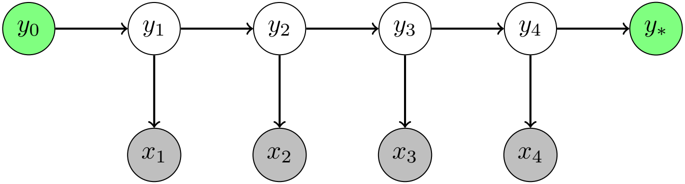
Lectures on Natural Language Processing

12. Conditional Random Fields

Karl Stratos

Review: Hidden Markov Model (HMM)

A generative model of sequence labeling

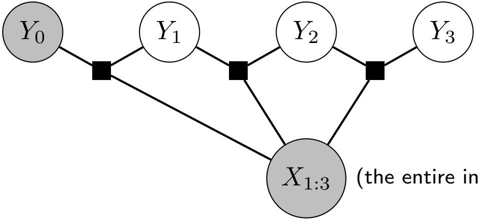


$$p(x_1 \ldots x_T,\ y_1 \ldots y_T) = \prod_{t=1}^{T} \underbrace{\tau(y_t | y_{t-1})}_{\text{transition prob}} \times \underbrace{o(x_t | y_t)}_{\text{emission prob}} \times \tau(y_* | y_T)$$

- **Forward/backward algorithm**: Marginalization in $O(T\,|\mathcal{Y}|^2)$
- **Viterbi decoding**: Best label sequence in $O(T\,|\mathcal{Y}|^2)$
- **Max marginal decoding**: Best label per position in $O(T\,|\mathcal{Y}|^2)$

Conditional Random Field (CRF) Tagger

A *discriminative* model of sequence labeling



(the entire input sequence: x_1, x_2, x_3)

$$p_\theta(y_1 \dots y_T | x_1 \dots x_T) = \frac{\prod_{t=1}^{T} \exp(\textbf{score}_\theta(x_1 \dots x_T, y_{t-1}, y_t, t))}{\sum_{y'_1 \dots y'_T \in \mathcal{Y}} \prod_{t=1}^{T} \exp(\textbf{score}_\theta(x_1 \dots x_T, y'_{t-1}, y'_t, t))}$$

Equivalently a "giant softmax" over label sequences where the score function is first-order Markovian

$$\textbf{score}_\theta(x_1 \dots x_T, y_1 \dots y_T) = \sum_{t=1}^{T} \textbf{score}_\theta(\underbrace{x_1 \dots x_T}_{\text{all inputs}}, \underbrace{y_{t-1}}_{\text{only prev. label}}, y_t, t)$$

Neural Parameterization of CRF Taggers

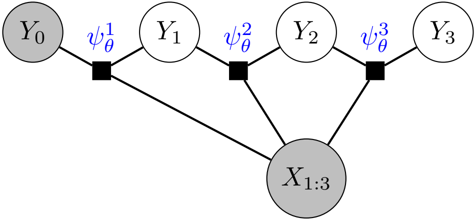
- $\textbf{score}_\theta(x_1 \ldots x_T, y, y', t) \in \mathbb{R}$ should capture how likely the t-th label is y', given $x_1 \ldots x_T$ and the previous label y
- Example: Assuming some contextual embeddings $\textbf{enc}_\theta(x_1 \ldots x_T) \in \mathbb{R}^{T \times d}$ (e.g., BiLSTM, BERT)

$$O = \textbf{enc}_\theta(x_1 \ldots x_T) W \in \mathbb{R}^{T \times |\mathcal{Y}|}$$
$$\textbf{score}_\theta(x_1 \ldots x_T, {\color{blue}y}, {\color{red}y'}, t) = T_{{\color{blue}y},{\color{red}y'}} + O_{t,{\color{red}y'}}$$

- Learnable parameters
  - $W \in \mathbb{R}^{d \times |\mathcal{Y}|}$ computes per-position label logits (i.e., O captures the "emission" scores)
  - $T \in \mathbb{R}^{|\mathcal{Y}| \times |\mathcal{Y}|}$ captures the "transition" scores
- Flexible, e.g., could define transition score $y \rightarrow y'$ to be $v_y^\top A v_{y'}$ where $v_y \in \mathbb{R}^{d'}$ is a learnable embedding of label y

Another View of CRF: Product of Potential Functions

Each maximal clique (i.e., fully connected subgraph) is associated with a nonnegative "potential function"



$$\psi_\theta^t(x_1 \ldots x_T, y, y') = \exp(\textbf{score}_\theta(x_1 \ldots x_T, y, y', t)) \geq 0$$

CRF distribution = normalized product of potential functions

$$p_\theta(y_1 \ldots y_T | x_1 \ldots x_T) \propto \prod_{t=1}^{T} \psi_\theta^t(x_1 \ldots x_T, y_{t-1}, y_t)$$

Implicit Backward Sampling in CRF Tagger

Lemma: The CRF tagger implies a first-order backward sampling procedure.

$$y_1 \dots y_T \sim p_\theta(\cdot | x_1 \dots x_T)$$

$$|||$$

$$\begin{aligned}
y_T &\sim q_\theta(\cdot | x_1 \dots x_T, T) \\
y_{T-1} &\sim q_\theta(\cdot | x_1 \dots x_T, y_T, T-1) \\
y_{T-2} &\sim q_\theta(\cdot | x_1 \dots x_T, y_{T-1}, T-2) \\
&\vdots \\
y_1 &\sim q_\theta(\cdot | x_1 \dots x_T, y_2, 1)
\end{aligned}$$

This fact allows us to efficiently sample a label sequence from the CRF tagger.

