Structured Probabilistic Models (chapter 16, part 2)

Monte Carlo Methods (chapter 17)

Learning about Dependencies

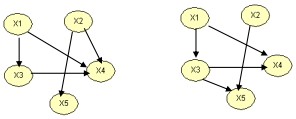
Learning Graph Structure

- Try out several graphs
- See which graph does best job of some criterion
  - Fitting training set with small model complexity
  - Fitting validation set
- Iterative search, propose new graphs similar to best graph so far (remove edge / add edge / flip edge)

Unknown structure – learn graph and parameters

➢Complete data:
optimization (search in space of graphs)

➢Incomplete data:
structural EM, mixture models

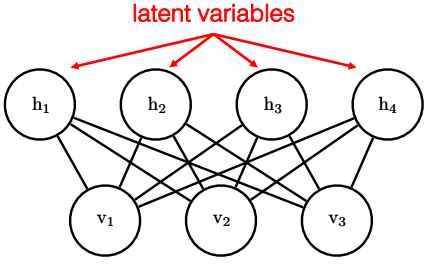


$S = \arg\max_{S} \text{Score}(S)$

1. Log-likelihood-based scores
   - MLC - Maximum Likelihood criterion
2. Penalized log-likelihood scores
   - BIC - Bayesian Information Criterion (Schwarz, 1978)
   - MDL - Minimum Description Length (Rissanen, 1978)
   - AIC - Akaike´s Information Criterion
3. Bayesian scores: Bayes, BDe, K2
4. Predictive scores
   - Cross-validation (k-Fold-CV)
   - Prequential (Preq)

Use Latent Variables

- Use one graph structure
- Many latent variables
- Dense connections of latent variables to observed variables
- Parameters learn that each latent variable interacts strongly with only a small subset of observed variables
- Trainable just with gradient descent; no discrete search over graphs

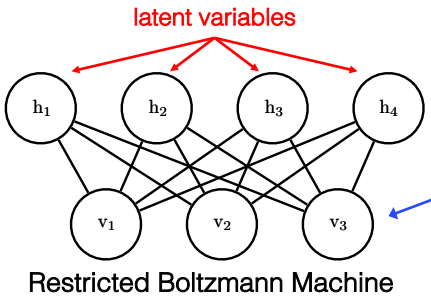


Restricted Boltzmann Machine

Inference and Approximate Inference

- Inferring marginal distribution over some nodes or conditional distribution of some nodes given other nodes is #P hard
  - NP-hardness describes decision problems
  - #P-hardness describes counting problems, e.g., how many solutions are there to a problem where finding one solution is NP-hard
- Deep Learning usually rely on approximate inference

Example: (shallow) Restricted Boltzmann Network



Restricted Boltzmann Machine

Separation (conditional independencies):
- no connections among h_j
- no connections among v_i

$P(\boldsymbol{v})$ is intractable, however both

$P(\boldsymbol{h} \mid \boldsymbol{v})$ and $P(\boldsymbol{v} \mid \boldsymbol{h})$

are *factorial* and easy to compute!

Energy-based Undirected Model

$$P(\mathbf{v} = \boldsymbol{v}, \mathbf{h} = \boldsymbol{h}) = \frac{1}{Z} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h})\right)$$

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h} - \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}$$

hidden units: not observable variables

visible units: observable variables

partition function

$$Z = \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \exp\left\{-E(\boldsymbol{v}, \boldsymbol{h})\right\}$$

$$P(\boldsymbol{h} \mid \boldsymbol{v}) = \frac{P(\boldsymbol{h}, \boldsymbol{v})}{P(\boldsymbol{v})}$$

$$= \frac{1}{P(\boldsymbol{v})} \frac{1}{Z} \exp\left\{\boldsymbol{b}^\top \boldsymbol{v} + \boldsymbol{c}^\top \boldsymbol{h} + \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}\right\}$$

$\boldsymbol{v}$ observed: constant

$\boldsymbol{v}$ observed: constant

$$= \frac{1}{Z'} \exp\left\{\boldsymbol{c}^\top \boldsymbol{h} + \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}\right\}$$

h_j separation

$$= \frac{1}{Z'} \exp\left\{\sum_{j=1}^{n_h} c_j h_j + \sum_{j=1}^{n_h} \boldsymbol{v}^\top \boldsymbol{W}_{:,j} h_j\right\}$$

$$= \frac{1}{Z'} \prod_{j=1}^{n_h} \exp\left\{c_j h_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j} h_j\right\}$$

