TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2019

Deep Graphical Models

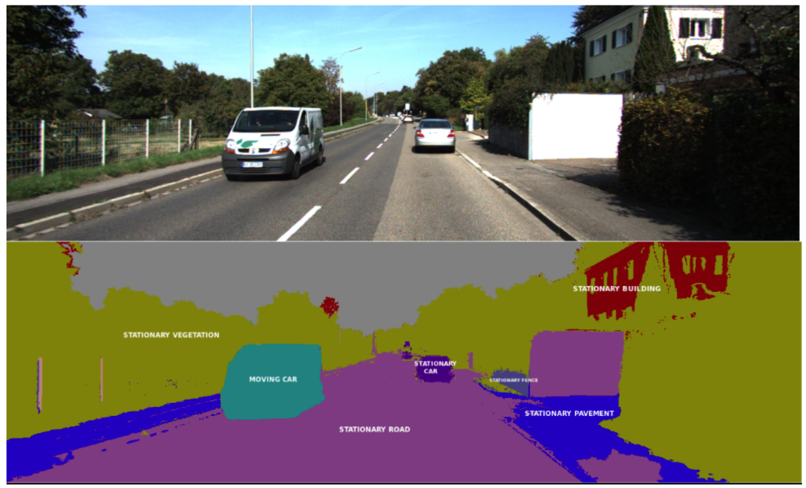
Distributions on Exponentially Large Sets

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

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

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

Semantic Segmentation



SLIC superpixels, Achanta et al.

We want to assign each pixel to one of C semantic classes.

For example “person”, “car”, “building”, “sky” or “other”.

Constructing a Graph

We construct a graph whose nodes are the pixels and where there is an edges between each pixel and its four nearest neighboring pixels.

$$\begin{array}{ccccc} & & j(i,\mathrm{up}) & & \\ & & | & & \\ j(i,\mathrm{right}) & - & i & - & j(i,\mathrm{left}) \\ & & | & & \\ & & j(i,\mathrm{down}) & & \end{array}$$

Labeling the Nodes of the Graph

$\hat{y}$ assigns a semantic class $\hat{y}[i]$ to each node (pixel) i.

We assign a score to $\hat{y}$ by assigning a score to each node and each edge of the graph.

$$s(\hat{y}) = \underbrace{\sum_{i \in \mathrm{Nodes}} s_n[i, \hat{y}[i]]}_{\text{Node Scores}} + \underbrace{\sum_{i \in \mathrm{Nodes},\ d \in \{L,R,U,D\}} s_e[i, d, \hat{y}[i],\ \hat{y}[j(i,d)]]}_{\text{Edge Scores}}$$

Computing the Node and Edge Scores

We assume a CNN computing node and edge score tensors

$$
\begin{array}{rr}
& \text{intput image} \\
& \vdots \\
& \text{CNN} \\
& \vdots \\
\text{for } i, c & s_n[i, c] = \ldots \\
\text{for } i, d, c, c' & s_e[i, d, c, c'] = \ldots
\end{array}
$$

The tensor $s_n[i, c]$ holds PC scores.

The tensor $s_e[i, j, c, c']$ holds $4PC^2$ scores.

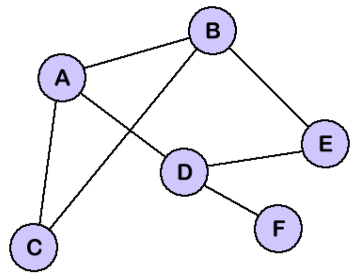
Computationally Infeasible Exponential Softmax

$$\text{intput image}$$
$$\vdots$$
$$\text{CNN}$$
$$\vdots$$

$$
\begin{aligned}
\text{for } i, c \qquad & s_n[i,c] = \ldots \\
\text{for } i, d, c, c' \qquad & s_e[i,d,c,c'] = \ldots
\end{aligned}
$$

$$
\begin{aligned}
\text{for } \hat{y} \quad s(\hat{y}) &= \textstyle\sum_i \; s_n[i, \hat{y}[i]] + \sum_{i,d} \; s_e[i, d, \hat{y}[i], \; \hat{y}[j[i,d)]] \\
\text{for } \hat{y} \quad P(\hat{y}) &= \text{expsoftmax}_{\hat{y}} \; s(\hat{y}) \quad \text{all possible } \hat{y} \\
\mathcal{L} &= -\ln P(y) \quad \text{gold label } y
\end{aligned}
$$

Exponential Softmax is Typically Intractable



$\hat{y}$ assigns a label $\hat{y}[i]$ to each node i.