The Marginalization and Inference Problems

How can we compute the normalizer, aka. **partition function**?

$$Z_\theta(x_1 \dots x_T) = \sum_{y_1 \dots y_T \in \mathcal{Y}} \prod_{t=1}^{T} \psi_\theta^t(x_1 \dots x_T, y_{t-1}, y_t)$$

How can we find the most likely label sequence?

$$y_1^\star \dots y_T^\star = \operatorname*{arg\,max}_{y_1 \dots y_T \in \mathcal{Y}} \prod_{t=1}^{T} \psi_\theta^t(x_1 \dots x_T, y_{t-1}, y_t)$$

(Normalization not necessary for inference)

Forward Algorithm for CRFs

Dynamic programming: Given $x_1 \ldots x_T$, we fill out a table $\alpha \in \mathbb{R}^{T \times |\mathcal{Y}|}$ left-to-right where

$$\alpha(t, y) = \sum_{y_1 \ldots y_t \in \mathcal{Y}:\ y_t = y} \prod_{l=1}^{t} \psi_\theta^l(x_1 \ldots x_T, y_{l-1}, y_l)$$

Base case?

$$\alpha(1, y) =$$

Forward Algorithm for CRFs

Dynamic programming: Given $x_1 \ldots x_T$, we fill out a table $\alpha \in \mathbb{R}^{T \times |\mathcal{Y}|}$ left-to-right where

$$\alpha(t, y) = \sum_{y_1 \ldots y_t \in \mathcal{Y}:\ y_t = y} \prod_{l=1}^{t} \psi_\theta^l(x_1 \ldots x_T, y_{l-1}, y_l)$$

Base case?

$$\alpha(1, y) = \psi_\theta^1(x_1 \ldots x_T, y_0, y)$$

Forward Algorithm for CRFs: Main Body ($t > 1$)

$$\begin{aligned}
\alpha(t, y') &= \sum_{y_1 \ldots y_t:\ y_t = y'} \prod_{l=1}^{t} \psi_\theta^l(x_1 \ldots x_T, y_{l-1}, y_l) \\
&= \sum_{y_1 \ldots y_{t-1}} \left(\prod_{l=1}^{t-1} \psi_\theta^l(x_1 \ldots x_T, y_{l-1}, y_l) \right) \psi_\theta^t(x_1 \ldots x_T, y_{t-1}, y') \\
&= \sum_{y} \sum_{y_1 \ldots y_{t-1}:\ y_{t-1} = y} \left(\prod_{l=1}^{t-1} \psi_\theta^l(x_1 \ldots x_T, y_{l-1}, y_l) \right) \psi_\theta^t(x_1 \ldots x_T, y, y') \\
&= \sum_{y} \alpha(t-1, y) \psi_\theta^t(x_1 \ldots x_T, y, y')
\end{aligned}$$

Viterbi Algorithm for CRFs

Given $x_1 \ldots x_T$, we fill out a table $\pi \in \mathbb{R}^{T \times |\mathcal{Y}|}$ left-to-right where

$$\pi(t, y) = \max_{y_1 \ldots y_t \in \mathcal{Y}:\, y_t = y} \prod_{l=1}^{t} \psi_\theta^l(x_1 \ldots x_T, y_{l-1}, y_l)$$

As in HMMs, same as forward except we switch sum with max!

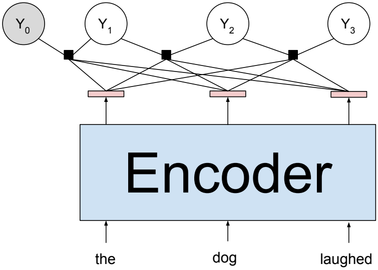
$$\pi(1, y) = \psi_\theta^1(x_1 \ldots x_T, y_0, y)$$
$$\pi(t, {\color{red}y'}) = \max_{\color{blue}y} \; \pi(t-1, {\color{blue}y})\psi_\theta^t(x_1 \ldots x_T, {\color{blue}y}, {\color{red}y'})$$
$$\mathbf{bp}(t, {\color{red}y'}) = \arg\max_{\color{blue}y} \; \pi(t-1, {\color{blue}y})\psi_\theta^t(x_1 \ldots x_T, {\color{blue}y}, {\color{red}y'})$$

Extract the argmax label sequence by backtracking

$$y_T^\star = \arg\max_{y \in \mathcal{Y}} \; \pi(T, y) \qquad y_{t-1}^\star = \mathbf{bp}(t, y_t^\star) \qquad \text{for } t = T \ldots 2$$

