TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2019

Deep Graphical Models

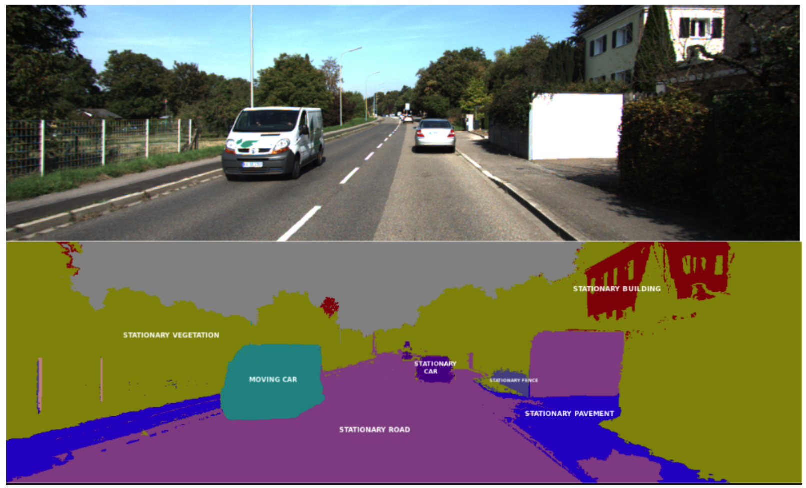
Distributions on Exponentially Large Sets

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

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

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

Semantic Segmentation



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.

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

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.

$$s(\hat{y}) = \underbrace{\sum_{i \in \text{Nodes}} s_n[i, \hat{y}[i]]}_{\text{Node Scores}} + \underbrace{\sum_{\langle i,j \rangle \in \text{Edges}} s_e[\underbrace{\langle i,j \rangle}_{\text{edge}}, \underbrace{\hat{y}[i], \hat{y}[j]}_{\text{pair}}]}_{\text{Edge Scores}}$$

Computing the Node and Edge Tensors

For input x we use a network to compute the score tensors.

$$s_n[I, C] = f^n_\Phi(x)$$

$$s_e[E, C, C] = f^e_\Phi(x)$$

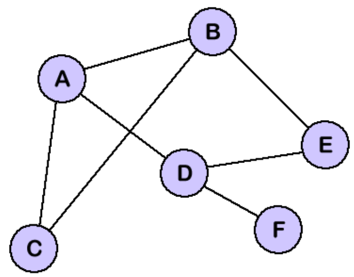
Exponential Softmax

for $\hat{y}$ $\quad s(\hat{y}) = \sum_i \; s_n[i, \hat{y}[i]] + \sum_{\langle i,\, j \rangle \in \text{Edges}} \; s_e[\langle i, j \rangle, \hat{y}[i], \; \hat{y}[j]]$

for $\hat{y}$ $\; P_s(\hat{y}) = \text{softmax}_{\hat{y}} \; s(\hat{y})$ all possible $\hat{y}$

$\mathcal{L} = -\ln P_s(y)$ gold label y

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.

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

Computing Z in general is #P hard (there is an easy direct reduction from SAT).

Compactly Representing Scores on Exponentially Many Labels

The tensor $s_n[I, C]$ holds IC scores.

The tensor $s_e[E, C, C]$ holds EC^2 scores where e ranges over edges $\langle i, j \rangle \in \text{Edges}$.

Back-Propagation Through Exponential Softmax

$$s_n[I, C] = f^n_\Phi(x)$$
$$s_e[E, C, C] = f^e_\Phi(x)$$

back-prop

$$s(\hat{y}) = \sum_i s_n[i, \hat{y}[i]] + \sum_{\langle i,j\rangle \in \text{Edges}} s_e[\langle i, j\rangle, \hat{y}[i],\ \hat{y}[j]]$$

$$P_s(\hat{y}) = \operatorname*{softmax}_{\hat{y}}\ s(\hat{y}) \quad \text{all possible } \hat{y}$$

$$\mathcal{L} = -\ln P_s(y) \quad \text{gold label } y$$

We want the gradients $s_n.\text{grad}[I, C]$ and $s_e.\text{grad}[E, C, C]$.

$y[i]$ — gold label

Model Marginals Theorem

$\hat{y}$: assign class

Theorem:

marginal dist on that node

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

$-1\,[y[i]=c]$

$$s_e.\text{grad}[\langle i,j\rangle,c,c'] = P_{\hat{y}\sim P_s}(\ \hat{y}[i]=c \wedge \hat{y}[j]=c'\ )$$
$$-\mathbf{1}[\ y[i]=c \wedge y[j]=c'\ ]$$

We need to compute (or approximate) the model marginals.

