TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2019

Deep Graphical Models

Expectation Gradient

Belief Propagation

Connectionist Temporal Classification (CTC)

The Big Picture I

Conditional vs. Unconditional

$$\Phi^* = \operatorname*{argmin}_{\Phi} E_{(x,y)\sim \mathrm{Pop}} \; -\ln \; P(y|x)$$

$$\Phi^* = \operatorname*{argmin}_{\Phi} E_{y\sim \mathrm{Pop}} \; -\ln \; P(y)$$

This is a non-distinction: the issues in the to the conditional case are exactly the same as in the unconditional case.

The Big Picture II

The structured case: $y \in \mathcal{Y}$ where $\mathcal{Y}$ is discrete but iteration over $\hat{y} \in \mathcal{Y}$ is infeasible.

Graphical models rely on assumptions about the structure of $P_\Phi(y)$.

Loopy Graphical Models can use weaker assumptions than those supporting exact computation of $P_\Phi(y)$.

For loopy graphical Models there are various methods of performing approximate gradient descent on cross-entropy loss.

Colorization — a Self-Supervised Problem

x is a black and white image.

y is a "color" image drawn from $\mathrm{Pop}(y|x)$ where the color has been rounded to one of, say, 100 color words.

$\hat{y}$ is an arbitrary color image.

$P_\Phi(\hat{y}|x)$ is the probability that model Φ assigns to the color image $\hat{y}$ given black and white image x.

Colorizing Superpixels

SLIC superpixels, Achanta et al.

$\hat{y}[i]$ is the color value of superpixel i in color image $\hat{y}$.

$\hat{y}[(i,j)]$ is the pair $(\hat{y}[i], \hat{y}[j])$ for neighbors i and j.

Exponential Softmax

$$P_\Phi(\hat{y}|x) = \operatorname*{softmax}_{\hat{y}} \; s_\Phi(\hat{y}|x)$$

Let $\mathcal{C}$ be colors, $\mathcal{I}$ be superpixels, and $\mathcal{E}$ be edges.

We will compute

a unary potential tensor $s_i[c] = s_\Phi(c|x,i)$

a binary potential tensor $s_e[c,c'] = s_\Phi(c,c'|x,e)$

$$s_\Phi(\hat{y}|x) = \sum_{i \in \mathcal{I}} s_i[\hat{y}[i]] + \sum_{e \in \mathcal{E}} \; s_e[\hat{y}[e.i], \hat{y}[e.j]]$$

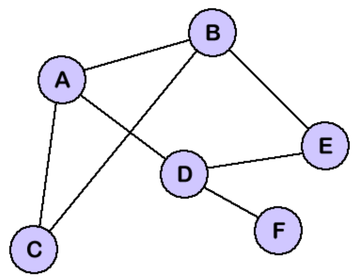
Backpropagation

The input is the image x and the parameter package Φ

$$\begin{aligned} &\vdots \\ s_i[c] &= \ldots \\ s_e[c, c'] &= \ldots \\ \mathcal{L} &= -\ln\; P(y \mid s_{\mathcal{I}}[\mathcal{C}],\; s_{\mathcal{E}}[\mathcal{C}, \mathcal{C}]) \end{aligned}$$

We need to compute $s_i.\mathrm{grad}[c]$ and $s_e.\mathrm{grad}[c, c']$.

General Markov Random Fields (MRFs)



$$s(\hat{y}) = \sum_{i \in \text{Nodes}} s_i[\hat{y}[i]] + \sum_{e \in \text{Edges}} s_e[\hat{y}[e.i], \hat{y}[e.j]]$$

Node Potentials Edge Potentials

An Example

Consider an image with three superpixels A, B and C where each superpixel is to labeled as either "foreground" or background.

