# Sequence Labeling

He He

New York University

October 6, 2021

# Last week

Goal: probabilistic modeling of (text) sequences

N-gram models

- Markov assumption: the next word depends on limited prior context
- Tackling sparsity
  - Discounting: allocate some probability mass to unseen events
  - backoff/interpolation: use dynamic context

LM as sequence classification

- Log-linear LM: represent context by a (handcrafted) feature vector
- Feed-forward neural LM: represent context by concatenated word vectors
- Recurrent neural LM: represent context by a recurrently updated state

# Table of Contents


Introduction

Maximum-entropy Markov Models

Conditional Random Field

Neural Sequence Modeling

# Sequence labeling

Language modeling as sequence labeling:

| | | | | | | |
|---|---|---|---|---|---|---|
| the | fox | jumped | over | the | dog | `STOP` |
| ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ |
| * | the | fox | jumped | over | the | dog |

**Part-of-speech (POS) tagging**:

| | | | | | |
|---|---|---|---|---|---|
| DT | NN | VBD | IN | DT | NN |
| ↑ | ↑ | ↑ | ↑ | ↑ | ↑ |
| the | fox | jumped | over | the | dog |

# Span prediction

**Named-entity recognition (NER)**:

| | | | | | | |
|---|---|---|---|---|---|---|
| B-org | I-org | I-org | O | O | O | B-date |
| New | York | University | is | founded | in | 1831 |
| | Organization | | | | | Date |

**BIO notation**:

- Reduce span prediction to sequence labeling
- B-<tag>: the first word in span <tag>
- I-<tag>: other words in span <tag>
- O: words not in any span

# POS tagging

**Part-of-speech**: the *syntactic* role of each word in a sentence

POS tagset:

- Universal dependency tagset
  - **Open class tags**: content words such as nouns, verbs, adjectives, adverbs etc.
  - **Closed class tags**: function words such as pronouns, determiners, auxiliary verbs etc.
- Penn Treebank tagset (developed for English, 45 tags)

Application:

- Often the first step in the NLP pipeline.
- Used as features for other NLP tasks.
- Included in tools such as Stanford CoreNLP and spaCy.

# The majority baseline

A dumb approach: look up each word in the dictionary and return the most common POS tag.

# The majority baseline

A dumb approach: look up each word in the dictionary and return the most common POS tag.

Problem: ambiguity. Example?

| | | WSJ | | Brown | |
|---|---|---|---|---|---|
| **Types:** | | | | | |
| **Unambiguous** | (1 tag) | 44,432 | (**86%**) | 45,799 | (**85%**) |
| **Ambiguous** | (2+ tags) | 7,025 | (**14%**) | 8,050 | (**15%**) |
| **Tokens:** | | | | | |
| **Unambiguous** | (1 tag) | 577,421 | (**45%**) | 384,349 | (**33%**) |
| **Ambiguous** | (2+ tags) | 711,780 | (**55%**) | 786,646 | (**67%**) |

**Figure 8.2** Tag ambiguity for word types in Brown and WSJ, using Treebank-3 (45-tag) tagging. Punctuation were treated as words, and words were kept in their original case.

Most types are unambiguous, but ambiguous ones are common words!

Most common tag: 92% accuracy on WSJ (vs 97% SOTA)
Always compare to the majority class baseline.

# Table of Contents


Introduction

Maximum-entropy Markov Models

Conditional Random Field

Neural Sequence Modeling

# Multiclass classifcation

Task: given $x = (x_1, \dots, x_m) \in \mathcal{X}^m$, predict $y = (y_1, \dots, y_m) \in \mathcal{Y}^m$.

