CSC 2535: 2013

Lecture 3b

Approximate inference in Energy-Based Models

Geoffrey Hinton

Two types of density model

Stochastic generative model using directed acyclic graph (e.g. Bayes Net)

$$p(v) = \sum_h p(h)\, p(v \mid h)$$

Generation from model is easy

Inference can be hard

Learning is easy after inference

Energy-based models that associate an energy with each data vector

$$p(v) = \frac{\sum_h e^{-E(v,h)}}{\sum_{u,g} e^{-E(u,g)}}$$

Generation from model is hard

Inference can be easy

Is learning hard?

Using energies to define probabilities

- The probability of a joint configuration over both visible and hidden units depends on the energy of that joint configuration compared with the energy of all other joint configurations.

$$p(v,h) = \frac{e^{-E(v,h)}}{\underbrace{\sum_{u,g} e^{-E(u,g)}}_{\text{partition function}}}$$

- The probability of a configuration of the visible units is the sum of the probabilities of all the joint configurations that contain it.

$$p(v) = \frac{\sum_{h} e^{-E(v,h)}}{\sum_{u,g} e^{-E(u,g)}}$$

Density models

Directed models

Tractable posterior

mixture models, sparse bayes nets

factor analysis

Compute exact posterior

Intractable posterior

Densely connected DAG’s

Markov Chain Monte Carlo

or

Minimize variational free energy

Energy-Based Models

Stochastic hidden units

Full MCMC

If the posterior over hidden variables is tractable

Minimize contrastive divergence

Deterministic hidden units

Hybrid MCMC for the visible variables

&

Minimize contrastive divergence

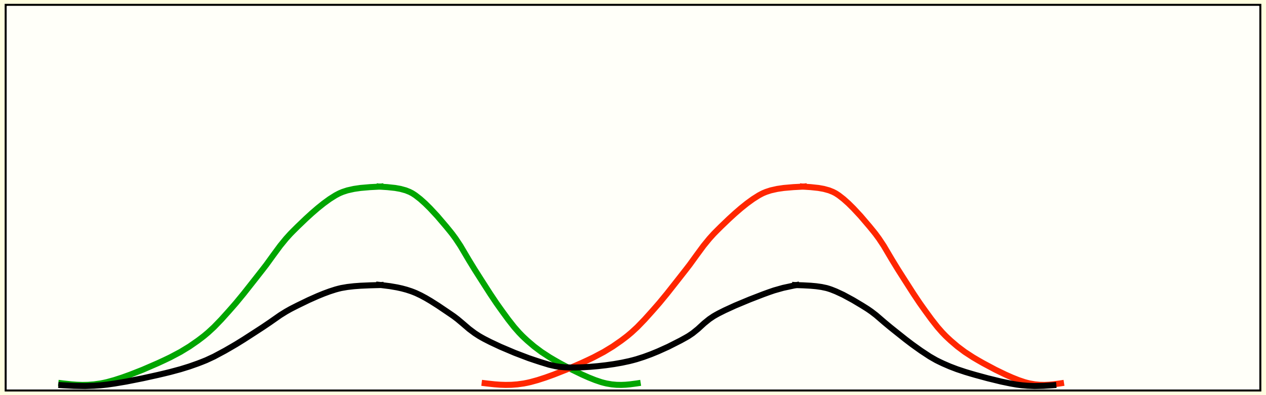
How to combine simple density models

- 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 can be much sharper than the individual distributions.
  - A nasty normalization term is needed to convert the product of the individual densities into a combined density.

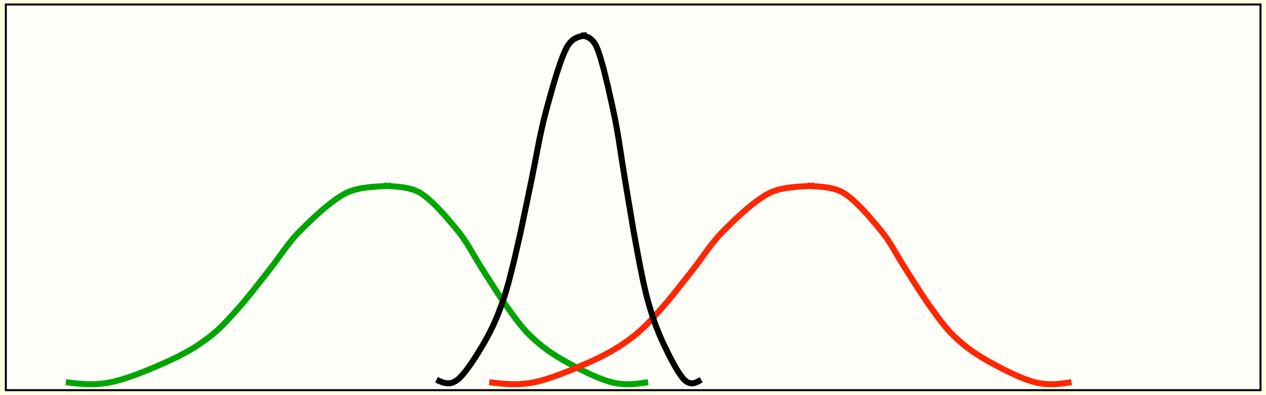
$$p(d) = \sum_m \overset{\text{mixing proportion}}{\pi_m} p_m(d)$$

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

A picture of the two combination methods



Mixture model: Scale each distribution down and add them together



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

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

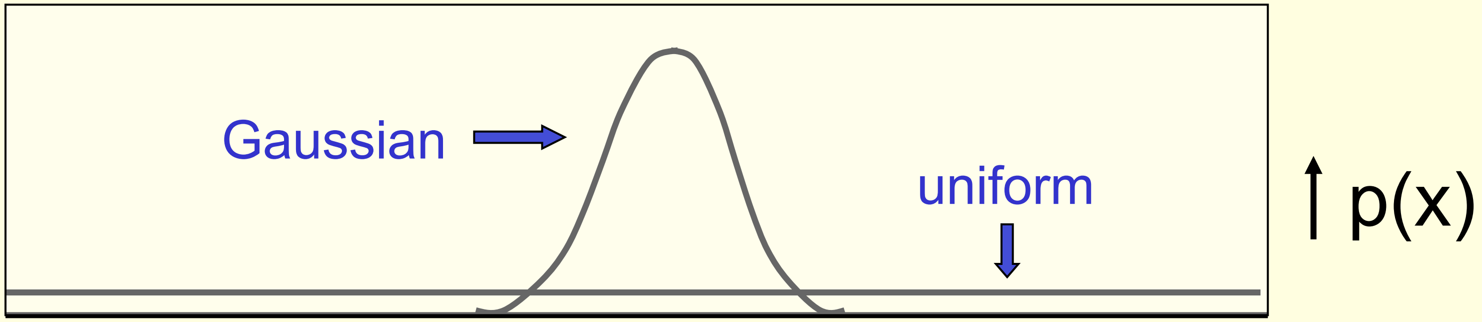
- Each expert is a mixture of a Gaussian and a uniform. This creates an energy dimple.

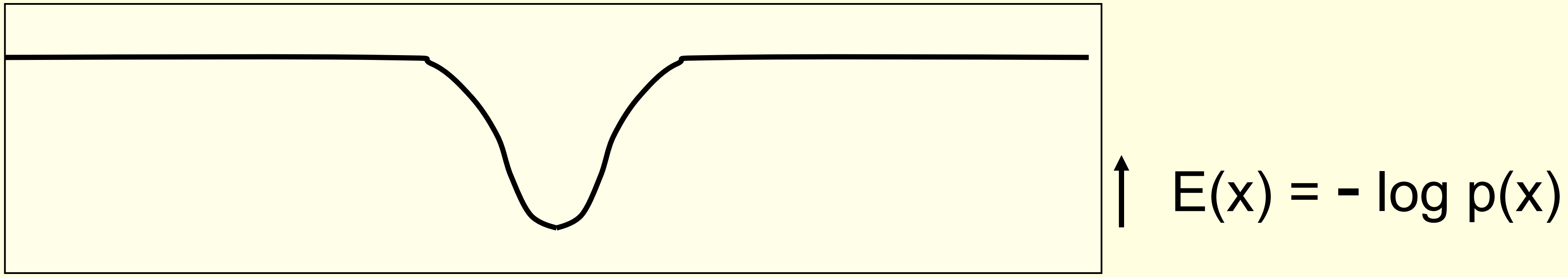
$$p_m(x) = \pi_m P(x \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \frac{1-\pi_m}{r}$$

Mixing proportion of Gaussian

Mean and variance of Gaussian

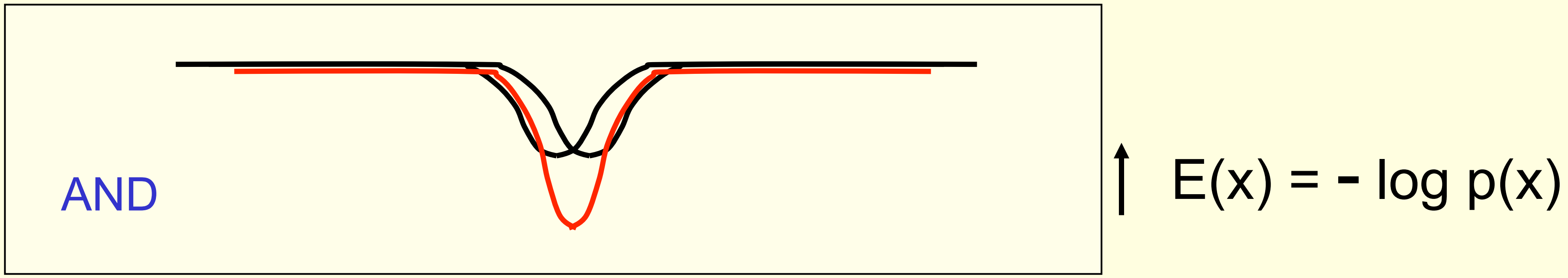
range of uniform

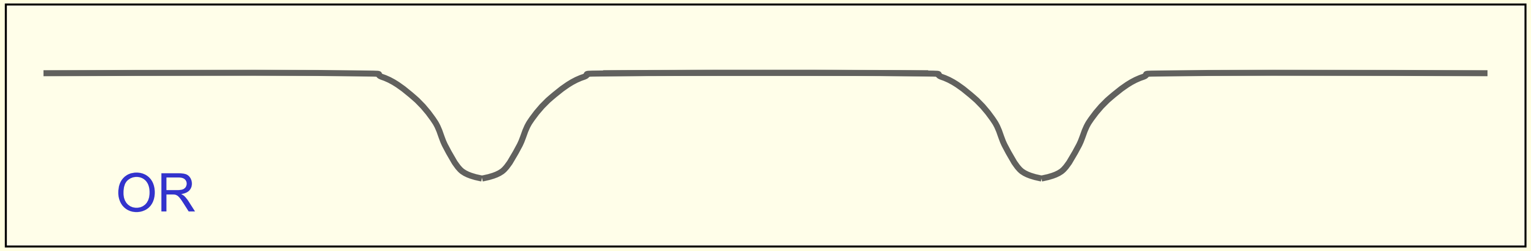




Combining energy dimples

- 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.





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 or latent variables:

	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)

Learning a Product of Experts

datavector

$$p(d \mid \theta) = \frac{\prod_{m \in Models} p_m(d \mid \theta_m)}{\sum_c \prod_m p_m(c \mid \theta_m)}$$

Normalization term to make the probabilities of all possible datavectors sum to 1

$$\log p(d \mid \theta) = \sum_m \log p_m(d \mid \theta_m) - \log \sum_c \prod_m p_m(c \mid \theta_m)$$

$$\frac{\partial \log p(d \mid \theta)}{\partial \theta_m} = \frac{\partial \log p_m(d \mid \theta_m)}{\partial \theta_m} - \sum_c p(c \mid \theta) \frac{\partial \log p_m(c \mid \theta_m)}{\partial \theta_m}$$

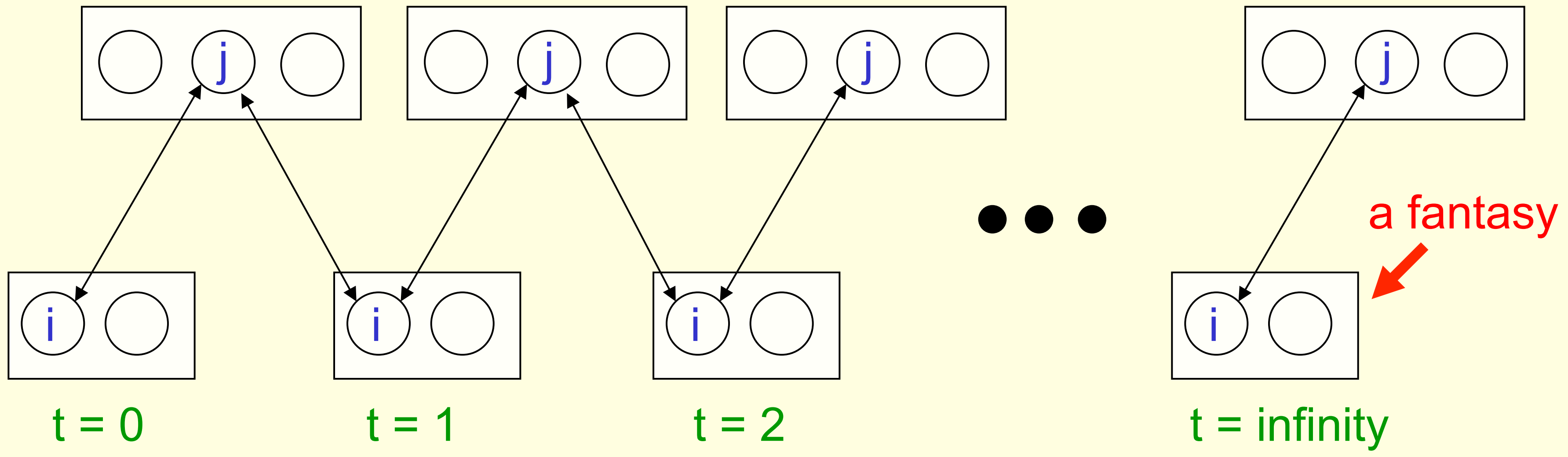
Sum over all possible datavectors

Probability of c under existing product model

Ways to deal with the intractable sum

- Set up a Markov Chain that samples from the existing model.
  - The samples can then be used to get a noisy estimate of the last term in the derivative
  - The chain may need to run for a long time before the fantasies it produces have the correct distribution.
- For uni-gauss experts we can set up a Markov chain by sampling the hidden state of each expert.
  - The hidden state is whether it used the Gaussian or the uniform.
  - The experts' hidden states can be sampled in parallel
    - This is a big advantage of products of experts.