Training a CRF Tagger

Given labeled sequences, calculate cross-entropy in $O(T|\mathcal{Y}|^2)$ using the forward algorithm (in log space for numerical stability):



$$\max_\theta \sum_{i=1}^{N} \log p_\theta(y_1^{(i)} \dots y_{T_i}^{(i)} | x_1^{(i)} \dots x_{T_i}^{(i)})$$

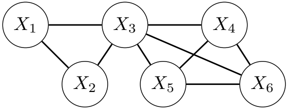
(Backprop "through structure")

After training, infer tags for any $x_1 \dots x_T$ in $O(T|\mathcal{Y}|^2)$ using the Viterbi algorithm:

$$y_1^\star \dots y_T^\star = \operatorname*{arg\,max}_{y_1 \dots y_T \in \mathcal{Y}} \log p_\theta(y_1 \dots y_T | x_1 \dots x_T)$$

Markov Random Fields (MRFs)

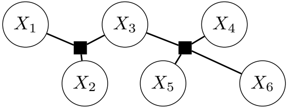
- CRF tagger is a special case of a **Markov random field** (MRF) (= *undirected* graphical model)
- MRF: Joint distribution that factorizes over *maximal* cliques C equipped with nonnegative **potential functions** ψ_C
  - Clique: Fully connected subgraph
  - Maximal clique: Clique that loses full connectivity if any node is added



$$\Pr(X_{1:6}) = \frac{1}{Z} \underbrace{\psi_{1:3}(X_{1:3})}_{\geq 0} \underbrace{\psi_{3:6}(X_{3:6})}_{\geq 0}$$

$$Z = \sum_{x_{1:6}} \psi_{1:3}(x_{1:3}) \psi_{3:6}(x_{3:6})$$

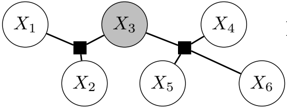
- Factor graph notation



$$\Pr(X_{1:6}) \propto \psi_{1:3}(X_{1:3}) \psi_{3:6}(X_{3:6})$$

General Marginalization and Inference in MRFs

- Some variables are observed in an MRF, work with conditional distributions (i.e., "conditional" random fields):

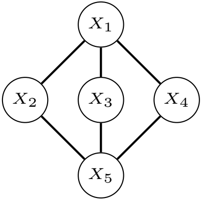


$$\Pr(X_{1:2}, X_{4:6}|X_3 = c) = \frac{1}{Z(X_3 = c)} \psi_{1:3}(X_1 X_2 c) \psi_{3:6}(X_{3:6})$$

$$Z(X_3 = c) = \sum_{x_{1:6}: x_3 = c} \psi_{1:3}(x_{1:3}) \psi_{3:6}(x_{3:6})$$

- MRF again poses general structured prediction problems, like
  - Marginalize: $\Pr(X_5 = c' | X_3 = c)$
  - Infer: $\arg\max_{x_{1:2}, x_{4:6}} \Pr(X_{1:2} = x_{1:2}, X_{4:6} = x_{4:6} | X_3 = c)$
- **Variable elimination (VE).** General "recipe" to solve these problems exactly in $O(n_{\text{infer}} K^{C_{\max}})$ time (assuming no cycles) where
  - n_{infer}: Number of variables in MRF that we're inferring
  - K: Number of possible values that variables can take
  - $C_{\max}$: Size of the *largest* maximal clique (e.g., 2 in taggers)
- Too abstract to be directly useful (e.g., must specify elimination ordering), but provides a unified framework of structured prediction (e.g., forward, Viterbi are VE on chains)

Example MRF

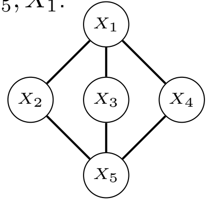


$$p(x_1, x_2, x_3, x_4, x_5) = \frac{\psi_{12}(x_1, x_2)\psi_{13}(x_1, x_3)\psi_{14}(x_1, x_4)\psi_{25}(x_2, x_5)\psi_{35}(x_3, x_5)\psi_{45}(x_4, x_5)}{\sum_{x' \in \{1 \ldots K\}^5} \psi_{12}(x'_1, x'_2)\psi_{13}(x'_1, x'_3)\psi_{14}(x'_1, x'_4)\psi_{25}(x'_2, x'_5)\psi_{35}(x'_3, x'_5)\psi_{45}(x'_4, x'_5)}$$

Computing the normalizer Z naively will take $O(K^5)$ time.

