Sequence Labeling

He He

New York University

October 5, 2020

Table of Contents

Sequence labeling

Language modeling as sequence labeling:

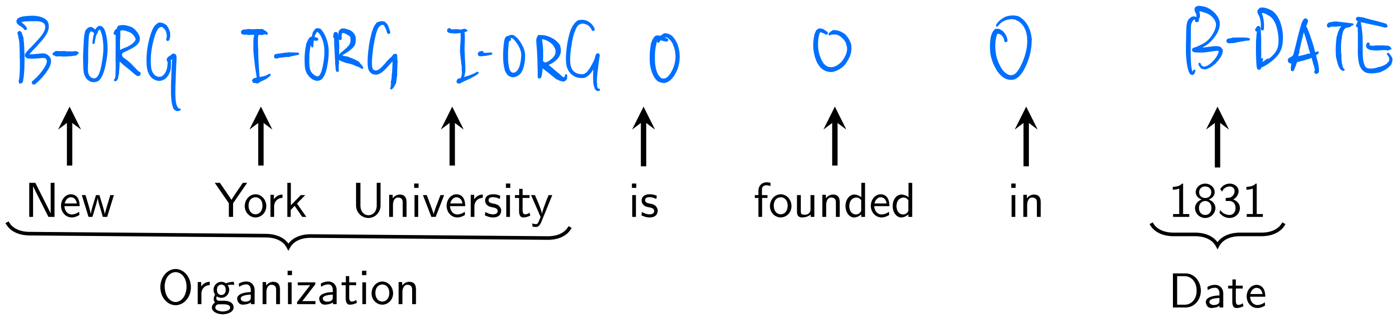
the	fox	jumped	over	the	dog	STOP
$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$
*	the	fox	jumped	over	the	dog

Part-of-speech (POS) tagging:

DT	NN	VBD	IN	DT	NN
$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$
the	fox	jumped	over	the	dog

Span prediction

Named-entity recognition (NER):



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?

Types:		WSJ		Brown	
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

Multiclass classifcation

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

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

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

Feature templates:

$T_1 = x[i], y$

Training data

x: Language is fun

y: NN VB ADJ

$\phi_1 = \mathbb{1}(x[i] = \text{language}, y = \text{NN})$

Multiclass classifcation

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

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?)
- Inference: trivial
- 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:

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

New feature templates? (See J&M 8.5.1)

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, $O(|\mathcal{Y}|^m)$
- Greedy: inexact, $O(m)$

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

last tag

seq len

$\pi[m, STOP]$

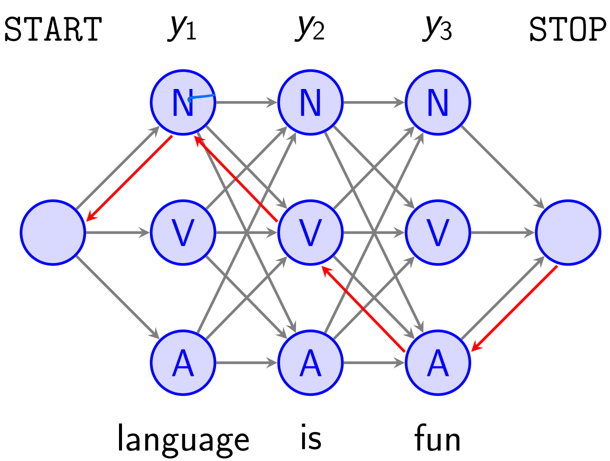
$= \max_{y_m} \pi[m,y_m]$

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: $\pi[0, t] = 0 \quad \forall t \in V$

