



Training Products of Experts by Minimizing Contrastive Divergence

Geoffrey E. Hinton

presented by Yuxiong Wang

10/21/2013

Goal

- Learn parameters for probability distribution models of high dimensional data
 - (Images, Population Firing Rates, Securities Data, NLP data, etc)

Mixture Model

$$p(\vec{d} \mid \theta_1, \dots, \theta_n) = \sum_m \alpha_m f_m(\vec{d} \mid \theta_m)$$

Use EM to learn parameters

Product of Experts

$$p(\vec{d} \mid \theta_1, \dots, \theta_n) = \frac{\prod_m f_m(\vec{d} \mid \theta_m)}{\sum_{\vec{c}} \prod_m f_m(\vec{c} \mid \theta_m)}$$

Use Contrastive Divergence to learn parameters

Outline

1

Products of Experts

2

What & Why is CD?

3

How does CD Work?

4

More concrete Analysis

5

Other Issues

Outline

1

Products of Experts

2

What & Why is CD?

3

How does CD Work?

4

More concrete Analysis

5

Other Issues


How to combine simple density models

mixing
proportion

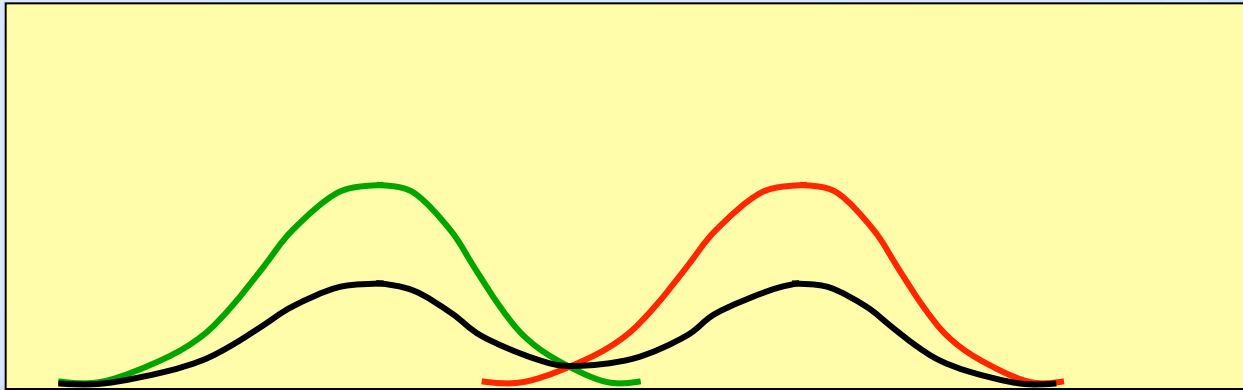


$$p(d) = \sum_m \alpha_m f_m(d)$$

- Suppose we want to build a model of a complicated data distribution by combining several simple models. What combination rule should we use?
- **Mixture models** take a weighted sum of the distributions
 - Easy to learn
 - The combination is always vaguer than the individual distributions
- **Products of Experts** multiply the distributions together and renormalize
 - The product is much sharper than the individual distributions
 - A nasty normalization term is needed to convert the product of the individual densities into a combined density


$$p(d) = \frac{\prod_m f_m(d)}{\sum_c \prod_m f_m(c)}$$

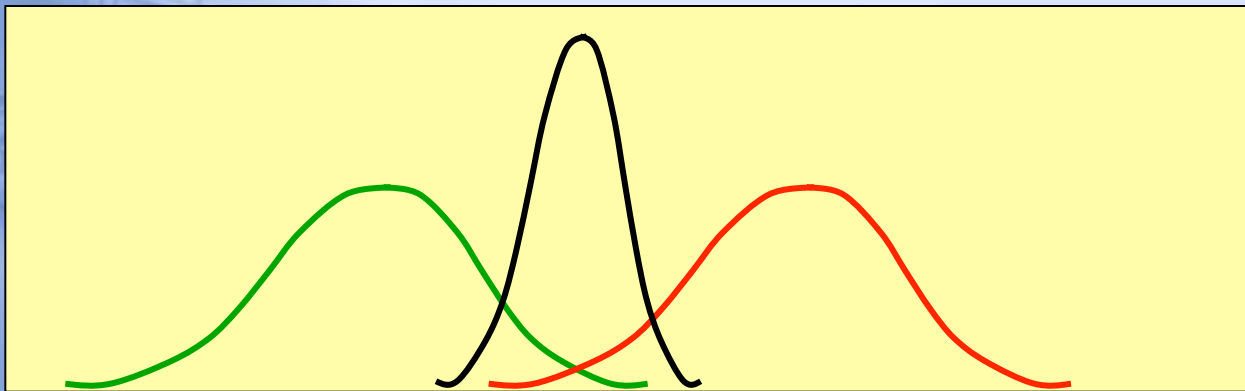
A picture of the two combination methods



Mixture model:
Scale each
distribution down
and add them
together

OR operation

Soft Template



Product model:
Multiply the two
densities together
at every point and
then renormalize

ADD Operation

Soft Constraint

Products of Experts and energies

- Products of Experts multiply probabilities together. This is equivalent to adding log probabilities.
 - Mixture models add contributions in the probability domain.
 - Product models add contributions in the log probability domain. The contributions are energies.
- In a mixture model, the only way a new component can reduce the density at a point is by stealing mixing proportion.
- In a product model, any expert can veto any point by giving that point a density of zero (i.e. an infinite energy)
 - So its important not to have overconfident experts in a product model.
 - Luckily, vague experts work well because their product can be sharp.

How sharp are products of experts?

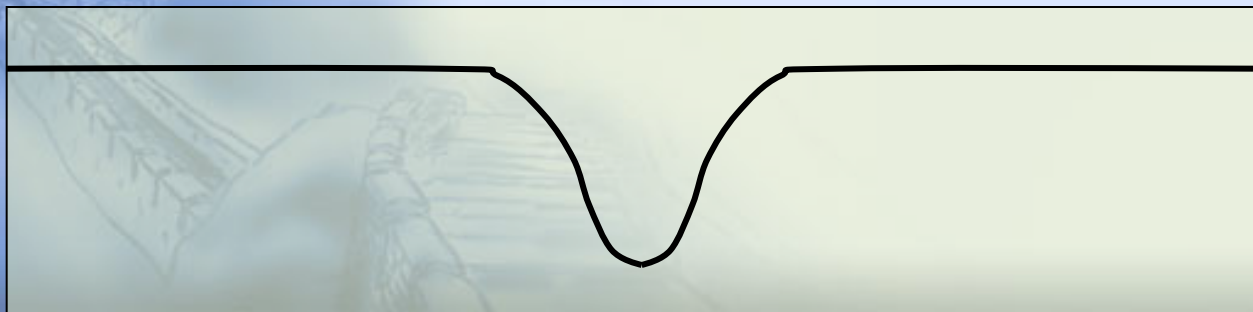
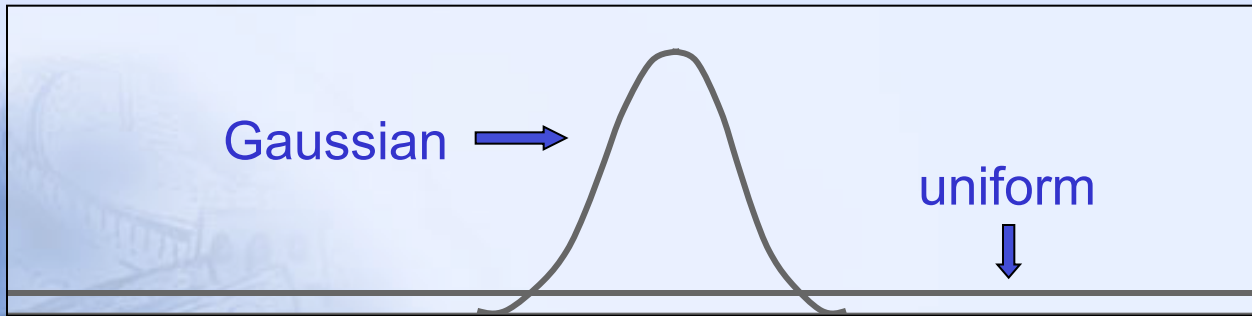
- If each of the M experts is a Gaussian with the same variance, the product is a Gaussian with a variance of $1/M$ on each dimension.
- But a product of lots of Gaussians is just a Gaussian
 - Adding Gaussians allows us to create arbitrarily complicated distributions.
 - Multiplying Gaussians doesn't.
 - So we need to multiply more complicated “experts” .