Predictor: *independent* classification problem at each position $y_i = h(x, i) \quad \forall i$

Multinomial logistic regression ($\theta \in \mathbb{R}^d$):

$$p(y_i \mid x) = \frac{\exp(\theta \cdot \phi(x, y_i, i))}{\sum_{y' \in \mathcal{Y}} \exp(\theta \cdot \phi(x, y', i))}$$

Feature templates:

$$T(x, i, y) = \begin{array}{l} x[i], y \\ \text{suff}(x[i]), y \\ x[i-1], y \end{array}$$

# Multiclass classifcation

Task: given $x = (x_1, \ldots, x_m) \in \mathcal{X}^m$, predict $y = (\overset{\in \mathcal{Y}}{y_1}, \ldots, \overset{\in \mathcal{Y}}{y_m}) \in \mathcal{Y}^m$. $\underbrace{\mathcal{Y} \times \cdots \times \mathcal{Y}}_{m}$

Predictor: *independent* classification problem at each position $y_i = h(x, i) \quad \forall i$

Multinomial logistic regression ($\theta \in \mathbb{R}^d$):

$$p(y_i \mid x) = \frac{\exp\left[\theta \cdot \phi(x, i, y_i)\right]}{\sum_{y' \in \mathcal{Y}} \exp\left[\theta \cdot \phi(x, i, y')\right]}$$

- Learning: MLE (is the objective ~~convex~~ concave?)
- Inference: trivial ($\arg\max_{y \in \mathcal{Y}} p(y \mid x)$)
- Does not consider dependency among $y_i$'s.

```
DT NN ?
B-<org> I-<org> ?
```

# Maximum-entropy markov model (MEMM)

Model the joint probability of $y_1, \ldots, y_m$:

$$p(y_1, \ldots, y_m \mid x) = \prod_{i=1}^{m} p(y_i \mid y_{i-1}, x) \; .$$

- Use the Markov assumption similar to n-gram LM.
- Insert start/end symbols: $y_0 = *$ and $y_m = \texttt{STOP}$.

Parametrization:

$$p(y_i \mid y_{i-1}, x) = \frac{\exp\left[\theta \cdot \phi(x, i, y_i, y_{i-1})\right]}{\sum_{y' \in \mathcal{Y}} \exp\left[\theta \cdot \phi(x, i, y', y_{i-1})\right]}$$

Learning: MLE (each sequence produces $m$ classification examples)

# Features for POS tagging

Interaction between word and tags:

$$\mathbb{1}\{x_i = \textit{the},\ y_i = \text{DET}\}$$
$$\mathbb{1}\{y_i = \text{PROPN}, x_{i+1} = \textit{Street}, y_{i-1} = \text{NUM}\}$$
$$\mathbb{1}\{y_i = \text{VERB}, y_{i-1} = \text{AUX}\}$$

Word shape feature that help with unknown words:

| |
|---|
| $x_i$ contains a particular prefix (perhaps from all prefixes of length $\leq 2$) |
| $x_i$ contains a particular suffix (perhaps from all suffixes of length $\leq 2$) |
| $x_i$'s word shape |
| $x_i$'s short word shape |

PM-abc-13 $\rightarrow$ X-x-d

# Inference

**Decoding / Inference**:

$$\begin{aligned}
&\arg\max_{y \in \mathcal{Y}^m} \prod_{i=1}^{m} p(y_i \mid y_{i-1}, x) \\
&= \arg\max_{y \in \mathcal{Y}^m} \sum_{i=1}^{m} \log p(y_i \mid y_{i-1}, x) \\
&= \arg\max_{y \in \mathcal{Y}^m} \sum_{i=1}^{m} \underbrace{s(y_i, y_{i-1})}_{\text{local score}},
\end{aligned}$$

where $s(y_i, y_{i-1}) = \theta \cdot \phi(x, i, y_i, y_{i-1})$.

