$
  - Take first-order CRF for example.
  - The feature vector locality allows the score to be decomposed:
    $\vec{\theta} \cdot \vec{\phi}(T_{1:n}, W_{1:n}) = \vec{\theta} \cdot \left(\sum_i \vec{\phi}(t_i, t_{i-1}, W_{1:n})\right)$
    $= \sum_{i=1}^{n} \vec{\theta} \cdot \vec{\phi}(t_i, t_{i-1}, W_{1:n})$
  - Thus the score can be computed incrementally from left to right

CRF decoding

WestlakeNLP

- Denote $T_{1:i}(t_i = t)$ as a tag sequence with the last tag being t .

  $\hat{T}_{1:i}(t_i = t)$ as the highest scored sequence among $T_{1:i}(t_i = t)$.

  $\hat{T}_{1:i}(t_i = t, t_{i-1} = t')$ must contain $\hat{T}_{1:i-1}(t_{i-1} = t')$
- Therefore $score\left(\hat{T}_{1:i}(t_i = t)\right) = argmax_{t' \in L} score\left(\hat{T}_{1:i}(t_i = t, t_{i-1} = t')\right)$

$$= argmax_{t' \in L} score(\hat{\mathrm{T}}_{1:\mathrm{i}-1}(\mathrm{t}_{\mathrm{i}-1} = \mathrm{t}')) + \vec{\theta} \cdot \vec{\phi}(t_i, t_{i-1}, W_{1:n})$$

- Therefore we can build score$(\hat{T}_{1:i}(t_i = t))$ incrementally from left to right.
- Also a back-pointer can be added.

Decoding algorithm

WestlakeNLP

$$tb \rightarrow score(\hat{T}_{1:i}(t_i = t))$$

$t\backslash i$	1	2	3	4	5	6	7
NN	$\hat{T}_{1:1}(t_1 = NN)$	$\hat{T}_{1:2}(t_2 = NN)$	…	$\hat{T}_{1:i-1}(t_{i-1} = NN)$ ①	$\hat{T}_{1:i}(t_i = NN)$	…	$\hat{T}_{1:n}(t_n = NN)$
VV	$\hat{T}_{1:1}(t_1 = VV)$	$\hat{T}_{1:2}(t_2 = VV)$	…	$\hat{T}_{1:i-1}(t_{i-1} = \mathrm{VV})$ ②	$\hat{T}_{1:i}(t_i = \mathrm{VV})$	…	$\hat{T}_{1:n}(t_n = \mathrm{VV})$
⋮	⋮	⋮	⋮	⋮ Ⓛ	⋮	⋮	⋮
AD	$\hat{T}_{1:1}(t_1 = AD)$	$\hat{T}_{1:2}(t_2 = \mathrm{AD})$	…	$\hat{T}_{1:i-1}(t_{i-1} = AD)$	$\hat{T}_{1:i}(t_i = AD)$	…	$\hat{T}_{1:n}(t_n = \mathrm{AD})$

$$score\left(\hat{T}_{1:i}(t_i = t)\right) = argmax_{t' \in L} score(\hat{\mathrm{T}}_{1:i-1}(\mathrm{t}_{i-1} = \mathrm{t}')) + \vec{\theta} \cdot \vec{\phi}(t_i, t_{i-1}, W_{1:n})$$

- bp stores the argmax, $|L| \times n$

Viterbi algorithm for CRF

Input: $s = W_{1:n}$, first-order POS tagging model with feature vector $\vec{\phi}(t_i, t_{i-1}, W_{1:n})$ and feature weight vector $\vec{\theta}$;
Variables: tb, bp;
Initialization: $tb\ [t][i] \leftarrow -\infty$; $bp[t][i] \leftarrow$ NULL for $t \in L \cup \{\langle \mathrm{B} \rangle\}, i \in [0, \ldots, n]$,
$tb[\langle B \rangle][0] \leftarrow 0$;
for $i \in [1, \ldots, n]$ **do**
  for $t \in L$ **do**
    for $t' \in L \cup \{\langle \mathrm{B} \rangle\}$ **do**
      $score \leftarrow \vec{\theta} \cdot \vec{\phi}(t_i = t, t_{i-1} = t', W_{1:n})$;
      if $tb[t'][i-1] + score > tb[t][i]$ **then**
        $tb[t][i] \leftarrow tb[t'][i-1] + score$;
        $bp[t][i] \leftarrow t'$;
$y_n \leftarrow \arg\max_t tb[t][n]$;
for $i \in [n, \ldots, 2]$ **do**
  $y_{i-1} \leftarrow bp[y_i][i]$;
Output: $y_1 \ldots y_n$;