Suppose the unary potentials are all zero.

$$s_A(\text{Foreground}) = s_A(\text{Background}) = 0$$
$$s_B(\text{Foreground}) = s_B(\text{Background}) = 0$$
$$s_C(\text{Foreground}) = s_C(\text{Background}) = 0$$

The Binary Potentials

Let F_A be the proposition that A is forground and similarly for F_B and F_C.

We can express $P_A \Rightarrow P_B$ with

$$s_{A,B}(\text{Foreground}, \text{Background}) = -1$$
$$s_{A,B}(\text{Foreground}, \text{Foreground}) = 1$$
$$s_{A,B}(\text{Background}, \text{Background}) = 1$$
$$s_{A,B}(\text{Background}, \text{Foreground}) = 1$$

The binary potentials are then given by $F_A \Rightarrow F_B$, $F_B \Rightarrow F_C$, $F_C \Rightarrow F_A$.

The Full Configuration Potential

For any configuration $\hat{y}$ we have that $s(\hat{y})$ is the sum of the unary and binary potentials.

If none are foreground we have $s(\hat{y}) = 3$

If one is foreground we have $s(\hat{y}) = -1 + 1 + 1 = 1$

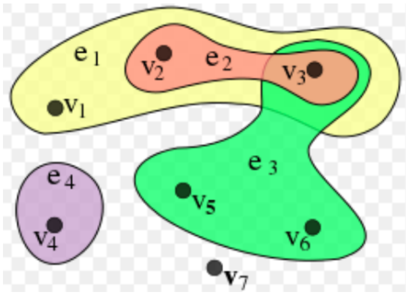
If two are foreground we also have $s(\hat{y}) = -1 + 1 + 1 = 1$

If all are foreground we have $s(\hat{y}) = 3$.

$$Z = 6 * 1 + 2 * 3 = 12 \quad P_A(\text{Foregound}) = \frac{3 * 1 + 3}{12} = \frac{1}{2}$$

Hyper-Graphs: More General and More Concise

A hyper-edge is a subset of nodes.



$$s(\hat{y}) = \sum_{i \in \text{Nodes}} s_i[\hat{y}[i]] + \sum_{e \in \text{Edges}} s_e[\hat{y}[e.i], \hat{y}[e.j]]$$

$$s(\hat{y}) = \sum_{e \in \text{HyperEdges}} s_e[\hat{y}[e]]$$

Backpropagation

The input is the image x and the parameter package Φ

$$
\begin{aligned}
&\vdots \\
s_e[\hat{y}] &= \ldots \\
\mathcal{L} &= -\ln\; P(y \mid s_{\mathcal{E}}[\mathcal{Y}])
\end{aligned}
$$

We abbreviate $P(\hat{y} \mid s_{\mathcal{E}}[\mathcal{Y}])$ as $P_s(\hat{y})$ — the distribution on $\hat{y}$ defined by the tensor s.

We need to compute $\nabla_s - \ln P_s(y)$, or equivalently, $s_e.\mathrm{grad}[\tilde{y}]$.

Back-Propagation Through An Exponential Softmax

$$\begin{aligned}\mathrm{loss}(s, y) &= -\ln\left(\frac{1}{Z(s)}\, e^{s(y)}\right) \\ &= \ln Z(s) - s(y)\end{aligned}$$

$$s_e.\mathrm{grad}[\tilde{y}] = \left(\frac{1}{Z}\sum_{\hat{y}} e^{s(\hat{y})}\,(\partial s(\hat{y})/\partial s_e[\tilde{y}])\right) - (\partial s(y)/\partial s_e[\tilde{y}])$$