- Bruteforce: exact, compute scores of all sequences, $O(|\mathcal{Y}|^m)$
- Greedy: inexact, predict $y_i$ sequentially, $O(m)$

# Viterbi decoding

$$\max_{y\in\mathcal{Y}^m}\sum_{i=1}^{m} s(y_i, y_{i-1})$$

$$= \max_{y\in\mathcal{Y}^m}\left(\sum_{i=1}^{m-1} s(y_i, y_{i-1}) + s(y_m, y_{m-1})\right)$$

$$= \max_{y_m\in\mathcal{Y}}\max_{y\in\mathcal{Y}^{m-1}}\left(\sum_{i=1}^{m-1} s(y_i, y_{i-1}) + s(y_m, y_{m-1})\right)$$

$$= \max_{y_m\in\mathcal{Y}}\max_{t\in\mathcal{Y}}\max_{y\in\mathcal{Y}^{m-1}, y_{m-1}=t}\left(\sum_{i=1}^{m-1} s(y_i, y_{i-1}) + s(y_m, y_{m-1}=t)\right)$$

$$= \max_{y_m\in\mathcal{Y}}\max_{t\in\mathcal{Y}}\left(s(y_m, t) + \boxed{\max_{y\in\mathcal{Y}^{m-1}, y_{m-1}=t}\sum_{i=1}^{m-1} s(y_i, y_{i-1})}\right)$$

$$= \max_{y_m\in\mathcal{Y}}\underbrace{\max_{t\in\mathcal{Y}}\left(s(y_m, t) + \boxed{\pi[m-1, t]}\right)}_{\pi[m, y_m]}$$

def

seq len

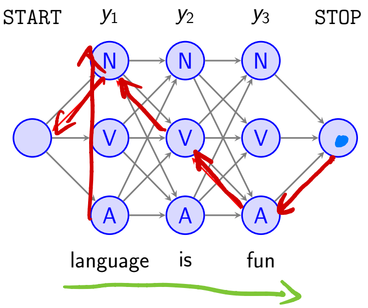
last tag

# Viterbi decoding

10 NN

DP: $\pi[j,t] = \max_{t' \in \mathcal{Y}} \pi[j-1, t'] + s(y_j = t, y_{j-1} = t')$

Backtracking: $p[j,t] = \arg\max_{t' \in \mathcal{Y}} \pi[j-1, t'] + s(y_j = t, y_{j-1} = t')$




Base: $\pi(0, t) = 0 \quad \forall\, t \in Y$

For $j = 1:m$ — length — $m$

For $t = 1:|Y|$ — tag — $|Y|$

$\pi[j,t] = \dots$ — $|Y|$

return $\pi[m, STOP]$

$O(m|Y|^2)$

Time complexity?

# Summary

Sequence labeling: $\mathcal{X}^m \to \mathcal{Y}^m$

- **Majority baseline**: $y_i = h(x_i)$ (no context)
- **Multiclass classification**: $y_i = h(x, i)$ (global input context)
- **MEMM**: $y_i = h(x, i, y_{i-1})$ (global input context, previous output context)

Problem: $y_t$ cannot be influenced by future evidence (more on this later)

Next: score $x$ and the output $y$ instead of local components $y_i$

# Table of Contents


Introduction

Maximum-entropy Markov Models

Conditional Random Field

Neural Sequence Modeling

# Structured prediction

$Y'$

Task: given $x = (x_1, \ldots, x_m) \in \mathcal{X}^m$, predict $y = (y_1, \ldots, y_m) \in \mathcal{Y}^m$.

$y \in \{pos, neg\}$

- Similar to multiclass classification except that $\mathcal{Y}$ is very large
- Compatibility score: $h \colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$
- Predictor: $\arg\max_{y \in \mathcal{Y}^m} h(x, y)$

General idea:

- $h(x, y) = f(\theta \cdot \Phi(x, y))$
- $\Phi$ should be decomposable so that inference is tractable
- Loss functions: structured hinge loss, negative log-likelihood etc.
- Inference: viterbi, interger linear programming (ILP)

# Graphical models

Graphical model:

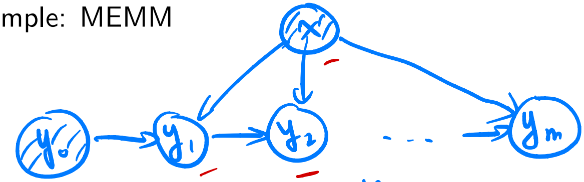
- A joint distribution of a set of random variables
- A graph represents conditional independence structure among random variables
  - Nodes: random variables
  - Edges: dependency relations
- Learning: estimate parameters of the distribution from data
- Inference: compute conditional/marginal distributions