Contents

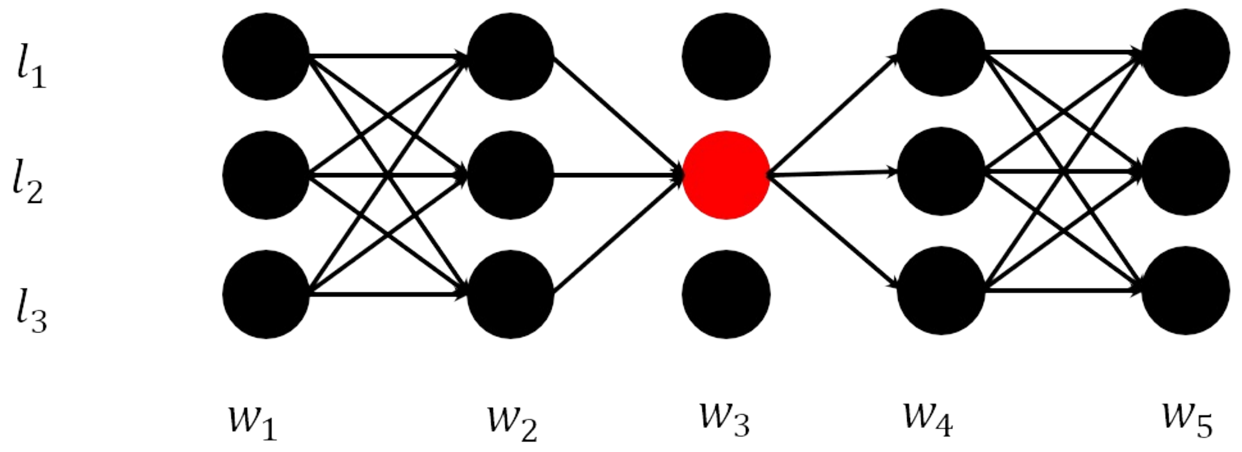
Calculating marginal probabilities

WestlakeNLP

- Given $\vec{\theta}$ and an input output pair $(W_{1:n}, T_{1:n})$, we want to calculate $P(t_i = t|W_{1:n})$ (during training)
- Definition

$$P(t_i = t|W_{1:n}) = \sum_{t_1 \in L} \sum_{t_2 \in L} \dots \sum_{t_{i-1} \in L} \sum_{t_{i+1} \in L} \dots \sum_{t_n \in L} P(T_{1:n}(t_i = t)|W_{1:n})$$

  - include $O(L^{n-1})$ summations



Solution

WestlakeNLP

- Again, leverage utilize the Markov properties in our features -- dynamic program

$$P(T_{1:n}|W_{1:n}) = \frac{exp\left(\vec{\theta}\cdot\vec{\phi}(T_{1:n},W_{1:n})\right)}{Z}$$

$$= \frac{exp\left(\vec{\theta}\cdot\left(\sum_i \vec{\phi}(t_i,t_{i-1},W_{1:n})\right)\right)}{Z}$$

$$= \frac{\left(exp\sum_i \vec{\theta}\cdot\vec{\phi}(t_i,t_{i-1},W_{1:n})\right)}{Z}$$

$$= \frac{\prod_i exp\left(\vec{\theta}\cdot\vec{\phi}(t_1,t_{i-1},W_{1:n})\right)}{Z}$$

- $Z = \sum_{T'_{1:n}} exp\left(\vec{\theta}\cdot\vec{\phi}(T'_{1:n},W_{1:n})\right)$