Back-Propagation Through An Exponential Softmax

$$
\begin{aligned}
s_e.\mathrm{grad}[\tilde{y}] &= \left(\frac{1}{Z}\sum_{\hat{y}} e^{s(\hat{y})}\left(\partial s(\hat{y})/\partial s_e[\tilde{y}]\right)\right) - \left(\partial s(y)/\partial s_e[\tilde{y}]\right) \\
&= \left(\sum_{\hat{y}} P_s(\hat{y})\left(\partial s(\hat{y})/\partial s_e[\tilde{y}]\right)\right) - \left(\partial s(y)/\partial s_e[\tilde{y}]\right) \\
&= E_{\hat{y}\sim P_s}\mathbb{1}[\hat{y}[e] = \tilde{y}] - \mathbb{1}[y[e] = \tilde{y}] \\
&= P_{\hat{y}\sim P_s}(\hat{y}[e] = \tilde{y}) - \mathbb{1}[y[e] = \tilde{y}]
\end{aligned}
$$

Hyperedge Marginals

$$s.\mathrm{grad}[e, \tilde{y}] = P_{\hat{y} \sim P_s}(\hat{y}[e] = \tilde{y}) - \mathbb{1}[y[e] = \tilde{y}]$$

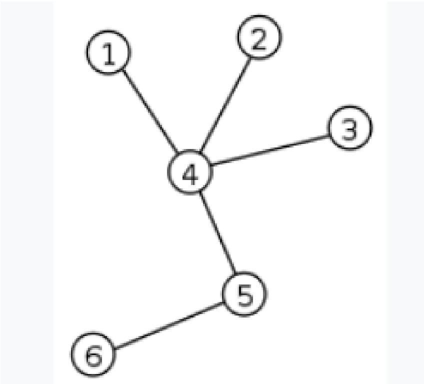
We will write $P_e(\tilde{y})$ for $P_{\hat{y} \sim P_s}(\hat{y}(e) = \tilde{y})$.

To compute s.grad it suffices to compute $P_e(\tilde{y})$.

We now focus on computing the hyperedge marginals for a given hyperedge score function (MRF) s.

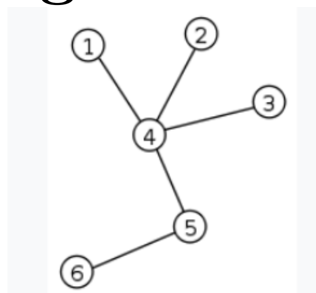
For Tree-Strutured Models
The Hyperedge Marginals Can be Computed Exactly

$$s.\mathrm{grad}[e, \tilde{y}] = \color{red}{P_e(\tilde{y})} - \mathbb{1}[y[e] = \tilde{y}]$$



For trees we can compute $P_e(\tilde{y})$ exactly by message passing, a.k.a., belief propagation.

Defining the Messages

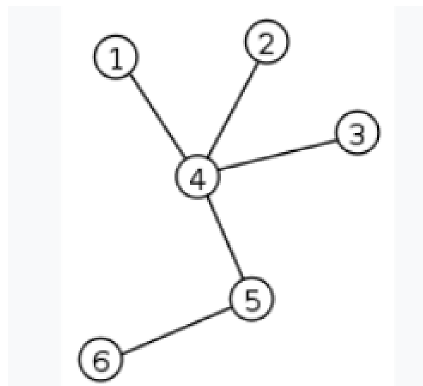


For each edge $\{i, j\}$ and possible value $\tilde{y}$ for node i we define $Z_{j \to i}[\tilde{y}]$ to be the partition function for the subtree attached to i through j and with $\hat{y}[i]$ restricted to $\tilde{y}$.

The function $Z_{j \to i}$ on the possible values of node i is called the **message** from j to i.

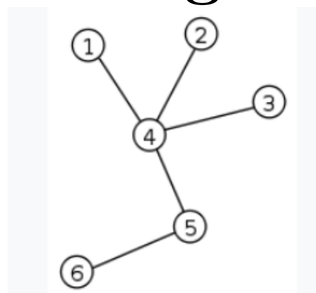
The reverse direction message $Z_{i \to j}$ is defined similarly.

