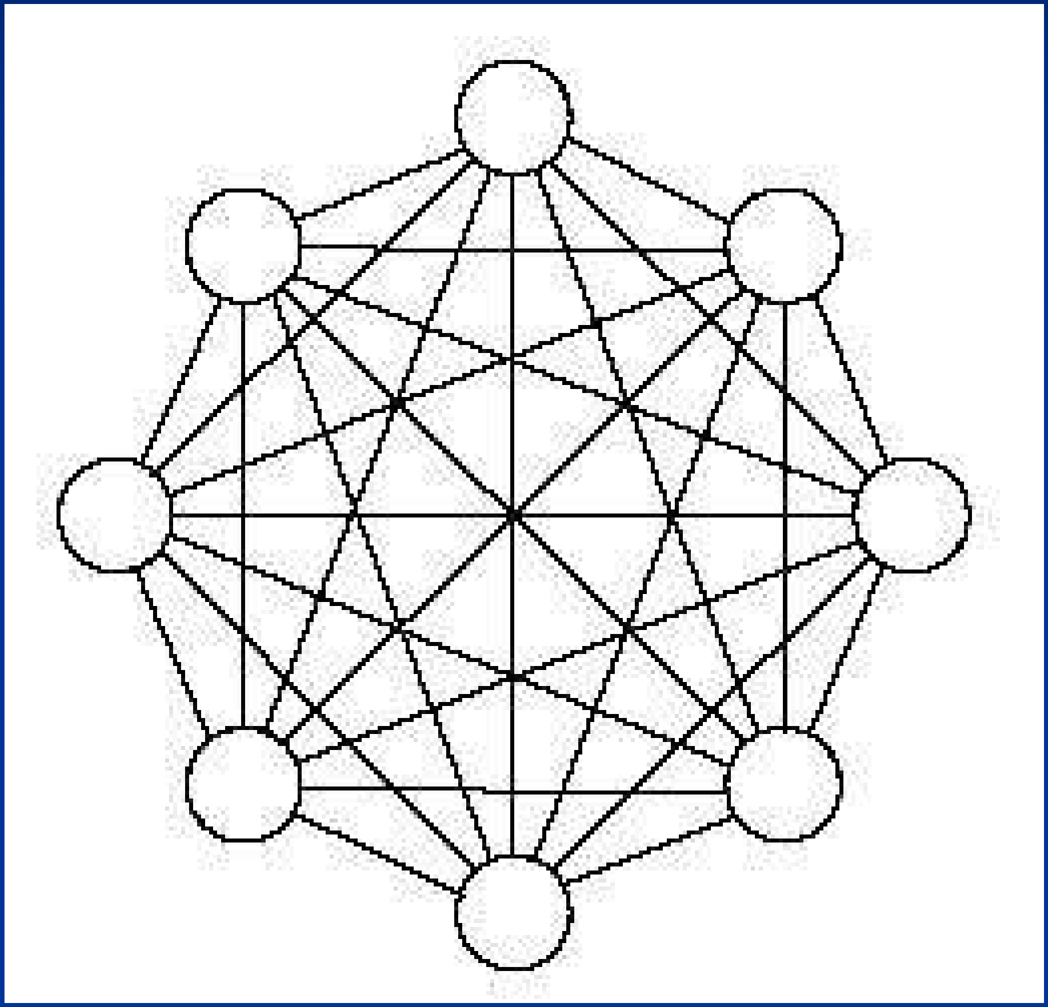
Introduction

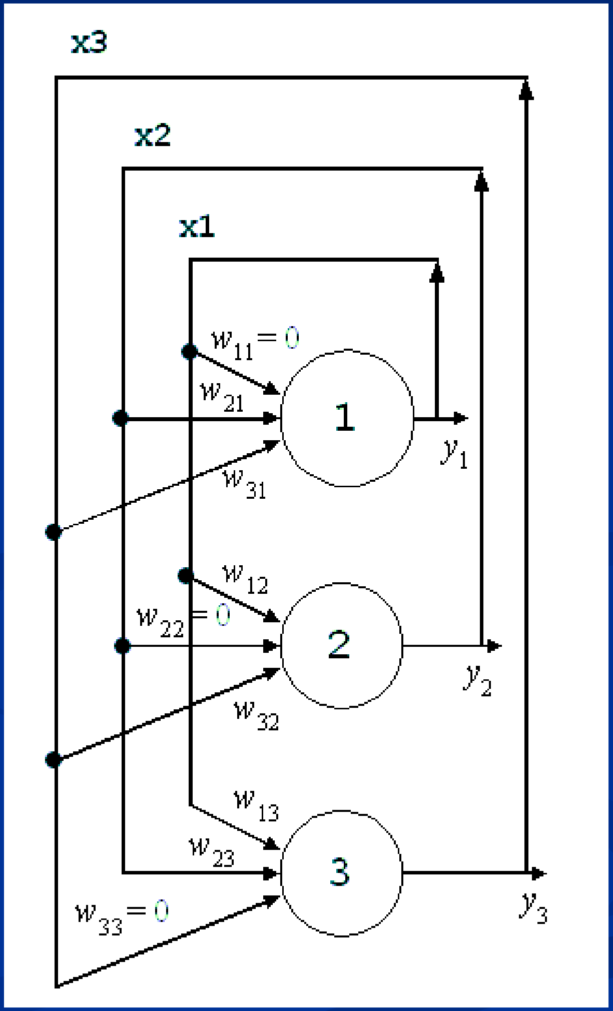
- "Boltzmann" invokes Stat Mech
- Ancestor is Hopfields Network
- Dynamics in terms of MCMC
- Learning
- Restricted Boltzmann Machines
- Example – Data Dimensionality Reduction

Boltzmann’s Machine

- Won’t win any popularity contest anytime soon
- NN (Neural Network) with intricate relationship to Stat Physics, arose from Hopfield’s NN (Hinton and Sejnowski, 1983)
- Locality of Learning Rule (Hebbianesque) + Generative Model (unsupervised learning), therefore more biologically plausible than back prop MLP
- Feedback + Dynamics
- Multiple Layers or “Deep” can be constructed using Restricted Boltzmann’s Machine (RBM)
- Hidden nodes and conditional inferencing via “Clamping”
- Relatively Slow

Visualizing Hopfield's Network





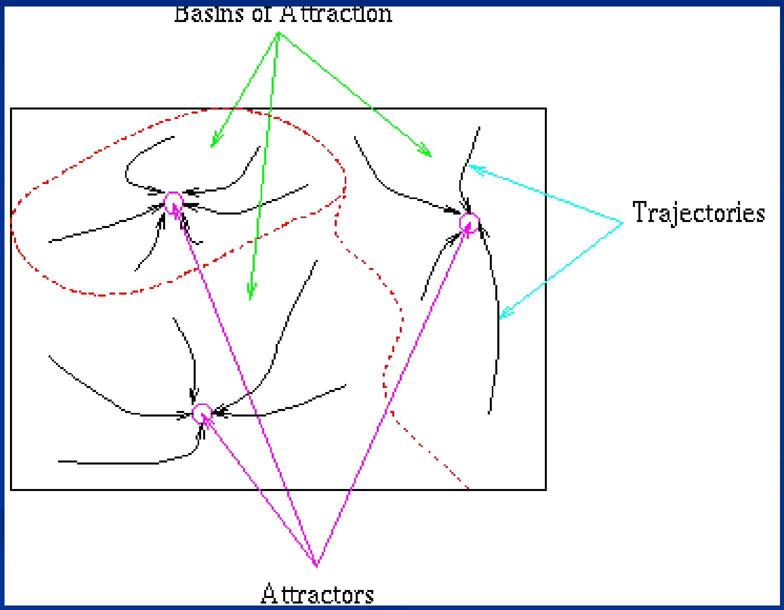
Hopfield's Network

- Equivalent to Spin Glass model, It is 2nd order energy network
- Ising Model is a specific form of Hopfield's network with only local neighborhood connections
- N binary nodes:

  $S_1 \ldots S_n,\ S_i = 0\ or\ 1\ W_{ij} = W_{ji},\ W_{ii} = 0$
- Weights are symmetrical, node can either be 0 or 1, network is fully connected, symmetry allows for local decision during every updates
- Energy function:

$$E = -\sum_{i<j} W_{ij} S_i S_j + \sum_i \theta_i S_i$$

- Hopfield's Net is an energy network, with dynamics implemented via asynchronous node activation, the dynamics makes it an recurrent attractor network



An Attractor Network

- Autoassociative/content addressable memory, categorization, noise suppression (Eliasmith, 2007)
- Dynamics: $S_i = \begin{cases} 1 & \sum_j W_{ij} S_j - \theta_i > 0 \\ 0 & else \end{cases}$
- By Construction, each update reduces global energy E

$$\Delta E_i = \sum_j W_{ij} S_j - \theta_i$$

- Fixed points in state/phase space are local minimums
- Each update can be seen as gradient descent in axial direction
- Often gets stuck in local min, but that's what we want for auto-associative memory, where each local minimum represent a memory!