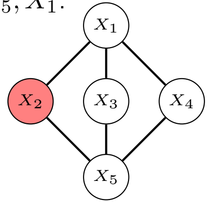
Example MRF: Variable Elimiation

VE ordering X_2, X_3, X_4, X_5, X_1:



Example MRF: Variable Elimiation

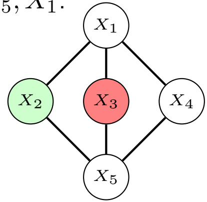
VE ordering X_2, X_3, X_4, X_5, X_1:



$$Z = \sum_{x_3, x_4, x_5, x_1} \psi_{13}(x_1, x_3)\psi_{14}(x_1, x_4)\psi_{35}(x_3, x_5)\psi_{45}(x_4, x_5) \underbrace{\left(\sum_{x_2} \psi_{12}(x_1, x_2)\psi_{25}(x_2, x_5)\right)}_{\phi_2(x_1, x_5)}$$

Example MRF: Variable Elimiation

VE ordering X_2, X_3, X_4, X_5, X_1:

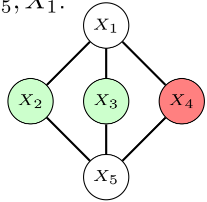


$$Z = \sum_{x_3, x_4, x_5, x_1} \psi_{13}(x_1, x_3)\psi_{14}(x_1, x_4)\psi_{35}(x_3, x_5)\psi_{45}(x_4, x_5) \underbrace{\left(\sum_{x_2} \psi_{12}(x_1, x_2)\psi_{25}(x_2, x_5)\right)}_{\phi_2(x_1, x_5)}$$

$$= \sum_{x_4, x_5, x_1} \psi_{14}(x_1, x_4)\psi_{45}(x_4, x_5)\phi^2(x_1, x_5) \underbrace{\left(\sum_{x_3} \psi_{13}(x_1, x_3)\psi_{35}(x_3, x_5)\right)}_{\phi_3(x_1, x_5)}$$

Example MRF: Variable Elimiation

VE ordering X_2, X_3, X_4, X_5, X_1:



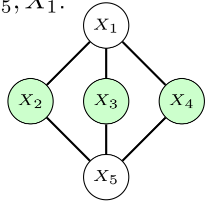
$$Z = \sum_{x_3, x_4, x_5, x_1} \psi_{13}(x_1, x_3)\psi_{14}(x_1, x_4)\psi_{35}(x_3, x_5)\psi_{45}(x_4, x_5) \underbrace{\left(\sum_{x_2} \psi_{12}(x_1, x_2)\psi_{25}(x_2, x_5)\right)}_{\phi_2(x_1, x_5)}$$

$$= \sum_{x_4, x_5, x_1} \psi_{14}(x_1, x_4)\psi_{45}(x_4, x_5)\phi^2(x_1, x_5) \underbrace{\left(\sum_{x_3} \psi_{13}(x_1, x_3)\psi_{35}(x_3, x_5)\right)}_{\phi_3(x_1, x_5)}$$

$$= \sum_{x_5, x_1} \phi^2(x_1, x_5)\phi^3(x_1, x_5) \underbrace{\left(\sum_{x_4} \psi_{14}(x_1, x_4)\psi_{45}(x_4, x_5)\right)}_{\phi_4(x_1, x_5)}$$

Example MRF: Variable Elimiation

VE ordering X_2, X_3, X_4, X_5, X_1:



$$Z = \sum_{x_3, x_4, x_5, x_1} \psi_{13}(x_1, x_3)\psi_{14}(x_1, x_4)\psi_{35}(x_3, x_5)\psi_{45}(x_4, x_5) \underbrace{\left(\sum_{x_2} \psi_{12}(x_1, x_2)\psi_{25}(x_2, x_5) \right)}_{\phi_2(x_1, x_5)}$$

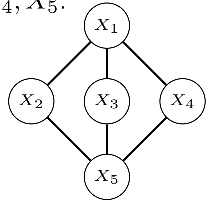
$$= \sum_{x_4, x_5, x_1} \psi_{14}(x_1, x_4)\psi_{45}(x_4, x_5)\phi^2(x_1, x_5) \underbrace{\left(\sum_{x_3} \psi_{13}(x_1, x_3)\psi_{35}(x_3, x_5) \right)}_{\phi_3(x_1, x_5)}$$

$$= \sum_{x_5, x_1} \phi^2(x_1, x_5)\phi^3(x_1, x_5) \underbrace{\left(\sum_{x_4} \psi_{14}(x_1, x_4)\psi_{45}(x_4, x_5) \right)}_{\phi_4(x_1, x_5)}$$

Runtime: $O(K^3)$

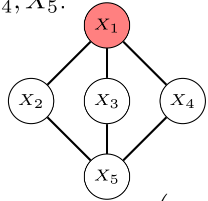
Example MRF: Variable Elimiation, Different Ordering

VE ordering X_1, X_2, X_3, X_4, X_5:



Example MRF: Variable Elimiation, Different Ordering

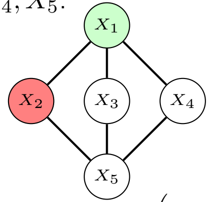
VE ordering X_1, X_2, X_3, X_4, X_5:



$$Z = \sum_{x_2, x_3, x_4, x_5} \psi_{25}(x_2, x_5)\psi_{35}(x_3, x_5)\psi_{45}(x_4, x_5) \underbrace{\left(\sum_{x_1} \psi_{12}(x_1, x_2)\psi_{13}(x_1, x_3)\psi_{14}(x_1, x_4) \right)}_{\psi^1(x_2, x_3, x_4)}$$