Example: (shallow) Restricted Boltzmann Network

From unnormalized to normalized distributions

$$
\begin{aligned}
P(h_j = 1 \mid \boldsymbol{v}) &= \frac{\tilde{P}(h_j = 1 \mid \boldsymbol{v})}{\tilde{P}(h_j = 0 \mid \boldsymbol{v}) + \tilde{P}(h_j = 1 \mid \boldsymbol{v})} \\
&= \frac{\exp\left\{c_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j}\right\}}{\exp\{0\} + \exp\left\{c_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j}\right\}} \\
&= \sigma\left(c_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j}\right).
\end{aligned}
$$

Recall that:

$$\sigma(x) = \frac{\exp(x)}{\exp(x) + \exp(0)}$$

Finally, for the hidden layer:

Recall that:

$$1 - \sigma(x) = \sigma(-x)$$

$$P(\boldsymbol{h} \mid \boldsymbol{v}) = \prod_{j=1}^{n_h} \sigma\left((2\boldsymbol{h} - 1) \odot (\boldsymbol{c} + \boldsymbol{W}^\top \boldsymbol{v})\right)_j$$

…and with a similar derivation, for the visible layer:

$$P(\boldsymbol{v} \mid \boldsymbol{h}) = \prod_{i=1}^{n_v} \sigma\left((2\boldsymbol{v} - 1) \odot (\boldsymbol{b} + \boldsymbol{W}\boldsymbol{h})\right)_i$$

Example: (shallow) Restricted Boltzmann Network

Why do we care about efficient computation of

$$P(\boldsymbol{h}|\boldsymbol{v}) \text{ and } P(\boldsymbol{v}|\boldsymbol{h})?$$

Because of learning:

- Given training data $\mathbf{x}$ (to be clamped to the visible units) we need to maximize $p(\mathbf{x}; \boldsymbol{\theta})$
- We know it is easier to maximize the Log-Likelihood $\log p(\mathbf{x}; \boldsymbol{\theta})$, however its gradient with respect to $\boldsymbol{\theta}$ is

$$\nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}; \boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x}; \boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} \log Z(\boldsymbol{\theta})$$

- Computing the term

$$\nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}; \boldsymbol{\theta})$$

is not a problem for RBMs, however the term

$$\nabla_{\boldsymbol{\theta}} \log Z(\boldsymbol{\theta})$$

is more problematic!

Monte Carlo Methods

Under some conditions satisfied by RBMs (as well as most ML models), it can be shown that

$$\nabla_{\boldsymbol{\theta}} \log Z = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x})$$

which is not possibile to compute exactly, however we can use

Monte Carlo methods

to compute a (good) approximation of it!

Randomized Algorithms	**Las Vegas**	**Monte Carlo**
Type of Answer	Exact	Random amount of error
Runtime	Random (until answer found)	Chosen by user (longer runtime gives less error)

Monte Carlo Methods: basic idea

$$s = \sum_{\boldsymbol{x}} p(\boldsymbol{x}) f(\boldsymbol{x}) = E_p[f(\mathbf{x})]$$

$$s = \int p(\boldsymbol{x}) f(\boldsymbol{x}) d\boldsymbol{x} = E_p[f(\mathbf{x})]$$

Estimating sums/integrals with samples

$$\hat{s}_n = \frac{1}{n} \sum_{i=1}^{n} f(\boldsymbol{x}^{(i)})$$

$$\boldsymbol{x}^{(i)} \sim p$$

Monte Carlo Methods: basic idea

Unbiased:

- The expected value for finite n is equal to the correct value

$$\mathbb{E}[\hat{s}_n] = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[f(\boldsymbol{x}^{(i)})] = \frac{1}{n}\sum_{i=1}^{n} s = s$$

- The value for any specific n samples will have random error, but the errors for different sample sets cancel out

Low variance:

- Variance is $O(\frac{1}{n})$
- For very large n, the error converges "almost surely" to 0