Hopfield Learning

- For N training data configurations, use gradient descent

$$E_{total} = \sum_{k=1}^{N} E_k \qquad E_k = -\sum_{i<j} W_{ij} S_i^k S_j^k$$

$$\frac{\partial E_{total}}{\partial W_{ij}} = \sum_{k=1}^{N} \frac{\partial}{\partial W_{ij}} \left(-\sum_{i<j} W_{ij} S_i^k S_j^k \right) = -\sum_{k=1}^{N} \sum_{i<j} S_i^k S_j^k$$

$$\frac{\partial E_{total}}{\partial W_{ij}} = -\langle S_i S_j \rangle_k$$

$$\Delta W_{ij} = -\eta \frac{\partial E_{total}}{\partial W_{ij}} = \eta \langle S_i S_j \rangle_k$$

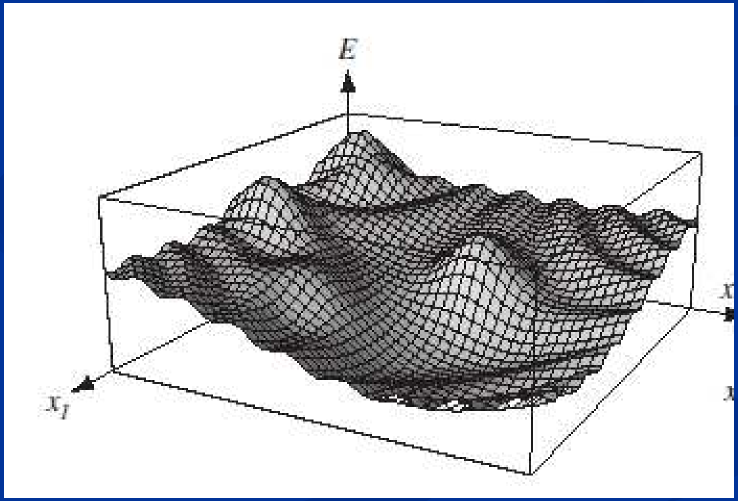
Same as Hebbian Learning Rule

What about Global Optimization?

- From optimization point of view, how do we find global min of

$$E = -\frac{1}{2}\sum_{i,j} W_{ij} S_i S_j$$

- The "bias node" can be eliminated assuming an extra node which is constantly on and connects to all other nodes with weights=bias



(Duda et al, 2001)

Annealing

- Physical annealing (slow cooling) is an analogy from metallurgy, for finding a low-energy configuration for systems of atoms of an alloy or many magnets. The system is heated repeatedly during cooling schedule, therefore allowing for configurations of higher energy at certain times during the annealing schedule.
- Simulated Annealing has 2 types:
- 1) gives the ability for escaping out of local minima, relationship to gradient descent optimization (Kirkpatrick et al, 1983)
- 2) in the context of BM simulation and MCMC sampling. In this case annealing leads faster to equilibrium and helps passage from ridges of high energy (low probability) in state space (Neal, 1993)

Simulated Annealing (Type I)

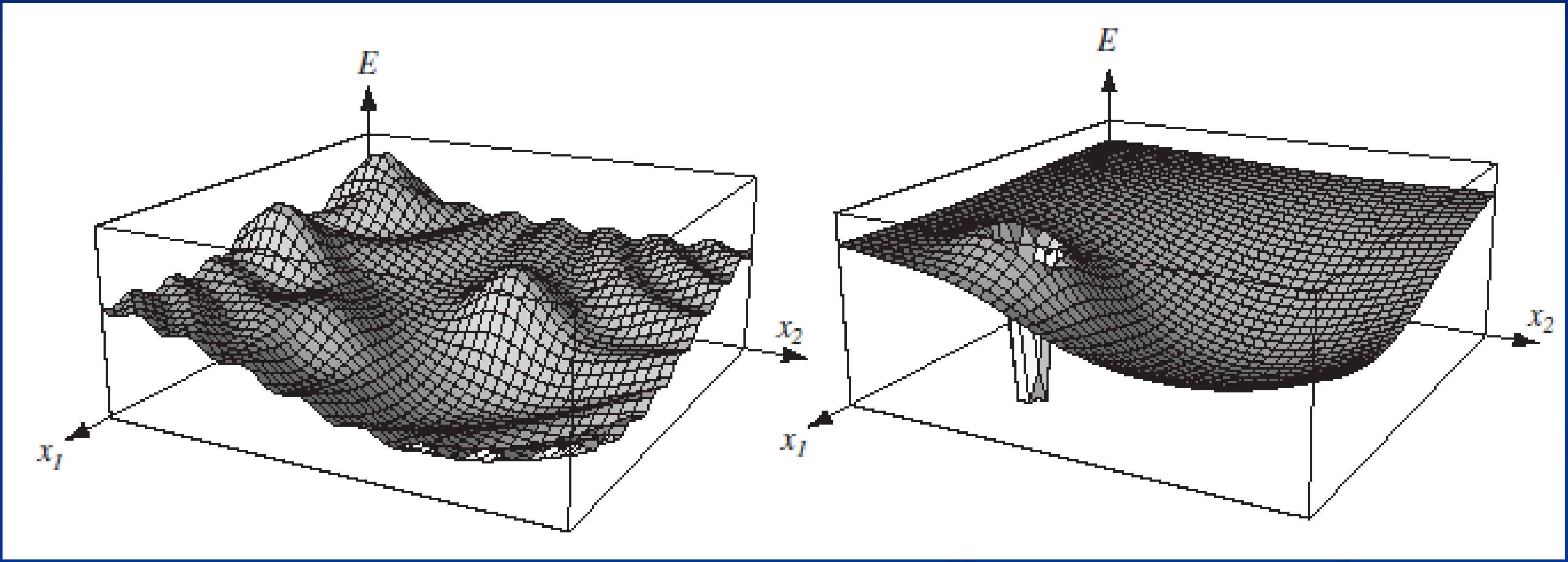
Simple Simulated Annealing Pseudocode

```
//x is a state vector
//t_max is the max # of iteration
//T[i] is the annealing schedule, T[t_max] = 0
//E(x) is the energy of state x,
//and the function trying to optimize

vector<float> x = rand_init(x);
for (int i = 0; i < t_max; ++i)
{
   x1 = rand_perturb(x, rho);
   if (E(x1) < E(x))
       x = x1;
   else if (exp(-(E(x1)-E(x))/T[i]) > rand(0,1))
       x = x1;
}
```

As T[i] -> 0, Pr(else if case = true) -> 0

Simulated Annealing (Type I)