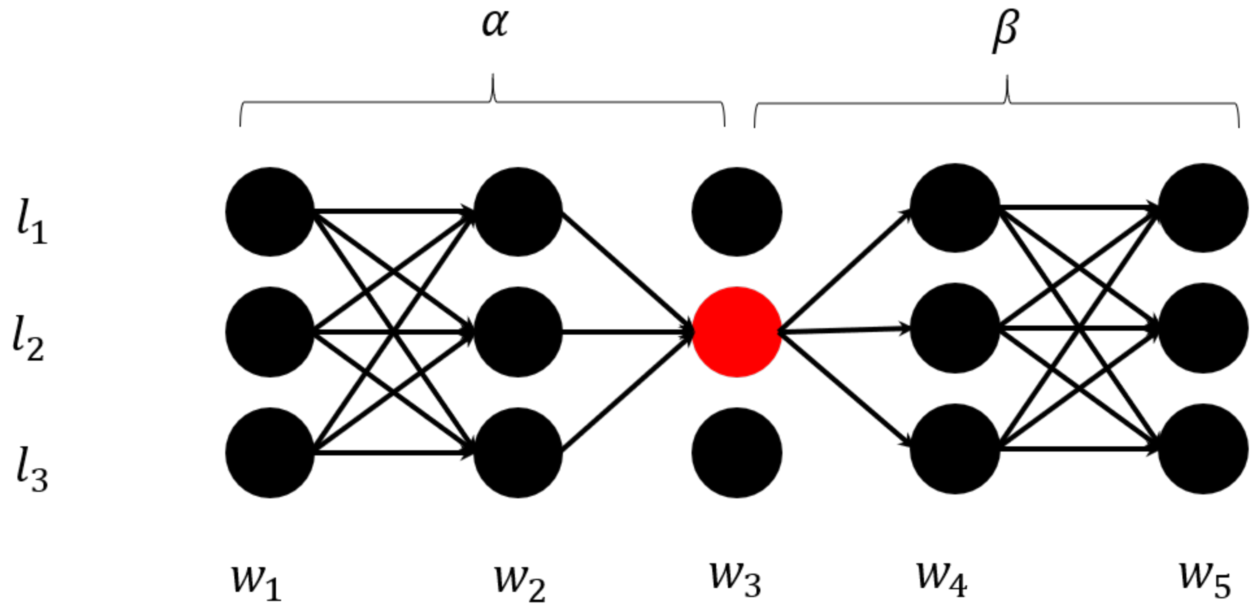
Solution

WestlakeNLP

- $$P(t_i = t|W_{1:n}) = \sum_{t_1 \in L} \cdots \sum_{t_{i-1} \in L} \sum_{t_{i+1} \in L} \cdots \sum_{t_n \in L} \frac{1}{Z} \prod_{j=1}^{n} \exp\left(\vec{\theta} \cdot \vec{\phi}(t_j, t_{j-1}, W_{1:n})\right), t_i = t$$

  cannot be calculated efficiently, due to exponential sum of product.

- Use dynamic programming to exploit feature locality. But there is a junction point in center.
- Cut the full summation equation into the product of two parts at i



Solution

- $$P(t_i = t|W_{1:n}) = \sum_{t_1 \in L} \cdots \sum_{t_{i-1} \in L} \sum_{t_{i+1} \in L} \cdots \sum_{t_n \in L} \frac{1}{Z} \prod_{j=1}^{n} \exp\left(\vec{\theta} \cdot \vec{\phi}(t_j, t_{j-1}, W_{1:n})\right), t_i = t$$

  cannot be calculated efficiently, due to exponential sum of product.
- Use dynamic programming to exploit feature locality. But there is a junction point in center.
- Cut the full summation equation into the product of two parts at i

$$P(t_i = t|W_{1:n}) = \sum_{t_1 \in L} \cdots \sum_{t_{i-1} \in L} \sum_{t_{i+1} \in L} \cdots \sum_{t_n \in L} \frac{1}{Z} \prod_{j=1}^{n} \exp\left(\vec{\theta} \cdot \vec{\phi}(t_j, t_{j-1}, W_{1:n})\right), t_i = t$$

$$= \frac{1}{Z}\left(\sum_{t_1 \in L} \cdots \sum_{t_{i-1} \in L} \prod_{j=1}^{i} \exp\left(\vec{\theta} \cdot \vec{\phi}(t_j, t_{j-1}, W_{1:n})\right)\right) \cdot$$

$$\left(\sum_{t_{i+1} \in L} \cdots \sum_{t_n \in L} \prod_{j=i+1}^{n} \exp\left(\vec{\theta} \cdot \vec{\phi}(t_j, t_{j-1}, W_{1:n})\right)\right),$$

where $t_i = t$ (distributivity).

Solution

WestlakeNLP

- Leverage the Markov properties in our features -- dynamic program
- the key is how to calculate each part efficiently
- similar computation method for forward part and backward part
- take forward part as example

Forward algorithm

- Our goal is to efficiently calculate:

$$\alpha(j,t) = \sum_{t_1 \in L} \dots \sum_{t_{j-1} \in L} \prod_{k=1}^{j} exp\left(\vec{\theta} \cdot \vec{\phi}(t_k, t_{k-1}, W_{1:n})\right), \text{where } t_j = t$$

- we can observe incrementally.

$$\prod_{k=1}^{j} \exp\left(\vec{\theta} \cdot \vec{\phi}(t_k, t_{k-1}, W_{1:n})\right) = \left(\prod_{k=1}^{j-1} \exp\left(\vec{\theta} \cdot \vec{\phi}(t_k, t_{k-1}, W_{1:n})\right)\right) \cdot \exp\left(\vec{\theta} \cdot \vec{\phi}(t_j, t_{j-1}, W_{1:n})\right)$$

- state transformation in dynamic programming

$$\alpha(j,t) = \Sigma_{t_{j-1} \in L}(\Sigma_{t_1 \in L} \dots \Sigma_{t_{j-2} \in L} \Pi_{k=1}^{j-1} \exp(\vec{\theta} \cdot \vec{\phi}(t_k, t_{k-1}, W_{1:n})))$$

$$\cdot \exp\left(\vec{\theta} \cdot \vec{\phi}(t_j = t_i, t_{j-1} = t', W_{1:n})\right) \text{ (distributivity)}$$