For $j = 1 : m$

For $t = 1 : |Y|$

$\pi[j, t] = \ldots$

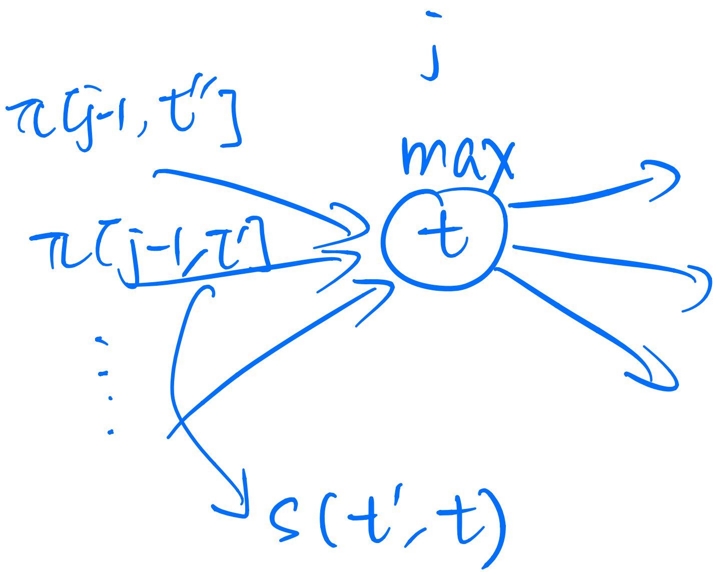
return $\pi[m, \text{STOP}]$

Time complexity? $\pi[j, t]$ $j \in \{1, \ldots m\}$ $t \in Y$

$m|Y|^2$

Viterbi decoding on the graph

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



Summary

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

- **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

Structured prediction

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

- 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:

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

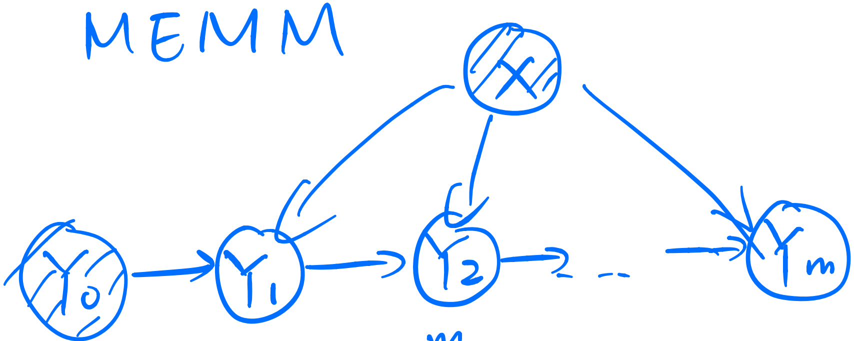
Graphical models

Graphical model:

- A joint distribution of a set of random variables
- Learn the distribution from data
- Inference: compute conditional/marginal distributions

Example of a directed graphical model (aka Bayes nets):

MEMM



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

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:

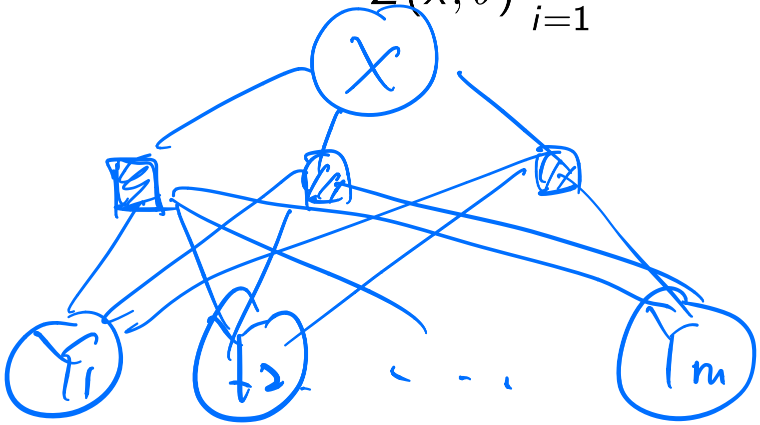
$$p(y \mid x; \theta) = \frac{1}{Z(x, \theta)} \prod_{c \in \mathcal{C}} \psi_c(y_c \mid x; \theta)$$

  - $Z(x, \theta)$: partition function (normalizer)
  - ψ_c: non-negative clique potential functions, also called factors