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.

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.

Good and bad properties of the shortcut

- Much less variance because a datavector and its confabulation form a matched pair.
- If the model is perfect and there is an infinite amount of data, the confabulations will be equilibrium samples.
  - So the shortcut will not cause learning to mess up a perfect model.
- What about regions far from the data that have high density under the model?
  - There is no pressure to raise their energy.
- Seems to be very biased
  - But maybe it is approximately optimizing a different objective function.

Contrastive divergence

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

data distribution

model's distribution

distribution after one step of Markov chain

$$CD = KL(P \| Q^{\infty}) - KL(Q^{1} \| Q^{\infty})$$

Minimize Contrastive Divergence

Minimize divergence between data distribution and model's distribution

Maximize the divergence between confabulations and model's distribution

Contrastive divergence

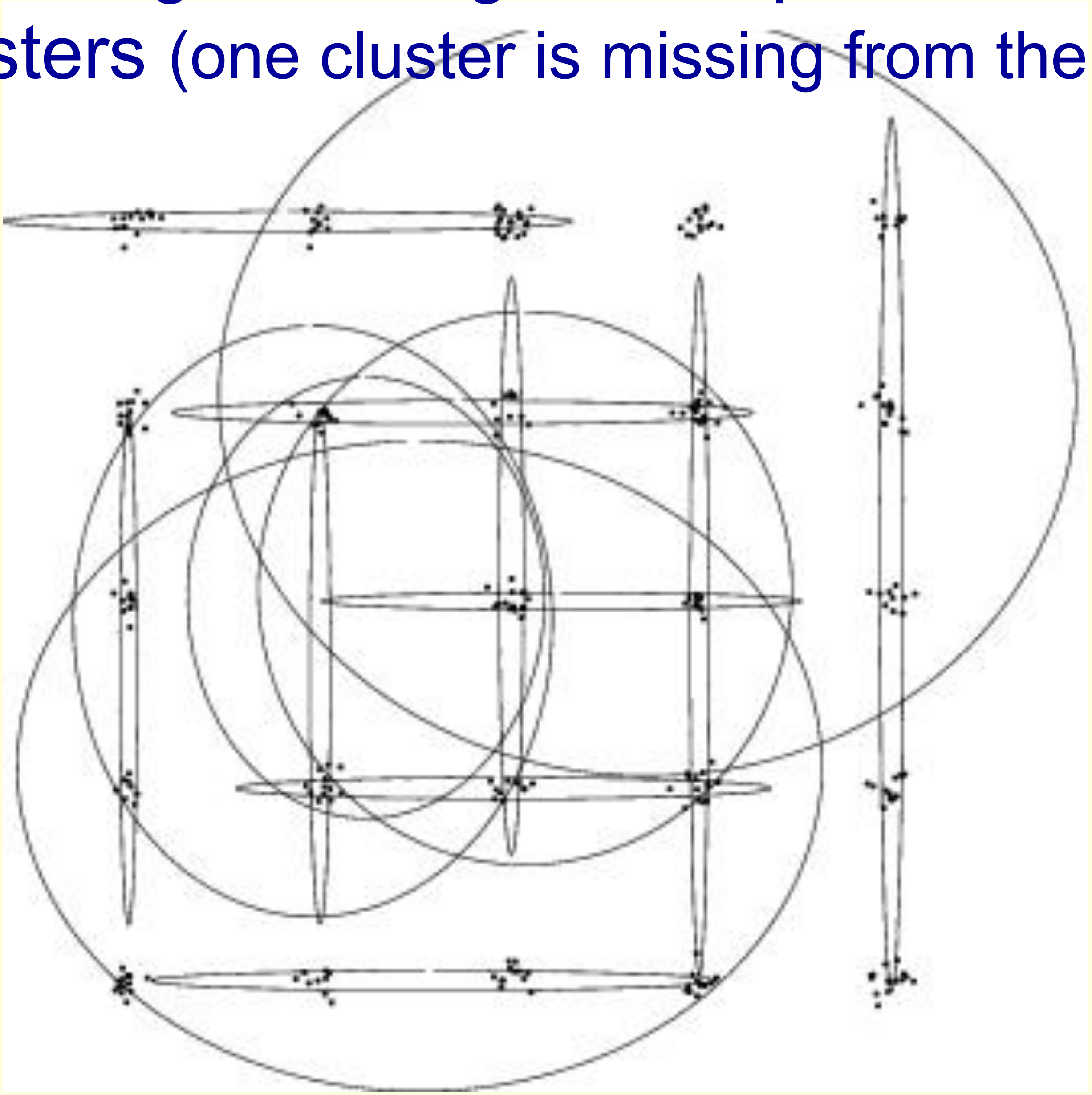
$$-\frac{\partial KL(P \| Q^\infty)}{\partial \theta} = -\left\langle \frac{\partial E}{\partial \theta} \right\rangle_P + \boxed{\left\langle \frac{\partial E}{\partial \theta} \right\rangle_{Q^\infty}}$$

$$-\frac{\partial KL(Q^1 \| Q^\infty)}{\partial \theta} = -\left\langle \frac{\partial E}{\partial \theta} \right\rangle_{Q^1} + \boxed{\left\langle \frac{\partial E}{\partial \theta} \right\rangle_{Q^\infty}} \boxed{-\frac{\partial Q^1}{\partial \theta} \frac{\partial KL(Q^1 \| Q^\infty)}{\partial Q^1}}$$

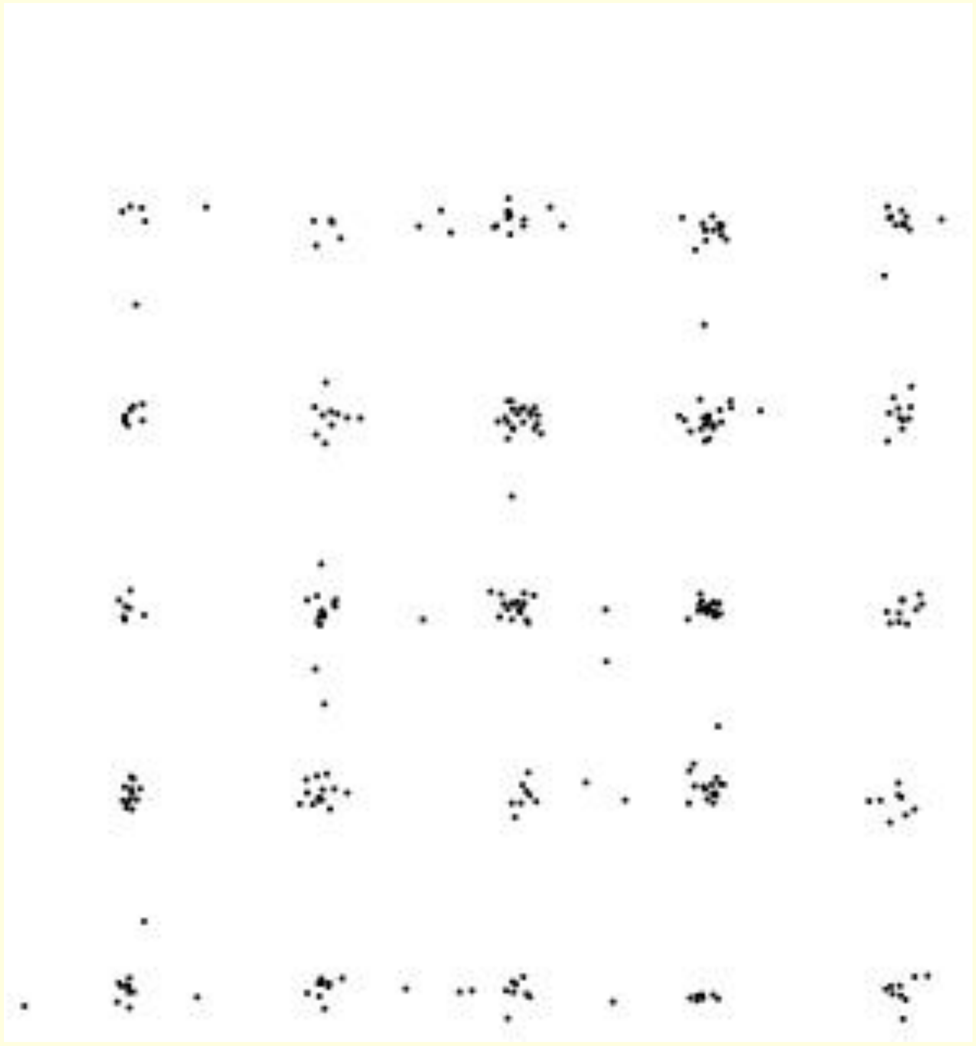
Contrastive divergence makes the awkward terms cancel

changing the parameters changes the distribution of confabulations

15 axis-aligned uni-gauss experts fitted to 24 clusters (one cluster is missing from the grid)



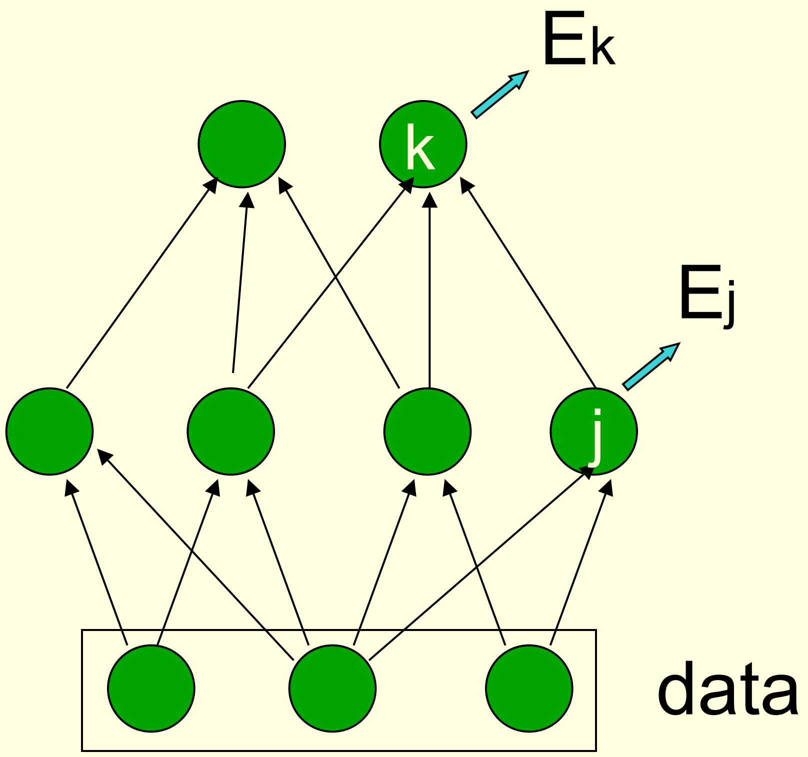
Fantasies from the model (it fills in the missing cluster)



Energy-Based Models with deterministic hidden units

- Use multiple layers of deterministic hidden units with non-linear activation functions.
- Hidden activities contribute additively to the global energy, E.
- Familiar features help, violated constraints hurt.

$$p(d) = \frac{e^{-E(d)}}{\sum_c e^{-E(c)}}$$



Reminder:
Maximum likelihood learning is hard

- To get high log probability for d we need low energy for d and high energy for its main rivals, c

$$\log p(d) = -E(d) - \log \sum_c e^{-E(c)}$$

$$\frac{\partial \log p(d)}{\partial \theta} = -\frac{\partial E(d)}{\partial \theta} + \sum_c \underset{\uparrow}{p(c)} \frac{\partial E(c)}{\partial \theta}$$