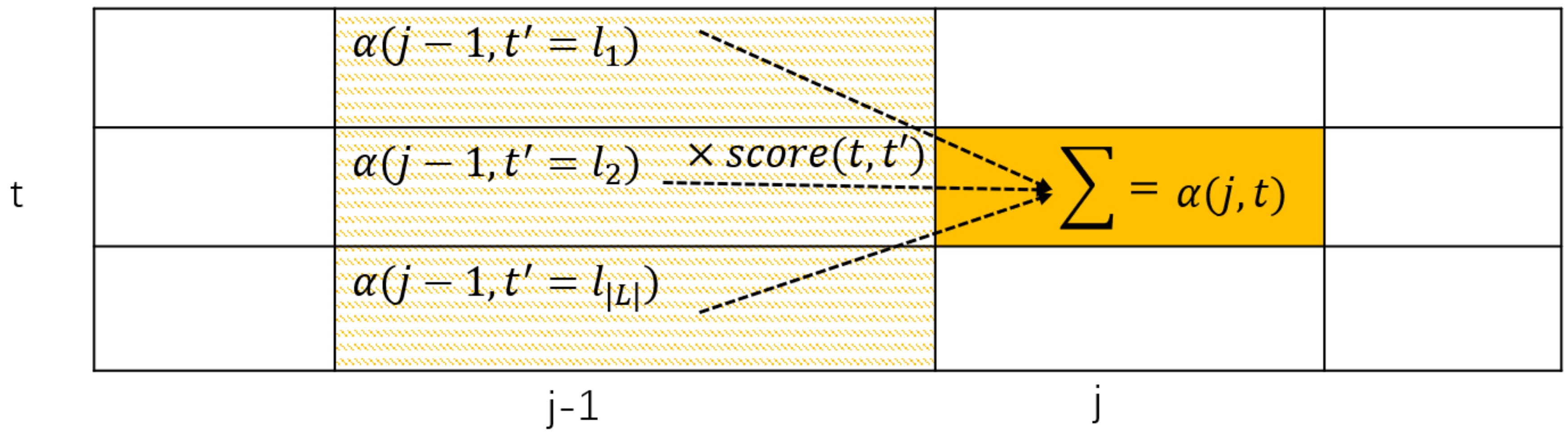
$$= \Sigma_{t' \in L}\left(\alpha(j-1, t') \cdot exp\left(\vec{\theta} \cdot \vec{\phi}(t_j = t, t_{j-1} = t', W_{1:n})\right)\right)$$

- Initialize $\alpha\ (0, \langle B \rangle) = 1$

Example

- Build a table of n columns and $|L|$ raws.

$$score(t, t') = \exp(\vec{\theta} \cdot \vec{\phi}(t, t', W_{1:n}))$$



$$\alpha(j, t) = \sum_{t' \in L} \left(\alpha(j-1, t') \cdot exp\left(\vec{\theta} \cdot \vec{\phi}(t_j = t, t_{j-1} = t', W_{1:n})\right)\right)$$

Forward algorithm

Forward algorithm for first-order CRF.

Inputs: $s = W_{1:n}$, first-order CRF model for POS tagging with with feature vector $\vec{\phi}(t_i, t_{i-1}, W_{1:n})$ and feature weight vector $\vec{\theta}$;

Variables: α;

Initialization: $\alpha[0][\langle \mathrm{B} \rangle] \leftarrow 1$, $\alpha[i][t] \leftarrow 0$ for $i \in [1 \dots n], t \in L$;

for $t \in L$ **do**

$\quad \alpha[1][t] \leftarrow \alpha[0][\langle \mathrm{B} \rangle] \cdot \exp\left(\vec{\theta} \cdot \vec{\phi}(t_1 = t, t_0 = \langle \mathrm{B} \rangle, W_{1:n})\right)$

for $i \in [2 \dots n]$ **do**

$\quad$ **for** $t \in L$ **do**

$\quad\quad$ **for** $t' \in L$ **do**

$\quad\quad\quad \alpha[i][t] \leftarrow \alpha[i][t] + \alpha[i-1][t'] \cdot \exp\left(\vec{\theta} \cdot \vec{\phi}(t_i = t, t_{i-1} = t', W_{1:n})\right)$;

Output: α;

Backward algorithm

WestlakeNLP

- Similar to Forward Algorithm, Backward Algorithm can be summarized as:

Inputs: $s = W_{1:n}$, first-order CRF model for POS tagging with with feature vector $\vec{\phi}(t_i, t_{i-1}, W_{1:n})$ and feature weight vector $\vec{\theta}$;
Variables: β;
Initialization: $\beta[i][t] \leftarrow 0$, $\beta[n+1][\langle \mathrm{B} \rangle] \leftarrow 1$ for $t \in L$, for $i \in [1 \ldots n], t \in L$;
for $t \in L$ **do**
  $\beta[n][t] \leftarrow \beta[n+1][\langle \mathrm{B} \rangle] \cdot \exp\left(\vec{\theta} \cdot \vec{\phi}(t_{n+1} = \langle \mathrm{B} \rangle, t_n = t, W_{1:n})\right)$;
for $i \in [n-1 \ldots 1]$ **do**
  for $t' \in L$ **do**
    for $t \in L$ **do**
      $\beta[i][t'] \leftarrow \beta[i][t'] + \beta[i+1][t] \cdot \exp\left(\vec{\theta} \cdot \vec{\phi}(t_{i+1} = t, t_i = t', W_{1:n})\right)$;
Output: β;