$y[i]=c$
the gold label

Proof of Model Marginals Theorem

We consider the case of node marginals, The case of edge marginals is similar.

$$
\begin{aligned}
s_n.\mathrm{grad}[i,c] &= \partial \mathcal{L}(\Phi, x, y) \;/\; \partial s_n[i,c] \\
&= \partial\left(-\ln \frac{1}{Z}\exp(s(y))\right) \;/\; \partial s_n[i,c] \\
&= \partial(\ln Z - s(y)) \;/\; \partial s_n[i,c] \\
&= \left(\frac{1}{Z}\sum_{\hat{y}} \underbrace{e^{s(\hat{y})}\,(\partial s(\hat{y})/\partial s_n[i,c])}_{\text{der of } Z = \text{sum of all labeling}}\right) - (\partial s(y)/\partial s_b[i,c])
\end{aligned}
$$

Proof of Model Marginals Theorem

"probability"

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

$$= \left(\sum_{\hat{y}} P_s(\hat{y})\,(\partial s(\hat{y})/\partial s_n[i,c])\right) - (\partial s(y)/\partial s_n[i,c])$$

$$s(\hat{y}) = \sum_i s_n[i,\hat{y}[i]]$$

→ sum of score for a node

expand

$$\frac{\partial s(\hat{y})}{\partial s_n[i,c]} = \mathbf{1}[\hat{y}[i] = c]$$

indicator
0 tenser X used
1, tenser entry is being used

Proof of Model Marginals Theorem

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

$$\left(\sum_{\hat{y}} P_s(\hat{y})\left(\partial s(\hat{y})/\partial s_n[i,c]\right)\right) - \left(\partial s(y)/\partial s_n[i,c]\right)$$

$$= E_{\hat{y}\sim P_s}\mathbf{1}[\hat{y}[i]=c] - \mathbf{1}[y[i]=c]$$

$$= P_{\hat{y}\sim P_s}(\hat{y}[i]=c) - \mathbf{1}[y[i]=c]$$

Model Marginals Theorem

Theorem:

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

$$s_e.\text{grad}[\langle i,j\rangle, c, c'] = P_{\hat{y}\sim P_s}(\ \hat{y}[i] = c \wedge \hat{y}[j] = c'\ ) \\ -\mathbf{1}[\ y[i] = c \wedge y[j] = c'\ ]$$

And now we need to compute P, marginal distributions.

Methods of Approximating Model Marginals

Monte Carlo Markov Chain (MCMC) Sampling

Pseudolikelihood

Contrastive Divergence

Loopy Belief Propagation (loopy BP)

MCMC Sampling

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

There are various ways to design a Markov process whose states are node labelings $\hat{y}$ and whose stationary distribution is P_s.

Softmax

Given such a process we can sample $\hat{y}$ from P_s by running the process past its mixing time.

We will consider Metropolis MCMC and the Gibbs MCMC. But there are more (like Hamiltonian MCMC).

measure marginal of edge & Nodes by MCMC

Metroplis MCMC

We assume a neighor relation on node assignments and let $N(\hat{y})$ be the set of neighbors of assignment $\hat{y}$.

For example, $N(\hat{y})$ can be taken to be the set of assignments $\hat{y}'$ that differ form $\hat{y}$ on exactly one node.

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

this has stationary distribution = softmax

Metropolis MCMC

Pick an initial state $\hat{y}_0$ and for $t \geq 0$ do

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

2. If $P_s(\hat{y}') > P_s(\hat{y}_t)$ then $\hat{y}_{t+1} = \hat{y}'$ → take it

   score up)

3. If $P_s(\hat{y}') \leq P_s(\hat{y})$ then with probability → score ↓

   $$e^{-\Delta s} = e^{-(s(\hat{y})-s(\hat{y}'))} = \frac{e^{s(\hat{y}')}}{e^{s(\hat{y})}} = \frac{P_s(\hat{y}')}{P_s(\hat{y})}$$