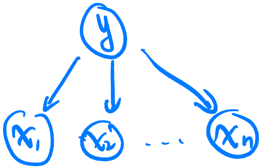
# Directed graphical model

Directed graphical model (aka Bayes nets):

- Edges represent conditional dependencies

Example: MEMM




*

$$P(y_{1:m} \mid x) = \prod_{i=1}^{m} P(y_i \mid Pa(y_i))$$

$$= \prod_{i=1}^{m} P(y_i \mid y_{i-1}, x)$$

# Undirected graphical models

Undirected graphical model (aka Markov random field):

- More natural for relational or spatial data

**Conditional random field**:

- MRF conditioned on observed data $x$
- Parameterization:

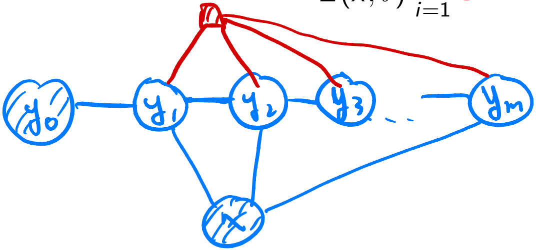
$$p(y \mid x; \theta) = \frac{1}{Z(x, \theta)} \prod_{c \in \mathcal{C}} \psi_c(y_c \mid x; \theta)$$

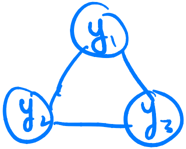
  - Clique $\mathcal{C}$: a subset of nodes/variables that form a complete graph
  - $\psi_c$: non-negative clique potential functions, also called factors
  - $Z(x, \theta)$: partition function (normalizer)

# Linear-chain CRF

Model dependence among $Y_i$'s

$$p(y \mid x; \theta) = \frac{1}{Z(x,\theta)} \prod_{i=1}^{m} \psi_i(y_1, \ldots, y_m \mid x; \theta)$$



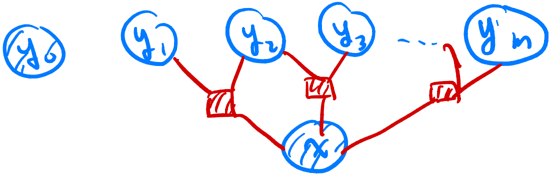

$-\ \psi(y_1, y_2, y_3)$

$-\ \psi_1(y_1, y_2)\,\psi_2(y_2, y_3)\,\psi_3(y_3, y_1)$

# Linear-chain CRF

Model dependence among *neighboring* $Y_i$'s

$$p(y \mid x; \theta) = \frac{1}{Z(x, \theta)} \prod_{i=1}^{m} \psi_i(y_i, y_{i-1} \mid x; \theta)$$



# Linear-chain CRF for sequence labeling

Log-linear potential function:

$$\psi_i(y_i, y_{i-1} \mid x; \theta) = \exp\left(\theta \cdot \phi(x, i, y_i, y_{i-1})\right)$$

$$p(y \mid x; \theta) \propto \prod_{i=1}^{m} \exp\left(\theta \cdot \phi(x, i, y_i, y_{i-1})\right)$$

$$= \exp\left(\sum_{i=1}^{m} \theta \cdot \phi(x, i, y_i, y_{i-1})\right)$$

Log-linear model with decomposable global feature function:

$$\Phi(x, y) \stackrel{\text{def}}{=} \sum_{i=1}^{m} \phi(x, i, y_i, y_{i-1})$$

$$p(y \mid x; \theta) = \frac{\exp\left(\sum_{i=1}^{m} \theta \cdot \phi(x, i, y_i, y_{i-1})\right)}{\sum_{y' \in \mathcal{Y}^m} \exp\left(\sum_{i=1}^{m} \theta \cdot \phi(x, i, y'_i, y'_{i-1})\right)}$$

$$= \frac{\exp\left(\theta \cdot \Phi(x, y)\right)}{\sum_{y' \in \mathcal{Y}^m} \exp\left(\theta \cdot \Phi(x, y)\right)}$$