- $\beta(j, t) = \sum_{t' \in L}\left(\beta(j+1, t') \cdot exp\left(\vec{\theta} \cdot \vec{\phi}(t_{j+1} = t', t_j = t, W_{1:n})\right)\right)$

Forward-Backward

- $P(t_i = t|W_{1:n}) = \frac{1}{Z}\alpha(j,t)\beta(j,t)$

- The normalization constant can be computed efficiently

Contents

Training

WestlakeNLP

- Given a set of training data $D = \{(W_i, T_i)\}|_{i=1}^{n}$, the CRF training objective is to maximize the log-likelihood of D:

$$\hat{\vec{\theta}} = argmax_{\vec{\theta}} \log P\,(D)$$

$$= argmax_{\vec{\theta}} \log \prod_i P(T_i|W_i) \;\; (i.i.d.)$$

$$= argmax_{\vec{\theta}} \sum_i \log P\,(T_i|W_i)$$

$$= argmax_{\vec{\theta}} \; \sum_i \log \frac{exp\left(\vec{\theta}\cdot\vec{\phi}(T_i,W_i)\right)}{\sum_{T'} exp\left(\vec{\theta}\cdot\vec{\phi}(T',W_i)\right)}$$

$$= argmax_{\vec{\theta}} \sum_i \left(\vec{\theta}\cdot\vec{\phi}(T_i,W_i) - \log \sum_{T'} exp\left(\vec{\theta}\cdot\vec{\phi}(T',W_i)\right)\right)$$

Local gradient

- For each training example, the local gradient is:

$$\frac{\partial \log P(T_i|W_i)}{\partial \vec{\theta}} = \vec{\phi}(T_i, W_i) - \frac{\sum_{T'} \exp\left(\vec{\theta} \cdot \vec{\phi}(T', W_i)\right) \cdot \vec{\phi}(T', W_i)}{\sum_{T''} exp\left(\vec{\theta} \cdot \vec{\phi}(T'', W_i)\right)}$$

$$= \vec{\phi}(T_i, W_i) - \sum_{T'} \frac{exp\left(\vec{\theta} \cdot \vec{\phi}(T', W_i)\right)}{\sum_{T''}\left(\vec{\theta} \cdot \vec{\phi}(T'', W_i)\right)} \cdot \vec{\phi}(T', W_i)$$

$$= \vec{\phi}(T_i, W_i) - \sum_{T'} P(T'|W_i) \vec{\phi}(T', W_i) \quad \left(\text{definition of } P(T'|W_i)\right)$$

- An exponential number of candidate outputs T', which makes the calculation of $\sum_{T'} P(T'|W_i) \vec{\phi}(T', W_i)$ intractable.

Efficiently calculating the expected global feature vector

- $\sum_{T'} P(T'|W_i)\vec{\phi}(T', W_i)$ is the expected global feature vector over all possible output label sequences
- resort to feature locality

$$\vec{\phi}(T', W_i) = \sum_{j=1}^{n_i} \vec{\phi}\left(t'_j, t'_{j-1}, W_i\right)$$

- rewrite $\sum_{T'} P(T'|W_i)\vec{\phi}(T', W_i)$ from a feature − centric perspective

$$\begin{aligned}
\sum_{T'} P(T'|W_i)\vec{\phi}(T', W_i) &= \sum_{T'} P(T'|W_i)\left(\sum_j \vec{\phi}\left(t'_j, t'_{j-1}, W_i\right)\right) \\
&= \sum_{T'} \sum_j P(T'|W_i)\vec{\phi}\left(t'_j, t'_{j-1}, W_i\right) \\
&= \sum_j \left(\sum_{T'} P(T'|W_i)\vec{\phi}\left(t'_j, t'_{j-1}, W_i\right)\right) \\
&= \sum_j E_{T' \sim P(T'|W_i)} \vec{\phi}\left(t'_j, t'_{j-1}, W_i\right)
\end{aligned}$$

Efficiently calculating the expected global feature vector

- Since a local feature $\vec{\phi}(t'_j, t'_{j-1}, W_i)$ is constrained to a context that consists of only t'_j and t'_{j-1}

$$\sum_j E_{T' \sim P(T'|W_i)} \vec{\phi}(t'_j, t'_{j-1}, W_i)$$

$$= \sum_j E_{t'_{j-1} t'_j \sim P(t'_{j-1} t'_j | W_i)} \vec{\phi}(t'_j, t'_{j-1}, W_i)$$

$$= \sum_j \left(\sum_{t'_{j \in L}, t'_{j-1 \in L}} P(t'_{j-1} t'_j | W_i) \vec{\phi}(t'_j, t'_{j-1}, W_i) \right)$$