Computing the Messages



$$Z_{j\to i}[\tilde{y}] = \sum_{\tilde{y}'} e^{s_j[\tilde{y}'] + s_{\{j,i\}}[\{\tilde{y}',\tilde{y}\}]} \left(\prod_{k\in N(j),\, k\neq i} Z_{k\to j}[\tilde{y}'] \right)$$

Computing Node Marginals from Messages



$$Z_i(\tilde{y}) \doteq \sum_{\hat{y}:\ \hat{y}[i]=\tilde{y}} e^{s(\hat{y})}$$

$$= e^{s_i[\tilde{y}]} \left(\prod_{j \in N(i)} Z_{j \to i}[\tilde{y}] \right)$$

$$P_i(\tilde{y}) = Z_i(\tilde{y})/Z, \qquad Z = \sum_{\tilde{y}} Z_i(\tilde{y})$$

Computing Edge Marginals from Messages

$$
\begin{aligned}
Z_{\{i,j\}}(\tilde{y}) &\doteq \sum_{\hat{y}:\ \hat{y}[\{i,j\}]=\tilde{y}} e^{s(\hat{y})} \\
&= e^{s[i,\tilde{y}[i]]+s[j,\tilde{y}[j]]+s[\{i,j\},\tilde{y}]} \\
&\quad \prod_{k\in N(i),\ k\neq j} Z_{k\rightarrow i}[\tilde{y}[i]] \\
&\quad \prod_{k\in N(j),\ k\neq i} Z_{k\rightarrow j}[\tilde{y}[j]] \\
{\color{red}P_{\{i,j\}}(\tilde{y})} &= Z_{\{i,j\}}(\tilde{y})/Z
\end{aligned}
$$

Loopy BP

Message passing is also called belief propagation (BP).

In a graph with cycles it is common to do **Loopy BP**.

This is done by initializing all message $Z_{i \to j}[\tilde{y}] = 1$ and then repeating (until convergence) the updates

$$P_{j \to i}[\tilde{y}] = \frac{1}{Z} Z_{j \to i}[\tilde{y}] \qquad Z = \sum_{\tilde{y}} Z_{j \to i}[\tilde{y}]$$

$$Z_{j \to i}[\tilde{y}] = \sum_{\tilde{y}'} e^{s[j,\tilde{y}'] + s[\{j,i\},\{\tilde{y}',\tilde{y}\}]} \left(\prod_{k \in N(j),\ k \neq i} P_{k \to j}[\tilde{y}'] \right)$$

Other Methods of Approximating Hyperedge Marginals

MCMC Sampling

Constrastive Divergence

Pseudo-Liklihood

Connectionist Temporal Classification (CTC)

Phonetic Transcription

A speech signal

$$x = x_1, \ldots, x_T$$

is labeled with a phone sequence

$$y = y_1, \ldots, y_N$$

with $N << T$ and with $y_n \in \mathcal{Y}$ for a set of phonemes $\mathcal{Y}$.

The length N of y is not determined by x and the alignment between x and y is not given.

CTC

The model defines $P_\Phi(\hat{z}|x)$ where $\hat{z}$ is latent and where y is determined by $\hat{z}$.

$$\hat{z} = \hat{z}_1, \ \ldots, \ \hat{z}_T, \quad \hat{z}_t \in \mathcal{Y} \cup \{\perp\}$$

The sequence

$$y(\hat{z}) = y_1, \ \ldots, \ y_N$$

is the result of removing all the occurrences of $\perp$ from $\hat{z}$.

$$\perp, a_1, \perp, \perp, \perp, a_2, \perp, \perp, a_3, \perp \Rightarrow a_1, a_2, a_3$$

The CTC Model

$$h_1, \ldots, h_T = \mathrm{RNN}_\Phi(x_1, \ldots, x_T)$$

$$P_\Phi(\hat{z}_t | x_1, \ldots, x_T) = \operatorname*{softmax}_{\hat{z}_t} e(\hat{z}_t)^\top h_t$$

Where $e(\hat{z}_t)$ is a vector embedding of the phoneme $\hat{z}_t$. The embedding is a parameter of the model.