Example MRF: Variable Elimiation, Different Ordering

VE ordering X_1, X_2, X_3, X_4, X_5:

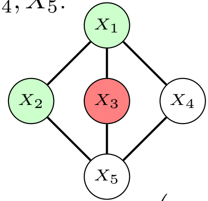


$$Z = \sum_{x_2,x_3,x_4,x_5} \psi_{25}(x_2,x_5)\psi_{35}(x_3,x_5)\psi_{45}(x_4,x_5) \underbrace{\left(\sum_{x_1} \psi_{12}(x_1,x_2)\psi_{13}(x_1,x_3)\psi_{14}(x_1,x_4)\right)}_{\psi^1(x_2,x_3,x_4)}$$

$$= \sum_{x_3,x_4,x_5} \psi_{35}(x_3,x_5)\psi_{45}(x_4,x_5) \underbrace{\left(\sum_{x_2} \psi_{25}(x_2,x_5)\psi^1(x_2,x_3,x_4)\right)}_{\psi^2(x_3,x_4,x_5)}$$

Example MRF: Variable Elimiation, Different Ordering

VE ordering X_1, X_2, X_3, X_4, X_5:



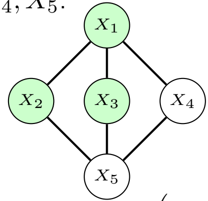
$$Z = \sum_{x_2, x_3, x_4, x_5} \psi_{25}(x_2, x_5)\psi_{35}(x_3, x_5)\psi_{45}(x_4, x_5) \underbrace{\left(\sum_{x_1} \psi_{12}(x_1, x_2)\psi_{13}(x_1, x_3)\psi_{14}(x_1, x_4) \right)}_{\psi^1(x_2, x_3, x_4)}$$

$$= \sum_{x_3, x_4, x_5} \psi_{35}(x_3, x_5)\psi_{45}(x_4, x_5) \underbrace{\left(\sum_{x_2} \psi_{25}(x_2, x_5)\psi^1(x_2, x_3, x_4) \right)}_{\psi^2(x_3, x_4, x_5)}$$

$$= \sum_{x_4, x_5} \psi_{45}(x_4, x_5) \underbrace{\left(\sum_{x_3} \psi_{35}(x_3, x_5)\psi^2(x_3, x_4, x_5) \right)}_{\psi^3(x_4, x_5)}$$

Example MRF: Variable Elimiation, Different Ordering

VE ordering X_1, X_2, X_3, X_4, X_5:



$$Z = \sum_{x_2, x_3, x_4, x_5} \psi_{25}(x_2, x_5)\psi_{35}(x_3, x_5)\psi_{45}(x_4, x_5) \underbrace{\left(\sum_{x_1} \psi_{12}(x_1, x_2)\psi_{13}(x_1, x_3)\psi_{14}(x_1, x_4)\right)}_{\psi^1(x_2, x_3, x_4)}$$

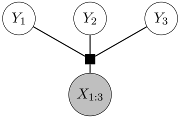
$$= \sum_{x_3, x_4, x_5} \psi_{35}(x_3, x_5)\psi_{45}(x_4, x_5) \underbrace{\left(\sum_{x_2} \psi_{25}(x_2, x_5)\psi^1(x_2, x_3, x_4)\right)}_{\psi^2(x_3, x_4, x_5)}$$

$$= \sum_{x_4, x_5} \psi_{45}(x_4, x_5) \underbrace{\left(\sum_{x_3} \psi_{35}(x_3, x_5)\psi^2(x_3, x_4, x_5)\right)}_{\psi^3(x_4, x_5)}$$

Runtime: $O(K^4)$

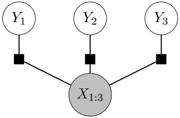
General Tagging with MRFs

- No independence assumptions: $O(T\,|\mathcal{Y}|^T)$



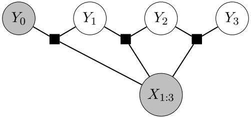
$$p_\theta(y_{1:3}|x_{1:3}) \propto \exp(\textbf{score}_\theta(x_{1:3}, y_{1:3}))$$
$$C_{\max} = 3$$

- Greedy tagging (i.e., softmax per position): $O(T\,|\mathcal{Y}|)$



$$p_\theta(y_{1:3}|x_{1:3}) \propto \prod_{t=1}^{3} \exp(\textbf{score}_\theta(x_{1:3}, y_t, t))$$
$$C_{\max} = 1$$

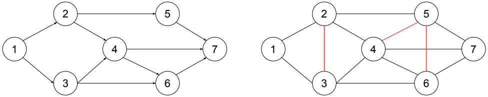
- First-order CRF: $O(T\,|\mathcal{Y}|^2)$



$$p_\theta(y_{1:3}|x_{1:3}) \propto \prod_{t=1}^{3} \exp(\textbf{score}_\theta(x_{1:3}, y_{t-1}, y_t, t))$$
$$C_{\max} = 2$$