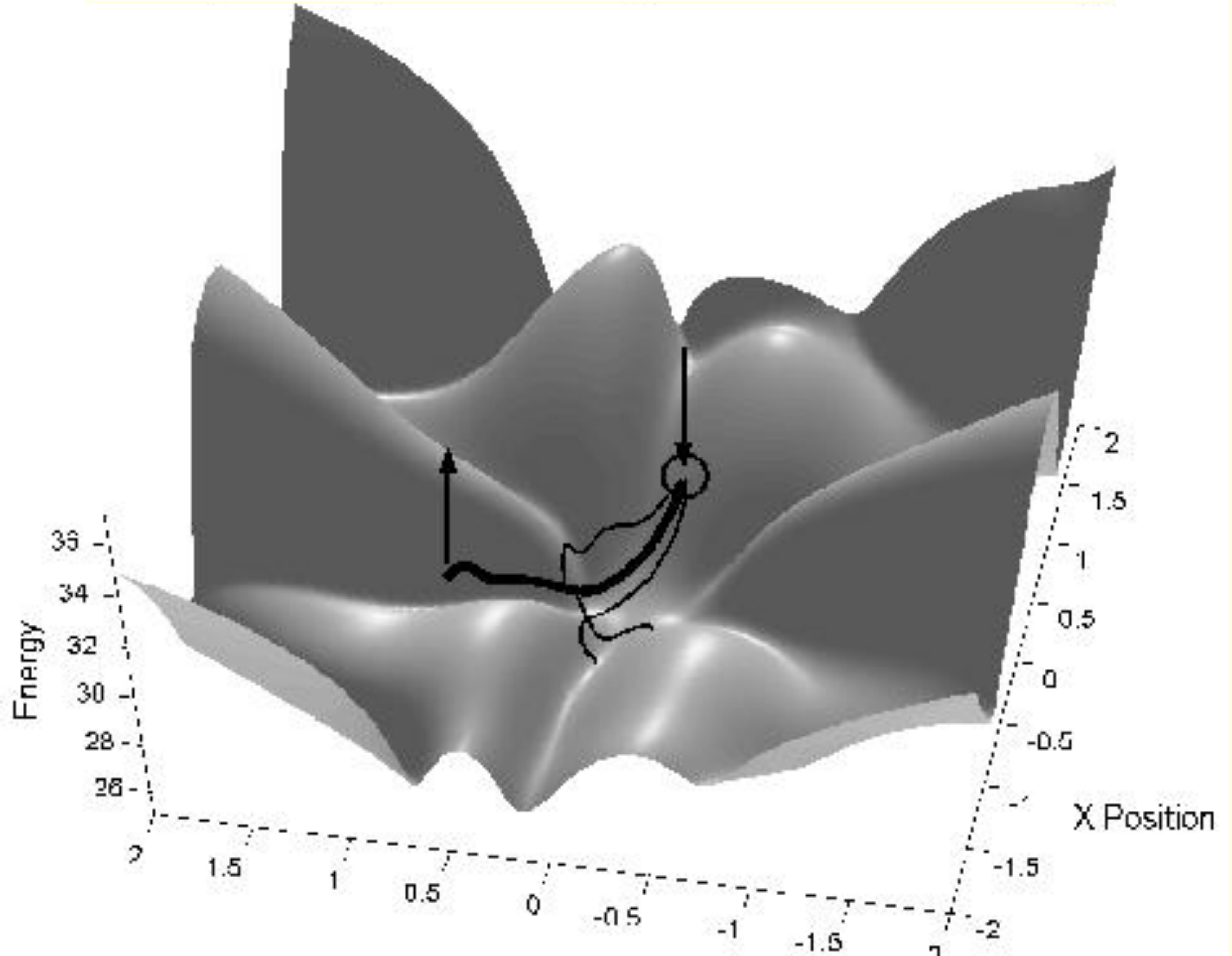
To sample from the model use Markov Chain Monte Carlo. But what kind of chain can we use when the hidden units are deterministic and the visible units are real-valued.

Hybrid Monte Carlo

- We could find good rivals by repeatedly making a random perturbation to the data and accepting the perturbation with a probability that depends on the energy change.
  - Diffuses very slowly over flat regions
  - Cannot cross energy barriers easily

- In high-dimensional spaces, it is much better to use the gradient to choose good directions.
- HMC adds a random momentum and then simulates a particle moving on an energy surface.
  - Beats diffusion. Scales well.
  - Can cross energy barriers.
  - Back-propagation can give us the gradient of the energy surface.

Trajectories with different initial momenta



Simulating the dynamics

- The total energy is the sum of the potential and kinetic energies.
  - This is called the Hamiltonian

$$H(\mathbf{q},\mathbf{p}) = E(\mathbf{q}) + \frac{1}{2}\sum_i p_i^2$$

- The rate of change of position, q, equals the velocity, p.

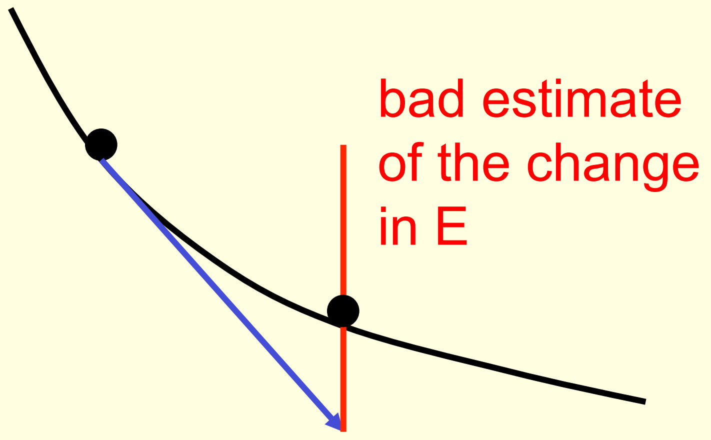
$$\frac{dq_i}{d\tau} = +\frac{\partial H}{\partial p_i} = p_i$$

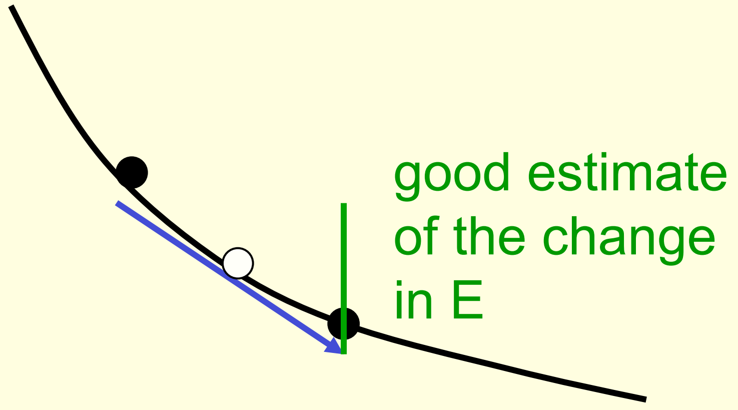
- The rate of change of the velocity is the negative gradient of the potential energy, E.

$$\frac{dp_i}{d\tau} = -\frac{\partial H}{\partial q_i} = -\frac{\partial E}{\partial q_i}$$

A numerical problem

- How can we minimize numerical errors while simulating the dynamics?
- We can use the same trick as we use for checking if we have got the right gradient of an objective function
  - Interpolation works much better than extrapolation
  - So use the gradient at the midpoint. This is the average gradient over the interval if the curvature is constant.





The leapfrog method for keeping numerical errors small.

- Update the velocity using the initial gradient.

$$p_i(\tau + \tfrac{\varepsilon}{2}) = p_i(\tau) - \frac{\varepsilon}{2} \frac{\partial E}{\partial q_i}\big(q(\tau)\big)$$

- Update the position using the velocity at the midpoint of the interval.

$$q_i(\tau + \varepsilon) = q_i(\tau) + \varepsilon\; p_i(\tau + \tfrac{\varepsilon}{2})$$

- Update the velocity again using the final gradient.

$$p_i(\tau + \varepsilon) = p_i(\tau + \tfrac{\varepsilon}{2}) - \frac{\varepsilon}{2} \frac{\partial E}{\partial q_i}\big(q(\tau + \varepsilon)\big)$$

Combining the last move of one interval with the first move of the next interval

$$q_i(\tau+\varepsilon) = q_i(\tau) + \varepsilon \; p_i(\tau + \frac{\varepsilon}{2})$$

$$p_i(\tau + 1.5\varepsilon) = p_i(\tau + \frac{\varepsilon}{2}) - \varepsilon \frac{\partial E}{\partial q_i}\big(q(\tau+\varepsilon)\big)$$

- Now we are using the gradient at the midpoint for updating both q and p. The updates leapfrog over each other.

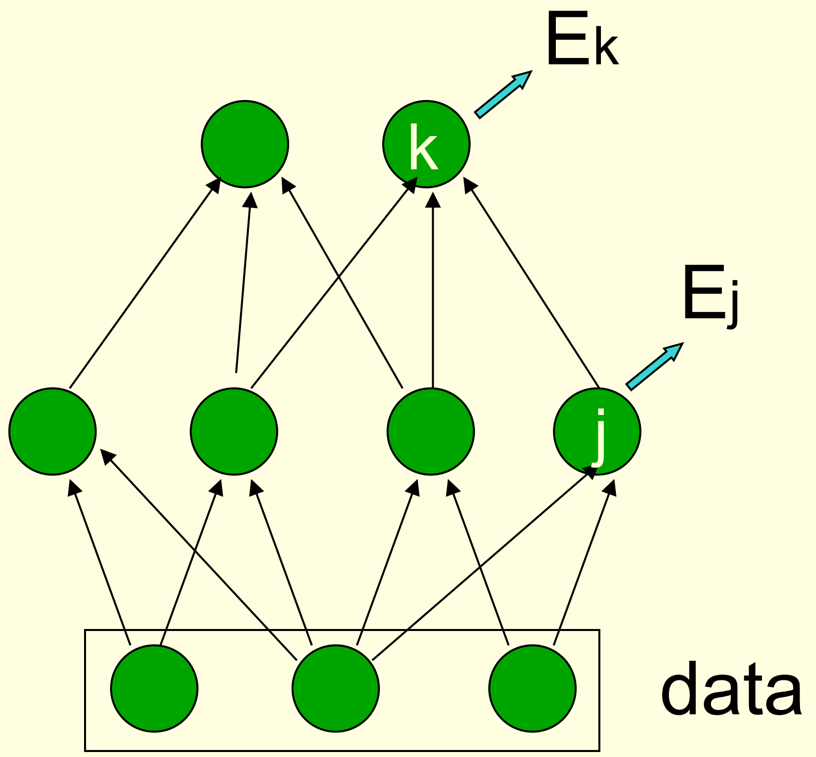
Dealing with the remaining numerical error

- Treat the whole trajectory as a proposed move for the Metropolis algorithm.
  - If the energy increases, only accept with probability exp(-increase).
- To decide on the size of the steps used for simulating the dynamics, look at the reject rate.
  - If its small, we could have used bigger steps and gone further.
  - If its big, we are wasting too many computed trajectories.

Backpropagation can compute the gradient that Hybrid Monte Carlo needs

1. Do a forward pass computing hidden activities.
2. Do a backward pass all the way to the data to compute the derivative of the global energy w.r.t each component of the data vector.

works with any smooth non-linearity



The online HMC learning procedure

1. Start at a datavector, d, and use backprop to compute for every parameter $\underline{\partial E(d)/\partial \theta}$

2. Run HMC for many steps with frequent renewal of the momentum to get equilibrium sample, c. Each step involves a forward and backward pass to get the gradient of the energy in dataspace.

The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

3. Use backprop to compute $\underline{\partial E(c)/\partial \theta}$

4. Update the parameters by :

$$\Delta\theta = \varepsilon \left(-\underline{\partial E(d)/\partial \theta} \; + \underline{\partial E(c)/\partial \theta} \right)$$

The shortcut

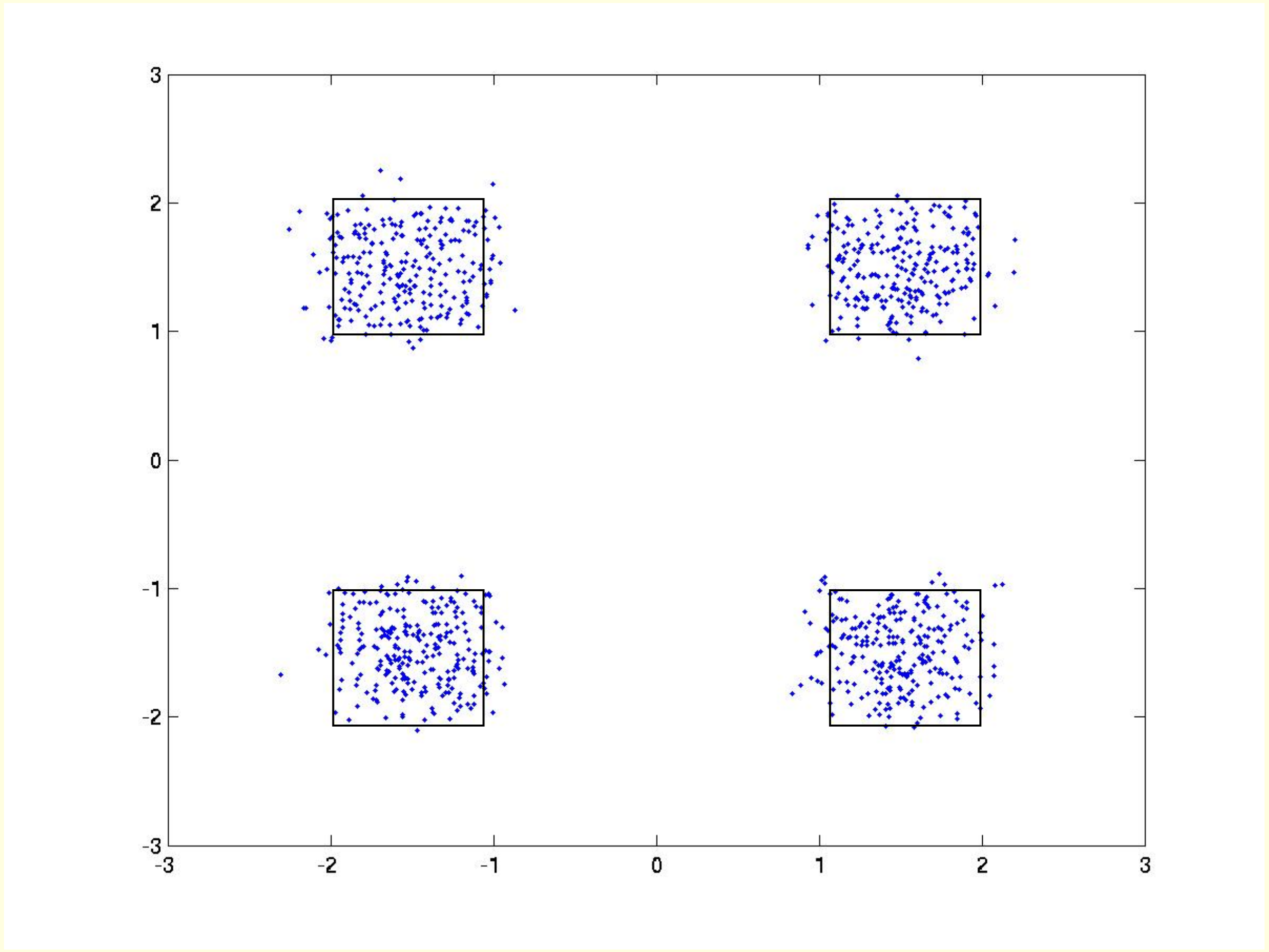
- Instead of taking the negative samples from the equilibrium distribution, use slight corruptions of the datavectors. Only add random momentum once, and only follow the dynamics for a few steps.
  - Much less variance because a datavector and its confabulation form a matched pair.
  - Gives a very biased estimate of the gradient of the log likelihood.
  - Gives a good estimate of the gradient of the contrastive divergence (i.e. the amount by which F falls during the brief HMC.)