- Thus, the key is to calculate the marginal probabilities $P(t'_{j-1} t'_j | W_i)$
  - Forward/Backward Algorithm

Efficiently calculating the expected global feature vector

- Similar to $P\left(t_j \middle| W_{1:n}\right)$:
  - define

$$\alpha(k,t) = \sum_{t'_1 \in L} \cdots \sum_{t'_{k-1} \in L} \prod_{m=1}^{k} exp\left(\vec{\theta} \cdot \vec{\phi}(t'_m, t'_{m-1}, W_i)\right)$$

  - define

$$\beta(k,t) = \sum_{t'_{k+1} \in L} \cdots \sum_{t'_{n_i} \in L} \prod_{m=k}^{n_i} exp\left(\vec{\theta} \cdot \vec{\phi}(t'_{m+1}, t'_m, W_i)\right)$$

$$P\left(t'_{j-1} t'_j \middle| W_i\right) = \left(\frac{1}{Z}\right) \alpha\left(j-1, t'_{j-1}\right) \beta\left(j, t'_j\right) exp\left(\vec{\theta} \cdot \vec{\phi}\left(t'_j, t'_{j-1}, W_i\right)\right)$$

Algorithm for training first-order CRF

WestlakeNLP

Inputs: $s = W_{1:n}$, first-order CRF model for POS tagging with with feature vector $\vec{\phi}(t_i, t_{i-1}, W_{1:n})$ and feature weight vector $\vec{\theta}$;

Variables: *table*, α, β;

$\alpha \leftarrow \text{FORWARD}(W_{1:n}, \vec{\phi}, \vec{\theta})$

$\beta \leftarrow \text{BACKWARD}(W_{1:n}, \vec{\phi}, \vec{\theta})$

for $j \in [1 \ldots n]$ **do**

    $total \leftarrow 0$;

    for $t \in L$ **do**

        for $t' \in L$ **do**

            $table[t'][t][j] \leftarrow \alpha[t'][j-1] \cdot \beta[t][j] \cdot \exp\left(\vec{\theta} \cdot \vec{\phi}(t, t', W_i)\right)$;

            $total \leftarrow total + table[t'][t][j]$;

    for $t \in L$ **do**

        for $t' \in L$ **do**

            $table[t'][t][j] \leftarrow \frac{table[t'][t][j]}{total}$;

Output: *table*;

Local gradient

- For each training example, the local gradient is:

$$\frac{\partial \log P(T_i|W_i)}{\partial \vec{\theta}} = \vec{\phi}(T_i, W_i) - \frac{\sum_{T'} \exp\left(\vec{\theta}\cdot\vec{\phi}(T',W_i)\right)\cdot\vec{\phi}(T',W_i)}{\sum_{T''} exp\left(\vec{\theta}\cdot\vec{\phi}(T'',W_i)\right)}$$

$$= \vec{\phi}(T_i, W_i) - \sum_{T'} \frac{exp\left(\vec{\theta}\cdot\vec{\phi}(T',W_i)\right)}{\sum_{T''}\left(\vec{\theta}\cdot\vec{\phi}(T'',W_i)\right)} \cdot \vec{\phi}(T', W_i)$$

$$= \vec{\phi}(T_i, W_i) - \sum_{T'} P(T'|W_i)\vec{\phi}(T', W_i) \quad \left(\text{definition of } P(T'|W_i)\right)$$

$$= \vec{\phi}(T_i, W_i) - \sum_j E_{t'_{j-1}t'_j \sim P\left(t'_{j-1}t'_j|W_i\right)} \vec{\phi}\left(t'_j, t'_{j-1}, W_i\right)$$

Contents

The perceptron algorithm

- Also a linear max-margin model to find a value for $(\vec{w}\,, b)$ such that y = SIGN $(\vec{w}^T\vec{v}(x_i) + b)$ for all training examples $(x_i\,, y_i\,)$
- Algorithm

Input: $D = \{(x_i, y_i)\}|_{i=1}^{N}, y_i \in \{-1, +1\}$
Initialization: $\vec{\omega} \leftarrow \vec{0}; b \leftarrow 0; t \leftarrow 0$
repeat
  for $i \in [1..N]$ **do**
    $z_i \leftarrow \mathrm{SIGN}(\vec{\omega}^T\vec{v}(x_i) + b)$;
    if $z_i \neq y_i$ **then**
      $\vec{\omega} \leftarrow \vec{\omega} + \vec{v}(x_i) \times y_i$;
      $b \leftarrow b + y_i$;
  $t \leftarrow t + 1$;
until $t = T$;

Structured perceptron

WestlakeNLP

- Perceptron is a discriminative linear model for classification, which can be adapted to structure prediction via
  - treating the whole label sequence structure as a single unit
  - global feature vector
  - $score(T_{1:n}, W_{1:n}) = \hat{\theta} \cdot \widehat{\Phi}(T_{1:n}, W_{1:n})$

