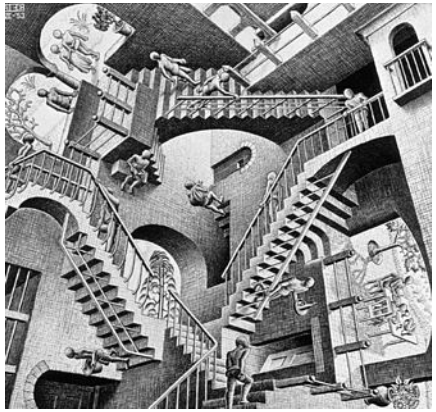
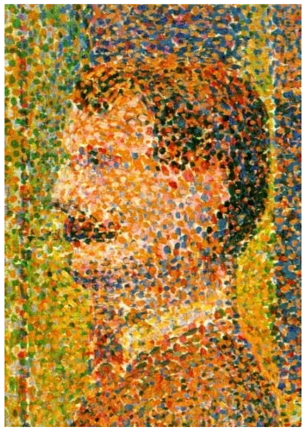
iPMR

Approximations

- Constrain Q to a family and optimize
  - Delta function (last lecture)
  - More general form. E.g. factorized distribution
- Unconstrain Q to impose only local consistency.
  - Loopy belief propagation
- Sample to obtain Q that is a mixture of points.
- Combine these methods.





Sampling

- Constraining to a delta function: bad.
- Constraining to a factorised distribution: inflexible.
- What about a *mixture* of delta functions?
- Will need many points. Costly to optimise positions.
  - Cheaper to have more delta functions, but be sloppier in positioning them.
  - Want it to be consistent: has the right distributional limit (by some means of assessment).
  - Want to do the right thing on average (unbiased).
- Instead of optimizing get positions via *sampling*.
  - Desirable properties e.g. Monte-Carlo estimates.
  - Monte-Carlo estimates are consistent (and unbiased).
  - Can obtain posterior samples from intractable distributions via Markov Chain methods.

- But how do we get samples? Use properties of Markov Chains:
  - Ergodicity: a Markov chain is ergodic if you would expect to get from each state to any other state in finite time, and if it is acyclic: its return time to any state is not always divisible by a number > 1.
  - Reversibility: a Markov chain is reversible iff it satisfies detailed balance: for some distribution P_B: $P_B(\theta)P_T(\phi|\theta) = P_B(\phi)P_T(\theta|\phi)$
  - Equilibrium Distribution: an ergodic Markov chain has a unique equilibrium distribution $P_\infty(\theta)$ such that

$$P_\infty(\theta) = \int d\theta' \; P_T(\theta|\theta')P_\infty(\theta')$$

  - An ergodic reversible Markov chain satisfying detailed balance wrt P_B has P_B as its unique equilibrium distribution.

How?

- Did not know how to sample from a distribution $P(\theta)$.
- Idea: Use a Markov chain. Design so $P(\theta)$ is equilibrium distribution.
- Run Markov chain sampling 'for long enough' to get samples from equilibrium distribution.
- How to design Markov chain? Ensure satisfies detailed balance wrt. $P(\theta)$,
- Sampling from a chain:
- Initialise state θ_0. Compute $P_T(\theta_1|\theta_0)$. Sample from this to get θ_1. Repeat ad infinitum (or until you get bored).
- Markov Chain Monte-Carlo (MCMC)

Markov Chain Sampling

- Want Posterior P(x|D)
- Need to approximate. Can we sample from it to get mixture of deltas approximation?
- Not directly but indirectly:
  - We can design a Markov chain to have limit distribution P(x|D), and sample from the chain.
- Markov chain:

$$P(x_t) = \sum_{x_{t-1}} P(x_t | x_{t-1}) P(x_{t-1})$$

Gibbs Sampling

- Gibbs sampling is one Markov Chain Monte Carlo method.
- Others discussed in more detail in MLPR

- Markov chain: Adapt θ_i keeping all $\theta_{j \neq i}$ fixed. i.e.
- Choose i uniformly from $i = 1, 2, \ldots, D$. Set $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t$. Then sample $\theta_{t+1,i}$ from the conditional probability $P(\theta_{t+1,i} | \theta_{t+1,\neq i})$ where $\theta_{t+1,\neq i}$ denotes the set $\{\theta_{t+1,j} | j \neq i\}$.
- Repeat.
- Can cycle through i either (this is not reversible, but can be shown to have a unique equilibrium distribution)

Example: The Boltzmann Machine

- Remember the good old Gaussian

$$P(\mathbf{x}) = \frac{1}{Z}\exp(-E(\mathbf{x}))$$

where

$$\begin{aligned} E(\mathbf{x}) &= \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Lambda}(\mathbf{x}-\boldsymbol{\mu}) \\ &= \frac{1}{2}\mathbf{x}^T\boldsymbol{\Lambda}\mathbf{x} + \mathbf{b}^T\mathbf{x} + \mathrm{const} \end{aligned}$$

- x is real valued.
- Does it have to be in these equations?
- What happens to Z if it isn't?

The Boltzmann Machine

- The Boltzmann Machine has the form

$$P(\mathbf{x}) = \frac{1}{Z}\exp(-E(\mathbf{x}))$$

where

$$E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x} + \mathbf{b}^T\mathbf{x}$$

$$x_i \in \{0, 1\}$$

- but where x is a binary vector
- Z is now not simple to compute.