# Learning

MLE:

$$\ell(\theta) = \sum_{(x,y)\in\mathcal{D}} \log p(y \mid x; \theta)$$

$$= \sum_{(x,y)\in\mathcal{D}} \log \frac{\exp\left(\theta \cdot \Phi(x, y)\right)}{\sum_{y' \in \mathcal{Y}^m} \exp\left(\theta \cdot \Phi(x, y)\right)}$$

$$\sum_i \underbrace{\phi(x, y_i, y_{i-1}, i; \theta)}_{s(y_i, y_{i-1})}$$

- Is the objective differentiable?
- Use back-propogation (autodiff) (equivalent to the forward-backward algorithm).
- Main challenge: compute the partition function.

max

$$\log \sum_{y\in\mathcal{Y}^m} \exp\left(\sum_{i=1}^{m} s(y_i, y_{i-1})\right)$$

$$= \log \sum_{y\in\mathcal{Y}^m}\left(\exp\left(\sum_{i=1}^{m-1} s(y_i, y_{i-1}) + s(y_m, y_{m-1})\right)\right)$$

$$= \log \sum_{y_m\in\mathcal{Y}} \sum_{t\in\mathcal{Y}} \sum_{y\in\mathcal{Y}^{m-1}, y_{m-1}=t} \exp\left(\sum_{i=1}^{m-1} s(y_i, y_{i-1}) + s(y_m, y_{m-1}=t)\right)$$

$$= \log \sum_{y_m\in\mathcal{Y}} \sum_{t\in\mathcal{Y}} \exp\left(s(y_m, y_{m-1}=t)\right) \boxed{\sum_{y\in\mathcal{Y}^{m-1}, y_{m-1}=t} \exp\left(\sum_{i=1}^{m-1} s(y_i, y_{i-1})\right)}$$

def

$$= \log \sum_{y_m\in\mathcal{Y}} \sum_{t\in\mathcal{Y}} \exp\left(s(y_m, y_{m-1}=t)\right) \boxed{\exp\left(\pi[m-1, t]\right)}$$

$$= \log \sum_{y_m\in\mathcal{Y}} \underbrace{\sum_{t\in\mathcal{Y}} \exp\left(s(y_m, y_{m-1}=t) + \pi[m-1, t]\right)}_{\exp\left(\pi[m, y_m]\right)}$$

# Compute the partition function

$$\log \exp(\pi[j,t]) \stackrel{\text{def}}{=} \log \sum_{y \in \mathcal{Y}^j, y_j = t} \exp\left(\sum_{i=1}^{j} s(y_i, y_{i-1})\right)$$

$$\pi[j,t] \stackrel{\text{def}}{=} \cancel{\log} \sum_{y \in \mathcal{Y}^j, y_j = t} \cancel{\exp}\left(\sum_{i=1}^{j} s(y_i, y_{i-1})\right)$$

seq len    last tag

$$\pi[j,t] = \cancel{\log} \sum_{t' \in \mathcal{Y}} \cancel{\exp}\big(s(y_j = t, y_{j-1} = t') + \pi[j-1, t']\big)$$

# Compute the "sum" partition function

$$\pi[j,t] = \log \sum_{\substack{y \in \mathcal{Y}^j \\ y_j = t}} \exp\left(\sum_{i=1}^{j} s(y_i, y_{i-1})\right)$$

$$= \log \sum_{\substack{y \in \mathcal{Y}^j \\ y_j = t}} \exp\Big(\underbrace{\sum_{i=1}^{j-1} s(y_i, y_{i-1})}_{\pi[j-1,\,?]} + s(y_j = t, y_{j-1})\Big)$$

$$= \log \sum_{t'} \sum_{\substack{y \in \mathcal{Y}^{j-1} \\ y_{j-1} = t'}} \exp\Big(\sum_{i=1}^{j-1} s(y_i, y_{i-1}) + \underbrace{s(y_j = t, y_{j-1} = t')}_{+\ \text{const}}\Big)$$