- Its very hard to say anything about what this method does to the log likelihood because it only looks at rivals in the vicinity of the data.

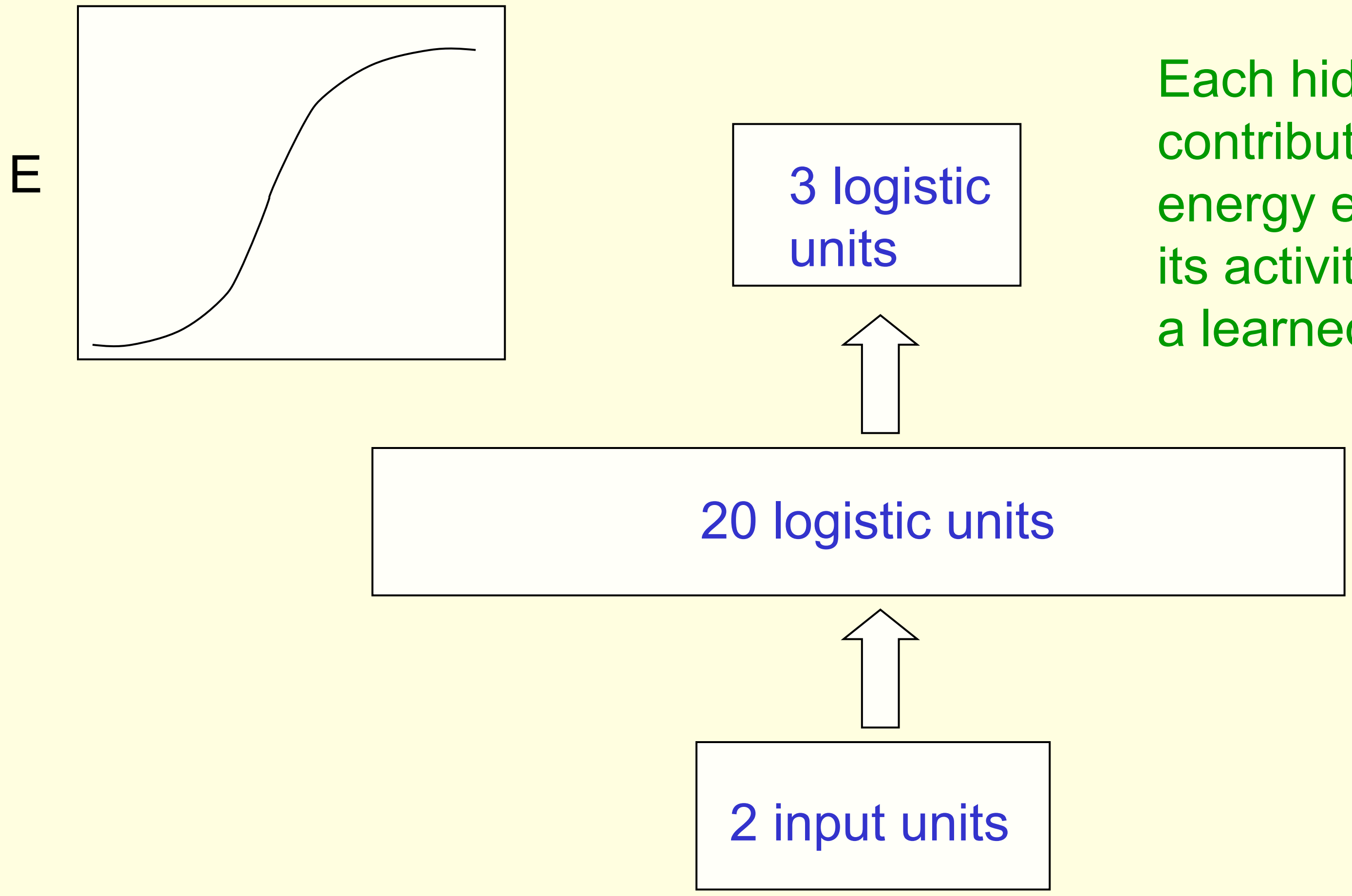
- Its hard to say exactly what this method does to the contrastive divergence because the Markov chain defines what we mean by "vicinity", and the chain keeps changing as the parameters change.
  - But its works well empirically, and it can be proved to work well in some very simple cases.

A simple 2-D dataset

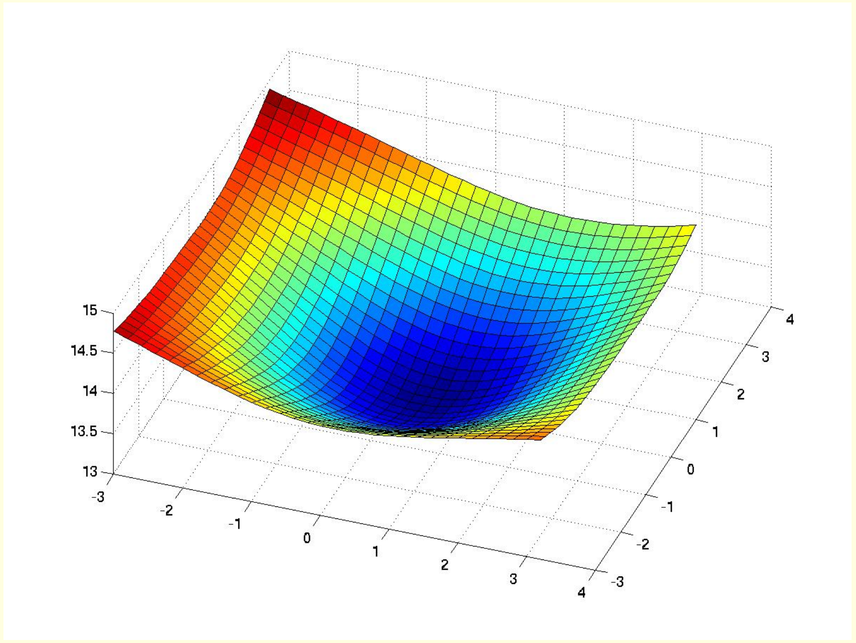
The true data is uniformly distributed within the 4 squares.
The blue dots are samples from the model.

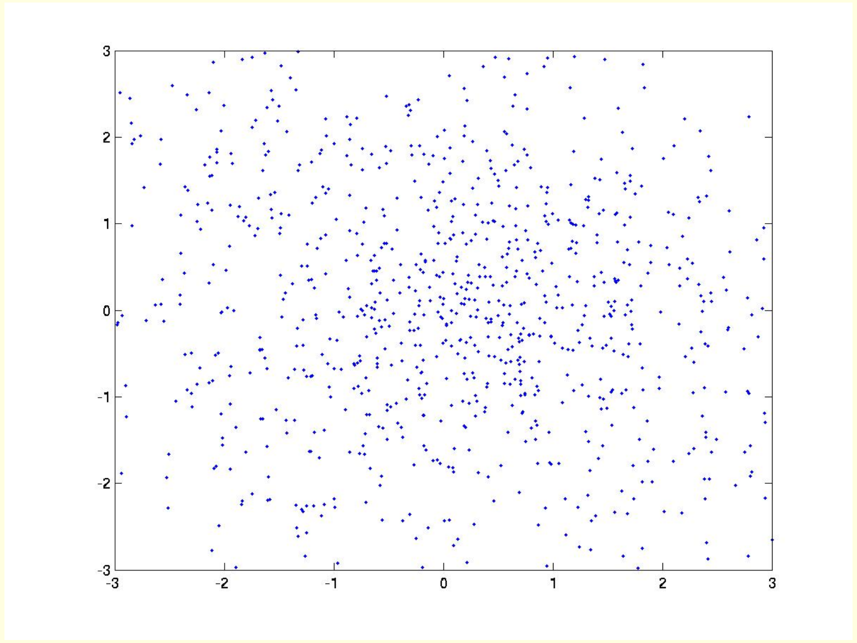


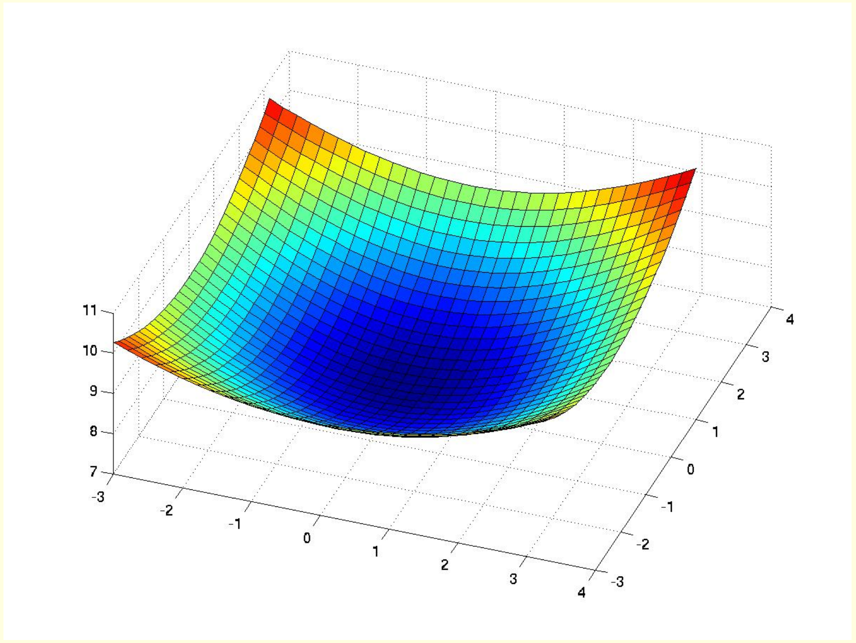
The network for the 4 squares task



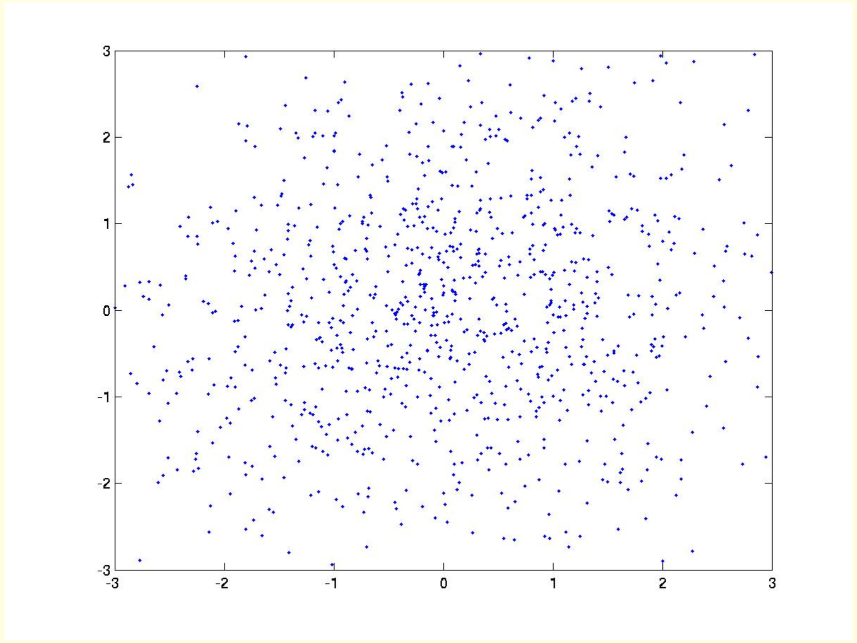
Each hidden unit contributes an energy equal to its activity times a learned scale.