More Facts About Graphical Models

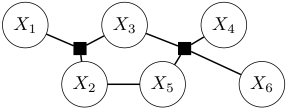
Any directed (acyclic) graph can be expressed by an equivalent MRF



- Forward algorithm for HMM: VE with left-to-right elimination ordering, generalizable to trees

VE applicable only if there's no cycle (e.g., sequences, trees)

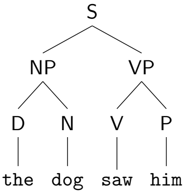
- If cycle between unobserved variables, $O(n_{\text{infer}} K^{C_{\max}})$ runtime guarantee doesn't hold, e.g., marginalization intractable in



- Can technically combine factors until there's no cycle and apply VE, but that's no better than brute-force
- Efficient approximations possible: loopy belief propagation

Review: PCFG

A top-down generative model of parsing

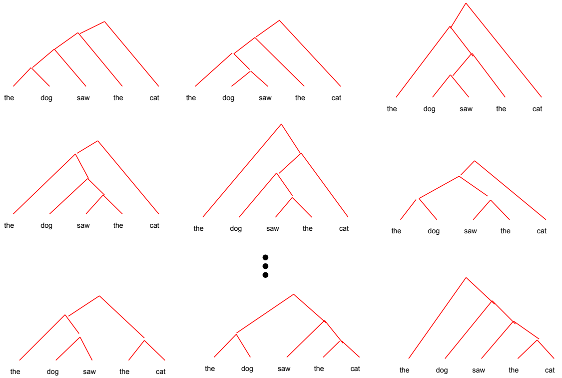


$$p(\quad) = q(\mathsf{S} \to \mathsf{NP\ VP}) \times q(\mathsf{NP} \to \mathsf{D\ N})$$
$$\times q(\mathsf{D} \to \mathsf{the}) \times q(\mathsf{N} \to \mathsf{dog})$$
$$\times q(\mathsf{VP} \to \mathsf{V\ P}) \times q(\mathsf{V} \to \mathsf{saw}) \times q(\mathsf{P} \to \mathsf{him})$$

- **Inside algorithm**: Marginalization in $O(T^3 |\mathcal{R}|)$
- **CKY decoding**: Best tree in $O(T^3 |\mathcal{R}|)$
- **Max marginal decoding**: Outside algorithm $O(T^3 |\mathcal{R}|)$, labeled recall algorithm $O(T^3)$

Discriminative Parsing

Given a sentence, define a conditional distribution over all possible (binary) trees

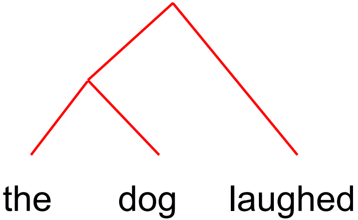


Questions:

1. How do we represent a tree?
2. How do we define the score of a tree to allow for efficient calculations?

CRF Parser

Unlabeled binary tree = a valid set of spans



$$\equiv \quad \{(1,3),(1,2)\}$$

Conditional tree probability as normalized product of potentials

$$p_\theta(\tau|x_1 \ldots x_T) \propto \prod_{(s,t)\in\tau} \exp(\textbf{score}_\theta(x_1 \ldots x_T, s, t))$$

Equivalently a "giant softmax" over trees, where the tree score is

$$\textbf{score}_\theta(x_1 \ldots x_T, \tau) = \sum_{(s,t)\in\tau} \textbf{score}_\theta(x_1 \ldots x_T, s, t)$$

Neural Parameterization of CRF Parsers

- **score**$_\theta(x_1 \ldots x_T, i, j) \in \mathbb{R}$ should capture how likely (i, j) is a span in the underlying tree
- Example: Given some text encoder (e.g., BERT)

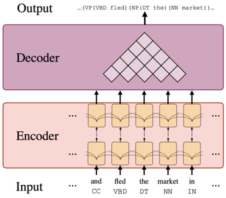
$$h_1 \ldots h_T = \textbf{enc}_\theta(x_1 \ldots x_T)$$

$$\textbf{score}_\theta(x_1 \ldots x_T, i, j) = v^\top \mathrm{ReLU}(W(h_j - h_i) + b) + b'$$

- Can also use labeled score (e.g., for constituency parsing)

$$\textbf{score}_\theta(x_1 \ldots x_T, i, j, l) = v_l^\top \mathrm{ReLU}(W(h_j - h_i) + b) + b'_l$$

- Given training data $(x^{(1)}, \tau^{(1)}) \ldots (x^{(N)}, \tau^{(N)})$ a CRF parser can be trained by cross-entropy or max-margin loss

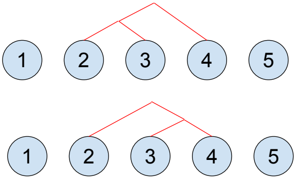


IF we can do marginalization/inference!