“Uni-gauss” experts

- Each expert is a mixture of a Gaussian and a uniform. This creates an energy dimple.

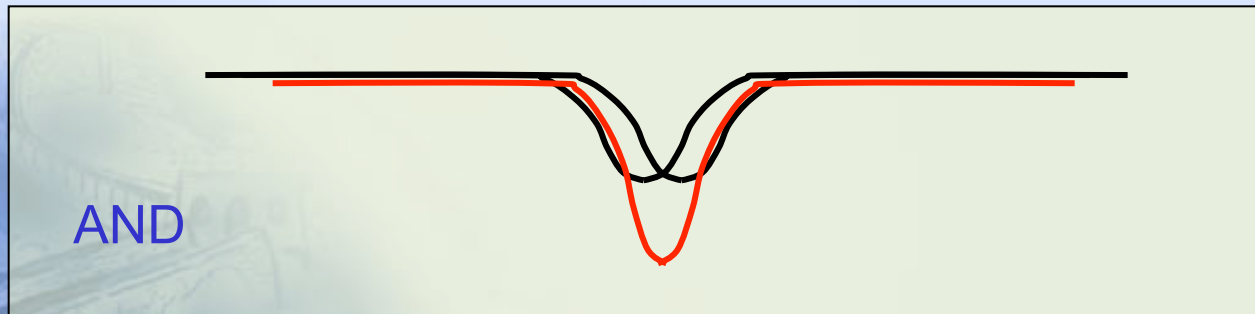
$$p_m(x) = \pi_m N(x | \mu, \Sigma) + \frac{1 - \pi_m}{r}$$

Mixing proportion of Gaussian Mean and variance of Gaussian range of uniform

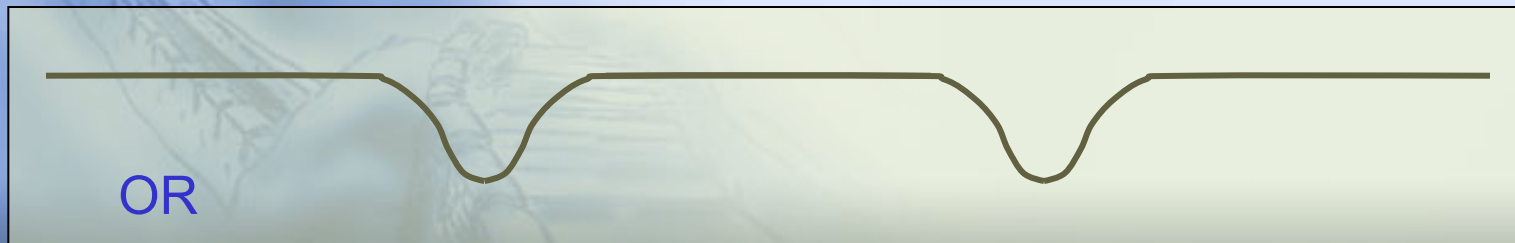


Combining energy dimples

- When we combine dimples, we get a sharper distribution if the dimples are close and a vaguer, multimodal distribution if they are further apart. We can get **both** multiplication and addition of probabilities.



↑ $E(x) = -\log p(x)$



Generating from a product of experts

- Here is a correct but inefficient way to generate an unbiased sample from a product of experts:
 - Let each expert produce a datavector **independently**.
 - If all the experts agree, output the datavector.
 - If they do not all agree, start again.
- The experts generate independently, but because of the rejections, their hidden states are not independent in the ensemble of accepted cases.
 - The proportion of rejected attempts implements the normalization term.

Relationship to causal generative models

- Consider the relationship between the hidden variables of two different experts:

	Causal model	Product of experts
Hidden states unconditional on data	independent (generation is easy)	dependent (rejecting away)
Hidden states conditional on data	dependent (explaining away)	independent (inference is easy)

Outline

1

Products of Experts

2

What & Why is CD?

3

How does CD Work?

4

More concrete Analysis

5

Other Issues

Learning a Product of Experts

- Let's first trace back to the general case of
 - Probability Modeling
 - Model Parameter Learning

Probability Modeling

- Model the probability of a data point x using a function of the form $f(x; \Theta)$
 - Θ is a vector of model parameters
 - $p(x; \Theta)$ must integrate to 1 over all x
 - partition function: $Z(\Theta)$

$$p(x; \Theta) = \frac{1}{Z(\Theta)} f(x; \Theta)$$

$$Z(\Theta) = \int f(x; \Theta) dx$$

Model Parameter Learning

- Maximizing the probability of a training set of data $X = x_{1,\dots,K}$

$$p(\mathbf{X}; \Theta) = \prod_{k=1}^K \frac{1}{Z(\Theta)} f(x_k; \Theta)$$

- Equivalently, minimizing energy $E(\mathbf{X}; \Theta)$: the negative log of $p(\mathbf{X}; \Theta)$

$$E(\mathbf{X}; \Theta) = \log Z(\Theta) - \frac{1}{K} \sum_{k=1}^K \log f(x_k; \Theta)$$

- Maximum-Likelihood learning

Energy Function Minimization

- Case I
 - Single model
 - exact minimization
 - $f(x; \Theta)$: normal distribution
 - $\log Z(\Theta) = 0$
- Case II
 - sum-of-experts or mixture model
 - Parameters from different models couple
 - use the partial differential equations and a gradient descent method with line search to find a local minimum of energy in the parameter space
 - Sum of N normal distributions
 - $\log Z(\Theta) = \log N$

$$f(x; \Theta) = \sum_{i=1}^N \mathcal{N}(x; \mu_i, \sigma_i)$$

Energy Function Minimization

- Case III
- product-of-experts model
- The partition function $Z(\Theta)$ is no longer a constant

$$f(x; \Theta) = \prod_{i=1}^N \mathcal{N}(x; \mu_i, \sigma_i)$$

- product of N normal distributions
- a model consisting of two normal distributions, both with $\sigma = 1$. If $\mu_1 = -\infty$ and $\mu_2 = \infty$ then $Z(\Theta) = 0$, while if $\mu_1 = \mu_2 = 0$ then $Z(\Theta) = \frac{1}{2}\sqrt{\pi}$
- Integration in $Z(\Theta)$ is not algebraically tractable

Energy Function Minimization

- Case III
- New situations:
- Need to
 - use a numerical integration method to evaluate $E(X; \Theta)$
 - use finite differences to calculate the gradient at a given point in parameter space
 - Use a gradient descent method to find a local minimum
- For high dimensional data spaces the integration time is crippling, and a high-dimensional parameter space compounds this problem
- This leads to a situation **where we are trying to minimize an energy function that we cannot evaluate.**

Solutions---CD!!!