$$= \log \sum_{t'} \exp\Big[s(y_j = t, y_{j-1} = t') + \underbrace{\log \sum_{\substack{y \in \mathcal{Y}^{j-1} \\ y_{j-1} = t'}} \exp\Big(\sum_{i=1}^{j-1} s(y_i, y_{i-1})\Big)}_{\pi[j-1,\,t'] \quad \text{"const"}}\Big]$$

$$= \log \sum_{t'} \exp\left[s(y_j = t, y_{j-1} = t') + \pi[j-1, t']\right]$$

Show that

$$\log \sum_{i=1}^{n} \exp(a_i + c) = \log \sum_{i=1}^{n} \exp(a_i) + c$$

# Compute the partition function

*DP*:

$$\exp(\pi[j,t]) = \sum_{t' \in \mathcal{Y}} \exp\left(s(y_j = t, y_{j-1} = t') + \pi[j-1, t']\right)$$

$$\pi[j,t] = \log \sum_{t' \in \mathcal{Y}} \exp\left(s(y_j = t, y_{j-1} = t') + \pi[j-1, t']\right)$$

*The logsumexp function*: $\log \sum_{i=1}^{n} \exp(x_i)$

$$\text{logsumexp}(x_1, \ldots, x_n) = \log\left(e^{x_1} + \ldots + e^{x_n}\right)$$

$$\text{logsumexp}(x_1 \overset{+c}{}, \ldots, x_n \overset{+c}{}) = x^* + \log\left(e^{x_1 - x^*} + \ldots + e^{x_n - x^*}\right)$$

$$= x^* + \log\left(\frac{e^{x_1}}{e^{x^*}} + \ldots \frac{e^{x_n}}{e^{x^*}}\right)$$

$$= \underbrace{x^* - \log e^{x^*}} + \log(e^{x_1} + \ldots + e^{x_n})$$

- Same as Viterbi except that max is replaced by logsumexp.
- Is this a coincidence?

$$\max(a+b, a+c) = a + \max(b,c)$$

$$\text{logsumexp}(a+b, a+c) = a + \text{logsumexp}(b,c)$$

# Learning

Use forward algorithm to compute:

$$\texttt{loss} = -\ell(\theta, x, y) = -\log \frac{\exp(\theta \cdot \Phi(x, y))}{\sum_{y' \in \mathcal{Y}^m} \exp(\theta \cdot \Phi(x, y))}$$

```
loss.backward()
```

Exercise: show that the optimal solution satisfies

$$\sum_{(x,y)\in\mathcal{D}} \Phi_k(x, y) = \sum_{(x,y)\in\mathcal{D}} \mathbb{E}_{y\sim p_\theta}[\Phi_k(x, y)] \qquad \frac{\partial \ell(\theta)}{\partial \theta_k} = 0$$

$1(x_i = \text{apple}, y_i = N)$

Interpretation: Observed counts of feature $k$ equals expected counts of feature $k$.

# Inference

$$
\begin{aligned}
&\operatorname*{arg\,max}_{y \in \mathcal{Y}^m} \log p(y \mid x; \theta) \\
&= \operatorname*{arg\,max}_{y \in \mathcal{Y}^m} \log \cancel{\exp} \underbrace{(\theta \cdot \Phi(x, y))} - \log Z(\theta) \\
&= \operatorname*{arg\,max}_{y \in \mathcal{Y}^m} \underbrace{\sum_{i=1}^{m} s(y_i, y_{i-1})}
\end{aligned}
$$

- Find highest-scoring sequence.
- Use Viterbi + backtracking.