15
14.5
14
13.5
13
-3
-2
-1
0
1
2
3
4
-3
-2
-1
0
1
2
3
4

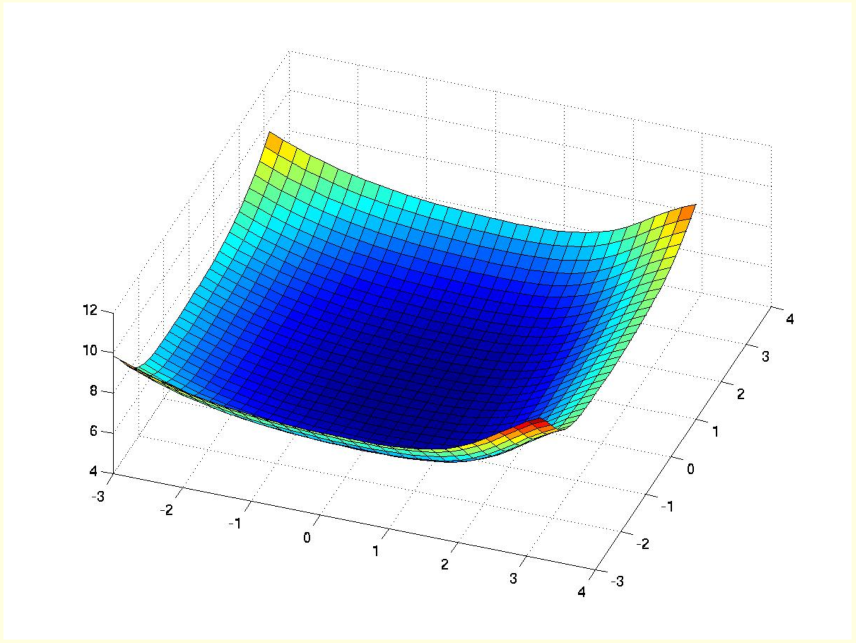

3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3



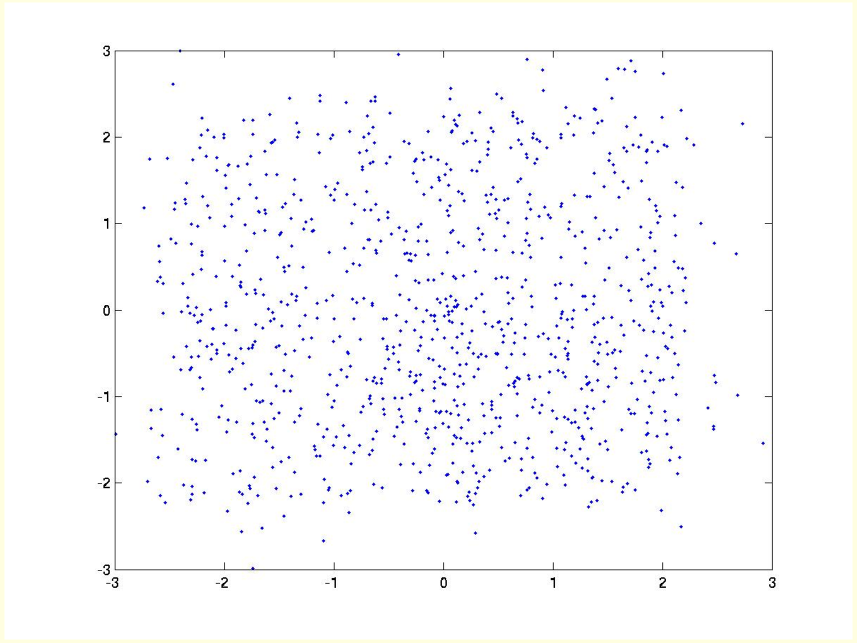
11
10
9
8
7
-3
-2
-1
0
1
2
3
4
4
3
2
1
0
-1
-2
-3



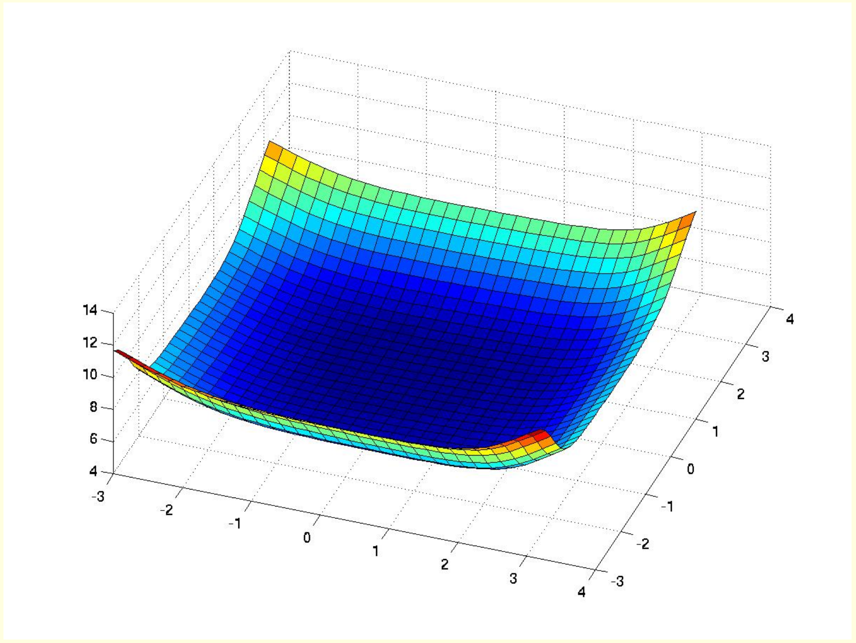
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3



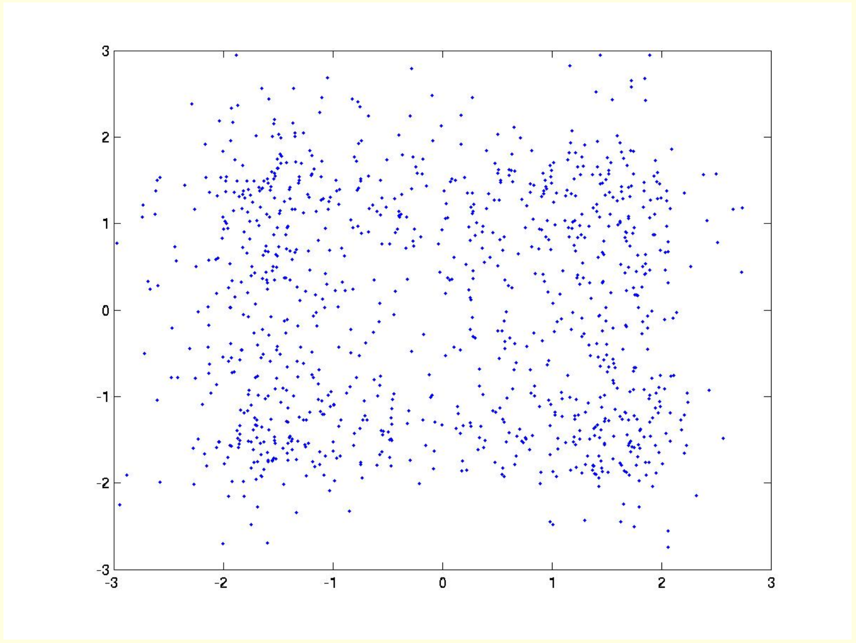
12
10
8
6
4
-3
-2
-1
0
1
2
3
4
4
3
2
1
0
-1
-2
-3



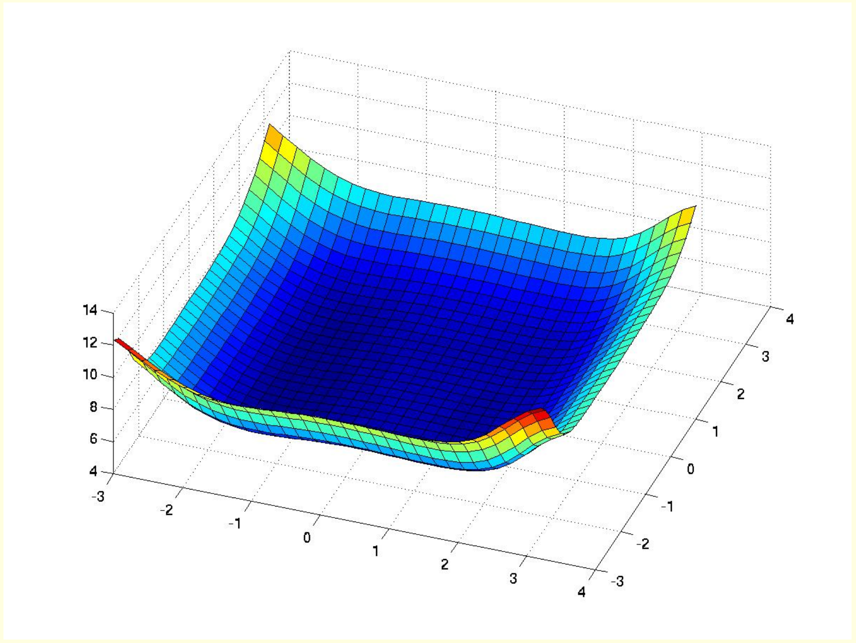
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3



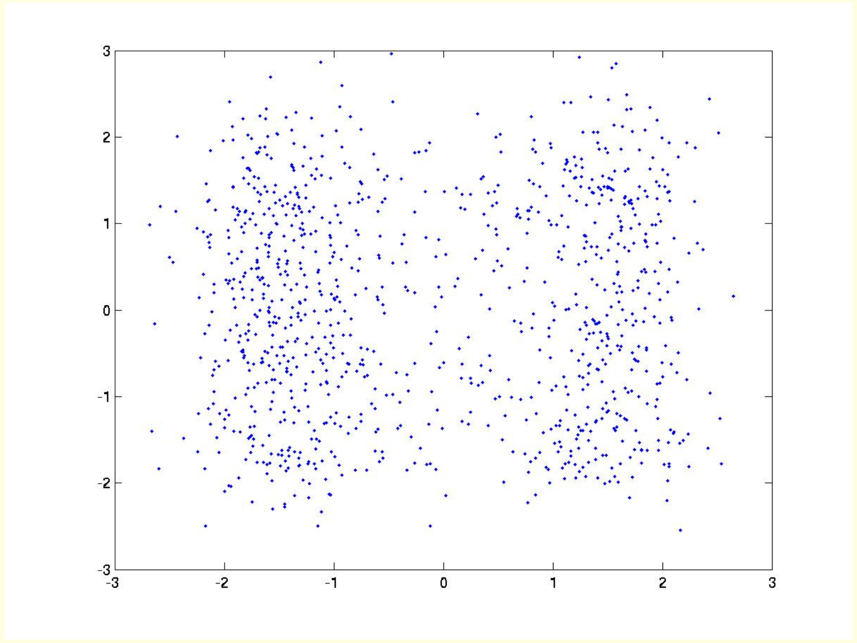
14
12
10
8
6
4
-3
-2
-1
0
1
2
3
4
-3
-2
-1
0
1
2
3
4



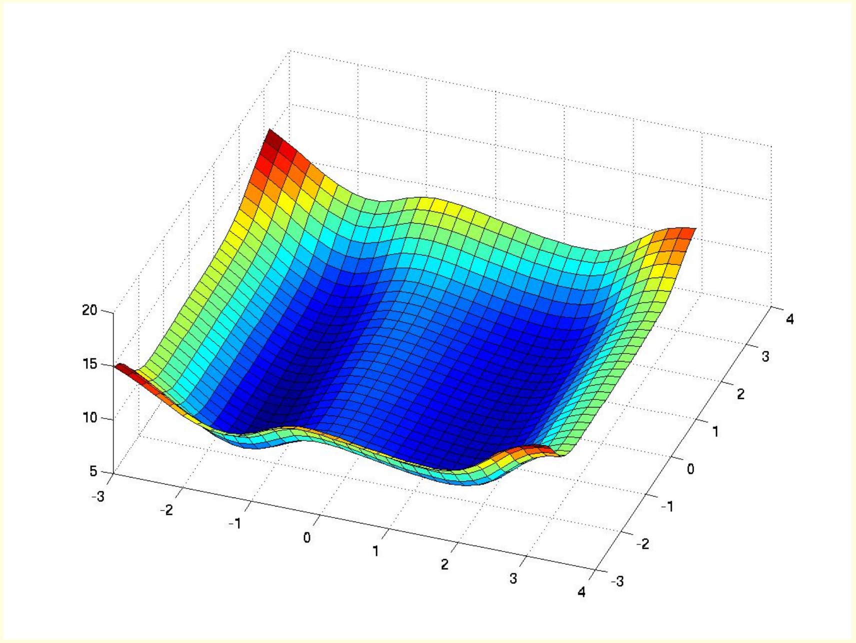
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