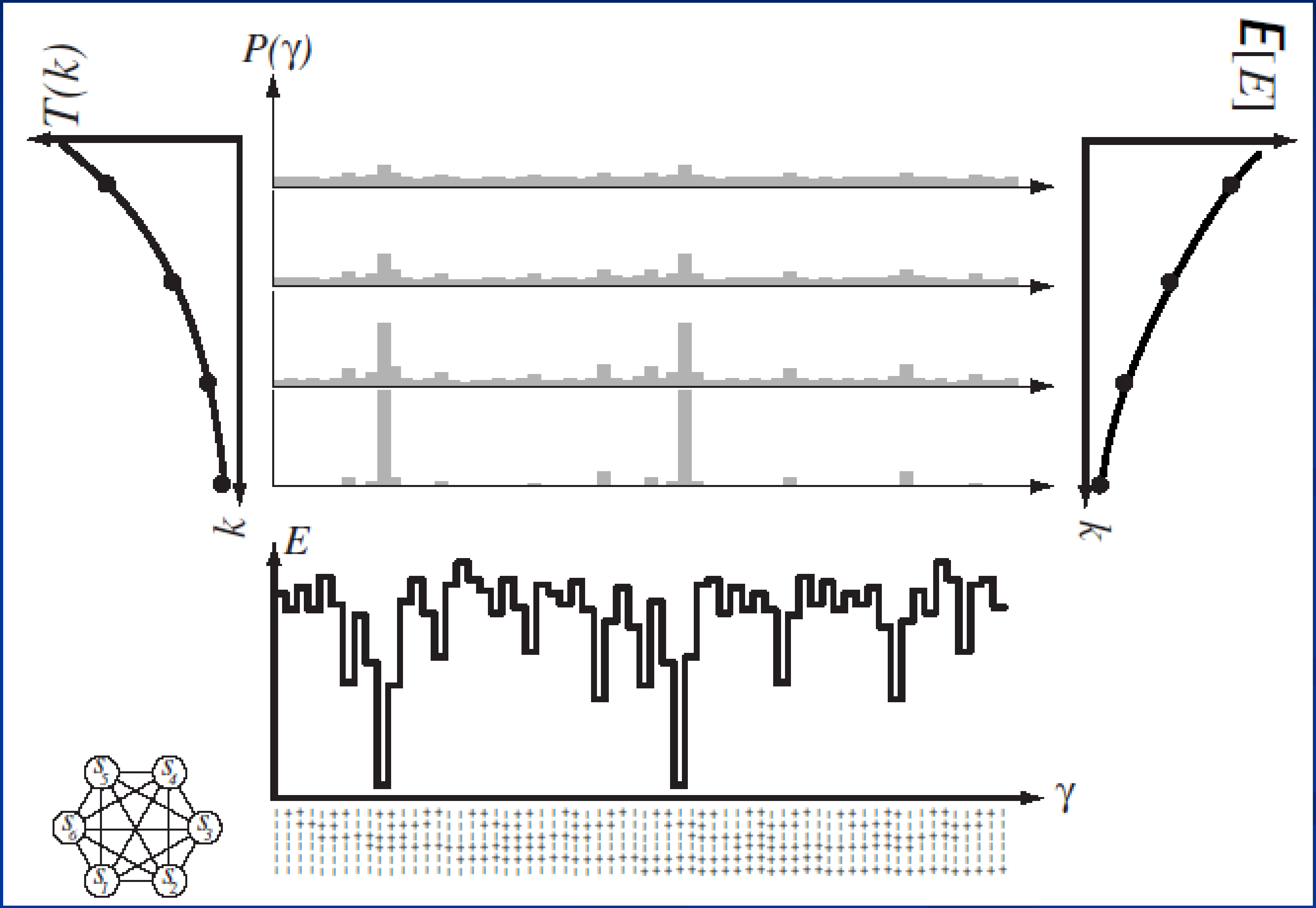
(Duda et al, 2001)

SA works well in the left energy landscape but not in the right energy landscape

Simulated Annealing (Type II)

- Kirkpatrick et al. formulated SA as sampling from the canonical distribution of a system at T=0, where probability is concentrated at states with minimum energy, therefore an optimization method.
- For Probability Inference, want to sample from a canonical distribution defined via an energy function that reproduces the pdf when T=1. In this case, SA can be used for easier reach of equilibrium. Such is the case with BM. (Ackley et al, 1985)
- E.g. If the state space of a certain distribution contains high energy boundaries, convergence of MCMC sampling (Metropolis or Gibbs) or reaching an thermodynamic equilibrium (Stat Mech formulation) would take much longer.

What happens to a Distribution?

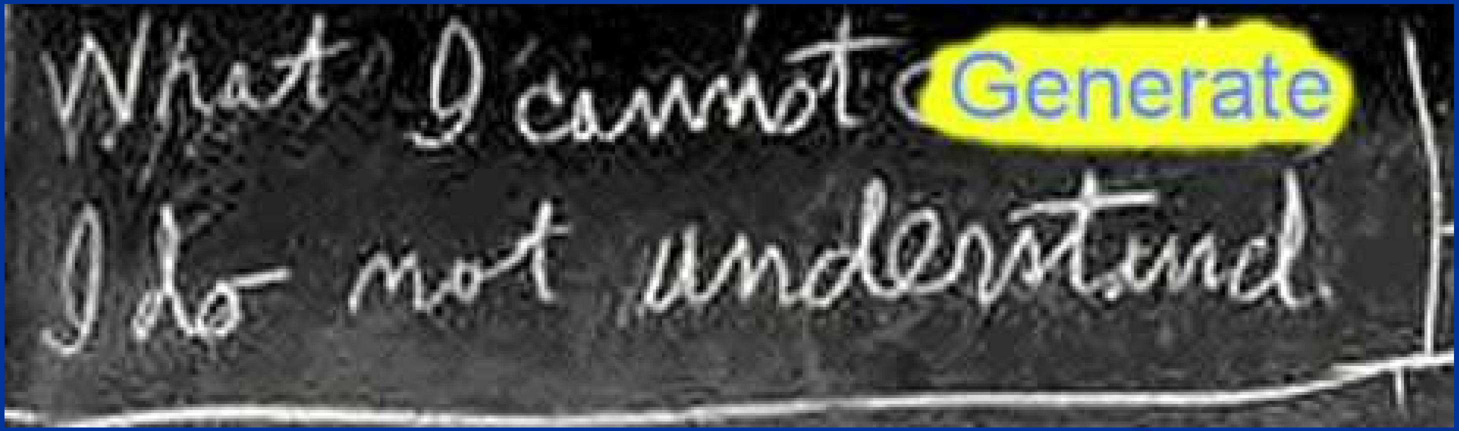


(Duda et al, 2001)

Boltzmann's Machine

- Paradigm Shift – We want to model the statistics of the input data
- This is known as a generative model: it is capable of generating the same distribution as the training input data on its own
- BM accomplish this by using probabilistic neurons
- $E[\alpha|\beta]$ easily computed via clamping!

A Boltzmann Machinist's Black Board



What I cannot Generate
I do not understand.

Boltzmann's Machine

$$E = -\frac{1}{2}\sum_{i,j} W_{ij} S_j S_i$$

$$Pr(\gamma) = \exp^{-E(\gamma)/T} / Z(T)$$

Boltzmann's Distribution

$$Z(T) = \sum_{\beta} \exp^{-E(\beta)/T}$$

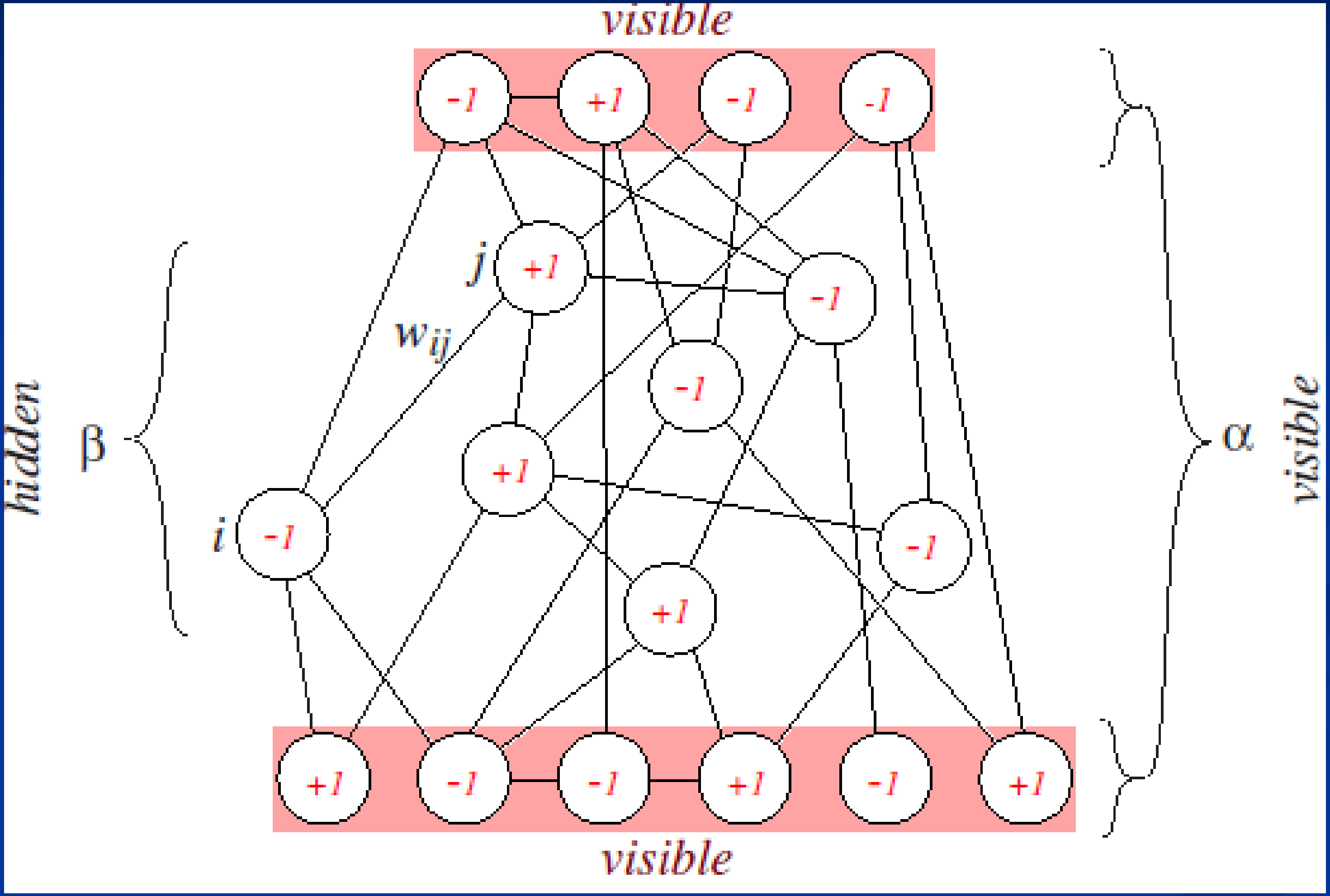
$$\frac{P_\alpha}{P_\beta} = \exp^{-(E_\alpha - E_\beta)/T}$$