- Even though we cannot evaluate the energy function itself
- Contrastive Divergence (CD) provides a way to estimate the gradient of the energy function!

Outline

1

Products of Experts

2

What & Why is CD?

3

How does CD Work?

4

More concrete Analysis

5

Other Issues

How does CD work?

- The partial derivative

$$\begin{aligned}\frac{\partial E(\mathbf{X}; \Theta)}{\partial \Theta} &= \frac{\partial \log Z(\Theta)}{\partial \Theta} - \frac{1}{K} \sum_{i=1}^K \frac{\partial \log f(x_i; \Theta)}{\partial \Theta} \\ &= \frac{\partial \log Z(\Theta)}{\partial \Theta} - \left\langle \frac{\partial \log f(x; \Theta)}{\partial \Theta} \right\rangle_{\mathbf{X}}\end{aligned}$$

$\langle \cdot \rangle_{\mathbf{X}}$ is the expectation of \cdot given the data distribution \mathbf{X} .

How does CD work?

- The first term

$$\begin{aligned}\frac{\partial \log Z(\Theta)}{\partial \Theta} &= \frac{1}{Z(\Theta)} \frac{\partial Z(\Theta)}{\partial \Theta} \\&= \frac{1}{Z(\Theta)} \frac{\partial}{\partial \Theta} \int f(x; \Theta) dx \\&= \frac{1}{Z(\Theta)} \int \frac{\partial f(x; \Theta)}{\partial \Theta} dx \\&= \frac{1}{Z(\Theta)} \int f(x; \Theta) \frac{\partial \log f(x; \Theta)}{\partial \Theta} dx \\&= \int p(x; \Theta) \frac{\partial \log f(x; \Theta)}{\partial \Theta} dx \\&= \left\langle \frac{\partial \log f(x; \Theta)}{\partial \Theta} \right\rangle_{p(x; \Theta)}\end{aligned}$$

How does CD work?

$$\begin{aligned}\frac{\partial \log Z(\Theta)}{\partial \Theta} &= \frac{1}{Z(\Theta)} \frac{\partial Z(\Theta)}{\partial \Theta} \\ &= \left\langle \frac{\partial \log f(x; \Theta)}{\partial \Theta} \right\rangle_{p(x; \Theta)}\end{aligned}$$

- This integration is generally algebraically intractable.
- **However**, here it can be numerically approximated by drawing samples from the proposed distribution, $p(x; \Theta)$
- But, wait ...

How does CD work?

- Samples cannot be drawn directly from $p(x; \Theta)$ as we do not know the value of the partition function
- Use many cycles of Markov Chain Monte Carlo (MCMC) sampling to transform our training data (drawn from the target distribution) into data drawn from the proposed distribution.
- This is possible as the transformation only involves calculating the ratio of two probabilities $p(x'; \Theta)/p(x; \Theta)$, so the partition function cancels out

How does CD work?

- MCMC
- X^n represents the training data transformed using n cycles of MCMC, such that $X^0 \equiv X$

$$\frac{\partial E(\mathbf{X}; \Theta)}{\partial \Theta} = \left\langle \frac{\partial \log f(x; \Theta)}{\partial \Theta} \right\rangle_{\mathbf{X}^\infty} - \left\langle \frac{\partial \log f(x; \Theta)}{\partial \Theta} \right\rangle_{\mathbf{X}^0}$$

- Still
- Many MCMC cycles required to compute an accurate gradient will take far too long

How does CD work?

- Hinton's assertion: only a few MCMC cycles would be needed to calculate an approximate gradient
 - After a few iterations the data will have moved from the target distribution (i.e. that of the training data) towards the proposed distribution
 - And so give an idea in which direction the proposed distribution should move to better model the training data.
- Empirically, even 1 cycle of MCMC is sufficient for the algorithm to converge

How does CD work?

- Parameter update equation can be written as

$$\Theta_{t+1} = \Theta_t + \eta \left(\left\langle \frac{\partial \log f(x; \Theta)}{\partial \Theta} \right\rangle_{\mathbf{x}^0} - \left\langle \frac{\partial \log f(x; \Theta)}{\partial \Theta} \right\rangle_{\mathbf{x}^1} \right)$$

Outline

1

Products of Experts

2

What & Why is CD?

3

How does CD Work?

4

More concrete Analysis

5

Other Issues

Brief Summary

- Contrastive divergence is a general MCMC gradient ascent learning algorithm particularly well suited to learning Product of Experts (PoE) and energy-based (Gibbs distributions, etc.) model parameters.
- The general algorithm:
 - Repeat Until “Convergence”
 - Draw samples from the current model *starting from the training data*.
 - Compute the expected gradient of the log probability w.r.t. all model parameters over both samples and the training data.
 - Update the model parameters according to the gradient.

Sampling – Critical to Understanding

■ Uniform

- rand() Linear Congruential Generator

- $x(n) = a * x(n-1) + b \bmod M$

0.2311 0.6068 0.4860 0.8913 0.7621 0.4565 0.0185

■ Normal

- randn() Box-Mueller

- $x_1, x_2 \sim U(0,1) \rightarrow y_1, y_2 \sim N(0,1)$

- $y_1 = \sqrt{-2 \ln(x_1)} \cos(2 \pi x_2)$

- $y_2 = \sqrt{-2 \ln(x_1)} \sin(2 \pi x_2)$

■ Binomial(p)

- if(rand()) < p)

■ More Complicated Distributions

- Mixture Model
 - Sample from a Gaussian
 - Sample from a multinomial (CDF + uniform)
- Product of Experts
 - Metropolis and/or Gibbs

The Flavor of Metropolis Sampling

- Given some distribution $p(\vec{d} | \theta)$, a random starting point \vec{d}_{t-1} , and a symmetric proposal distribution $J(\vec{d}_t | \vec{d}_{t-1})$.
- Calculate the ratio of densities $r = \frac{p(\vec{d}_t | \theta)}{p(\vec{d}_{t-1} | \theta)}$ where \vec{d}_t is sampled from the proposal distribution.
- With probability $\min(r, 1)$ accept \vec{d}_t .
- Given sufficiently many iterations $\{\vec{d}_n, \vec{d}_{n+1}, \vec{d}_{n+2}, \dots\} \sim p(\vec{d} | \theta)$

Only need to know the distribution up to a proportionality!