$s(\hat{y})$ is defined by a sum over node and edge tensor scores.

$P(\hat{y})$ is defined by an exponential softmax over $s(\hat{y})$.

Computing Z in general is #P hard.

Back-Propagation Through Intractable Softmax

$$\begin{array}{lrl} & & \vdots \\ \text{for } i, c & s_n[i, c] & = \ldots \\ \text{for } i, d, c, c' & s_e[i, d, c, c'] & = \ldots \end{array}$$

$$\begin{array}{lrl} \text{for } \hat{y} & s(\hat{y}) & = \sum_i \; s_n[i, \hat{y}[i]] + \sum_{i,d} \; s_e[i, d, \hat{y}[i], \; \hat{y}[j[i, d)]] \\ \text{for } \hat{y} & P(\hat{y}) & = \text{expsoftmax}_{\hat{y}} \; s(\hat{y}) \;\; \text{all possible } \hat{y} \\ & \mathcal{L} & = -\ln P(y) \quad \text{gold label } y \end{array}$$

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

Although exact calculation is intractable, in practice the gradients can be approximated.

Model Marginals Theorem

Theorem:

$$s_n.\mathrm{grad}[i,c] = P_{\hat{y}\sim P_s}(\ \hat{y}[i]=c\ ) \\ -\mathbb{1}[\ y[i]=c\ ]$$

$$s_e.\mathrm{grad}[i,d,c,c'] = P_{\hat{y}\sim P_s}(\ \hat{y}[i]=c \wedge \hat{y}[j(i,d)]=c'\ ) \\ -\mathbb{1}[\ y[i]=c \wedge y[j(i,d)]=c'\ ]$$

To approximately back-propagate log loss of an intractable graphical model it suffices to approximate the model marginals in red above.

Proof of Model Marginals Theorem

We consider the case of node marginals.

$$
\begin{aligned}
s_n.\mathrm{grad}[i,c] &= \partial(\ln Z - s(y)) \;/\; \partial s_n[i,c] \\
&= \left(\frac{1}{Z}\sum_{\hat{y}} e^{s(\hat{y})}\,(\partial s(\hat{y})/\partial s_n[i,c])\right) - (\partial s(y)/\partial s_b[i,c]) \\
&= \left(\sum_{\hat{y}} P_s(\hat{y})\,(\partial s(\hat{y})/\partial s_n[i,c])\right) - (\partial s(y)/\partial s_n[i,c]) \\
&= E_{\hat{y}\sim P_s}\mathbb{1}[\hat{y}[i]=c] - \mathbb{1}[y[i]=c] \\
&= P_{\hat{y}\sim P_s}(\hat{y}[i]=c) - \mathbb{1}[y[i]=c]
\end{aligned}
$$

Methods of Approximating Model Marginals

MCMC Sampling

Loopy Belief Propagation

Pseudo-Liklihood

Constrastive Divergence

MCMC Sampling

The model marginals, such as the node marginals $P_s(\hat{y}[i] = c)$, can be estimated by sampling $\hat{y}$ from $P_s(\hat{y})$.

We will design a Markov process whose states are segmentations $\hat{y}$ and whose stationary distribution is P_s.

We will run the process past its mixing time to get a sample $\hat{y}$ from P_s.

A Neighbor Relation on States

We will say that segmentations (states) $\hat{y}$ and $\hat{y}'$ are neighbors if they differ in exactly one pixel.

Note that the number of neighbors of $\hat{y}$ is $P(C-1)$.

We will write $N(\hat{y})$ for the set of neighbors of $\hat{y}$.

For the correctness of the Metropolis algorithm we need that all states have the same number of neighbors and $\hat{y}' \in N(\hat{y})$ if and only if $\hat{y} \in N(\hat{y}')$.

The Metropolis Markov Chain

We need to define state transition probabilities.

In the Metropolis algorithm we do the following.

