TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2019

Connectionist Temporal Classification (CTC)

and Deep Graphical Models

The Fundamental Equation:
Conditional vs. Unconditional

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

$$\Phi^* = \underset{\Phi}{\mathrm{argmin}}\, E_{y\sim\mathrm{Pop}}\ -\ln\ P(y)$$

This is a non-distinction: the analysis of the conditional case is exactly the same as that of the unconditional case.

The Fundamental Equation:

Distributions on Exponentially Large Sets

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

Language modeling (unconditional) and machine translation (conditional) are distributions on exponentially large (even infinite) sets.

Friendly and Unfriendly Distributions

A model $P_\Phi(y)$ will be called friendly if we can efficiently sample from it and, for any given y, can efficiently compute $P_\Phi(y)$.

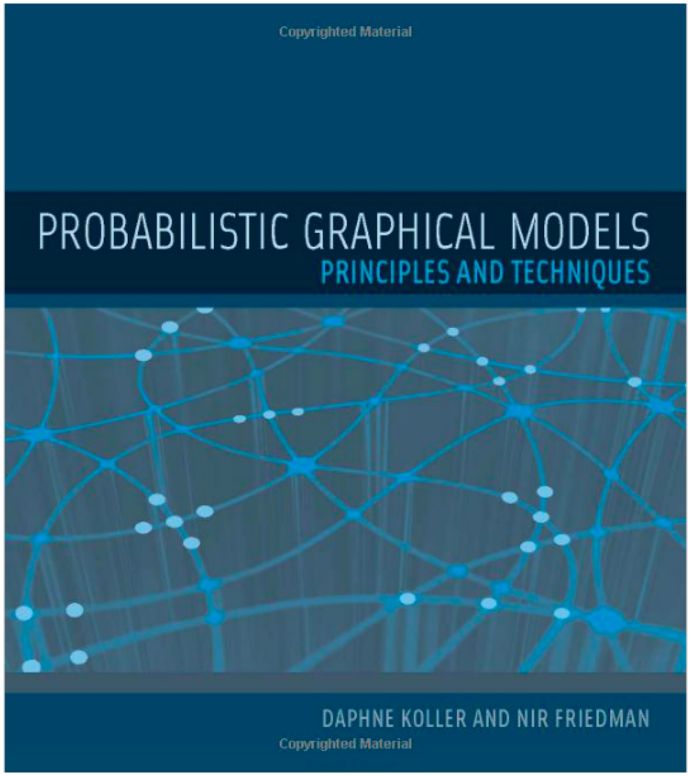
Autoregressive language models (unconditional) and autoregressive machine translation models (conditional) are friendly.

Distributions which are not friendly in this sense will be called unfriendly.

The Importance of Being Friendly

If $P_\Phi(y|x)$ can be computed (a friendly model) we can do SGD on cross-entropy loss $-\ln P_\Phi(y|x)$ by back-propagating through the computation of $P_\Phi(y|x)$.

Graphical Models



Koller and Friedman, MIT Press, 2009, 1270 pages

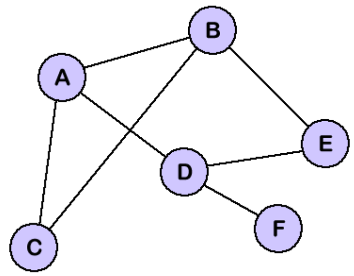
Semantic Segmentation

SLIC superpixels, Achanta et al.

We want to assign each superpixel one of k semantic classes.

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

General Markov Random Fields (MRFs)



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

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

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

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

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 $s(\hat{y}) = 3$

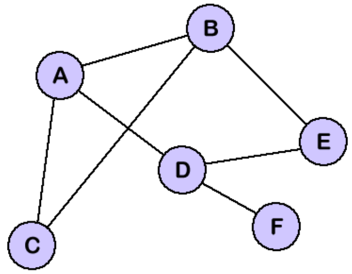
If one is foreground we have $s(\hat{y}) = -1 + 1 + 1 = 1$

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

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

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

Exponential Softmax

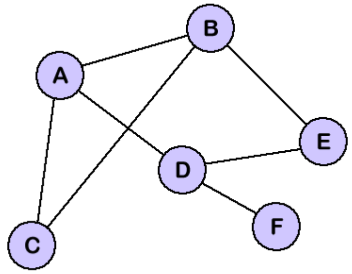


$\hat{y}$ assigns a class $\hat{y}[i]$ to each node (superpixel) i.