Contrastive Divergence (Final Result!)

Model parameters.

Training data
(empirical distribution).

$$\Delta \theta_m \propto \left\langle \frac{\partial \log f_{\theta_m}}{\partial \theta_m} \right\rangle_{P^0} - \left\langle \frac{\partial \log f_{\theta_m}}{\partial \theta_m} \right\rangle_{P_{\theta}^1}$$

Samples from model.

Law of Large Numbers, compute expectations using samples.

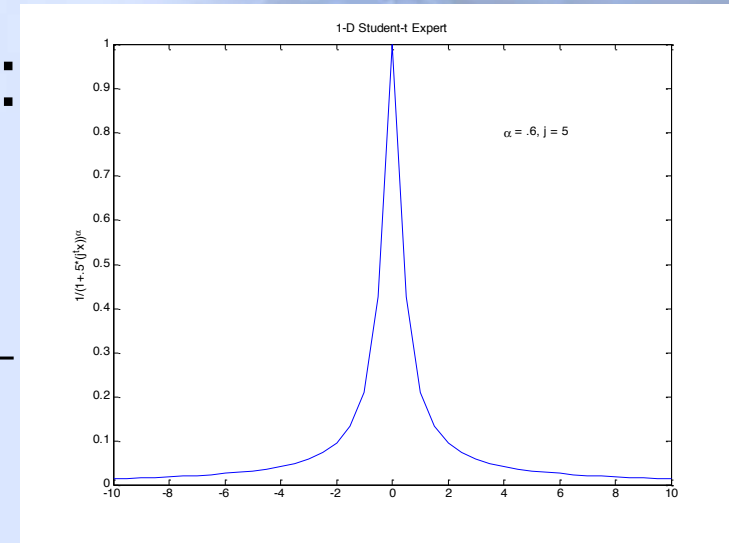
$$\Delta \theta_m \propto \frac{1}{N} \sum_{d \in D} \frac{\partial \log f_{\theta_m}(d)}{\partial \theta_m} - \frac{1}{N} \sum_{c \sim P_{\theta}^1} \frac{\partial \log f_{\theta_m}(c)}{\partial \theta_m}$$

Now you know how to do it, let's see why this works!

But First: The last vestige of concreteness.

- Looking towards the future:
 - Take f to be a Student-t.

$$f_{\theta^m}(\vec{d}) = f_{\alpha_m, \vec{j}_m}(\vec{d}) = \frac{1}{\left(1 + \frac{1}{2}(\vec{j}_m^T \vec{d})\right)^{\alpha_m}}$$



- Then (for instance)

$$\frac{\partial \log f_{\alpha_m, \vec{j}_m}(\vec{d})}{\partial \alpha_m} = \frac{-\partial \alpha_m \log\left(1 + \frac{1}{2}(\vec{j}_m^T \vec{d})\right)}{\partial \alpha_m} = -\log\left(1 + \frac{1}{2}(\vec{j}_m^T \vec{d})\right)$$

Dot product \Leftrightarrow Projection \Leftrightarrow 1-D Marginal

Maximizing the training data log likelihood

- We want maximizing parameters

Standard PoE form

$$\arg \max_{\theta_1, \dots, \theta_n} \log p(D | \theta_1, \dots, \theta_n) = \arg \max_{\theta_1, \dots, \theta_n} \log \prod_{\vec{d} \in D} \left(\frac{\prod_m f_m(\vec{d} | \theta_m)}{\sum_{\vec{c}} \prod_m f_m(\vec{c} | \theta_m)} \right)$$

Over all training data.

Assuming d's drawn independently from $p()$

- Differentiate w.r.t. to all parameters and perform gradient ascent to find optimal parameters.
- The derivation is somewhat nasty.

Maximizing the training data log likelihood

$$\begin{aligned}\frac{\partial \log p(D | \theta_1, \dots, \theta_n)}{\partial \theta_m} &= \frac{\partial \log \prod_{\vec{d} \in D} p(\vec{d} | \theta_1, \dots, \theta_n)}{\partial \theta_m} \\ &= \sum_{\vec{d} \in D} \frac{\partial}{\partial \theta_m} \log p(\vec{d} | \theta_1, \dots, \theta_n) \\ &= N \left\langle \frac{\partial \log p(\vec{d} | \theta_1, \dots, \theta_n)}{\partial \theta_m} \right\rangle_{P_{\theta}^{\infty} \star}\end{aligned}$$

Remember this
equivalence!

Maximizing the training data log likelihood

$$\begin{aligned}\frac{1}{N} \frac{\partial \log p(D | \theta_1, \dots, \theta_n)}{\partial \theta_m} &= \frac{1}{N} \frac{\partial}{\partial \theta_m} \log \prod_{\vec{d} \in D} \frac{\prod_m f_m(\vec{d} | \theta_m)}{\sum_{\vec{c}} \prod_m f_m(\vec{c} | \theta_m)} \\&= \frac{1}{N} \sum_{\vec{d} \in D} \frac{\partial \log f_m(\vec{d} | \theta_m)}{\partial \theta_m} - \frac{1}{N} \sum_{\vec{d} \in D} \frac{\partial \log \sum_{\vec{c}} \prod_m f_m(\vec{c} | \theta_m)}{\partial \theta_m} \\&= \frac{1}{N} \sum_{\vec{d} \in D} \frac{\partial \log f_m(\vec{d} | \theta_m)}{\partial \theta_m} - \frac{\partial \log \sum_{\vec{c}} \prod_m f_m(\vec{c} | \theta_m)}{\partial \theta_m}\end{aligned}$$

Maximizing the training data log likelihood

$$\begin{aligned}
 &= \frac{1}{N} \sum_{d \in D} \frac{\partial \log f_m(\vec{d} | \theta_m)}{\partial \theta_m} - \frac{\partial \log \sum_{\vec{c}} \prod_m f_m(\vec{c} | \theta_m)}{\partial \theta_m} \\
 &= \left\langle \frac{\partial \log f_m(\vec{d} | \theta_m)}{\partial \theta_m} \right\rangle_{P^0} - \frac{\partial \log \sum_{\vec{c}} \prod_m f_m(\vec{c} | \theta_m)}{\partial \theta_m} \\
 &= \left\langle \frac{\partial \log f_m(\vec{d} | \theta_m)}{\partial \theta_m} \right\rangle_{P^0} - \frac{1}{\sum_{\vec{c}} \prod_m f_m(\vec{c} | \theta_m)} \frac{\partial \sum_{\vec{c}} \prod_m f_m(\vec{c} | \theta_m)}{\partial \theta_m}
 \end{aligned}$$