Pick an initial state $\hat{y}$ and then repeat:

1. Pick a neighbor $\hat{y}' \in N(\hat{y})$ uniformly at random.

2. If $P_s(\hat{y}') \geq P_s(\hat{y})$ then do $\hat{y} = \hat{y}'$

3. If $P_s(\hat{y}') < P_s(\hat{y})$ then do $\hat{y} = \hat{y}'$ with probability $\frac{P_s(\hat{y}')}{P_s(\hat{y})}$

The Metropolis Markov Chain

Pick an initial state $\hat{y}$ and then repeat:

1. Pick a neighbor $\hat{y}' \in N(\hat{y})$ uniformly at random.
2. If $P_s(\hat{y}') \geq P_s(\hat{y})$ then do $\hat{y} = \hat{y}'$
3. If $P_s(\hat{y}') < P_s(\hat{y})$ then do $\hat{y} = \hat{y}'$ with probability $\frac{P_s(\hat{y}')}{P_s(\hat{y})}$

Note that we can determine which probability is larger just by comparing scores — we do not need to know Z.

Also note that the ratio $P_s(\hat{y}')/P_s(\hat{y})$ can be computed from the scores without knowledge of Z.

The Metropolis Markov Chain

We need to show that P_s is a stationary distribution of this process.

We must show that if we select $\hat{y}_t$ from P_s and then select $\hat{y}_{t+1}$ using the transition probabilities for the process from $\hat{y}_t$ then the distribution on $\hat{y}_{t+1}$ is also P_s.

The Metropolis Markov Chain

$$
\begin{aligned}
P'(\hat{y}) &= \sum_{\hat{y}'} P_s(\hat{y}')P_{\text{Trans}}(\hat{y} \mid \hat{y}') \\
&= P_s(\hat{y})P_{\text{Trans}}(\hat{y} \mid \hat{y}) + \sum_{\hat{y}' \in N(\hat{y})} P_s(\hat{y}')P_{\text{Trans}}(\hat{y} \mid \hat{y}') \\
&= P_s(\hat{y})\left(1 - \sum_{\hat{y}' \in N(\hat{y})} P_{\text{Trans}}(\hat{y}' \mid \hat{y})\right) + \sum_{\hat{y}' \in N(\hat{y})} P_s(\hat{y}')P_{\text{Trans}}(\hat{y} \mid \hat{y}') \\
&= P_s(\hat{y}) + \sum_{\hat{y}' \in N(\hat{y})} P_s(\hat{y}')P_{\text{Trans}}(\hat{y} \mid \hat{y}') - P_s(\hat{y})P_{\text{Trans}}(\hat{y}' \mid \hat{y})
\end{aligned}
$$

Detailed Balance

$$P'(\hat{y}) \;=\; P_s(\hat{y}) + \sum_{\hat{y}' \in N(\hat{y})} P_s(\hat{y}')P_{\text{Trans}}(\hat{y} \mid \hat{y}') - P_s(\hat{y})P_{\text{Trans}}(\hat{y}' \mid \hat{y})$$

$P_s(\hat{y}')P_{\text{Trans}}(\hat{y} \mid \hat{y}')$ is an amount of probability mass that moves from $\hat{y}'$ to $\hat{y}$.

$P_s(\hat{y})P_{\text{Trans}}(\hat{y}' \mid \hat{y})$ is an amount of probability mass that moves from $\hat{y}$ to $\hat{y}'$.

If these two flows are equal we call it detailed balance.

Detailed balance implies $P'(\hat{y}) = P_s(\hat{y})$ and hence P_s is the stationary distribution.

Detailed Balance

For $P_s(\hat{y}) \geq P_s(\hat{y}')$ we have

$$\begin{aligned}
P_s(\hat{y}) P_{\mathrm{Trans}}(\hat{y}' \mid \hat{y}) &= P_s(\hat{y}) \frac{1}{|N(\hat{y})|} \frac{P_s(\hat{y}')}{P_s(\hat{y})} \\
&= \frac{P_s(\hat{y}')}{|N(\hat{y})|} \\
&= P_s(\hat{y}') P_{\mathrm{Trans}}(\hat{y} \mid \hat{y}')
\end{aligned}$$

Gibbs Sampling

The Metropolis algorithm wastes time by rejecting proposed moves.

Gibbs sampling avoids this move rejection.

In Gibbs sampling we select a node i at random and change that node by drawing a new node value conditioned on the current values of the other nodes.