γ, β – Global Configuration States

$Z(T)$ – Partition Function

Each global state has an associated energy and also a probability according to the Boltzmann's Distribution

Visualizing BM



(Duda et al, 2001)

Notice that the hidden units are now possible with BM, which allows for

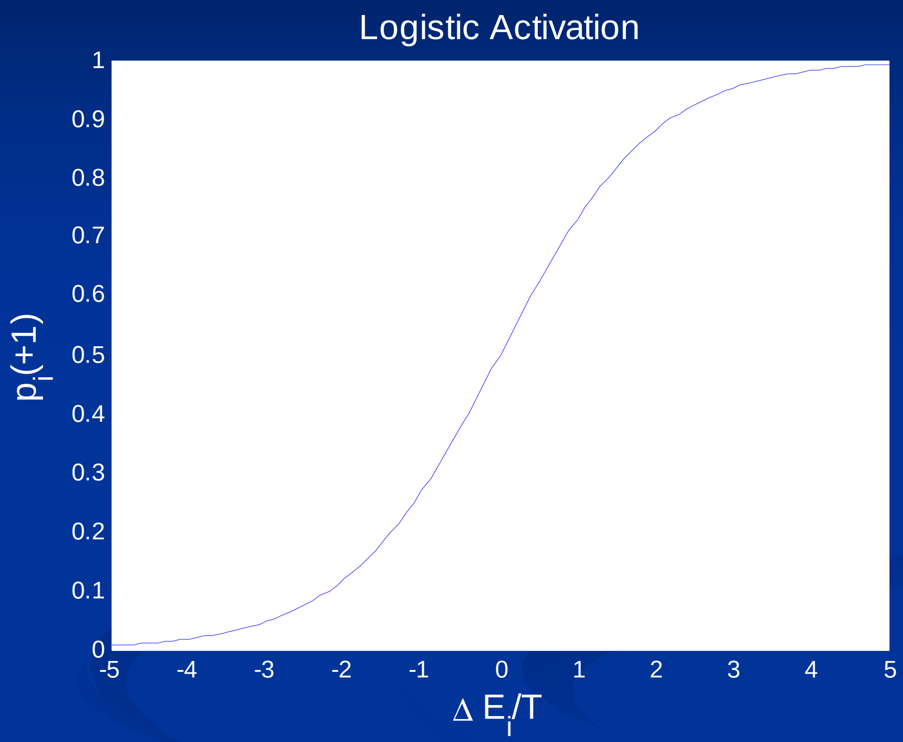
• a richer representation of Pr – higher order regularities

• think of it as unsupervised feature extraction

• increases the capacity of the network (big problem for Hopfield's Network ~.15d)

Stochastic Boltzmann's Machine

- Update Rule (Stochastic, unlike Hopfields):



$$p_i(+1) = \frac{1}{1 + \exp^{-\Delta E_i/T}}$$

$$\Delta E_i = E(S_i = 0) - E(S_i = +1) = \sum_j W_{ij} S_j$$

Note: the probability of the i-th neuron firing is monotonically increasing with its activation. Only possible due to the symmetric W

What the?

Well known NNs are
Input->Bl[redacted]ox->Output
(MLP for pattern recognition)
or
Noisy Data->Bl[redacted]ox->Memory Item
(Autoassociative Memory)
or
x->Bla[redacted]x->f(x)
(RBF for regression)
or
x->Encoding->Transformation->Decoding->f(x)
(NEF)

Where does binary neurons firing probabilistically fit into all this?

MCMC Gibbs Sampler!

- It turns out the update rule for BM is simply a Gibbs Sampler for the distribution defined by $Pr(\gamma)$ $$Pr(\gamma) = \exp^{-E(\gamma)/T} / Z(T)$$
- Gibbs Sampler (Geman and Geman, 1984) works well when the domain of the variables are small and finite, or that the conditional distributions are parametric and easy to sample from (Neal, 1993)
- Gibbs Sampler: replaces each component of a random vector with a value selected from its distribution conditional on other components remaining the same

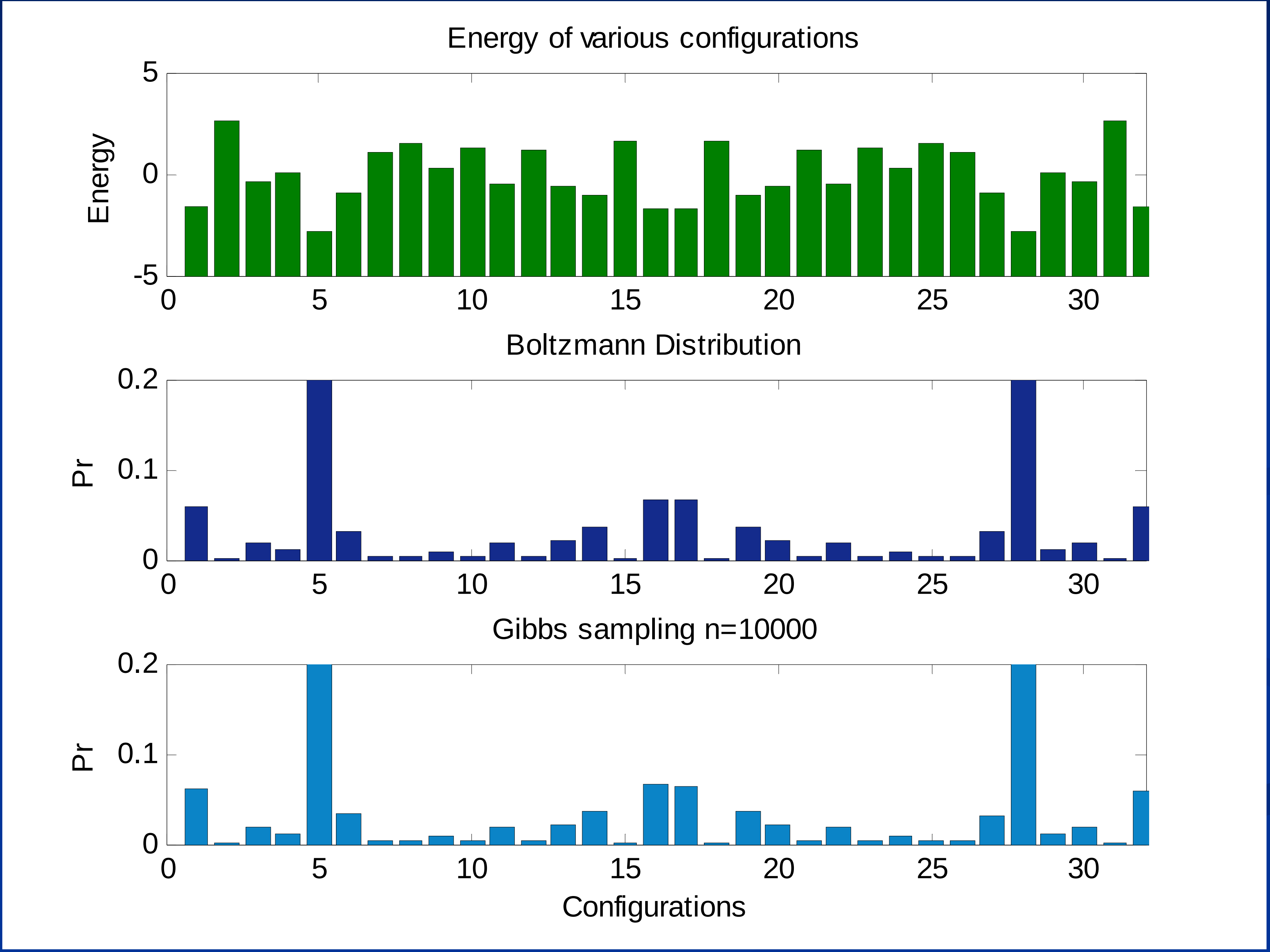
$$P(s_i = +1 |\ \{s_j : j \neq i\}) = \frac{P(s_i = +1,\ s_j : j \neq i)}{P(s_i = +1,\ s_j : j \neq i) + P(s_i = -1,\ s_j : j \neq i)}$$

$$P(s_i = +1 |\ \{s_j : j \neq i\}) = \frac{e^{-E_\alpha/T}}{e^{-E_\alpha/T} + e^{-E_\beta/T}}$$

$$P(s_i = +1 |\ \{s_j : j \neq i\}) = \frac{1}{1 + e^{-(E_\beta - E_\alpha)/T}}$$