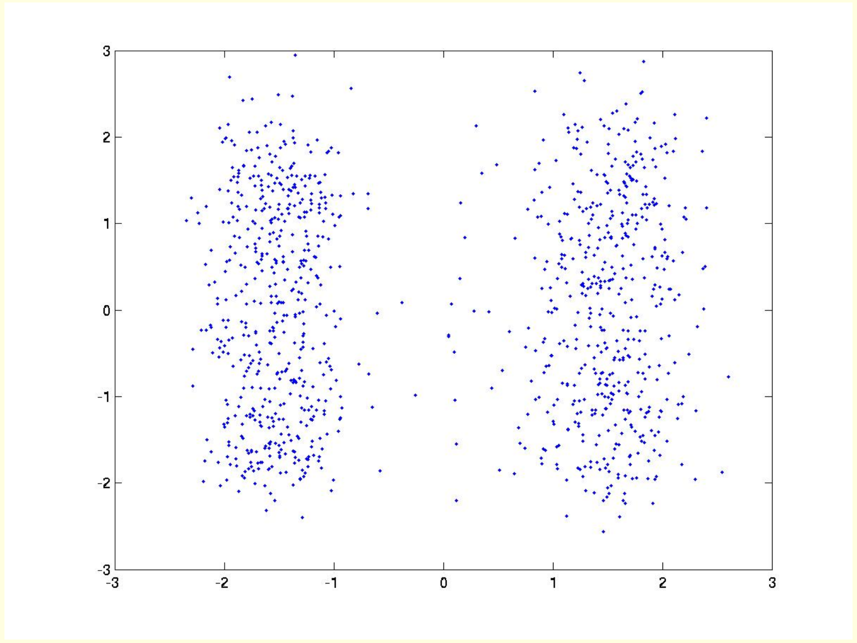


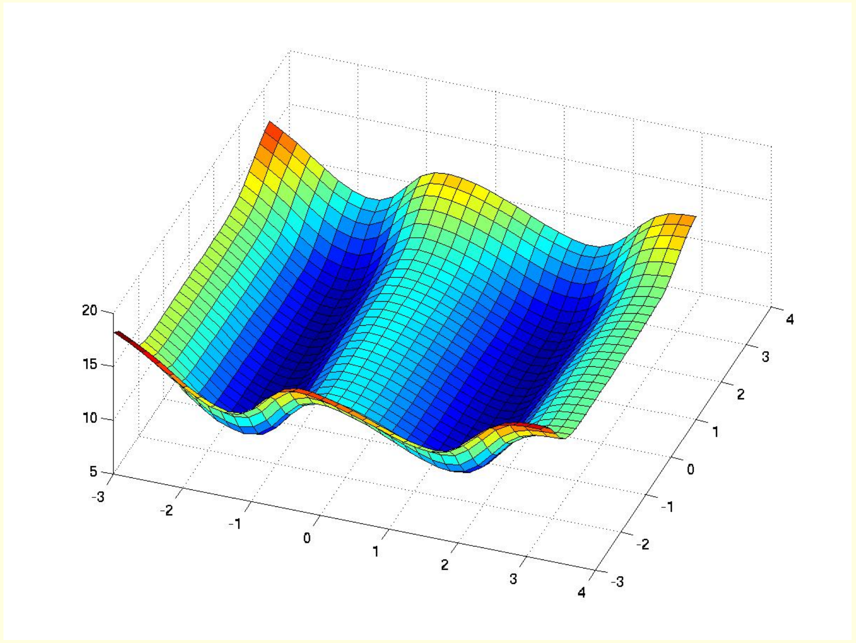
14
12
10
8
6
4
-3
-2
-1
0
1
2
3
4
-3
-2
-1
0
1
2
3
4



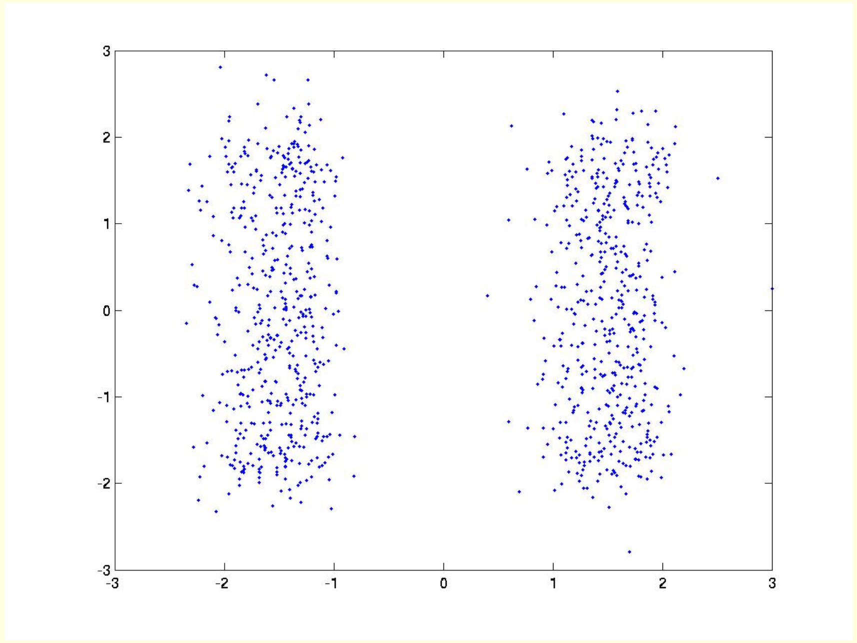
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3


20
15
10
5
-3
-2
-1
0
1
2
3
4
-3
-2
-1
0
1
2
3
4

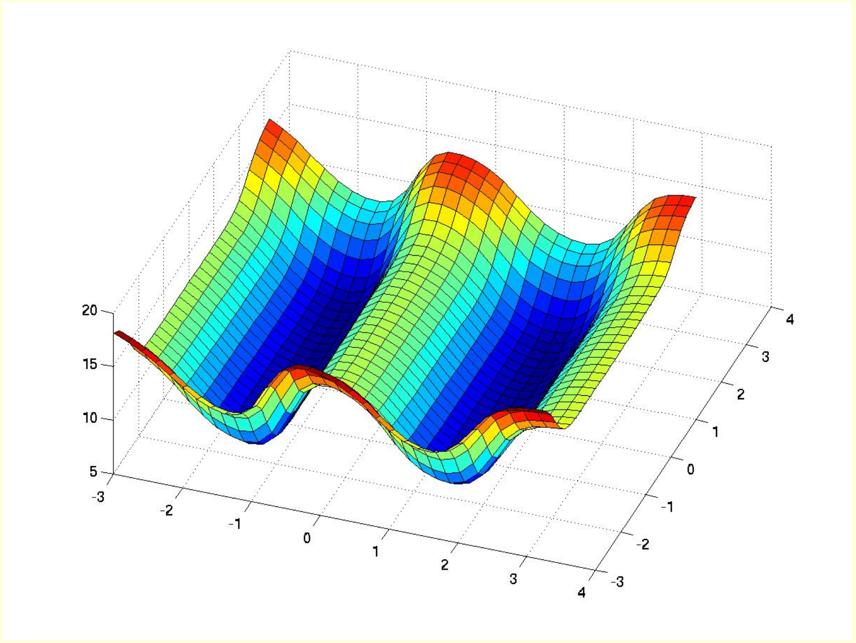

3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

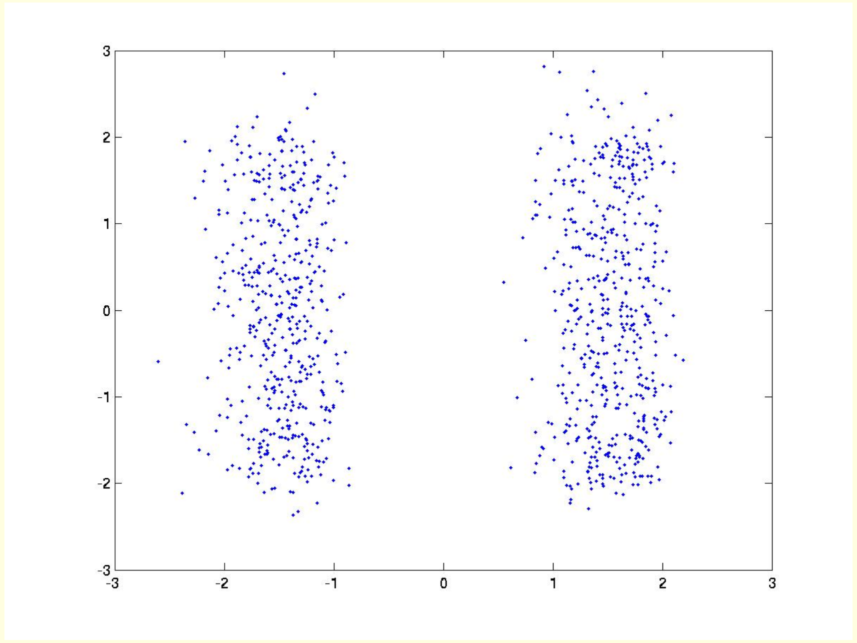


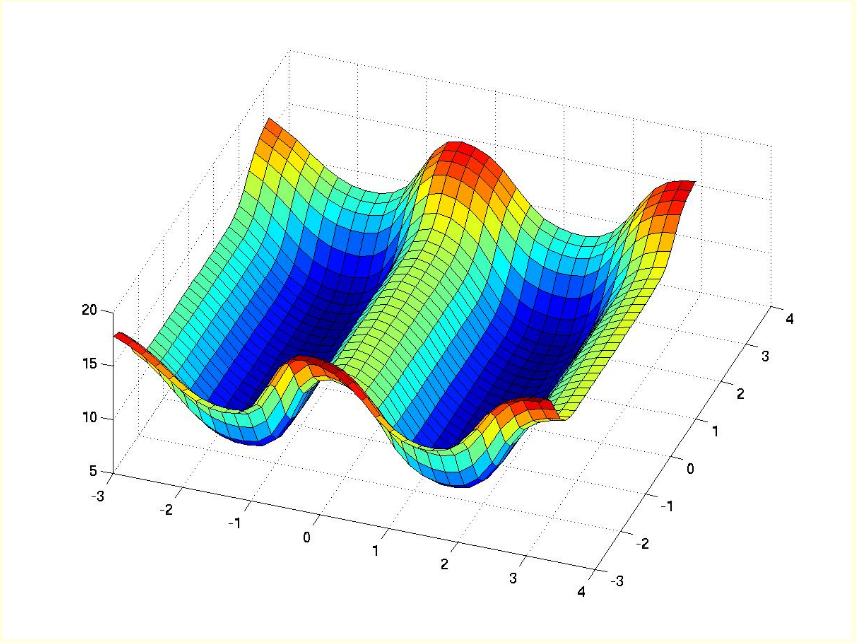
20
15
10
5
-3
-2
-1
0
1
2
3
4
-3
-2
-1
0
1
2
3
4

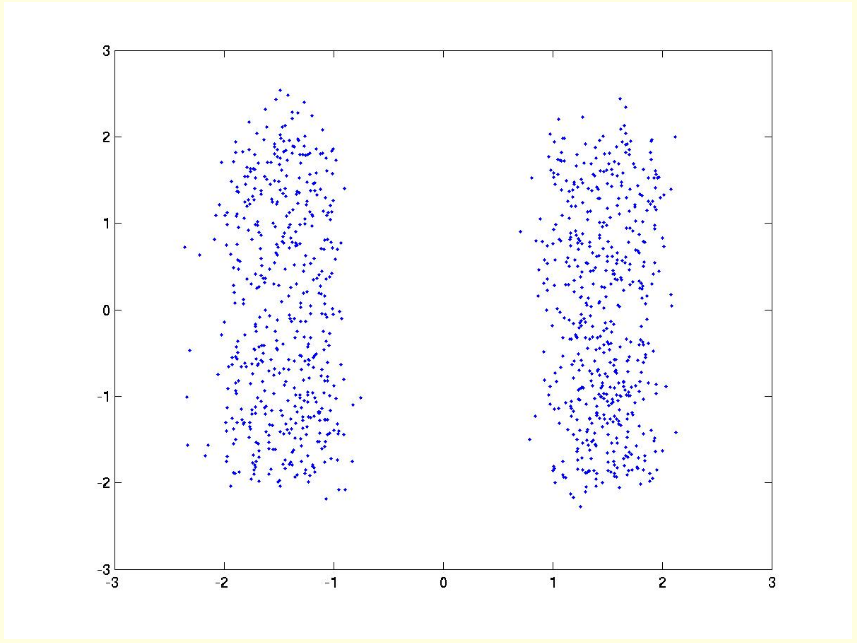


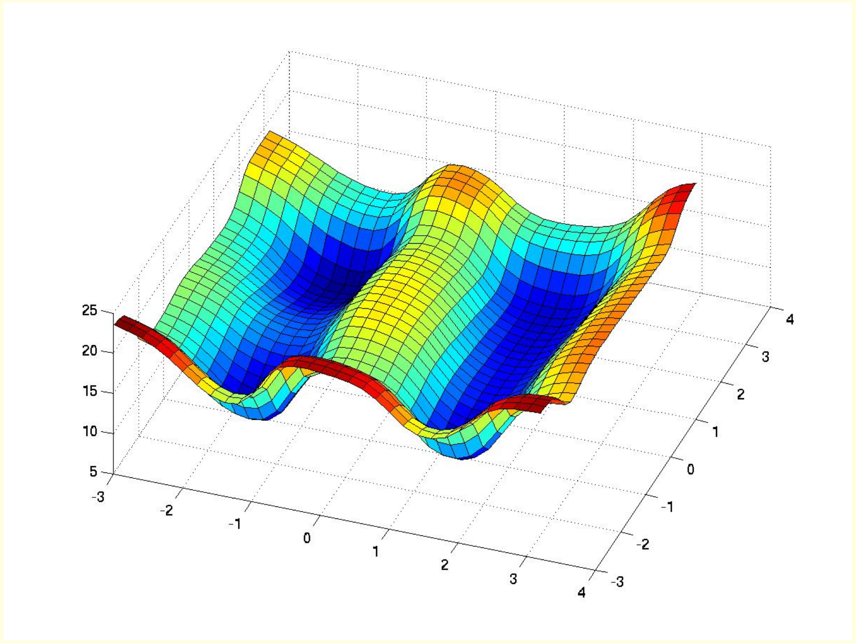
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