- Consider the following questions:
  - What is the graphical model for a Boltzmann Machine?
  - What does a Boltzmann Machine model that a Gaussian doesn't?
  - What sort of information can be captured?
  - How can we do learning and inference in a Boltzmann Machine?
  - What form does the Energy function take.
- We will discuss

ML and Graphical Models

- Remember: need to be able to compute with both prior and posterior.
- Previously we wrote

Let $\mathbf{x}^n = ((\mathbf{v}^n)^T, (\mathbf{h}^n)^T)^T$, and $P(\mathbf{x}|\boldsymbol{\theta}) = \frac{1}{Z}\exp(\sum_i \phi_i(\mathbf{x}_{C_i}|\theta_i))$
Using trick from previous slide,

$$\frac{\partial}{\partial\theta_i}\sum_n \log\sum_{\mathbf{h}^n} P(\mathbf{x}^n|\boldsymbol{\theta}) = \left[\sum_n\sum_{\mathbf{h}^n} P(\mathbf{x}^n_{C_i}|\boldsymbol{\theta},\mathbf{v}^n)\frac{\partial}{\partial\theta_i}\phi(\mathbf{x}^n_{C_i}|\theta_i)\right] - N\frac{\partial}{\partial\theta_i}\log Z(\theta).$$

But $Z(\boldsymbol{\theta})$ is also a log sum as on previous slide, so we can rewrite

Posterior over latent vars

$$\frac{\partial}{\partial\theta_i}\sum_n \log\sum_{\mathbf{h}^n} P(\mathbf{x}^n|\boldsymbol{\theta}) = \sum_n \left[\sum_{\mathbf{h}^n_{C_i}} P(\mathbf{x}^n_{C_i}|\boldsymbol{\theta},\mathbf{v}^n)\frac{\partial}{\partial\theta_i}\phi(\mathbf{x}^n_{C_i}|\theta_i) - \sum_{\mathbf{x}'_{C_i}} P(\mathbf{x}'_{C_i}|\boldsymbol{\theta})\frac{\partial}{\partial\theta_i}\phi(\mathbf{x}'_{C_i}|\theta_i)\right]$$

Prior over latent vars

The Boltzmann Machine

- Learning and Inference is hard.
- Inference is tough due to high connectivity, and no tractability.
- Can sample using Gibbs sampling.

- But even sampling is tough as disconnected regions of high probability.
- Learning is hard because we don't have either the prior or posterior to use to get our gradient.

The Restricted Boltzmann Machine

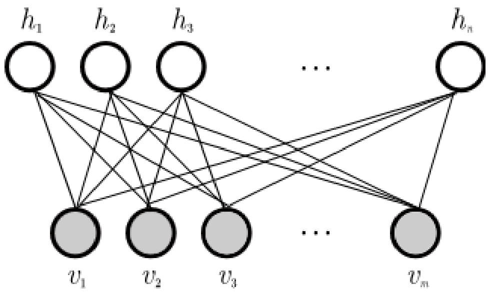
- Let us make things easier.
- The Restricted Boltzmann Machine has the form

$$P(\mathbf{x}) = \frac{1}{Z}\exp(-E(\mathbf{x}))$$

where

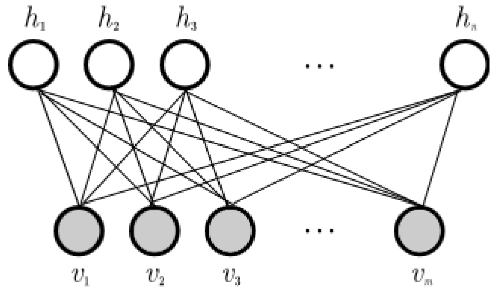
$$E(\mathbf{x}) = \mathbf{v}^T\mathbf{W}\mathbf{h} + \mathbf{a}^T\mathbf{v} + \mathbf{b}^T\mathbf{h}$$

- What is its graphical structure?



- What are its conditional independence relationships?
- For the RBM the posterior is tractable but the prior isn't!

Gibbs sampling the latent prior



- Given the model
- Start at any v.
- Iterate P(h|v), P(v|h)
- Keep iterating until sufficiently converged
- Draw samples of v,h.
- Use samples in gradient updates.

- Takes a long time.
- Can cheat: start v at data. Sample h. Do small n number of iterations of Gibbs sampling.
- Use these is gradient updates.
- *Contrastive Divergence.*

Learning with samples

- Remember: Gibbs sampling for inference?
- But how do we do learning?
- Can just sample jointly from parameters and latent variables: learning as inference.
  - But that can be hard to get good mixing.
- Can we do gradient ascent?
  - Tough because gradient estimate is noisy (e.g. Contrastive Divergence). That effects some gradient method
- Use stochastic gradient ascent.

Stochastic Gradient

- Use the sampling methods to get a noisy gradient.
- Use noisy gradients to gradually improve the parameters.

Stochastic Gradient Methods

- Take dataset and split it into minibatches.
- Now select a minibatch (sequentially or at random)
- Compute the gradient for the minibatch.
- Update the parameters.
- Move on to the next minibatch.
- Reduce the learning rate through time.
- Lots of details…
- Benefit – make parameter changes on minibatches not whole datasets. More steps, faster, but noisier learning.
- For large datasets, the minibatch may contain all the info you need to get the right gradient direction.

Stacked RBMs

- Having learnt an RBM. We have a mapping from visible to hidden units.
- Given the visibles we can obtain a hidden representation.
- In fact we could just focus on this representation as a summary for the data.
- And we could learn another RBM for that representation
- And so on.
- The basis for early models of unsupervised deep learning
- Also used as a pretraining method for supervised deep learning.
  - Train a deep unsupervised model.
  - Leverage the learnt parameters as a model for

Summary

- Sampling
- Boltzmann Machine
- Restricted Boltzmann Machine
- Deep Learning
- Stochastic Gradient Methods.