Linear-chain CRF

Model dependence among Y_i's

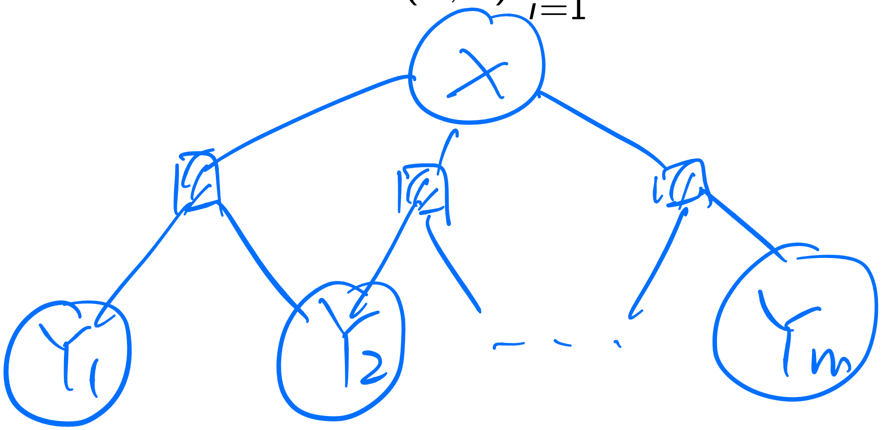
$$p(y \mid x; \theta) = \frac{1}{Z(x, \theta)} \prod_{i=1}^{m} \psi_i(y_1, \ldots, y_m \mid x; \theta)$$



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:

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

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

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

Learning

MLE:

$$
\begin{aligned}
\ell(\theta) &= \sum_{(x,y)\in\mathcal{D}} \log p(y \mid x; \theta) \\
&= \sum_{(x,y)\in\mathcal{D}} \log \frac{\exp\left(\theta \cdot \Phi(x, y)\right)}{\color{red}{\sum_{y'\in\mathcal{Y}^m}} \exp\left(\theta \cdot \Phi(x, y)\right)}
\end{aligned}
$$

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

Compute the partition function

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

$$
\begin{aligned}
&\log \sum_{y \in \mathcal{Y}^m} \exp\left(\sum_{i=1}^{m} \underline{s(y_i, y_{i-1})}\right) \\
&= \log \sum_{y \in \mathcal{Y}^m} \left(\exp\left(\sum_{i=1}^{m-1} s(y_i, y_{i-1})\right) \exp\left(s(y_m, y_{m-1})\right) \right) \\
&= \log \sum_{y_m \in \mathcal{Y}} \sum_{t \in \mathcal{Y}} \sum_{y \in \mathcal{Y}^{m-1}, y_{m-1}=t} \exp\left(\sum_{i=1}^{m-1} s(y_i, y_{i-1})\right) \exp\left(s(y_m, y_{m-1}=t)\right) \\
&= \log \sum_{y_m \in \mathcal{Y}} \sum_{t \in \mathcal{Y}} \exp\left(s(y_m, y_{m-1}=t)\right) \boxed{\sum_{y \in \mathcal{Y}^{m-1}, y_{m-1}=t} \exp\left(\sum_{i=1}^{m-1} s(y_i, y_{i-1})\right)} \\
&= \log \sum_{y_m \in \mathcal{Y}} \sum_{t \in \mathcal{Y}} \exp\left(s(y_m, y_{m-1}=t)\right) \boxed{\exp\left(\pi[m-1, y_m]\right)} \\
&= \log \sum_{y_m \in \mathcal{Y}} \underbrace{\sum_{t \in \mathcal{Y}} \exp\left(s(y_m, y_{m-1}=t) + (\pi[m-1, y_m])\right)}_{\exp(\pi[m, y_m])}
\end{aligned}
$$

def

Compute the partition function

DP:

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

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

The logsumexp function:

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

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

$x^* = \max(x_1 \ldots x_n)$

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

$a \otimes b \oplus a \otimes c = a \otimes (b \oplus c)$

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

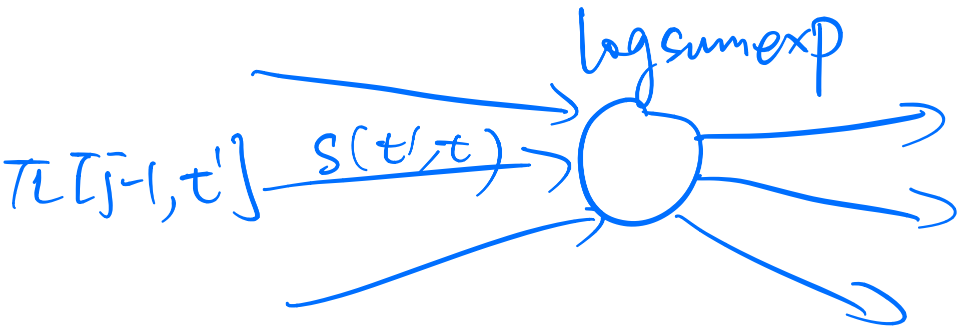
$$\text{logsumexp}(a+b, a+c) = a + \text{logsumexp}(b, c)$$

Forward algorithm on the graph

DP:

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

$\pi[m, \text{STOP}]$



Learning

Use forward algorithm to compute:

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

```
loss.backward()
```

Exercise: show that the optimal solution satisfies

$$\sum_{(x,y) \in \mathcal{D}} \Phi_k(x, y) = \sum_{(x,y) \in \mathcal{D}} \mathbb{E}_{y \sim p_\theta}\left[\Phi_k(x, y)\right]$$

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

Inference

$$
\begin{aligned}
&\arg\max_{y \in \mathcal{Y}^m} \log p(y \mid x; \theta) \\
&= \arg\max_{y \in \mathcal{Y}^m} \log \exp\left(\theta \cdot \Phi(x, y)\right) \\
&= \arg\max_{y \in \mathcal{Y}^m} \sum_{i=1}^{m} s(y_i, y_{i-1})
\end{aligned}
$$

- 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

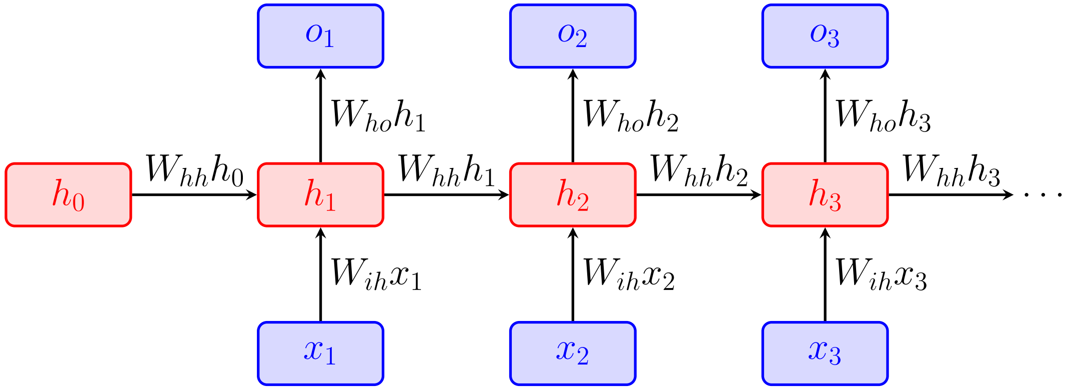
- Models dependence between neighboring outputs
- Learning: forward algorithm + backpropagation
- Inference: Viterbi algorithm

Table of Contents

Classification using recurrent neural networks

Logistic regression with h_t as the features:

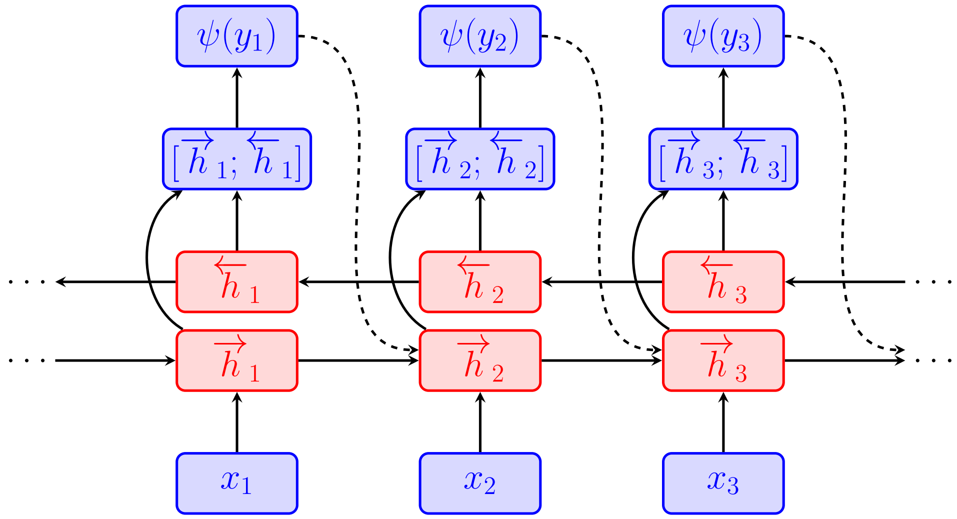
$$p(y_i \mid x) = \text{softmax}(W_{ho} h_i + b)$$



What is the problem?

Bi-directional RNN

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



- 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:

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

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:

$\phi(x_i, y_i, y_{1:i-1})$

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

$x_{1:i}$

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