Note: This is also same as The Metropolis with Boltzmann's acceptance function

Example


Energy of various configurations
Energy
5
0
-5
0
5
10
15
20
25
30
Boltzmann Distribution
Pr
0.2
0.1
0
Gibbs sampling n=10000
Pr
0.2
0.1
0
Configurations

Learning w/o Hidden Nodes

- If the full Pr(.) for entire configurations of all nodes are known, then learning is easy
- Let the Pr(.) unclamped by the environment (BM running freely) be: P_α^- $\quad P_\alpha^- = e^{-E_\alpha/T}/Z$

$$\frac{\partial\, ln(P_\alpha^-)}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_{i<j} w_{ij} s_i^\alpha s_j^\alpha / T - \frac{1}{Z}\frac{\partial}{\partial w_{ij}} \sum_\gamma e^{-E_\gamma/T}$$

$$\frac{\partial\, ln(P_\alpha^-)}{\partial w_{ij}} = \frac{1}{T}(s_i^\alpha s_j^\alpha) - \sum_\gamma \frac{e^{-E_\gamma/T}}{Z}\frac{1}{T} s_i^\gamma s_j^\gamma$$

$$\frac{\partial\, ln(P_\alpha^-)}{\partial w_{ij}} = \frac{1}{T}\left(s_i^\alpha s_j^\alpha - \sum_\gamma P_\gamma^- s_i^\gamma s_j^\gamma \right)$$

Note: the second term is just the probability of finding the ith and jth neuron on at same time

Learning w/ Hidden Nodes

- The previous equation and Hopfield's learning rule are inappropriate with the inclusion of hidden nodes, since their states as well as probability distribution are unknown
- The environment/teacher can only present the states and the probability distribution of the visible nodes
- The difficult problem becomes learning how to use the hidden nodes such that the visible nodes exhibits the required probabilities
- Need to adapt all connection weights in the network when given only the Pr distribution (or training samples) over the visible nodes

Learning w/ Hidden Nodes

- Minimize relative entropy of the environment clamped data and the model generated data
- We also assume there are two phases phase + and phase -;
- During phase +, the environment clamps a particular configuration over the visible nodes long enough to reach thermo equilibrium
- During phase -, the visible units are unclamped and the configurations are freely generated by the network

$$D_{KL}(P^+||P^-) = \sum_\alpha P^+(V_\alpha)\ ln\frac{P^+(V_\alpha)}{P^-(V_\alpha)}$$

+ve; zero iff both distributions are the same

$$\frac{\partial D_{KL}}{\partial w_{ij}} = -\sum_\alpha \frac{P^+(V_\alpha)}{P^-(V_\alpha)} \frac{\partial P^-(V_\alpha)}{\partial w_{ij}}$$

Vision Analogy

$$P^-(V_\alpha) = \sum_\beta P^-(V_\alpha \wedge H_\beta) = \frac{\sum_\beta \exp^{-E_{\alpha\beta}/T}}{\sum_{\lambda\mu} \exp^{-E_{\lambda\mu}/T}}$$

Marginalization over beta

$$E_{\alpha\beta} = -\sum_{i<j} w_{ij} s_i^{\alpha\beta} s_j^{\alpha\beta}$$

Energy of a given joint configuration

$$\frac{\partial e^{-E_{\alpha\beta}/T}}{\partial w_{ij}} = \frac{1}{T} s_i^{\alpha\beta} s_j^{\alpha\beta} e^{-E_{\alpha\beta}/T}$$

Learning w/ Hidden Nodes

$$\frac{\partial P^-(V_\alpha)}{\partial w_{ij}} = \frac{\frac{1}{T}\sum_\beta e^{-E_{\alpha\beta}/T} s_i^{\alpha\beta} s_j^{\alpha\beta}}{\sum_{\alpha\beta} e^{-E_{\alpha\beta}/T}} - \frac{\sum_\beta e^{-E_{\alpha\beta}/T} \sum_{\lambda\mu} e^{-E_{\lambda\mu}/T} s_i^{\lambda\mu} s_j^{\lambda\mu}}{\left(\sum_{\lambda\mu} e^{-E_{\lambda\mu}/T}\right)^2}$$

$$= \frac{1}{T}\left[\sum_\beta P^-(V_\alpha \wedge H_\beta) s_i^{\alpha\beta} s_j^{\alpha\beta} - P^-(V_\alpha)\sum_{\lambda\mu} P^-(V_\lambda \wedge H_\mu) s_i^{\lambda\mu} s_j^{\lambda\mu}\right]$$

This is the change of model generated probability of visible state alpha as w_ij changes

$$\frac{\partial D_{KL}}{\partial w_{ij}} = -\frac{1}{T}\left[\sum_\alpha \frac{P^+(V_\alpha)}{P^-(V_\alpha)} \sum_\beta P^-(V_\alpha \wedge H_\beta) s_i^{\alpha\beta} s_j^{\alpha\beta} - \sum_\alpha \frac{P^+(V_\alpha)}{P^-(V_\alpha)} P^-(V_\alpha) \sum_{\lambda\mu} P^-(V_\lambda \wedge H_\mu) s_i^{\lambda\mu} s_j^{\lambda\mu}\right]$$

$$P^+(V_\alpha \wedge H_\beta) = P^+(H_\beta | V_\alpha) P^+(V_\alpha) \qquad P^-(V_\alpha \wedge H_\beta) = P^-(H_\beta | V_\alpha) P^-(V_\alpha)$$

$$P^-(H_\beta | V_\alpha) = P^+(H_\beta | V_\alpha)$$

The Pr of hidden states at equilibrium must be the same given the visible state whether or not that visible state is reached by environment clamping or free-running!

Learning w/ Hidden Nodes

$$P^-(V_\alpha \wedge H_\beta)\frac{P^+(V_\alpha)}{P^-(V_\alpha)} = P^+(V_\alpha \wedge H_\beta)$$

$$\sum_\alpha P^+(V_\alpha) = 1$$

$$\frac{\partial D_{KL}}{\partial w_{ij}} = -\frac{1}{T}[p_{ij}^+ - p_{ij}^-]$$

Works for any pair of nodes, hidden or visible!

Where

$$p_{ij}^+ \equiv \sum_{\alpha\beta} P^+(V_\alpha \wedge H_\beta) s_i^{\alpha\beta} s_j^{\alpha\beta}$$

Clamped by the environment (Learning)