Gibbs Sampling

Markov Blanket Property:

$$P_s(\hat{y}[i] \mid \hat{y} \backslash i) = P_s(\hat{y}[i] \mid \hat{y}[N(i)])$$

Gibbs Sampling, Repeat:

- Select i at random
- draw $\tilde{y}$ from $P_s(\hat{y}[i] \mid \hat{y} \backslash i)$
- $\hat{y}[i] = \tilde{y}$

Gibbs Sampling Theorem

$P_s(\hat{y})$ is a stationary distribution of Gibbs Sampling.

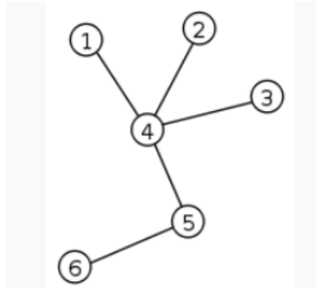
- Select i at random
- draw $\tilde{y}$ from $P_s(\hat{y}[i] \mid \hat{y} \backslash i)$
- $\hat{y}[i] = \tilde{y}$

The distribution before the update equals the distribution after the update.

Loopy Belief Propagation (Loopy BP)

We design an algorithm that is correct for tree graphs and use it on non-tree (loopy) graphs.

Belief Propagation on Trees



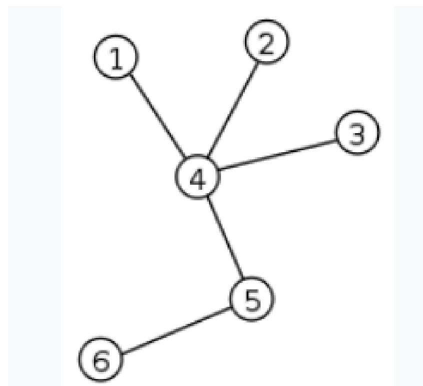
Belief Propagation is a message passing procedure (actually dynamic programming).

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

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

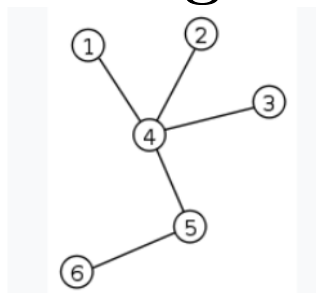
The reverse direction message $Z_{i \rightarrow j}$ is defined similarly.

Computing the Messages



$$Z_{j \to i}[c] = \sum_{c'} e^{s_n[j,c'] + s_e[j,i,c',c]} \left(\prod_{k \in N(j),\, k \neq i} Z_{k \to j}[c'] \right)$$

Computing Node Marginals from Messages



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

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

$$P_i(c) = Z_i(c)/Z, \qquad Z = \sum_c Z_i(c) = \sum_{\hat{y}} e^{s(\hat{y})}$$

Computing Edge Marginals from Messages

$$
\begin{aligned}
Z_{i,j}(c, c') &\doteq \sum_{\hat{y}:\ \hat{y}[i]=c,\ \hat{y}[j]=c'} e^{s(\hat{y})} \\
&= e^{s_n[i,c]+s_n[j,c']+s_e[i,j,c,c']} \\
&\quad \prod_{k \in N(i),\ k \neq j} Z_{k \to i}[c] \\
&\quad \prod_{k \in N(j),\ k \neq i} Z_{k \to j}[c'] \\
P_{i,j}(c, c') &= Z_{i,j}(c, c')/Z
\end{aligned}
$$

Loopy BP

In a Loopy Graph we can initializing all message $Z_{i\to j}[c] = 1$ and then repeating (until convergence) the updates

$$\tilde{Z}_{j\to i}[c] = \frac{1}{Z_{j\to i}} Z_{j\to i}[c] \qquad Z_{j\to i} = \sum_c Z_{j\to i}[c]$$

$$Z_{j\to i}[c] = \sum_{c'} e^{s_n[j,c'] + s_e[j,i,c',c]} \left(\prod_{k\in N(j),\ k\neq i} \tilde{Z}_{k\to j}[c'] \right)$$