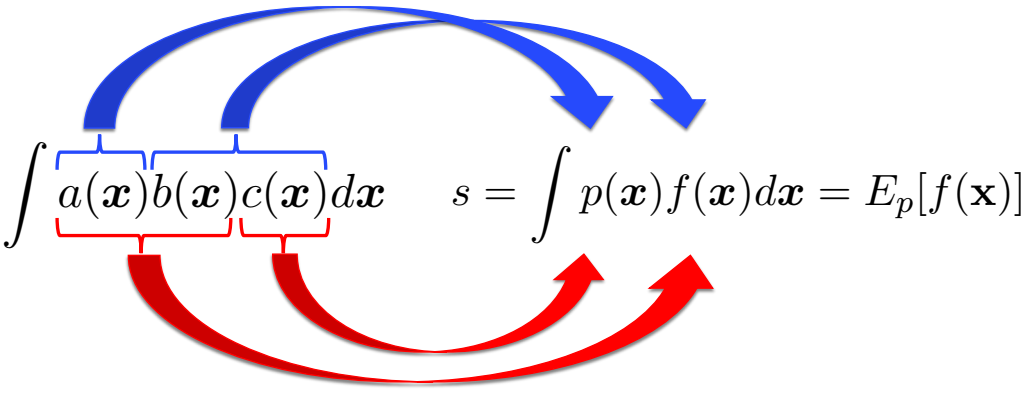
However sampling from $p(\boldsymbol{x})$ (i.e., $\boldsymbol{x}^{(i)} \sim p$) is not always possible!

Importance Sampling

Suppose we want to compute (an approximation of)

$$\int a(\boldsymbol{x})b(\boldsymbol{x})c(\boldsymbol{x})d\boldsymbol{x}$$

? Non-unique decomposition!



...and other possible combinations

Importance Sampling

$$p(\boldsymbol{x})f(\boldsymbol{x}) = q(\boldsymbol{x})\frac{p(\boldsymbol{x})f(\boldsymbol{x})}{q(\boldsymbol{x})}$$

This is our new p, meaning it is the distribution we will draw samples from

This ratio is our new f, meaning we will evaluate it at each sample

Why is this useful ?

- Maybe it is feasible to sample from q but not from p
- A good q can reduce the variance of the estimate
- Importance sampling is still unbiased for every q

Optimal q

(optimal importance sampling)

importance sampling estimator

$$\hat{s}_q = \frac{1}{n}\sum_{i=1,\mathbf{x}^{(i)}\sim q}^{n}\frac{p(\boldsymbol{x}^{(i)})f(\boldsymbol{x}^{(i)})}{q(\boldsymbol{x}^{(i)})}$$

$$\mathrm{Var}[\hat{s}_q] = \mathrm{Var}[\frac{p(\mathbf{x})f(\mathbf{x})}{q(\mathbf{x})}]/n$$

Unbiased!

$$\mathbb{E}_q[\hat{s}_q] = \mathbb{E}_q[\hat{s}_p] = s$$

We can choose q so to reduce variance!!

The minimum variance occurs when q is

$$q^*(\boldsymbol{x}) = \frac{p(\boldsymbol{x})|f(\boldsymbol{x})|}{Z}$$

- Determining the optimal q requires solving the original integral, so not useful in practice
- Useful to understand intuition behind importance sampling
- Places more mass on points where the weighted function is larger

Biased Importance Sampling

If we need to use only the unnormalized distribution

$$\hat{s}_{BIS} = \frac{\sum_{i=1}^{n} \frac{p(\boldsymbol{x}^{(i)})}{q(\boldsymbol{x}^{(i)})} f(\boldsymbol{x}^{(i)})}{\sum_{i=1}^{n} \frac{p(\boldsymbol{x}^{(i)})}{q(\boldsymbol{x}^{(i)})}}$$

$$= \frac{\sum_{i=1}^{n} \frac{p(\boldsymbol{x}^{(i)})}{\tilde{q}(\boldsymbol{x}^{(i)})} f(\boldsymbol{x}^{(i)})}{\sum_{i=1}^{n} \frac{p(\boldsymbol{x}^{(i)})}{\tilde{q}(\boldsymbol{x}^{(i)})}}$$