   do $\hat{y}_{t+1} = \hat{y}'$ and otherwise $\hat{y}_{t+1} = \hat{y}_t$

take that $e^{-\Delta s}$ to discourage going down

The Metropolis Markov Chain

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

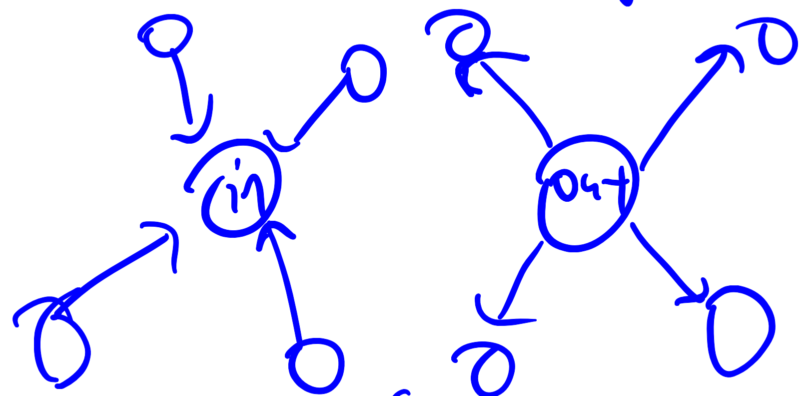
We must show that if we select $\hat{y}_t$ from P_s, and then select $\hat{y}_{t+1}$ using the transition probabilities, then the distribution on $\hat{y}_{t+1}$ is also P_s.

prove the stationary distribution $= P_s(\hat{y})$

Stationarity Condition

if draw from stationary, apply this

$\Rightarrow$ get same thing back



$$P'(\hat{y}) = \sum_{\hat{y}'} P_s(\hat{y}') P_{\text{Trans}}(\hat{y} \mid \hat{y}')$$

$$= P_s(\hat{y}) + \text{flow in} - \text{flow-out}$$

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

each pipe
balance out

Detailed Balance

Detailed balance means that for each pair of neighboring assignments $\hat{y}$, $\hat{y}'$ we have equal flows in both directions.

$$P_s(\hat{y}')P_{\text{Trans}}(\hat{y} \mid \hat{y}') = P_s(\hat{y})P_{\text{Trans}}(\hat{y}' \mid \hat{y})$$

Without loss generality assume $P_s(\hat{y}') \geq P_s(\hat{y})$.

Metropolis is defined by

$$P_{\text{Trans}}(\hat{y} \mid \hat{y}') = e^{-\Delta s}\, P_{\text{Trans}}(\hat{y}' \mid \hat{y}) = \frac{P_s(\hat{y})}{P_s(\hat{y}')}\, P_{\text{Trans}}(\hat{y}' \mid \hat{y})$$

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.

$P_s(\hat{y}[i] \mid \hat{y} \setminus i)$

We let $\hat{y}\backslash i$ be the assignment of labels given by $\hat{y}$ except that no label is assigned to node i.

We let $\hat{y}[N(i)]$ be the assignment that $\hat{y}$ gives to the nodes (pixels) that are the neighbors of node i (connected to i by an edge.)

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

This algorithm does not require knowledge of Z.

The stationary distribution is P_s.

y: global labeling

$\hat{y}[N(i)]$: given Neighbor of i

Pseudolikelihood

$\hat{y}/i$: all the other nodes

For any distribution Q on assignments of labels to nodes (segmentations), and any assignment $\hat{y}$, we define $\tilde{Q}(\hat{y})$ as follows.

$$\tilde{Q}(\hat{y}) = \prod_i Q(\hat{y}[i] \mid \hat{y}/i) = \prod_i Q(\hat{y}[i] \mid \hat{y}[N(i)])$$

We then train a graphical model with pseudolikelyhood loss.

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

Pseudolikelihood

$$\mathcal{L}_{\mathrm{PL}} = -\ln \tilde{P}_s(y)$$

We note that by the Markov blanket property for Markov random fields we have

$$\tilde{P}_s(\hat{y}) = \prod_i P_s(\hat{y}[i] \mid \hat{y}[N(i)])$$

Since the loss is directly computed we can directly back-propagate on the loss.

Pseudolikelihood Theorem

Q : any distribution

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

Sample from the population

≃ population

or equivalently

$$\min_{Q} \; E_{y\sim\text{Pop}} \; -\ln \tilde{Q}(y) = E_{y\sim\text{Pop}} \; -\ln \widetilde{\text{Pop}}(y)$$

if min , solution of this minimization

is just pop

Proof I

We have

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

So it suffices to show

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

Proof II

We will prove the case of two nodes.

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

Contrastive Divergence (CDk)

In contrastive divergence we first construct an MCMC process whose stationary distribution is P_s. This could be Metropolis or Gibbs or something else.