(Image credit: Kitaev and Klein (2018))

Notation: Set of All Trees Over Some Span

$B_T[i, j]$: all binary trees over the span $[i, j]$ in a sequence of length T



$$B_5[2, 4] = \{\{[2, 4], [2, 3]\}, \{[2, 4], [3, 4]\}\}$$

Span-conditional tree distribution by

$$p_\theta(\tau | x_1 \ldots x_T, i, j) = \frac{\exp(\textbf{score}_\theta(x_1 \ldots x_T, \tau))}{\sum_{\tau' \in B_T[i,j]} \exp(\textbf{score}_\theta(x_1 \ldots x_T, \tau'))} \quad \forall \tau \in B_T[i, j]$$

Special case: final CRF parser

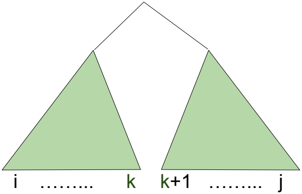
$$p_\theta(\tau | x_1 \ldots x_T) = p_\theta(\tau | x_1 \ldots x_T, 1, T)$$

Implicit Top-Down Sampling in CRF Parser

Lemma: The CRF parser implies a top-down sampling procedure.

$$\tau \sim p_\theta(\cdot | x_1 \ldots x_T, i, j)$$

$$|||$$



$$k \sim q_\theta(\cdot | x_1 \ldots x_T, i, j)$$
$$\tau_{\text{left}} \sim p_\theta(\cdot | x_1 \ldots x_T, i, k)$$
$$\tau_{\text{right}} \sim p_\theta(\cdot | x_1 \ldots x_T, k+1, T)$$

Marginalization and Inference

Given a sentence $x_1 \dots x_T$,

1. How can we compute the partition function?

$$\sum_{\tau \in B_T[1,T]} \exp(\textbf{score}_\theta(x_1 \dots x_T, \tau))$$

2. What is the most likely tree?

$$\tau^\star = \arg\max_{\tau \in B_T[1,T]} \textbf{score}_\theta(x_1 \dots x_T, \tau)$$

Can use variants of the inside/CKY algorithm for PCFGs

Inside Algorithm for CRF Parser

Dynamic programming: Given $x_1 \ldots x_T$, we fill out a table $\alpha \in \mathbb{R}^{T \times T}$ bottom up:

$$\alpha(i,j) = \sum_{\tau \in B_T[i,j]} \exp(\textbf{score}_\theta(x_1 \ldots x_T, \tau))$$

Base case?

$$\alpha(i,i) = \exp(\textbf{score}_\theta(x_1 \ldots x_T, i, i))$$

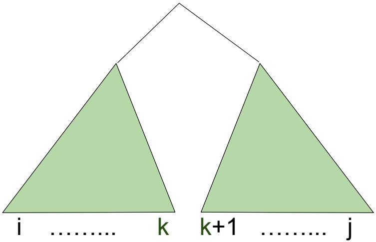
Inside Algoirthm for CRF Parser: Main Body

$$\alpha(i,j) = \sum_{\tau \in B_T[i,j]} \prod_{(s,t)\in\tau} \exp(\textbf{score}_\theta(x_1 \ldots x_T, s, t))$$

$$= \sum_{\substack{i \leq k < j \\ l \in B_T[i,k] \\ r \in B_T[k+1,j]}} \exp(\textbf{score}_\theta(x_1 \ldots x_T, i, j)) \times \left(\prod_{(s,t)\in l} \exp(\textbf{score}_\theta(x_1 \ldots x_T, s, t)) \right)$$

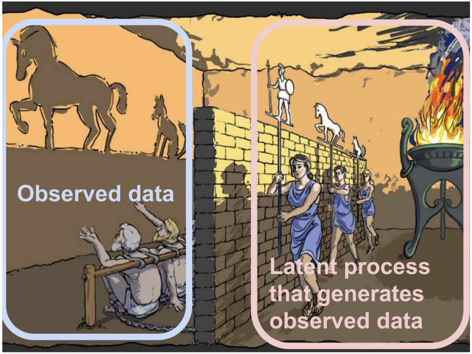
$$\times \left(\prod_{(a,b)\in r} \exp(\textbf{score}_\theta(x_1 \ldots x_T, a, b)) \right)$$

$$= \sum_{i \leq k < j} \exp(\textbf{score}_\theta(x_1 \ldots x_T, i, j)) \times \alpha(i,k) \times \alpha(k+1,j)$$



Introducing Latent Variables in Generative Models

- Generative models (e.g., LMs) define $p_\theta(x)$
  - The only random variable is observation x
- Idea: Introduce additional variable z and explicitly model an unseen generative process
  - We believe the process to be true (or at least useful for something), even though we don't observe it



(Original Image: 4edges/Wikimedia Commons)

Latent-Variable Generative Models (LVGMs)