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

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

Exponential Softmax is Typically Unfriendly



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

$$P_\Phi(\hat{y}|x) = \underset{\hat{y}}{\text{expsoftmax}}\; s_\Phi(\hat{y}|x)$$

Computing Z in general is #P hard.

But special cases can be friendly and approximations can be made in unfriendly cases.

Latent Variables

We are often interested in models of the form

$$P_\Phi(y) = \sum_z P_\Phi(z) P_\Phi(y|z).$$

Probabilistic grammar models have this form where y is a sentence and z is a parse tree and $P(y|z)$ is deterministic.

Exponential Softmax as Intermediate Computation

$$P_\Phi(y) = \sum_z P_\Phi(z) P_\Phi(y|z).$$

$$\begin{aligned}
&\text{intput } x \\
&\vdots \\
z \;=\; &\operatorname*{expsoftmax}_z \ldots \\
&\text{input } z \\
&\vdots \\
y \;=\; &\operatorname*{expsoftmax}_y \ \ldots
\end{aligned}$$

A Composition of Friendlies is Typically Unfriendly

$$P_\Phi(y) = \sum_z P_\Phi(z) P_\Phi(y|z).$$

It is often the case that $P_\Phi(z)$ is friendly, and $P_\Phi(y|z)$ is friendly, but $P_\Phi(y)$ is not friendly (the sum over z is intractible).

For example z might be uniformly distributed over assignments of truth values to Boolean variables (which is friendly) and y might be the value of a fixed Boolean formula Φ (which is friendly given z). In this case determining if $P_\Phi(y) > 0$ is the SAT problem which is NP hard.

Connectionist Temporal Classification (CTC)

A Successful Deep Latent Variable Model

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 $y_n \in \mathcal{P}$ for a set of phonemes $\mathcal{P}$.

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

CTC: A Friendly Compositions of Friendlies

$$P_\Phi(y|x) = \sum_z P_\Phi(z|x) P_\Phi(y|z).$$

Input Signal: $x = x_1, \ldots, x_T$

Latent Label: $z = z_1, \ldots, z_T, \quad z_t \in \mathcal{P} \cup \{\perp\}$

Output: $y(z) = y_1, \ldots, y_N$

$y(z)$ is the result of removing all the occurrences of $\perp$ from z:

$$z \quad \Rightarrow \quad y$$

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

The CTC Model

For $z \in \mathcal{P} \cup \{\perp\}$ we have an embedding $e(z)$. The embedding is a parameter of the model.

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

$$P_\Phi(z_t|x_1,\ldots,x_T) = \operatorname{softmax}_z\ e(z)^\top h_t$$

$z_1, \ldots\ z_T$ are all independent given x (very friendly).

$P_\Phi(y|z)$ is deterministic (very friendly).

But it is not obvious whether $P_\Phi(y|x)$ is friendly.

Dynamic Programming

$$
\begin{aligned}
x &= x_1, \ \ldots, \ x_T \\
z &= z_1, \ \ldots, \ z_T, \quad z_t \in \mathcal{P} \cup \{\perp\} \\
y &= y_1, \ \ldots, \ y_N, \quad y_n \in \mathcal{P}, \quad N << T \\
y(z) &= (z_1, \ \ldots, \ z_T) - \perp
\end{aligned}
$$

$$
\begin{aligned}
\vec{y}_t &= (z_1, \ \ldots, \ z_t) - \perp \\
F[n, t] &= P(\vec{y}_t = y_1, \ \ldots, \ y_n) \\
P(y) &= F[N, T]
\end{aligned}
$$

Dynamic Programming

$$\begin{aligned} \vec{y}_t &= (z_1, \ \ldots, \ z_t) - \bot \\ F[n,t] &= P(\vec{y}_t = y_1, \ \ldots, \ y_n) \end{aligned}$$

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

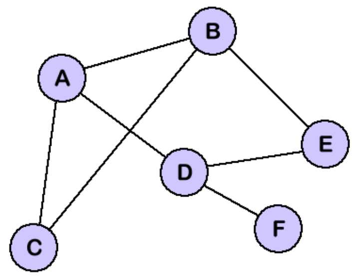
Semantic Segmentation

SLIC superpixels, Achanta et al.