$\log(x)' = x' / x$

Maximizing the training data log likelihood

$$\begin{aligned}
 &= \left\langle \frac{\partial \log f_m(\vec{d} | \theta_m)}{\partial \theta_m} \right\rangle_{P^0} - \frac{1}{\sum_{\vec{c}} \prod_m f_m(\vec{c} | \theta_m)} \frac{\partial \sum_{\vec{c}} \prod_m f_m(\vec{c} | \theta_m)}{\partial \theta_m} \\
 &= \left\langle \frac{\partial \log f_m(\vec{d} | \theta_m)}{\partial \theta_m} \right\rangle_{P^0} - \frac{1}{\sum_{\vec{c}} \prod_m f_m(\vec{c} | \theta_m)} \frac{\sum_{\vec{c}} \prod_{j \neq m} f_j(\vec{c} | \theta_j) \partial f_m(\vec{c} | \theta_m)}{\partial \theta_m} \\
 &= \left\langle \frac{\partial \log f_m(\vec{d} | \theta_m)}{\partial \theta_m} \right\rangle_{P^0} - \frac{1}{\sum_{\vec{c}} \prod_m f_m(\vec{c} | \theta_m)} \frac{\sum_{\vec{c}} \prod_m f_m(\vec{c} | \theta_m) \partial \log f_m(\vec{c} | \theta_m)}{\partial \theta_m}
 \end{aligned}$$

$\log(x)' = x'/x$

Maximizing the training data log likelihood

$$= \left\langle \frac{\partial \log f_m(\vec{d} | \theta_m)}{\partial \theta_m} \right\rangle_{P^0} - \frac{1}{\sum_{\vec{c}} \prod_m f_m(\vec{c} | \theta_m)} \frac{\sum_{\vec{c}} \prod_m f_m(\vec{c} | \theta_m) \partial \log f_m(\vec{c} | \theta_m)}{\partial \theta_m}$$

$$= \left\langle \frac{\partial \log f_m(\vec{d} | \theta_m)}{\partial \theta_m} \right\rangle_{P^0} - \sum_{\vec{c}} \left(\frac{\prod_m f_m(\vec{c} | \theta_m)}{\sum_{\vec{c}} \prod_m f_m(\vec{c} | \theta_m)} \frac{\partial \log f_m(\vec{c} | \theta_m)}{\partial \theta_m} \right)$$

$$= \left\langle \frac{\partial \log f_m(\vec{d} | \theta_m)}{\partial \theta_m} \right\rangle_{P^0} - \sum_{\vec{c}} p(\vec{c} | \theta_1, \dots, \theta_n) \frac{\partial \log f_m(\vec{c} | \theta_m)}{\partial \theta_m}$$

Maximizing the training data log likelihood

$$\begin{aligned}
 &= \left\langle \frac{\partial \log f_m(\vec{d} \mid \theta_m)}{\partial \theta_m} \right\rangle_{P^0} - \sum_c p(c \mid \theta_1, \dots, \theta_n) \frac{\partial \log f_m(\vec{c} \mid \theta_m)}{\partial \theta_m} \\
 &= \left\langle \frac{\partial \log f_m(\vec{d} \mid \theta_m)}{\partial \theta_m} \right\rangle_{P^0} - \left\langle \frac{\partial \log f_m(\vec{c} \mid \theta_m)}{\partial \theta_m} \right\rangle_{P_{\theta^\infty}}
 \end{aligned}$$

Phew! We're done! So:

$$\begin{aligned}
 \frac{\partial \log p(D \mid \theta_1, \dots, \theta_n)}{\partial \theta_m} &\Leftrightarrow N \left\langle \frac{\partial \log p_\theta^\infty(\vec{d} \mid \theta_m)}{\partial \theta_m} \right\rangle_{P^0} \\
 &\propto \left\langle \frac{\partial \log f_m(\vec{d} \mid \theta_m)}{\partial \theta_m} \right\rangle_{P^0} - \left\langle \frac{\partial \log f_m(\vec{c} \mid \theta_m)}{\partial \theta_m} \right\rangle_{P_{\theta^\infty}}
 \end{aligned}$$

Equilibrium Is Hard to Achieve

- With:

$$\frac{\partial \log p(D | \theta_1, \dots, \theta_n)}{\partial \theta_m} \propto \left\langle \frac{\partial \log f_m(\vec{d} | \theta_m)}{\partial \theta_m} \right\rangle_{P^0} - \left\langle \frac{\partial \log f_m(\vec{c} | \theta_m)}{\partial \theta_m} \right\rangle_{P_{\theta}^{\infty}}$$

we can now train our PoE model.

- But... there's a problem:

- P_{θ}^{∞} is computationally infeasible to obtain (esp. in an inner gradient ascent loop).
- Sampling Markov Chain must converge to target distribution. Often this takes a *very* long time!

Solution: Contrastive Divergence!

$$\frac{\partial \log p(D | \theta_1, \dots, \theta_n)}{\partial \theta_m} \propto \left\langle \frac{\partial \log f_m(\vec{d} | \theta_m)}{\partial \theta_m} \right\rangle_{P^0} - \left\langle \frac{\partial \log f_m(\vec{c} | \theta_m)}{\partial \theta_m} \right\rangle_{P_\theta^1}$$

- Now we don't have to run the sampling Markov Chain to convergence, instead we can stop after 1 iteration (or perhaps a few iterations more typically)
- Why does this work?
 - Attempts to minimize the ways that the model distorts the data.

Equivalence of argmax log P() and argmin KL()

$$\begin{aligned} P^0 \| P_\theta^\infty &= \sum_{\vec{d}} P^0(\vec{d}) \log \frac{P^0(\vec{d})}{P_\theta^\infty(\vec{d})} \\ &= \sum_{\vec{d}} P^0(\vec{d}) \log P^0(\vec{d}) - \sum_{\vec{d}} P^0(\vec{d}) \log P_\theta^\infty(\vec{d}) \\ &= -H(P^0) - \left\langle \log P_\theta^\infty(\vec{d}) \right\rangle_{P^0} \end{aligned}$$

$$\frac{\partial P^0 \| P_\theta^\infty}{\partial \theta_m} = - \left\langle \frac{\partial \log P_\theta^\infty(\vec{d})}{\partial \theta_m} \right\rangle_{P^0}$$

This is what
we got out of
the nasty
derivation!

$\log p(D | \theta_1, \dots, \theta_n)$

Contrastive divergence

Aim is to minimize the amount by which a step toward equilibrium improves the data distribution.

The diagram illustrates the Contrastive Divergence (CD) formula with several annotations:

- data distribution**: A green arrow points from this text to the P in the first KL divergence term.
- model's distribution**: A green arrow points from this text to the Q^∞ in the first KL divergence term.
- distribution after one step of Markov chain**: A green arrow points from this text to the Q^1 in the second KL divergence term.
- Minimize Contrastive Divergence**: A blue arrow points from this text to the CD variable.
- Minimize divergence between data distribution and model's distribution**: A blue arrow points from this text to the first KL divergence term $KL(P \parallel Q^\infty)$.
- Maximize the divergence between confabulations and model's distribution**: A blue arrow points from this text to the second KL divergence term $KL(Q^1 \parallel Q^\infty)$.

$$CD = KL(P \parallel Q^\infty) - KL(Q^1 \parallel Q^\infty)$$

Contrastive Divergence

- We want to “update the parameters to reduce the tendency of the chain to wander away from the initial distribution on the first step” .