Algorithm CDk: Given a gold segmentation y, start the MCMC process from initial state y and run the process for k steps to get $\hat{y}$. Then take the loss to be

push up

$s(y) \uparrow$

$$\mathcal{L}_{\mathrm{CD}} = s(\hat{y}) - s(y)$$

$s(\hat{y}) \downarrow$

If $P_s = \mathrm{Pop}$ then the the distribution on $\hat{y}$ is the same as the distribution on y and the expected loss gradient is zero.

Gibbs CD1

CD1 for the Gibbs MCMC process is a particularly interesting special case.

Algorithm (Gibbs CD1): Given y, select a node i at random and draw $c \sim P(y[i] \mid y[N(i)])$. Define $y[i = c]$ to be the assignment (segmentation) which is the same as y except that node i is assigned label c. Take the loss to be

$$\mathcal{L}_{\mathrm{CD}} = s(y[i = c]) - s(y)$$

use the theorem before

MCMC takes so long to converge

Sudo likelihood: linear time to update

Gibbs CD1 Theorem

Gibbs CD1 is equivalent in expectation to pseudolikelihood.

$$\mathcal{L}_{\mathrm{PL}} = \sum_i -\ln \frac{e^{s(y)}}{\sum_c e^{s(y[i=c])}}$$

$$= \sum_i \left(\ln \left(\sum_c e^{s(y[i=c])} \right) - s(y) \right)$$

?

$$\nabla_\Phi \mathcal{L}_{\mathrm{PL}} = \sum_i \left(E_{c|i} \; \nabla_\Phi \; s(y[i=c]) - \nabla_\Phi \; s(y) \right)$$

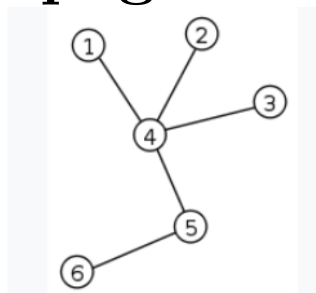
$$= N \; E_{i,c} \; \nabla_\Phi \; \mathcal{L}_{\mathrm{CD}}$$

Loopy Belief Propagation (Loopy BP)

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

$$\frac{e^{\beta z_i}}{\sum e^{\beta z_j}} \Rightarrow Z : \text{partition function}$$

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 $Z_{j \to i}[c]$ to be the partition function for the subtree attached to i through j and with $\hat{y}[i]$ restricted to c.

The function $Z_{j \to i}$ 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.

$\{4, 5\}$;

fix 5, function for the tree that connected to 5

↳ $i = 5$, $j = 4$

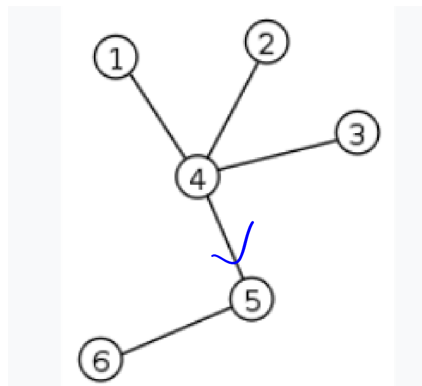
eg: message from $4 \to 5$

Dynamic Programming Computes the Messages

c: possible value of 5

$i := 5$

$j = 4$



① to ② to ③

independent

c' = all possible value of 4

$$Z_{j\to i}[c] = \sum_{c'} e^{s_n[j,c'] + s_e[j,i,c',c]} \left(\prod_{k\in N(j),\, k\neq i} \underset{4}{Z_{k\to j}[c']} \right)$$

message come into 5 for some particular value of 5

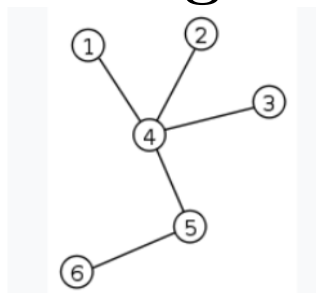
Loopy BP

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

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

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

Computing Node Marginals from Messages



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

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

$$P_i(c) = Z_i(c)/Z, \qquad Z = \sum_c Z_i(c)$$

Computing Edge Marginals from Messages

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

$$
P_{i,j}(c,c') = Z_{i,j}(c,c')/Z \quad Z = \sum_{c,c'} Z_{i,j}(c,c')
$$

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, pseudolikelihood provides a reasonable training algorithm and loopy BP can be used for both training time and test time inference.

END