We want to assign each superpixel one of k semantic classes.

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

Unfriendly Exponential Softmax



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

$$P_\Phi(\hat{y}|x) \;=\; \underset{\hat{y}}{\text{expsoftmax}}\;\; s_\Phi(\hat{y}|x)$$

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

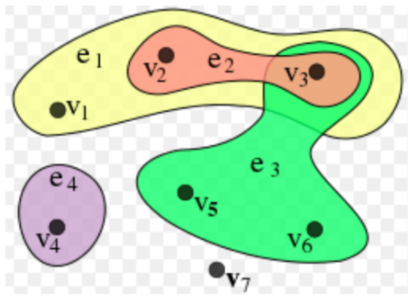
Back-Propagation Through Unfriendly Softmax

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

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

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, $s_e.\mathrm{grad}[\hat{y}[e]]$.

Back-Propagation Through An Exponential Softmax

We will abbreviate $s_e[\hat{y}[e]]$ as $s_e[\tilde{y}]$.

$\tilde{y}$ has a small number of possible values.

We will similarly write $s_e.\mathrm{grad}[\tilde{y}]$.

We need to compute the tensor values $s_e.\mathrm{grad}[\tilde{y}]$

Back-Propagation Through An Exponential Softmax

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

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

Back-Propagation Through An Exponential Softmax

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

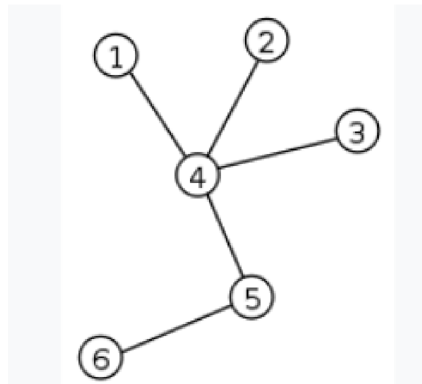
Hyperedge Marginals

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

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

To back-propagate log loss on a labeling of an unfriendly MRF it suffices to compute (or perhaps approximate) the current model's hyperedge marginals $P_e(\tilde{y})$.

Tree-Structured Models are Friendly

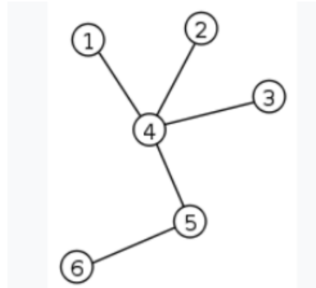


Tree structure models can always be locally renormalized to form "autoregressive" models that predict one node at a time.

Also, the hyperedge marginals can be computed exactly.

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

Belief Propagation



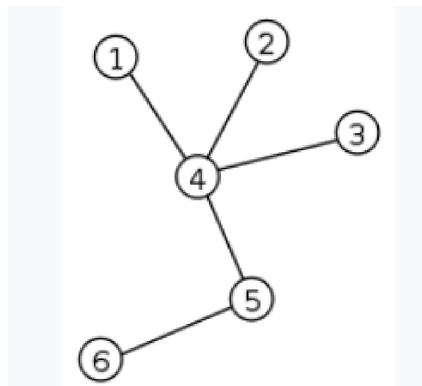
Belief Propagation is a message passing procedure (actually dynamic programming).

For each edge $\{i, j\}$ and possible value $\tilde{y}$ for node i we define $Z_{j \to i}[\tilde{y}]$ 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.

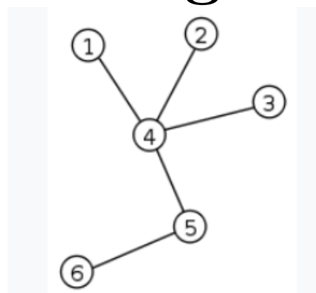
The reverse direction message $Z_{i \to j}$ is defined similarly.

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

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

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

Sampling

The quantities $P_e(\tilde{e})$ are **hyperedge marginals**.

We can estimate the hyperedge marginals by sampling $\hat{y}$ from $P_s(\hat{y})$.

Monte Carlo Markov Chain (MCMC) Sampling

Metropolis Algorithm

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

1. Pick a "neighbor" $\hat{y}'$ of $\hat{y}$ uniformly at random. The neighbor relation must be symmetric. Perhaps Hamming distance one.