# Summary

**Conditional random field**

- Undirected graphical model
- Use factors to capture dependence among random variables
- Need to trade-off modeling and inference

**Linear-chain CRF for sequence labeling**

- Models dependence between neighboring outputs
- Learning: forward algorithm + backpropagation
- Inference: Viterbi algorithm

# Table of Contents

Introduction

Maximum-entropy Markov Models

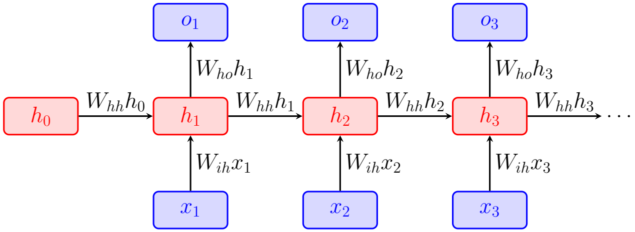
Conditional Random Field

Neural Sequence Modeling

# Classification using recurrent neural networks

Logistic regression with $h_t$ as the features:

$$p(y_i \mid x) = \mathrm{softmax}(W_{ho} h_i + b)$$

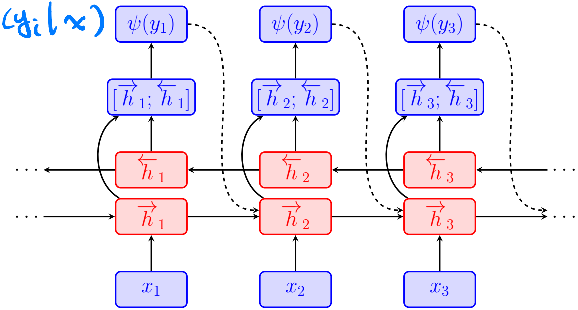



What is the problem?

# Bi-directional RNN

Use two RNNs to summarize the "past" and the "future":

$p(y_i | x)$




- Concatenated hidden states: $h_i = [\overrightarrow{h}_{1:m}; \overleftarrow{h}_{1:m}]$
- Optional: use $y_{i-1}$ as inputs: $\overrightarrow{h}'_i = [\overrightarrow{h}_i; \underbrace{W_{yh} y_{i-1}}_{\text{label embedding}}]$

MEMM

# Bi-LSTM CRF

Use neural nets to compute the local scores:

$$\boxed{s(y_i, y_{i-1})} = s_{\text{unigram}}(y_i) + s_{\text{bigram}}(y_i, y_{i-1})$$

Bi-LSTM

Basic implementation:

$$s_{\text{unigram}}(y_i) = (W_{ho}h_i + b)[y_i]$$
$$s_{\text{bigram}}(y_i, y_{i-1}) = \theta_{y_i, y_{i-1}} \quad (|\mathcal{Y}|^2 \text{ parameters })$$

Context-dependent scores:

$$s_{\text{unigram}}(y_i) = (W_{ho}h_i + b)[y_i]$$
$$s_{\text{bigram}}(y_i, y_{i-1}) = w_{y_i, y_{i-1}} \cdot h_i + b_{y_i, y_{i-1}}$$

# Does it worth it?

Typical neural sequence models:

$$p(y \mid x; \theta) = \prod_{i=1}^{m} p(y_i \mid x, y_{1:i-1}; \theta)$$

*Exposure bias*: a learning problem

- Conditions on gold $y_{1:i-1}$ during training but predicted $\hat{y}_{1:i-1}$ during test
- Solution: search-aware training

*Label bias*: a model problem

- Locally normalized models are strictly less expressive than globally normalized *given partial inputs* [Andor+ 16]
- Solution: globally normalized models or better encoder

# Does it worth it?

Empirical results from [Goyal+ 19]

| | Unidirectional | Bidirectional |
|---|---|---|
| pretrain-greedy | 76.54 | 92.59 |
| pretrain-beam | 77.76 | 93.29 |
| locally normalized | 83.9 | **93.76** |
| globally normalized | **83.93** | 93.73 |

Table 2: **Accuracy results on CCG supertagging when initialized with a regular teacher-forcing model**. Reported using *Unidirectional* and *Bidirectional* encoders respectively with fixed attention tagging decoder. *pretrain-greedy* and *pretrain-beam* refer to the output of decoding the initializer model. *locally normalized* and *globally normalized* refer to search-aware soft-beam models