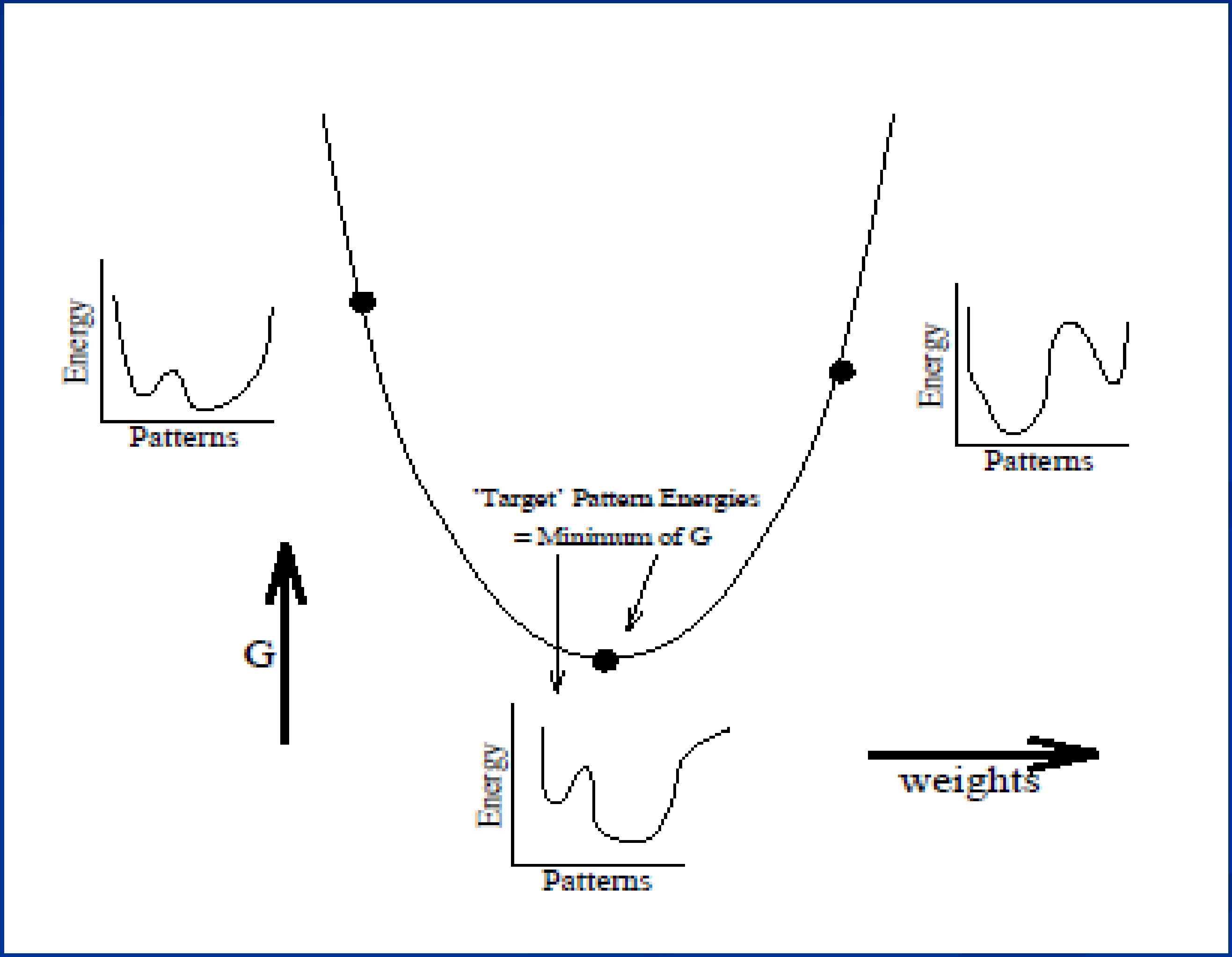
$$p_{ij}^- \equiv \sum_{\lambda\mu} P^-(V_\lambda \wedge H_\mu) s_i^{\lambda\mu} s_j^{\lambda\mu}$$

Generated freely (Unlearning)

$$P^+(V_\alpha \wedge H_\beta)$$

This represents the Pr of a certain global configuration when the environment set the marginal $P(V_\alpha)$, and the network generates $P^-(H_\beta | V_\alpha)$

Learning w/ Hidden Nodes



(Roweis, [6])

Learning w/ Hidden Nodes

- The training algorithm alternates between two phases, phase+ and phase-
- Phase+ is known as the learning phase
- Phase- is known as the unlearning phase
- Hinton and Sejnowski came up with the name “unlearning” from Crick and Mitchson (1983) where REM sleep was proposed as when reverse learning might occurr!
- In BM, it is minimizing an info theoretic measure of the difference b/t environment and model distribution
- Also look at as Hebbian learning w/ +ve weights when info from environment is captured
- Hebbian learning w/ -ve weights when system randomly generates samples from Boltzmann’s distribution

Phase+ (Learning Phase)

- Visible nodes are clamped by a particular pattern.
- The network is allowed to reach thermal equilibrium as temperature is reduced to T_0(e.g. 10), following an annealing schedule.
- The hidden nodes are the only ones changing its states. Thermo equilibrium == MCMC convergence; e.g.$\langle s_i \rangle$ becomes stationary
- This simulated annealing is type II and speeds up MCMC convergence to the distribution $P^{-}(H_\beta | V_\alpha)$
- Upon convergence (not before), more iterations (e.g. 10) are run in which the correlation statistics are collected $\langle s_i s_j \rangle_{data}$
- The N (e.g. 20) runs are performed as different patterns are clamped to the input of the network, the $\langle s_i s_j \rangle_{data}$ averaged over these trials are p_{ij}^{+}

Phase- (Unlearning Phase)

- None of the nodes are clamped and the annealing process is the same as phase+
- The statistics collected is p_{ij}^{-}
- The set of 2N annealing process are called 1 sweep
- For each sweep, the correlation for all connections i,j are collected
- Weight update according to gradient descent on the KL-divergence
- Weight decay [w(n+1) = w(n)*.9995] can also be used to serve as a form of regularization and helps convergence to equilibrium

Nested Loops

```
while $D_{KL} > \epsilon$ , calculate $\Delta W_{ij}$
    Phase+, for all patterns $V_\alpha$
        Anneal to equilibrium or convergence
            Collect corre stat $\langle s_i s_j \rangle_{data}$

    Phase-, for all starting patterns $V_\alpha$
        Anneal to equilibrium or convergence
            Collect corre stat $\langle s_i s_j \rangle_{model}$
```

Really, really slow…
However, it's entirely biologically plausible, especially if REM sleep performs Phase- unlearning

Pros and Cons

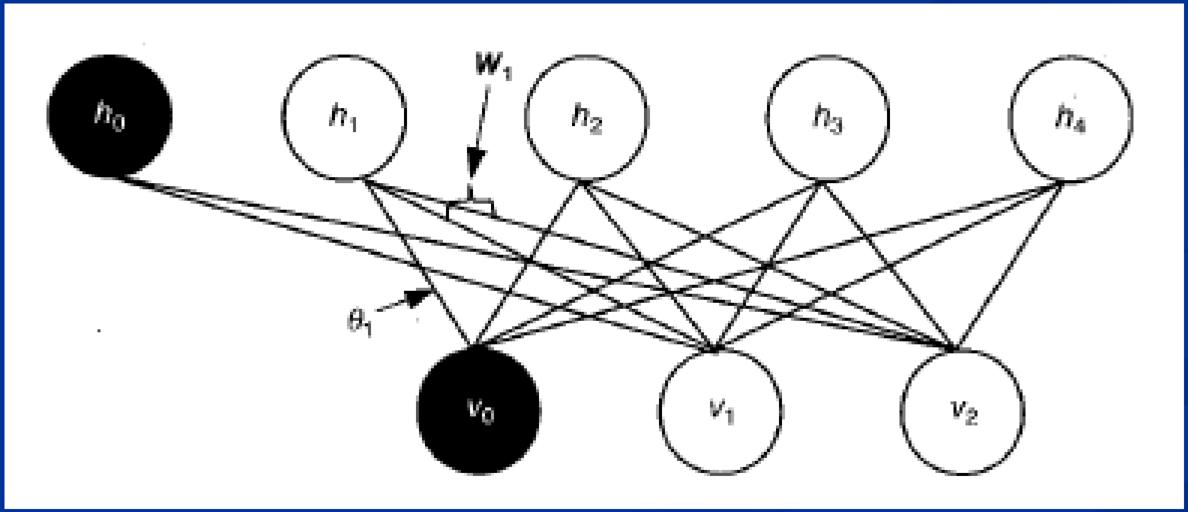
- Hidden nodes allows for encoding of higher than 2nd order regularities
- Represent probabilities *directly* via Gibbs sampler, trying to emulate the actual process
- Unlike other generative models which uses functions to represent distributions, e.g. PPCA
- Conditional inference via clamping: a BM can used for classification by learning the joint pdf of input and output; Inferencing takes place by clamping the input with test data and arriving at the MAP solution!
- Really Slow - O(hours -> days)

Extensions

- Meanfield approximation speeds up the learning process at a cost of accuracy (Petersen and Andersen, 1987)
- Units take on continuous values (Welling et. al., 2005)
- Rate coded BM (Teh and Hinton 2001): estimates the ‘firing rate’ by these binary stochastic nodes, rate represent continuous values
- Higher ordered BMs (Sejnowski, 1986)
- Many more … since BM is intricately related to MRF, which are part of undirected graphical models

Restricted Boltzmann Machine

- First introduced by (Smolensky, 1986), named "Harmoniums"
- 1 layer hidden, 1 layer visible; No hidden-hidden or visible-visible connections
- Hidden nodes are conditionally independent given visible nodes



(Chen. Murray. [11])