2. If $s(\hat{y}') > s(\hat{y})$ update $\hat{y} = \hat{y}'$

3. If $s(\hat{y}') \leq s(\hat{y})$ then update $\hat{y} = \hat{y}'$ with probability $e^{-(s(\hat{y}) - s(\hat{y}'))}$

Markov Processes and Stationary Distributions

A Markov process is a process defined by a fixed state transition probability $P(\hat{y}'|\hat{y}) = M_{\hat{y}',\hat{y}}$.

Let P^t the probability distribution for time t.

$$P^{t+1} = MP^t$$

If every state can be reached form every state (ergodic process) then P^t converges to a unique **stationary distribution** P^∞

$$P^\infty = MP^\infty$$

Metropolis Theorem

To verify that the Metropolis process has the correct stationary distribution we simply verify that $MP = P$ where P is the desired distribution.

This can be done by checking that under the desired distribution the flow from $\hat{y}$ to $\hat{y}'$ equals the flow from $\hat{y}'$ to $\hat{y}$ (**detailed balance**).

Metropolis Theorem

For $s(\hat{y}) \geq s(\hat{y}')$

$$\begin{aligned}
\text{flow}(\hat{y}' \to \hat{y}) &= \frac{1}{Z} e^{s(\hat{y}')} \frac{1}{N} \\
\text{flow}(\hat{y} \to \hat{y}') &= \frac{1}{Z} e^{s(\hat{y})} \frac{1}{N} e^{-\Delta f} = \frac{1}{Z} e^{s(\hat{y}')} \frac{1}{N}
\end{aligned}$$

But detailed balance is not required in general (see Hamiltonian MCMC).

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

$$P_s(i = \tilde{y} \mid \hat{y}) \doteq P_s(\hat{y}[i] = \tilde{y} \mid \hat{y}[1], \ldots, \hat{y}[i-1], \hat{y}[i+1], \ldots, \hat{y}[I])$$

Markov Blanket Property:

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

Gibbs Sampling, Repeat:

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

Gibbs Sampling

Let $\hat{y}[i = \tilde{y}]$ be the assignment $\hat{y}'$ equal to $\hat{y}$ except $\hat{y}'[i] = \tilde{y}$.

$$P_s(i = \tilde{y} \mid \hat{y}) = \frac{P_s(\hat{y}[i] = \tilde{y})}{\sum_{\tilde{y}} P_s(\hat{y}[i] = \tilde{y})}$$

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

Gibbs Sampling Theorem

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

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

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

Pseudolikelihood

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

$$
\begin{aligned}
\tilde{Q}_s(\hat{y}) &\doteq \prod_i P_s(i = \hat{y}[i] \mid \hat{y}) \\
\mathrm{loss}(f) &\doteq -\log \tilde{Q}(y) \\
s.\mathrm{grad}[e, \tilde{y}] &= \sum_i -\partial \log P_s[i = \hat{y}[i] \mid \hat{y}]/\partial s[e, \tilde{y}]
\end{aligned}
$$

Pseudolikelihood Theorem

$$\operatorname*{argmin}_{Q} \; E_{y \sim \mathrm{Pop}} \; -\log \tilde{Q}(y) = \mathrm{Pop}$$

Proof I

We have

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

If we can show

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

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

Proof II

We will prove the case of two nodes.

$$\min_{Q} \; E_{y\sim\mathrm{Pop}} - \log Q(y[1]|y[2]) \; Q(y[2]|y[1])$$

$$\geq \; \min_{P_1,P_2} E_{y\sim\mathrm{Pop}} - \log P_1(y[1]|y[2]) \; P_2(y[2]|y[1])$$

$$= \; \min_{P_1} E_{y\sim\mathrm{Pop}} - \log P_1(y[1]|y[2]) + \min_{P_2} E_{y\sim\mathrm{Pop}} - \log P_2(y[2]|y[1])$$

$$= \; E_{y\sim\mathrm{Pop}} - \log \mathrm{Pop}(y[1]|y[2]) + E_{y\sim\mathrm{Pop}} - \log \mathrm{Pop}(y[2]|y[1])$$

$$= \; E_{y\sim\mathrm{Pop}} - \log \widetilde{\mathrm{Pop}}(y|x)$$

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**.

END