Asymptotically unbiased

$$= \frac{\sum_{i=1}^{n} \frac{\tilde{p}(\boldsymbol{x}^{(i)})}{\tilde{q}(\boldsymbol{x}^{(i)})} f(\boldsymbol{x}^{(i)})}{\sum_{i=1}^{n} \frac{\tilde{p}(\boldsymbol{x}^{(i)})}{\tilde{q}(\boldsymbol{x}^{(i)})}}$$

poor choice of *q* can make the efficiency much worse

Markov Chain

So far we have assumed we can sample from p or q easily

- This is true when p or q has a directed graphical model representation
  - Use ancestral sampling
  - Sample each node given its parents, moving from roots to leaves
- Sampling from undirected models is more difficult
  - Cannot get a fair sample in one pass
  - Use a Monte Carlo algorithm that incrementally updates samples, comes closer to sampling from the right distribution at each step
  - This is called a Markov Chain

Markov Chain

A Markov chain is defined by

- a random state $\boldsymbol{x}$
- a transition distribution $T(\boldsymbol{x}'|\boldsymbol{x})$ specifying the probability that a random update will go to state $\boldsymbol{x}'$ if it starts in state $\boldsymbol{x}$

Running the Markov chain means repeatedly updating the state $\boldsymbol{x}$ to a value $\boldsymbol{x}'$ sampled from $T(\boldsymbol{x}'|\boldsymbol{x})$

Single Chain (can have more chains running in parallel!)

- $q^{(t)}(x)$: distribution at iteration t from which x is drwan
- Aim: $q^{(t)}(x) \xrightarrow{t} p(x)$

Markov Chain

- positive integer i to represent a state (countably many states)
- can describe $q(x)$ as a vector $\boldsymbol{v}$ where $v_i = q(x = i)$
- probability of a single state landing in state x' is given by

$$q^{(t+1)}(x') = \sum_x q^{(t)}(x) T(x'|x)$$

- can represent the effect of the transition operator T using a *stochastic matrix* $\boldsymbol{A}$ (each of its columns represents a probability distribution)

$$A_{i,j} = T(x' = i|x = j)$$

Perron-Frobenius theorem $\Rightarrow$ the largest eigenvalue of $\boldsymbol{A}$ is real and equal to 1

- then $\boldsymbol{v}^{(t)} = \boldsymbol{A}\boldsymbol{v}^{(t-1)} \Rightarrow \boldsymbol{v}^{(t)} = \boldsymbol{A}^t\boldsymbol{v}^{(0)} \Rightarrow \boldsymbol{v}^{(t)} = \boldsymbol{V} diag(\boldsymbol{\lambda})^t \boldsymbol{V}^{-1}\boldsymbol{v}^{(0)}$
- largest eigenvalue is $1 \Rightarrow$ converges to a stationary distribution (equilibrium distribution)

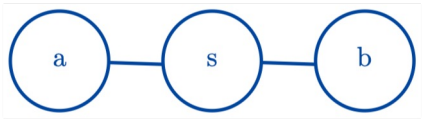
$$\boldsymbol{v}' = \boldsymbol{A}\boldsymbol{v} = \boldsymbol{v}$$

converges to p for the right choice of T

Simple Markov Chain: Gibbs sampling

- Repeatedly cycle through all variables
  - For each variable, randomly sample that variable given its Markov blanket
  - For an undirected model, the Markov blanket is just the neighbors in the graph
- *Block Gibbs* trick: conditionally independent variables may be sampled simultaneously (used in RBMs!)

Example:



- Initialize $\mathtt{a}$, $\mathtt{s}$, and $\mathtt{b}$
- For n repetitions
  - Sample $\mathtt{a}$ from $P(\mathtt{a}|\mathtt{s})$ and $\mathtt{b}$, from $P(\mathtt{b}|\mathtt{s})$
  - Sample $\mathtt{s}$ from $P(\mathtt{s}|\mathtt{a},\mathtt{b})$

Block Gibbs trick lets us sample a and b in parallel

Problems…

Generally infeasible to...

- ... know ahead of time how long *mixing* will take
- ... know how far a chain is from equilibrium
- ... know whether a chain is at equilibrium

Usually in deep learning we just run for n steps, for some n that we think will be big enough, and hope for the best

In practice

- Mixing can take an infeasibly long time
- This is especially true for
  - High-dimensional distributions
  - Distributions with strong correlations between variables
  - Distributions with multiple highly separated modes