Restricted Boltzmann Machine

$$P(h1, h2|\vec{V}) = \frac{1}{Z} e^{\sum w_{1j} v_j h_1} e^{\sum w_{2j} v_j h_2} = P(h1|\vec{V}) P(h2|\vec{V})$$

$$\frac{\partial D_{KL}}{\partial w_{ij}} = -\frac{1}{T}[p_{ij}^{+} - p_{ij}^{-}] = -\frac{1}{T}[\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}]$$

$$\langle v_i h_j \rangle_{data} = \langle v_i h_j \rangle_{data, h_k : k=1 \ to \ n}$$

$$\langle v_i h_j \rangle_{data} = \langle v_i h_j \rangle_{data, h_j}$$

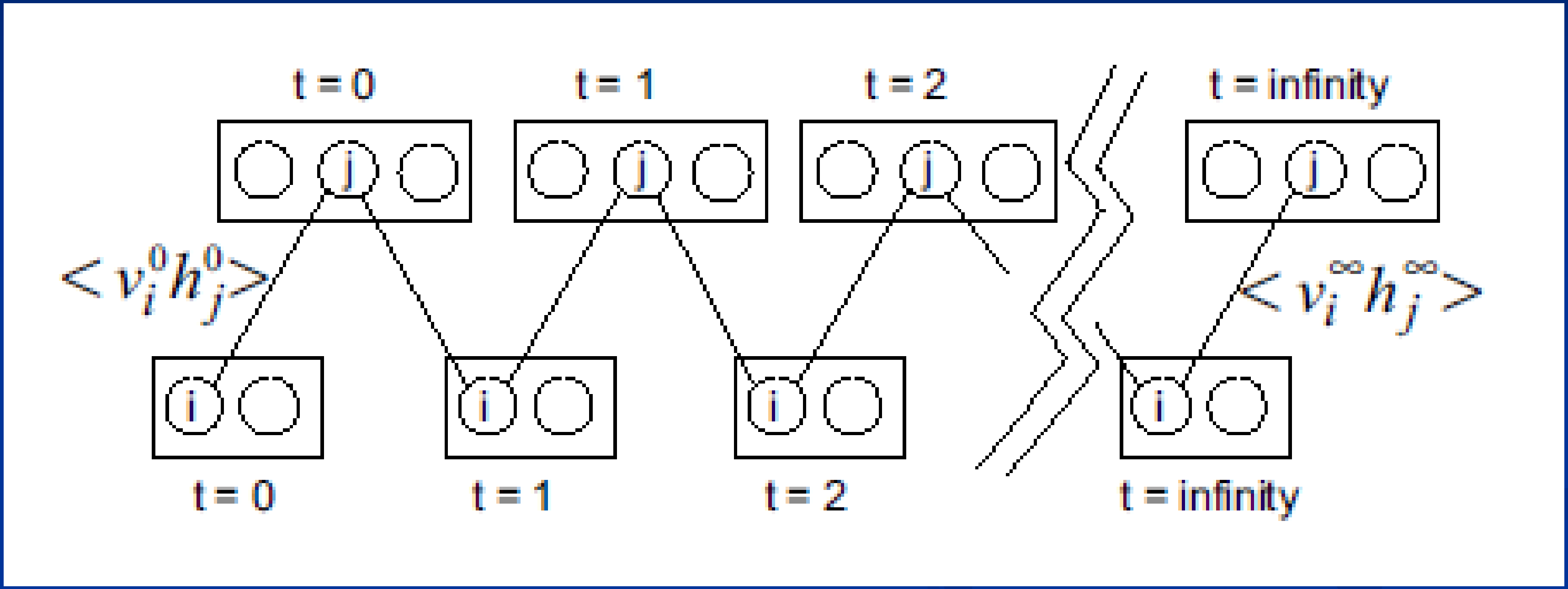
Because of independence of hidden nodes

$$\langle v_i h_j \rangle_{data} = \langle v_i \times [1 \times P(h_j = +1|\vec{V}) + 0 \times P(h_j = 0|\vec{V})] \rangle_{data}$$

Easy to calculate

$\langle v_i h_j \rangle_{model}$ still needs Gibbs sampling

Restricted Boltzmann Machine



(Hinton, 2001)

- Start at a random state in one of the layers
- Perform alternate Gibbs sampling
- All the nodes in one layer is updated in parallel given the other layer
- Repeated until the distribution reaches equilibrium or convergence

Contrastive Divergence Learning

- (Hinton, 2001) To speed things up

Instead of $\Delta w_{ij} = \eta\big(\langle v_i h_j\rangle_{data} - \langle v_i h_j\rangle_{model}\big)$

$$\Delta w_{ij} = \eta\big(\langle v_i h_j\rangle_{data} - \langle v_i h_j\rangle_{recons}\big)$$

$\langle v_i h_j\rangle_{model}$ is replaced with $\langle v_i h_j\rangle_{recons}$

1. Starting with the visible units clamped to the data, update all the hidden units in parallel
2. Holding the hidden units constant, update all the visible units – "reconstruction"
3. Update the hidden units again

Contrastive Divergence Learning

Let $P^{+} \equiv Q^{0}$ and $P^{-} \equiv Q^{\infty}$

Think Q^{0} as a Markov Chain with the data distribution at time step 0

$$D_{KL} \equiv Q^{0} || Q^{\infty}$$

$$CD \equiv (Q^{0} || Q^{\infty} - Q^{1} || Q^{\infty})$$

Hence the name

$$\Delta w_{ij} = \eta \big(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{recons} \big) \approx \frac{\partial CD}{\partial w_{ij}}$$

Intuition: To minimize the urges of the chain to wander away from initial distribution in the first step

Contrastive Divergence Learning

$Q^0 || Q^\infty \geq Q^1 || Q^\infty$ Since we'd be one step closer to equilibrium distribution

$Q^0 == Q^1$ implies $Q^0 == Q^\infty$ and the model would be perfect

It is reported to work pretty well and justifies the speed up in not computing Q^∞
(Hinton, 2002)

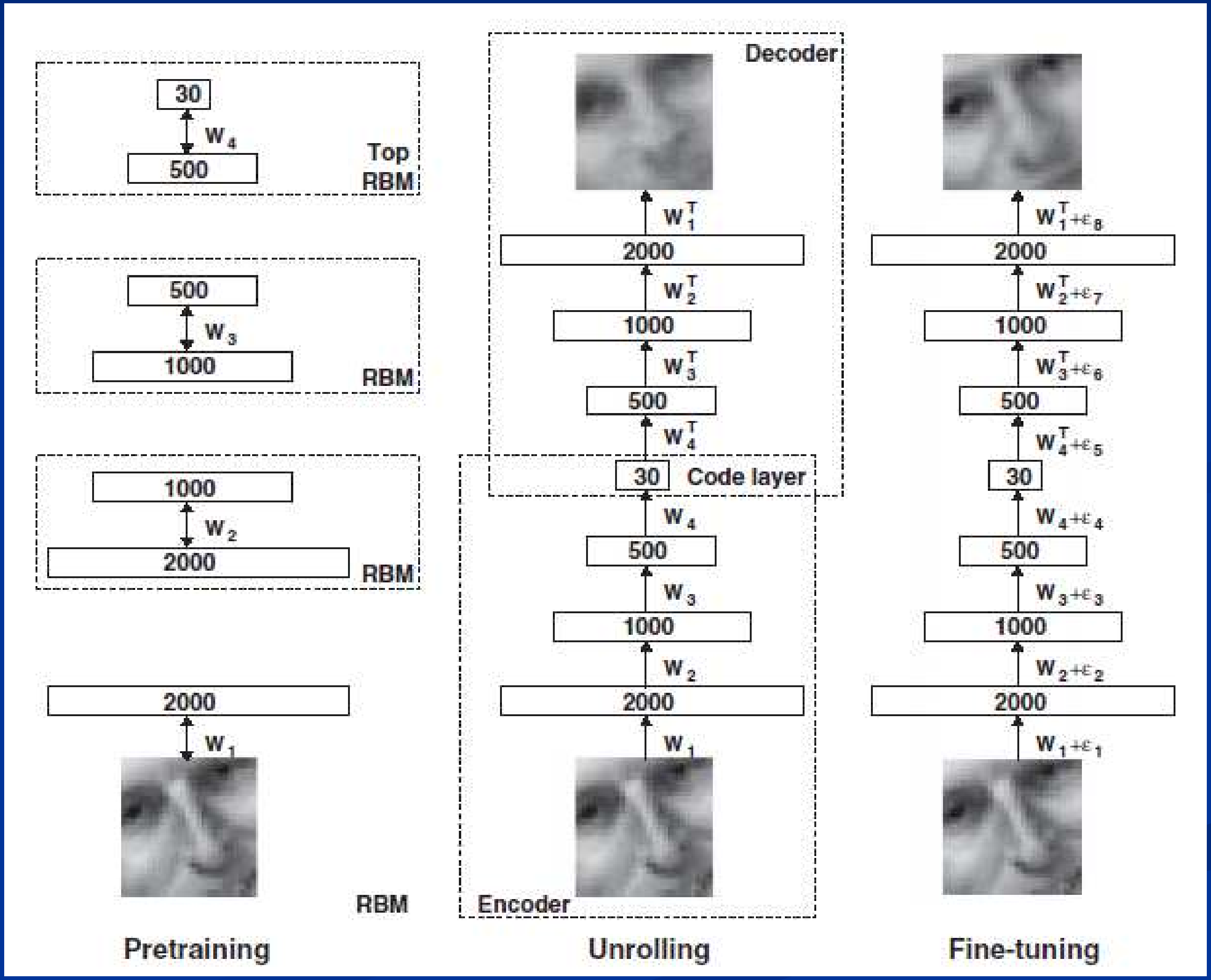
So we can learn 2 layer RBM, now let's learn more layers!