$$\begin{aligned}
 -\frac{\partial}{\partial \theta_m} (P^0 \| P_\theta^\infty - P_\theta^1 \| P_\theta^\infty) &= \left\langle \frac{\partial \log f_m(\vec{d} | \theta_m)}{\partial \theta_m} \right\rangle_{P^0} - \left\langle \frac{\partial \log f_m(\vec{d} | \theta_m)}{\partial \theta_m} \right\rangle_{P_\theta^1} \\
 &\quad + \frac{\partial P_\theta^1}{\partial \theta_m} \frac{\partial (P_\theta^1 \| P_\theta^\infty)}{\partial P_\theta^1} \\
 &\propto \left\langle \frac{\partial \log f_m(\vec{d} | \theta_m)}{\partial \theta_m} \right\rangle_{P^0} - \left\langle \frac{\partial \log f_m(\vec{d} | \theta_m)}{\partial \theta_m} \right\rangle_{P_\theta^1}
 \end{aligned}$$

Contrastive divergence

$$\begin{aligned}
 -\frac{\partial KL(Q^0 \parallel Q^\infty)}{\partial \theta} &= -\left\langle \frac{\partial E}{\partial \theta} \right\rangle_{Q^0} + \left\langle \frac{\partial E}{\partial \theta} \right\rangle_{Q^\infty} \\
 -\frac{\partial KL(Q^1 \parallel Q^\infty)}{\partial \theta} &= -\left\langle \frac{\partial E}{\partial \theta} \right\rangle_{Q^1} + \left\langle \frac{\partial E}{\partial \theta} \right\rangle_{Q^\infty} - \frac{\partial Q^1}{\partial \theta} \frac{\partial KL(Q^1 \parallel Q^\infty)}{\partial Q^1}
 \end{aligned}$$

Contrastive divergence makes the awkward terms cancel

changing the parameters changes the distribution of confabulations

Contrastive Divergence (Final Result!)

Model parameters.

Training data (empirical distribution).

Samples from model.

$$\Delta \theta_m \propto \left\langle \frac{\partial \log f_{\theta_m}}{\partial \theta_m} \right\rangle_{P^0} - \left\langle \frac{\partial \log f_{\theta_m}}{\partial \theta_m} \right\rangle_{P_\theta^1}$$

Law of Large Numbers, compute expectations using samples.

$$\Delta \theta_m \propto \frac{1}{N} \sum_{d \in D} \frac{\partial \log f_{\theta_m}(d)}{\partial \theta_m} - \frac{1}{N} \sum_{c \sim P_\theta^1} \frac{\partial \log f_{\theta_m}(c)}{\partial \theta_m}$$

Now you know how to do it and why it works!

A shortcut

- Only run the Markov chain for a few time steps.
 - This gets negative samples very quickly.
 - It works well in practice.
- Why does it work?
 - If we start at the data, the Markov chain wanders away from them data and towards things that it likes more.
 - We can see what direction it is wandering in after only a few steps. It's a big waste of time to let it go all the way to equilibrium.
 - All we need to do is lower the probability of the “confabulations” it produces and raise the probability of the data. Then it will stop wandering away.
 - The learning cancels out once the confabulations and the data have the same distribution.

A shortcut

- A cheaper, lower-variance alternative
- This approximation as trading variance for bias
- Thus, at convergence, we do not expect
- that the estimates of the parameters are equal to those of maximum likelihood learning, but will be slightly biased
- To correct this, one can increase k close to convergence

Outline

1

Products of Experts

2

What & Why is CD?

3

How does CD Work?

4

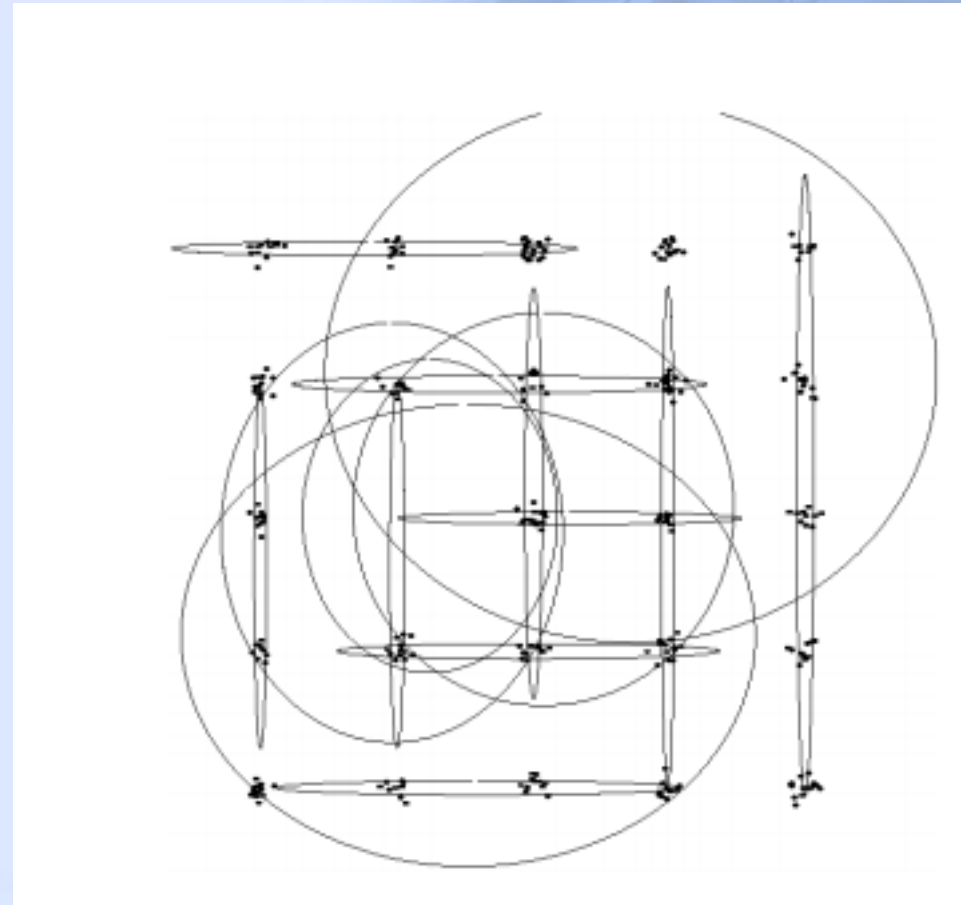
More concrete Analysis

5

Other Issues

Simple Cases

- Data distributions that can be factorized into a product of lower-dimensional distributions
- Each expert is quite broadly tuned on every dimension