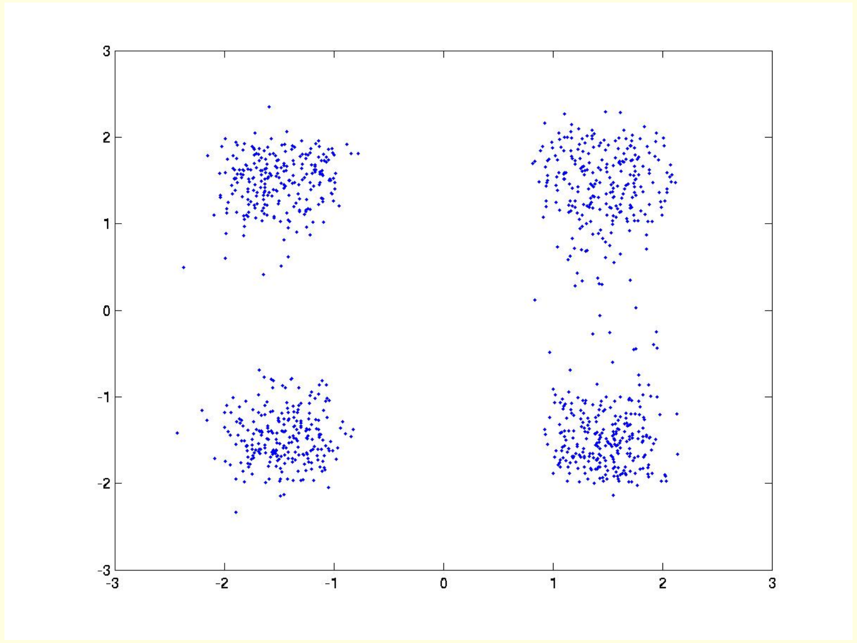

20
15
10
5
-3
-2
-1
0
1
2
3
4
4
3
2
1
0
-1
-2
-3

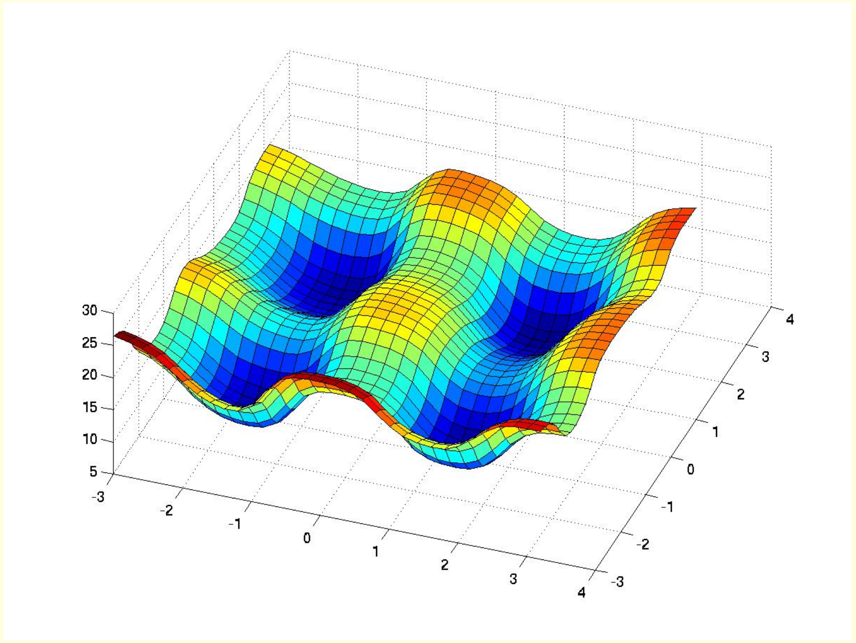

3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

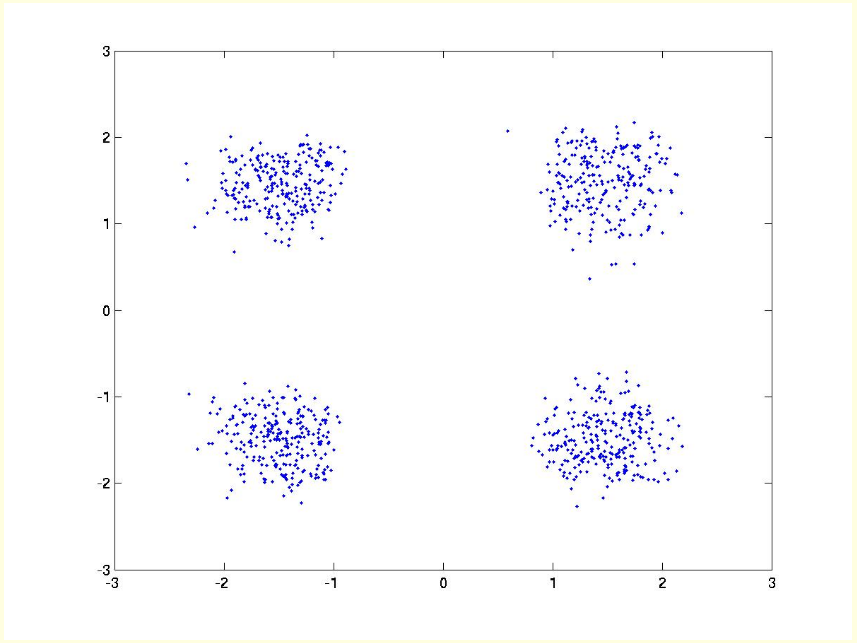

20
15
10
5
-3
-2
-1
0
1
2
3
4
-3
-2
-1
0
1
2
3
4

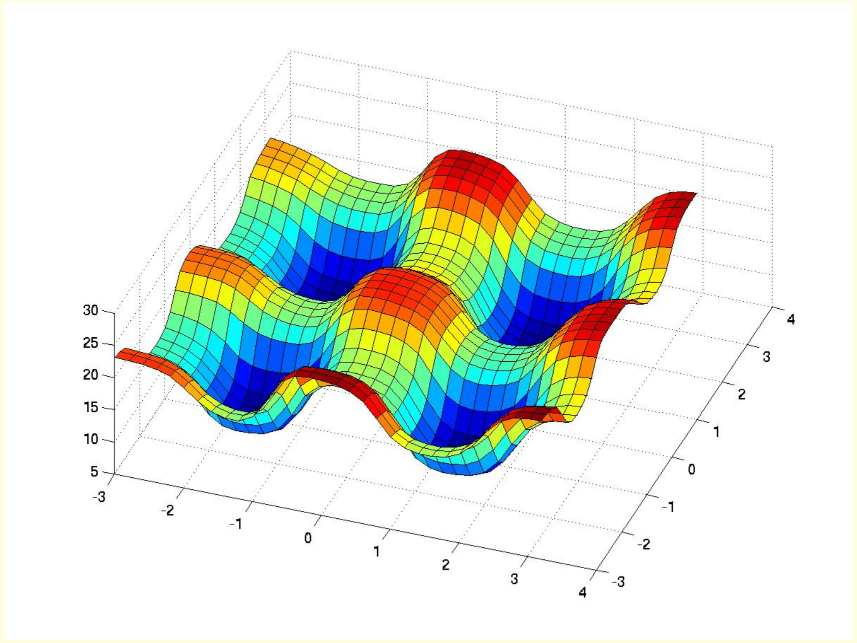

3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

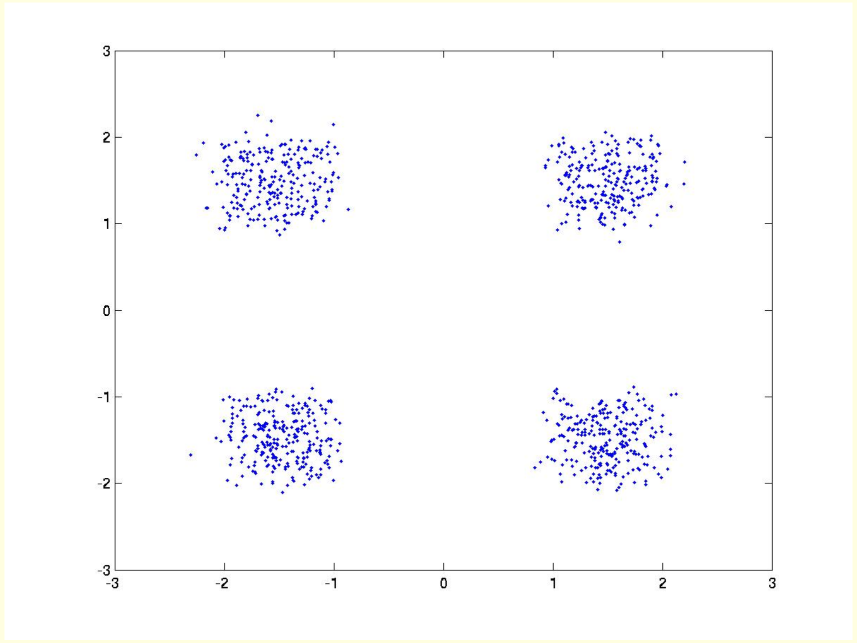

25
20
15
10
5
-3
-2
-1
0
1
2
3
4
-3
-2
-1
0
1
2
3
4


3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3


30
25
20
15
10
5
-3
-2
-1
0
1
2
3
4
-3
-2
-1
0
1
2
3
4


3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3


30
25
20
15
10
5
-3
-2
-1
0
1
2
3
4
-3
-2
-1
0
1
2
3
4


3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

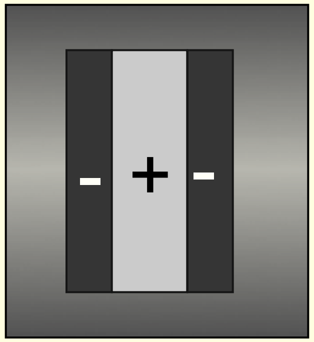
A different kind of hidden structure

Data is often characterized by saying which directions have high variance. But we can also capture structure by finding constraints that are Frequently Approximately Satisfied. If the constrints are linear they represent directions of low variance.

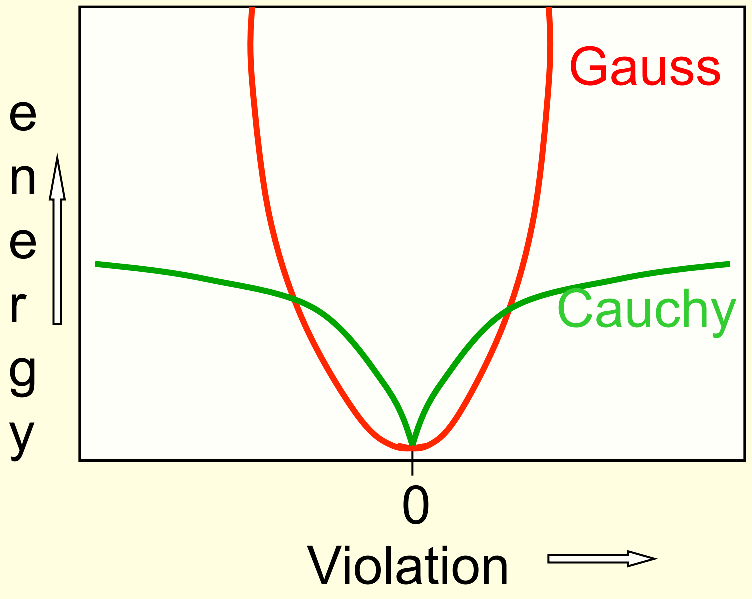
Violations of FAS constraints reduce the probability of a data vector. If a constraint already has a big violation, violating it more does not make the data vector much worse (i.e. assume the distribution of violations is heavy-tailed.)

Frequently Approximately Satisfied constraints

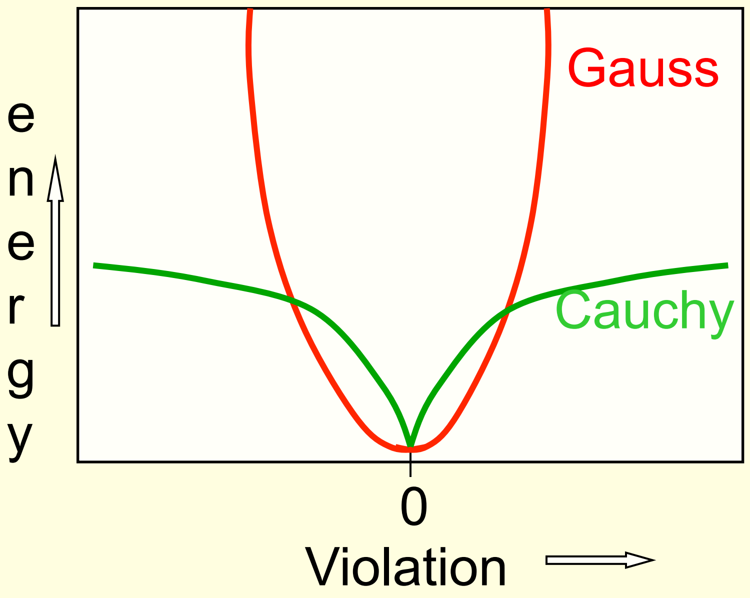
- The intensities in a typical image satisfy many different linear constraints very accurately, and violate a few constraints by a lot.
- The constraint violations fit a heavy-tailed distribution.
- The negative log probabilities of constraint violations can be used as energies.