“Deep” Network

- Deep network composed of successive RBM layers can be stacked one on top of another
- The hidden layer of the bottom RBM serves as the visible layer of the top RBM
- After training the combined network is a multilayer generative network
- Learning is unsupervised, with the final level features typically much more useful for classification

Example – Dimension Reduction

- Autoencoders. Hinton and Salakhutdinov, Science, 2006.
- With 2-4 hidden layers, they are hard to train. Easily stuck in local minimum if initial weights are too large
- Want to find "good" weights before beginning gradient descent
- Use Deep RBM to find these good weights – Pretraining
- Progressively reveal low-dimensional and non-linear feature in data



(Hinton, 2006)

Dimension Reduction

- After pretraining, the RBMs are stacked to form an encoder and decoder network
- Stochastic nodes are replaced by deterministic probabilities
- Backprop is used to fine-tune the network for optimum reconstruction
- Visible nodes of the first layer RBM replaced with linear units with gaussian noise
- All other visible nodes had [0 1] continuous value and were set to the probability of the hidden nodes below
- All hidden nodes except the “code layer” are binary stochastic nodes
- The hidden nodes in the “code layer” took on value drawn from unit variance gaussian, mean determined by the input to it

Results

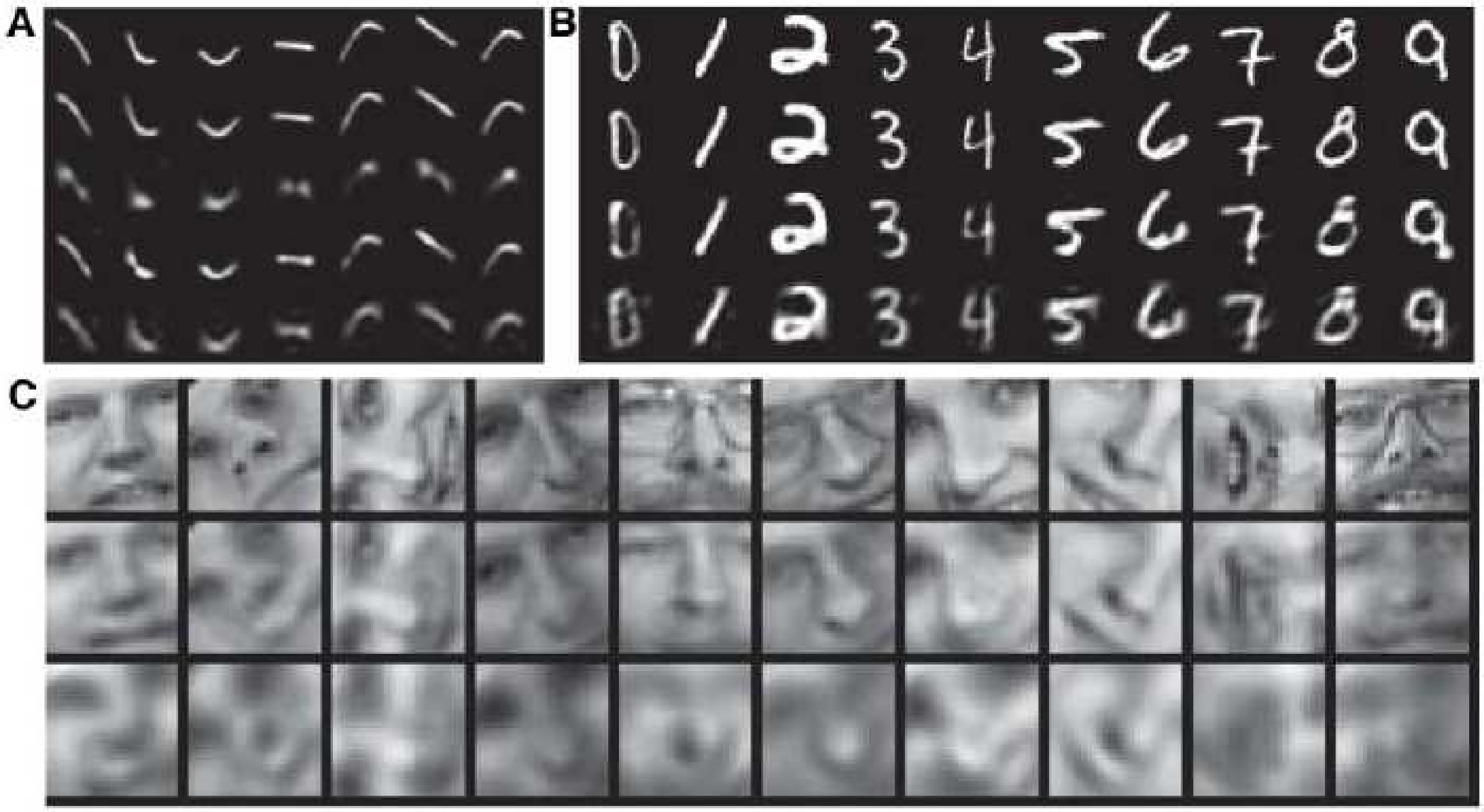


Fig. 2. (**A**) Top to bottom: Random samples of curves from the test data set; reconstructions produced by the six-dimensional deep autoencoder; reconstructions by "logistic PCA" (*8*) using six components; reconstructions by logistic PCA and standard PCA using 18 components. The average squared error per image for the last four rows is 1.44, 7.64, 2.45, 5.90. (**B**) Top to bottom: A random test image from each class; reconstructions by the 30-dimensional autoencoder; reconstructions by 30-dimensional logistic PCA and standard PCA. The average squared errors for the last three rows are 3.00, 8.01, and 13.87. (**C**) Top to bottom: Random samples from the test data set; reconstructions by the 30-dimensional autoencoder; reconstructions by 30-dimensional PCA. The average squared errors are 126 and 135.

(Hinton, 2006)

Thoughts

- Pretraining is useful because it make sure that most information of weights comes from modeling the data
- Unlike LLE and ISOMAP (Both also published in Science in 2000!), it can encode and also decode
- Reported 1.2% error rate, beating SVMs and Backprops (No mentioning of LeNet) on MNIST data set, using 784-500-500-2000-10 classificator
- Shares similar Encoder->Representation->Decoder semantic with NEF.
- The code layer could represent higher ventral stream areas such as the anterior inferior temporal lobe, or in the case of faces, the fusiform gyrus

Conclusion

- Hopfield's Net and BM intricately connected to Stat Mech
- Update in Hopfield's Net: gradient descent
- Update in BM: Gibbs sampling
- BM is a generative model and represents distributions *directly*
- BM is unsupervised, feedbacks, local learning **->** biologically plausible
- Slowness and ways to speedup using approximations
- Though old, the ideas from BM shows up in many recent graphical model researches

References

- Ackley et al. “A Learning Algorithm for Boltzmann Machines”. 1985
- Hinton, Sejnowski. “Learning and Relearning In Boltzmann Machines”. 1986
- Peterson. Andersen. “A mean field theory learning algorithm for neural networks”, 1987
- Neal. “Probability Inferencing using Markov chain Monte Carlo.” 1993
- Duda. Hart. Stork. “Pattern Classification”. 2001
- Roweis. “Boltzmann Machines”.
- Hinton. “Training Products of Experts by Minimizing Contrastive
- Divergence”. 2002
- Hinton. Salakhutdinov. “Reducing the Dimensionality of Data using Neural Networks”. 2006
- Hinton et al. “A Fast Learning Algorithm For Deep Belief Nets”. 2006
- Chen. Murry. “Continuous Restricted Boltzmann Machine with implementable training algorithm”
- Eliasmith. “Attractor Network”. Scholarpedia. 2007
- Hinton. “Boltzmann Machine”. Scholarpedia. 2007
- Salakhutdinov et al. “Restricted Boltzmann Machine for Collabrative Filtering”. 2007