Simple Cases

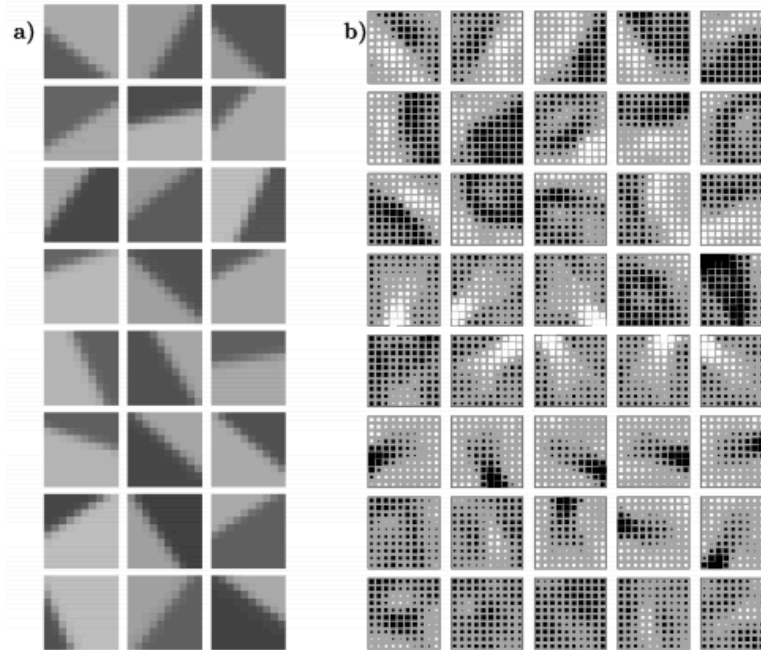
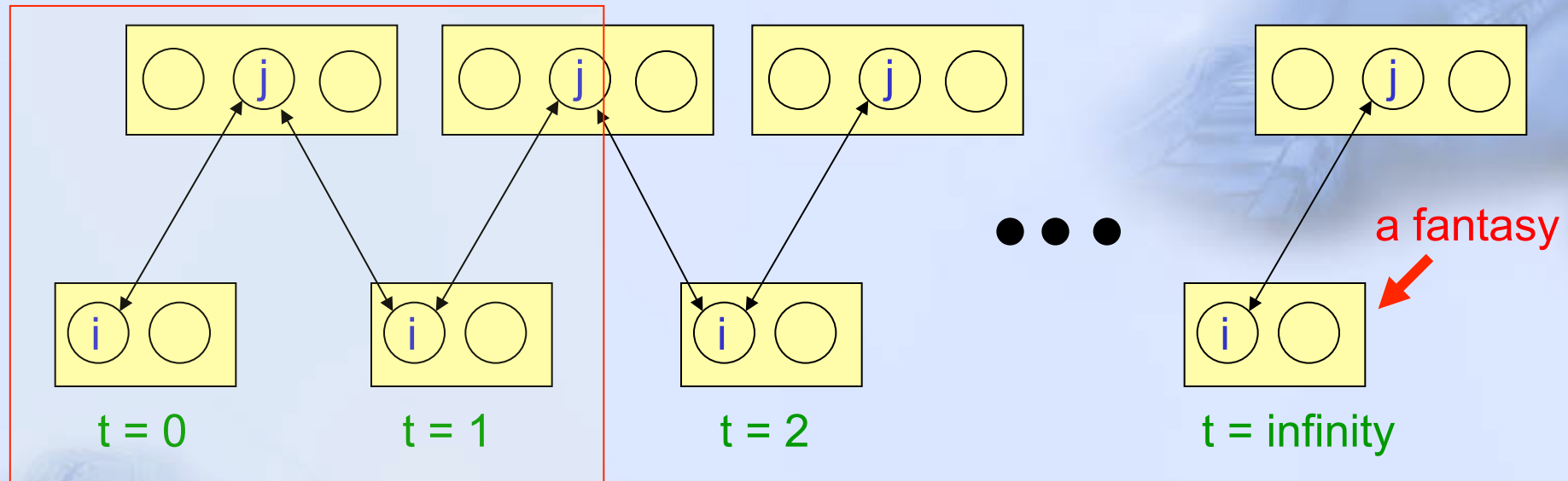


Figure 4: (a) Some 10×10 images that each contain a single intensity edge. The location, orientation, and contrast of the edge all vary. (b) The means of all the 100-dimensional Gaussians in a product of 40 experts, each of which is a mixture of a gaussian and a uniform. The PoE was fitted to 500 images of the type shown on the left. The experts have been ordered by hand so that qualitatively similar experts are adjacent.

The Markov chain for unigauss experts



Each hidden unit has a binary state which is 1 if the unigauss chose its Gaussian. Start with a training vector on the visible units. Then alternate between updating all the hidden units in parallel and updating all the visible units in parallel.

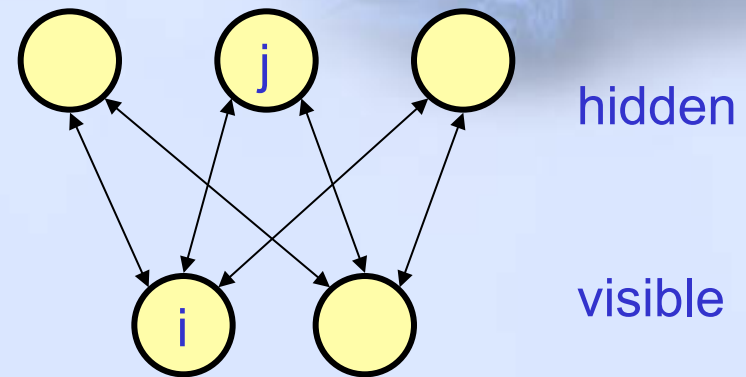
Update the hidden states by picking from the posterior.

Update the visible states by picking from the Gaussian you get when you multiply together all the Gaussians for the active hidden units.

Restricted Boltzmann Machines

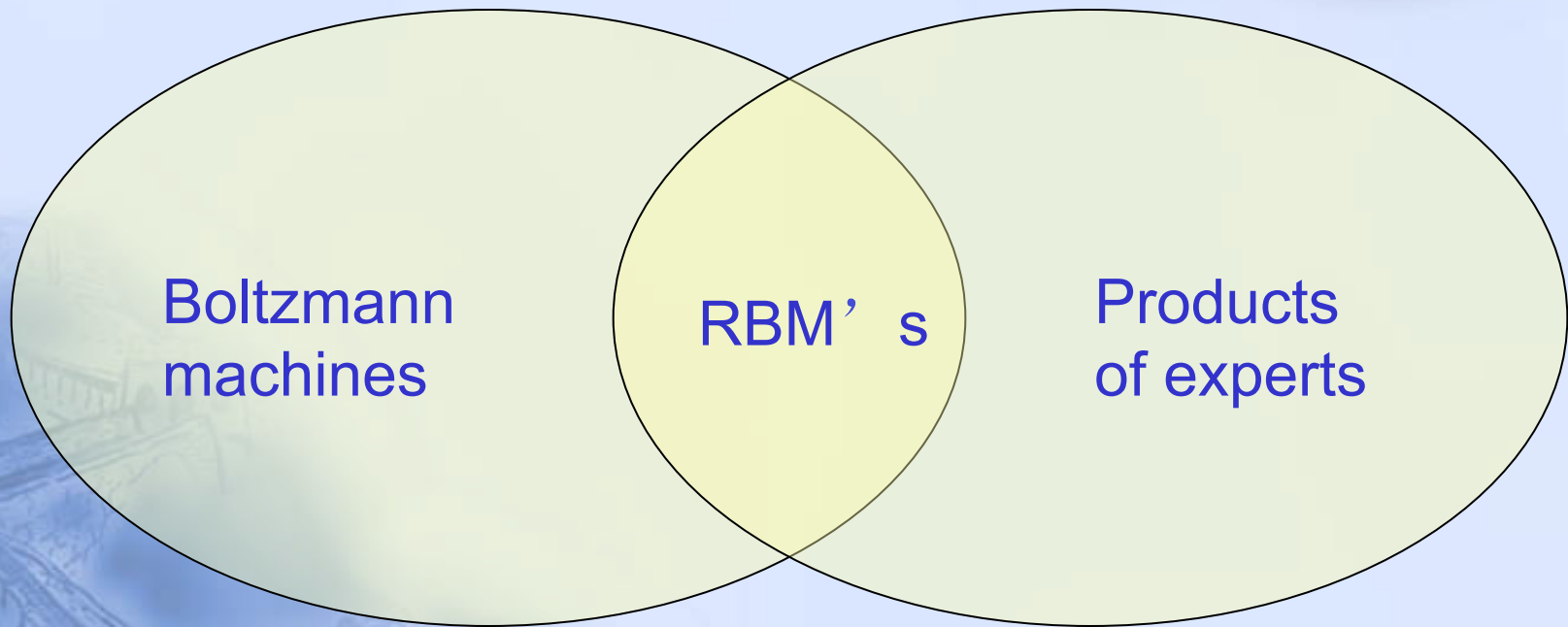
- We restrict the connectivity to make inference and learning easier.
 - Only one layer of hidden units.
 - No connections between hidden units.
- In an RBM it only takes one step to reach thermal equilibrium when the visible units are clamped.
 - So we can quickly get the exact value of :