Algorithm of structured perceptron

Inputs: $D = \{(W_i, T_i)\}|_{i=1:N}$
Initialization: $\vec{\theta} \leftarrow \vec{\mathbf{0}}$; $\vec{\sigma} \leftarrow \vec{\mathbf{0}}$; $t \leftarrow 0$;
repeat
  for $i \in [1 \ldots N]$ **do**
    $Z_i \leftarrow \arg\max_{\mathfrak{z}} \vec{\theta} \cdot \vec{\phi}(W_i, \mathfrak{z})$;
    if $Z_i \neq T_i$ **then**
      $\vec{\theta} \leftarrow \vec{\theta} + \vec{\phi}(W_i, T_i) - \vec{\phi}(W_i, Z_i)$;
  $t \leftarrow t + 1$;
until $t = T$;

Algorithm of structured perceptron

WestlakeNLP

Inputs: $D = \{(W_i, T_i)\}|_{i=1:N}$
Initialization: $\vec{\theta} \leftarrow \vec{\mathbf{0}}$; $\vec{\sigma} \leftarrow \vec{\mathbf{0}}$; $t \leftarrow 0$;
repeat
  for $i \in [1 \ldots N]$ **do**
    $Z_i \leftarrow \arg\max_{\mathfrak{z}} \vec{\theta} \cdot \vec{\phi}(W_i, \mathfrak{z})$; ← Viterbi
    if $Z_i \neq T_i$ **then**
      $\vec{\theta} \leftarrow \vec{\theta} + \vec{\phi}(W_i, T_i) - \vec{\phi}(W_i, Z_i)$;
  $t \leftarrow t + 1$;
until $t = T$;

Relationship with CRF

WestlakeNLP

- Common with CRF
  - Model: $score(T_{1:n}, W_{1:n}) = \vec{\theta} \cdot \vec{\phi}(T_{1:n}, W_{1:n})$
  - Decoding: Viterbi Algorithm
- Difference from CRF
  - Training objective, minimizing:

$$\sum_{i=1}^{N} (\max_{Z'_{1:n}} \vec{\theta} \cdot \vec{\phi}(W_{1:n}, Z'_{1:n}) - \vec{\theta} \cdot \vec{\phi}(W_{1:n}, T_{1:n}))$$

Contents

The average perceptron

WestlakeNLP

A variation of the standard perceptron algorithm

- record the values of $\vec{\theta}$ after each training example
- taking the average value as the final model, instead of the last updated value of $\vec{\theta}$

$$\vec{\gamma} = \frac{1}{NT} \sum_{i \in \{1 \ldots N\} t \in \{1 \ldots T\}} \overrightarrow{\theta^{i,t}}$$

- N is the number of training examples
- T is the number of training iterations

Nature of averaged perceptron

WestlakeNLP

- The score given by the averaged parameter vector is:

$$\overline{score}(x,y) = \left(\frac{1}{NT}\sum_{i,t}\overrightarrow{\theta^{i,t}}\right)\cdot\vec{\phi}(x,y)$$
$$= \frac{1}{NT}\sum_{i,t}\left(\overrightarrow{\theta^{i,t}}\cdot\vec{\phi}(x,y)\right)$$
$$= \frac{1}{NT}\sum_{i,t} score^{i,t}(x,y)\,,$$

- an effective voting strategy
- avoid overfitting

Algorithm of averaged perceptron

WestlakeNLP

Inputs: $D = \{(W_i, T_i)\}|_{i=1:N}$

Initialization: $\vec{\theta} \leftarrow \vec{\mathbf{0}}$; $\vec{\sigma} \leftarrow \vec{\mathbf{0}}$; $t \leftarrow 0$;

repeat

  for $i \in [1 \ldots N]$ **do**

    $Z_i \leftarrow \arg\max_{\mathfrak{z}} \vec{\theta} \cdot \vec{\phi}(W_i, \mathfrak{z})$;

    if $Z_i \neq T_i$ **then**

      $\vec{\theta} \leftarrow \vec{\theta} + \vec{\phi}(W_i, T_i) - \vec{\phi}(W_i, Z_i)$;

    $\vec{\sigma} \leftarrow \vec{\sigma} + \vec{\theta}$;

  $t \leftarrow t + 1$;

until $t = T$;

$\vec{\sigma} \leftarrow \frac{\vec{\sigma}}{NT}$;

Contents

Structured SVM

SVM is a **large-margin** discriminative linear model for classification, which can be adapted to structure prediction

- Common with CRF and structured perceptron
  - Model: $score(T_{1:n}, W_{1:n}) = \vec{\theta} \cdot \vec{\phi}(T_{1:n}, W_{1:n})$
  - Decoding: Viterbi Algorithm
- Difference from CRF or structured perceptron
  - Training objective

$$\min_{\vec{\theta}} \frac{1}{2} \left|\left|\vec{\theta}\right|\right|^2 + C \left(\sum_{i=1}^{N} \max\left(0, 1 - \vec{\theta} \cdot \vec{\phi}\left(W^{(i)}{}_{1:n}, T^{(i)}{}_{1:n}\right) + \max_{T'_{1:n} \neq T^{(i)}{}_{1:n}} \left(\vec{\theta} \cdot \vec{\phi}\left(W^{(i)}{}_{1:n}, T'{}_{1:n}\right)\right)\right)\right)$$

Contents

Cost-sensitive training

WestlakeNLP

Which candidate do you think is better?