Pseudolikelihood

In pseudolikelihood we replace the objective $-\ln P_s(\hat{y})$ with the objective $-\ln \tilde{Q}_s(\hat{y})$ where

$$\tilde{Q}_s(y) \doteq \prod_i P_s(y[i] \mid y \backslash i)$$

$$\mathcal{L}(s) \doteq -\ln \tilde{Q}_s(y)$$

$$s.\mathrm{grad}[e, \tilde{y}] = \sum_i \frac{-\partial \ln P_s(y[i] \mid y \backslash i)}{\partial s[e, \tilde{y}]} \qquad \text{immediate gradient!}$$

Pseudolikelihood Theorem

$$\underset{Q}{\operatorname{argmin}} \; E_{y \sim \mathrm{Pop}} \; -\ln \tilde{Q}(y) = \mathrm{Pop}$$

Proof I

$$E_{y \sim \mathrm{Pop}} \; \ln \widetilde{\mathrm{Pop}}(y) = E_{y \sim \mathrm{Pop}} \; \ln \mathrm{Pop}(y)$$

Proof:

$$\begin{aligned}
\mathrm{Pop}(y) &= \prod_i P(y[i] \mid y[< \; i]) \\
\ln \mathrm{Pop}(y) &= \sum_i \; \ln P(y[i] \mid y[< \; i]) \\
E_{y \sim \mathrm{Pop}} \; \ln \mathrm{Pop}(y) &= \sum_i \; E_{y \sim \mathrm{Pop}} \; \ln P(y[i] \mid y[< \; i]) \\
&= \sum_i \; E_{y \sim \mathrm{Pop}} \; \ln P(y[i] \mid y \backslash i) \\
&= E_{y \sim \mathrm{Pop}} \, \widetilde{\mathrm{Pop}}(y)
\end{aligned}$$

Proof II

$$\min_Q \; E_{y\sim\mathrm{Pop}} \; -\ln \tilde{Q}(y) \;\; \leq \;\; E_{y\sim\mathrm{Pop}} \; -\ln \widetilde{\mathrm{Pop}}(y)$$

If we can show

$$\min_Q \; E_{y\sim\mathrm{Pop}} \; -\ln \tilde{Q}(y) \;\; \geq \;\; E_{y\sim\mathrm{Pop}} \; -\ln \widetilde{\mathrm{Pop}}(y)$$

Then the minimizer (the argmin) is Pop as desired.

Proof III

We will prove the case of two nodes.

$$
\begin{aligned}
& \min_{Q} \; E_{y\sim \mathrm{Pop}} - \ln Q(y[1]|y[2]) \; Q(y[2]|y[1]) \\
\geq \; & \min_{P_1,P_2} E_{y\sim \mathrm{Pop}} - \ln P_1(y[1]|y[2]) \; P_2(y[2]|y[1]) \\
= \; & \min_{P_1} E_{y\sim \mathrm{Pop}} - \ln P_1(y[1]|y[2]) + \min_{P_2} E_{y\sim \mathrm{Pop}} - \ln P_2(y[2]|y[1]) \\
= \; & E_{y\sim \mathrm{Pop}} - \ln \mathrm{Pop}(y[1]|y[2]) + E_{y\sim \mathrm{Pop}} - \ln \mathrm{Pop}(y[2]|y[1]) \\
= \; & E_{y\sim \mathrm{Pop}} - \ln \widetilde{\mathrm{Pop}}(y|x)
\end{aligned}
$$

Contrastive Divergence

Algorithm (CDk): Run k steps of MCMC for $P_s(\hat{y})$ **starting from** y to get $\hat{y}$.

Then set

$$s.\mathrm{grad}[e, \tilde{y}] = \mathbb{1}[\hat{y}[e] = \tilde{y}] - \mathbb{1}[y[e] = \tilde{y}]$$

CD Theorem: If $P_s(\hat{y}) = \mathrm{Pop}$ then

$$E_{y \sim \mathrm{Pop}}\ \mathbb{1}[\hat{y}[e] = \tilde{y}] - \mathbb{1}[y[e] = \tilde{y}] = 0$$

Here we can take $k = 1$ **— no mixing time required**.

Summary

We are often interested in probability distributions on structured objects such as sentence or images.

Graphical models define softmax distributions on structured values.

It is infeasible to enumerate all sentences or all images.

However, some graphical models sometimes yield friendly distributions and methods exist for training unfriendly graphical models.

END