Note that $\hat{z}_1, \ldots \hat{z}_T$ are all independent given x.

The Expectation Gradient (EG) Algorithm

CTC is a special case of a latent variable graphical model.

In cases where $P_\Phi(y|x)$ can be computed from dynamic programming we can backpropagate through the dynamic programming algorithm to get the gradient of cross-entropy loss.

This general technique is the **expectation gradient** (EG) algorithm.

We will first show that for the CTC model we can compute $P_\Phi(y|x)$ by dynamic programming.

We will then note that it is not necessary to backpropagate through the dynamic programming algorithm.

Dynamic Programming (Forward-Backward)

$$
\begin{aligned}
x &= x_1, \ \ldots, \ x_T \\
\hat{z} &= \hat{z}_1, \ \ldots, \ \hat{z}_T, \quad \hat{z}_t \in \mathcal{Y} \cup \{\perp\} \\
y &= y_1, \ \ldots, \ y_N, \quad y_n \in \mathcal{Y}, \quad N << T \\
y(\hat{z}) &= (\hat{z}_1, \ \ldots, \ \hat{z}_T) - \perp
\end{aligned}
$$

Forward-Backward

$$
\begin{aligned}
\vec{y}_t &= (\hat{z}_1, \ \ldots, \ \hat{z}_t) - \perp \\
F[n,t] &= P(\vec{y}_t = y_1, \ \ldots, \ y_n) \\
B[n,t] &= P(y_{n+1}, \ \ldots, y_N | \vec{y}_t = y_1, \ \ldots, \ y_n) \\
P(y) &= F[N,T] = B[0,0]
\end{aligned}
$$

Dynamic Programming (Forward-Backward)

$$
\begin{aligned}
\vec{y}_t &= (\hat{z}_1,\ \ldots,\ \hat{z}_t) - \perp \\
F[n,t] &= P(\vec{y}_t = y_1,\ \ldots,\ y_n) \\
B[n,t] &= P(y_{n+1},\ \ldots, y_N | \vec{y}_t = y_1,\ \ldots,\ y_n)
\end{aligned}
$$

$$
\begin{aligned}
F[0,0] &= 1 \\
F[n,0] &= 0 \quad \text{for } n > 0 \\
F[n+1,t+1] &= P(\hat{z}_{t+1} = \perp)F[n+1,t] + P(\hat{z}_{t+1} = y_{n+1})F[n,t]
\end{aligned}
$$

$$
\begin{aligned}
B[N,T] &= 1 \\
B[n,T] &= 0, \quad \text{for } n < N \\
B[n-1,t-1] &= P(\hat{z}_t = \perp)B[n-1,t] + P(\hat{z}_t = y_n)B[n,t]
\end{aligned}
$$

The Big Picture I

Conditional vs. Unconditional

$$\Phi^* = \operatorname*{argmin}_{\Phi} E_{(x,y)\sim\mathrm{Pop}} \; -\ln\; P(y|x)$$

$$\Phi^* = \operatorname*{argmin}_{\Phi} E_{y\sim\mathrm{Pop}} \; -\ln\; P(y)$$

This is a non-distinction: the issues in the to the conditional case are exactly the same as in the unconditional case.

The Big Picture II

The binary case: $y \in \{-1, 1\}$ (cancer screening).

The multiclass case: $y \in \mathcal{Y}$ where iteration over $\hat{y} \in \mathcal{Y}$ is feasible (MNIST, CFAR, ImageNet).

The structured case: $y \in \mathcal{Y}$ where $\mathcal{Y}$ is discrete but iteration over $\hat{y} \in \mathcal{Y}$ is infeasible (language modeling, speech recognition).

Graphical models (such as CTC) rely on assumptions about the structure of $P_\Phi(y)$.

END