Input	The	man	went	to	the	park	.
Gold	DT	NN	VBD	TO	DT	NN	.
Cand1	DT	NN	VBD	TO	DT	VV	.
Cand2	NN	NN	VBD	TO	NN	VV	.

$\vec{\theta} \cdot \vec{\varphi}(W, Cand1) = 1,$
$\Delta = 6 - 1 = 5$

$\vec{\theta} \cdot \vec{\varphi}(W, Cand2) = 2,$
$\Delta = 6 - 3 = 3$

- All incorrect structures are **NOT** equally incorrect
- If the model has to make a mistake, we would rather choose Cand1 than Cand2

Cost-sensitive training objective

use $\Delta(T'_{1:n}, T_{1:n})$ to denote the **cost** of mistakenly predicting $T'_{1:n}$ when the gold-standard output is $T_{1:n}$

- Δ can be any measure function, we use **Hamming distance** here
- Hamming distance refers to the number of different labels between a pair of label sequences
- Assign not only a high score for correct output, but also less costly incorrect output compared to a more costly one

Cost-sensitive training objective

Define the cost-sensitive structured SVM training objective:

$$\min_{\vec{\theta}} \frac{1}{2}\left|\left|\vec{\theta}\right|\right|^2 +$$

$$C\left(\sum_{i=1}^{N} \max\left(0, \Delta\left(\widehat{T'^{(i)}}, {T^{(i)}}_{1:n}\right) - \vec{\theta}\cdot\vec{\phi}\left({W^{(i)}}_{1:n}, {T^{(i)}}_{1:n}\right) + \vec{\theta}\cdot\vec{\phi}\left({W^{(i)}}_{1:n}, \widehat{T'^{(i)}}\right)\right)\right)$$

where

$$\widehat{T'^{(i)}} = \max_{T' \neq {T^{(i)}}_{1:n}} \left(\Delta\left(T:', {T^{(i)}}_{1:n}\right) + \vec{\theta}\cdot\vec{\phi}\left({W^{(i)}}_{1:n}, T'\right)\right)$$

The essential problem:

- If $\text{Score}(T_1) + \Delta(T_1, T) > \text{Score}(T_2) + \Delta(T_2, T), \text{Score}(T_1) > \text{Score}(T_2)$?

Cost augmented decoding - Challenge

The margin violation $\Delta(T', T_i) - \vec{\theta} \cdot \vec{\phi}(T_i, W_i) + \vec{\theta} \cdot \vec{\phi}(T', W_i)$ scales with $\Delta(T', T_i)$.

How to find the maximum?

- the highest-scored output may not coincide with the max violated margin
- Viterbi decoder (Algorithm in MEMM) cannot be directly used

Relationship with past objective

- Past: Decoder finds $argmax_{T'} Score(T')$
- Now: Decoder fins $argmax_{T'}(Score(T') + \Delta)$

Cost augmented decoding - Solution

Fortunately, Hamming distant cost $\Delta(T'_{1:n}, T_{1:n})$ can be decomposed into local components

- $\Delta(T'_{1:n}, T_{1:n}) = \sum_{i=1:n} \delta\left(t'_i, t_i\right)$, where $\delta(t'_i, t_i) = 1$ if and only if $t'_i = t_i$
- As a result, $\widehat{T'}_{1:i}$ must contain $\widehat{T'}_{1:i-1}$
- Thus $\widehat{T'}_{1:i}$ can be computed incrementally

$$\widehat{T'}_{1:i}(t'_i = t) = argmax_{T'_{1:i-1}(t'_{i-1}=t')} \left(\begin{array}{c} \left(\vec{\theta} \cdot \left(\sum_{j=1:i-1} \vec{\phi}\ (t'_j, t'_{j-1}, W_{1:n}\right) + \Delta T'_{1:j-1}, T_{1:j-1}\ )\right) \\ + \left(\vec{\theta} \cdot \vec{\phi}(t'_i = t, t'_{i-1} = t', W_{1:n}) + \delta(t, t')\right) \end{array} \right)$$

- We can use Viterbi algorithm by adding $\delta(t'_i, t_i)$

Contents

WestlakeNLP

Summary

- Maximum Entropy Markov Models
- the label bias problem
- Conditional random fields
- Structured perceptron, averaged perceptron
- Structured SVM, cost-sensitive training