$$\langle S_i S_j \rangle_{\mathbf{v}}$$

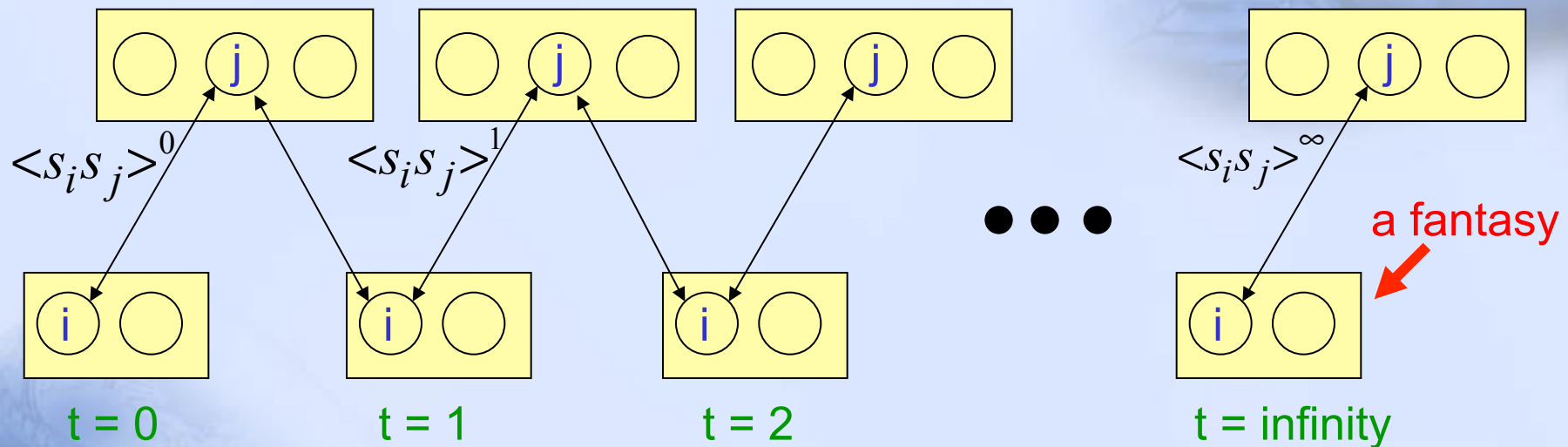


$$p(s_j = 1) = \frac{1}{1 + e^{-\left(b_j + \sum_{i \in \text{vis}} s_i w_{ij}\right)}}$$

Restricted Boltzmann Machines and products of experts



A picture of the Boltzmann machine learning algorithm for an RBM

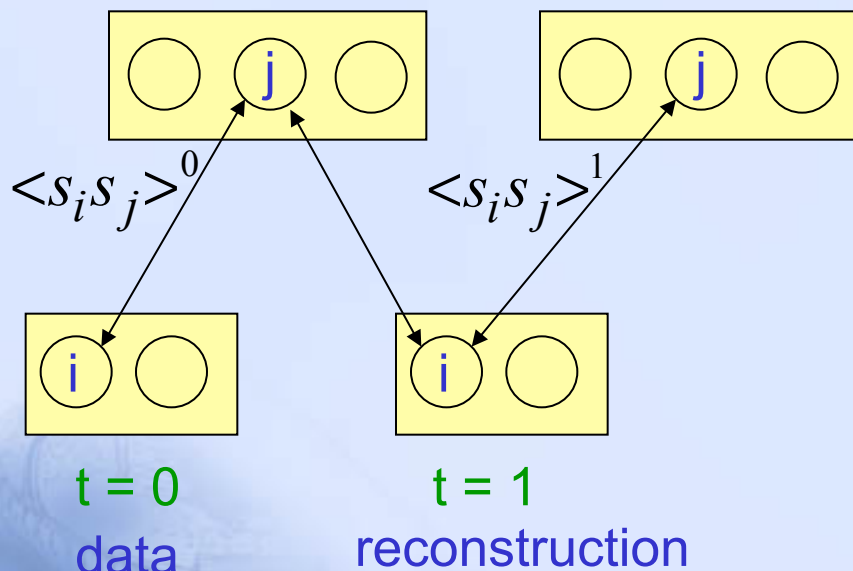


Start with a training vector on the visible units.

Then alternate between updating all the hidden units in parallel and updating all the visible units in parallel.

$$\Delta w_{ij} = \varepsilon (\langle s_i s_j \rangle^0 - \langle s_i s_j \rangle^\infty)$$

The short-cut



Start with a training vector on the visible units.

Update all the hidden units in parallel

Update the all the visible units in parallel to get a “reconstruction” .

Update the hidden units again.

$$\Delta w_{ij} = \mathcal{E} (\langle s_i s_j \rangle^0 - \langle s_i s_j \rangle^1)$$

This is not following the gradient of the log likelihood. But it works very well.

Relationship to Independent and Extreme Components Analysis

- Noiseless Independent Components Analysis (ICA) with an equal number of input dimensions and source distributions can be written as a PoE model


$$P(x|\{w_j\}) = |\det(W)| \prod_{j=1}^M p_j \left(\sum_i w_{ji} x_i \right)$$

- Choosing the heavy tailed Student-T distributions as the experts one obtains the general form of the "Products of Student-T" distribution (PoT)

$$P(x, h) = \frac{1}{Z} \prod_{j=1}^M \exp \left(-h_j \left[1 + \frac{1}{2} \left(\sum_i w_{ji} x_i \right)^2 \right] + (1 - \alpha_j) \log h_j \right)$$

Relationships to Others

- Products of Hidden Markov Models
- Relationship to Boosting
- Relationship to Analysis-by-Synthesis
-



Thanks!