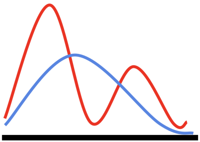
- p_θ defining a *joint* distribution over observation $x \in \mathcal{X}$ and latent variable $z \in \mathcal{Z}$

$$p_\theta(x, z) = \underbrace{\kappa_\theta(x|z)}_{\text{conditional likelihood}} \times \underbrace{\pi_\theta(z)}_{\text{prior}}$$

- Very general definition
  - Can be discrete, continuous, or mixed
  - x can be structured, z can be structured, or both
- Why introduce latent variables?
  1. Clear generative story: Sample $z \sim \pi_\theta(z)$, then $x \sim \kappa_\theta(\cdot|z)$
  2. *Marginal* observation distribution can be more expressive
  3. Latent variables can be useful: Controllable generation (i.e., change z to get x we want), z natural representation of x

Marginal Observation Distribution

- LVGM defines a marginal distribution m_θ over $\mathcal{X}$
  - If z is discrete: $m_\theta(x) = \sum_{z \in \mathcal{Z}} p_\theta(x, z)$
  - If z is continuous: $m_\theta(x) = \int_{z \in \mathcal{Z}} p_\theta(x, z) dz$
  - If z is mixed: sum/integrate out appropriate dimensions
- m_θ can express a larger family of distributions
- Example: Bimodal distribution over $\mathcal{X} = \mathbb{R}$ cannot be expressed by any single Gaussian $\mathcal{N}(\mu, \sigma^2)$



- But can be expressed by a mixture of two Gaussians:

$$m_\theta(x) = \pi_1 \mathcal{N}(\mu_1, \sigma_1^2)(x) + \pi_2 \mathcal{N}(\mu_2, \sigma_2^2)(x)$$

Discrete latent variable $\mathcal{Z} = \{1, 2\}$

Better Explanation of Data

- Suppose iid samples from unknown $\mathbf{pop}$ over $\{a, b\}^{10}$ look like

$$x^{(1)} = (a, a, a, a, a, a, a, a, a, a) \qquad x^{(2)} = (b, b, b, b, b, b, b, b, b, b)$$
$$x^{(3)} = (a, a, a, a, a, a, a, a, a, a) \qquad x^{(4)} = (b, b, b, b, b, b, b, b, b, b)$$
$$x^{(5)} = (a, a, a, a, a, a, a, a, a, a) \qquad x^{(6)} = (b, b, b, b, b, b, b, b, b, b)$$

- Bag-of-words model $p_\theta(x) = \prod_{j=1}^{10} p_\theta(x_j)$?
  - The model's independence assumption is clearly wrong!
  - Poor data fit: At most $p_\theta(x^{(i)}) = 2^{-10} < 0.001$ for each i
- LVGM $m_\theta(x) = \sum_{z \in \{1,2\}} \pi_\theta(z) \times \prod_{j=1}^{10} \kappa_\theta(x_j|z)$
  - The model makes the right assumption (draw a latent "topic" z and draw observation conditioned on z).
  - Can achieve $m_\theta(x^{(i)}) = 2^{-1}$ for each i with only twice more parameters
  - Also likely to generalize better (i.e., higher log liklihood of future samples)

Example LVGMs

- **HMMs**: $z \in \mathcal{Z}^T$ (unobserved label sequence), $x \in \mathcal{V}^T$ (sentence)

$$p_\theta(x, z) = \prod_{t=1}^{T+1} t_\theta(z_t|z_{t-1}) \times \prod_{t=1}^{T} o_\theta(x_t|z_t)$$

- **Gaussian LM**: $z \in \mathbb{R}^d$ ("thought vector"), $x \in \mathcal{V}^T$ (sentence)

$$p_\theta(x, z) = \mathcal{N}(0_d, I_{d\times d})(z) \times \prod_{t=1}^{T+1} p_\theta(x_t|x_{<t}, z)$$

- **Document hashing**: $z \in \{0,1\}^d$ ("hash code"), $x \in \mathbb{R}^V$ (TFIDF document encoding)

$$p_\theta(x, z) = \prod_{j=1}^{d} \mathrm{Bernoulli}(\lambda_j)(z_j) \times \prod_{k=1}^{V} p_\theta(x_k|z)$$

Marginal Log-Likelihood

- Training objective: Maximize marginal log-likelihood (MLL)

$$L(\theta) = \mathop{\mathbf{E}}_{x \sim \mathbf{pop}} [\log m_\theta(x)]$$

  (Equivalent to cross entropy minimization, but convenient to frame as maximization for later)

- Requires the ability to calculate marginal probability of x!

$$m_\theta(x) = \mathop{\mathbf{E}}_{z \sim \pi_\theta} [\kappa_\theta(x|z)]$$

- Sometimes we can calculate it exactly (best scenario)
  - z is discrete and $\mathcal{Z}$ is small: $m_\theta(x) = \sum_{z \in \mathcal{Z}} p_\theta(x, z)$ directly computable
  - p_θ makes Markov assumptions: $m_\theta(x)$ computable by dynamic programming (e.g., forward algorithm for HMMs)
- In general, we need to approximate by sampling