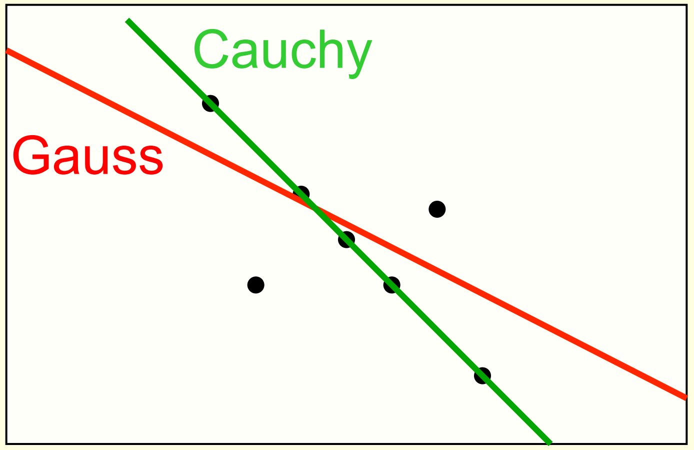


On a smooth intensity patch the sides balance the middle



Frequently Approximately Satisfied constraints



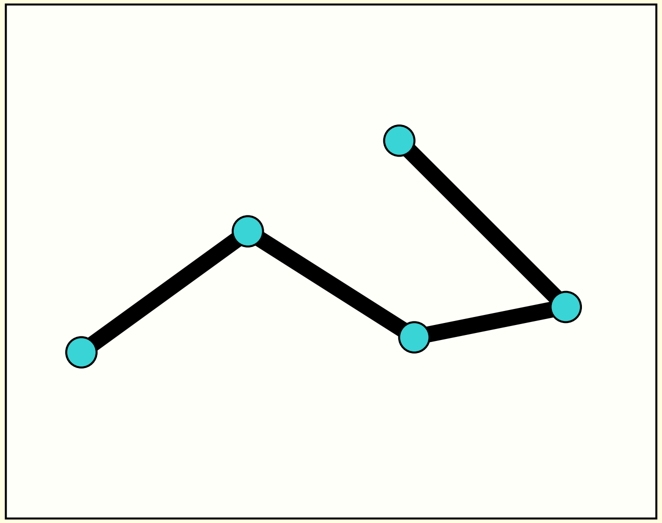


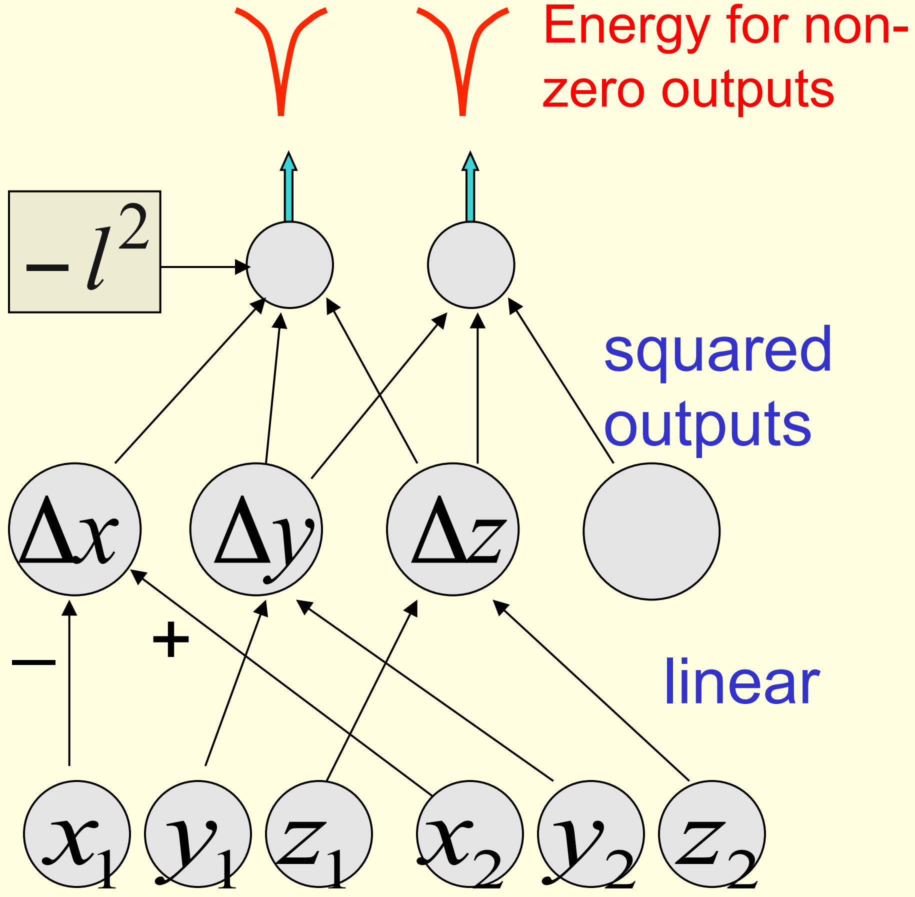
what is the best line?

The energy contributed by a violation is the negative log probability of the violation

Learning the constraints on an arm

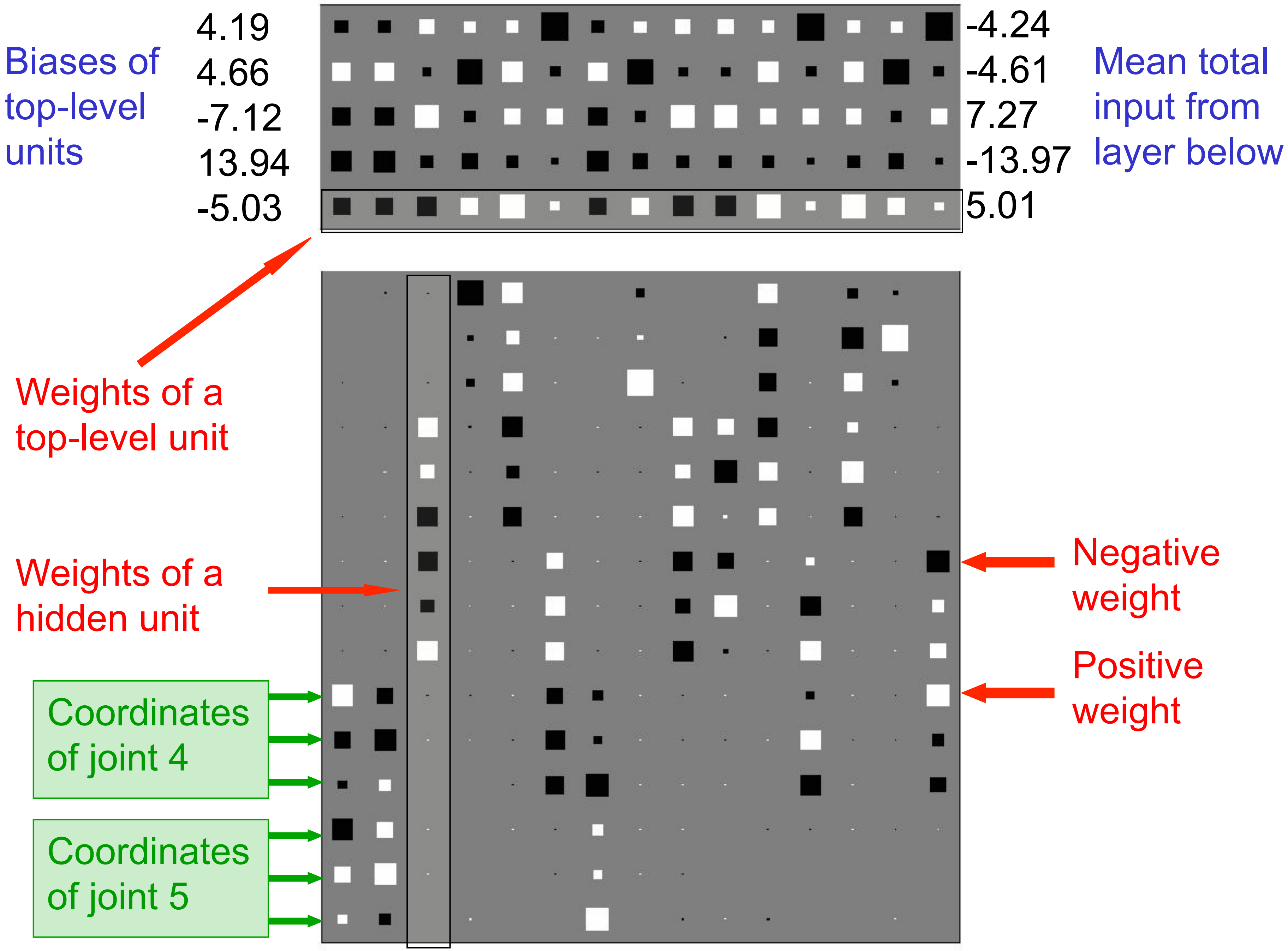
3-D arm with 4 links and 5 joints





For each link:

$$\Delta x^2 + \Delta y^2 + \Delta z^2 - l^2 = 0$$


Biases of top-level units
4.19
4.66
-7.12
13.94
-5.03
-4.24
-4.61
7.27
-13.97
5.01
Mean total input from layer below
Weights of a top-level unit
Weights of a hidden unit
Negative weight
Positive weight
Coordinates of joint 4
Coordinates of joint 5

Superimposing constraints

- A unit in the second layer could represent a single constraint.
- But it can model the data just as well by representing a linear combination of constraints.

$$a\Delta x_{34}^2 + a\Delta y_{34}^2 + a\Delta z_{34}^2 - al_{34}^2 = 0$$

$$b\Delta x_{45}^2 + b\Delta y_{45}^2 + b\Delta z_{45}^2 - bl_{45}^2 = 0$$

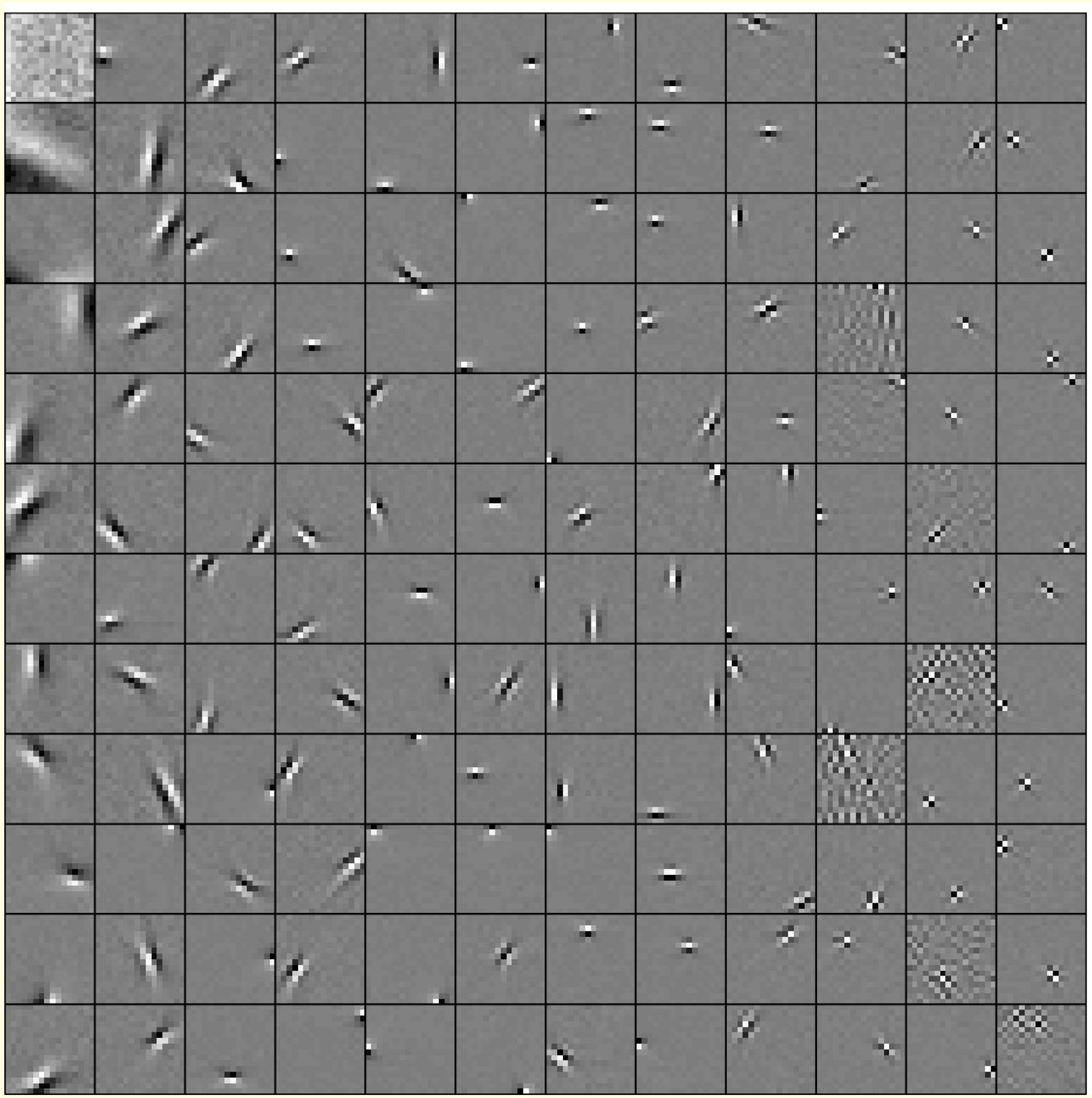
Dealing with missing inputs

- The network learns the constraints even if 10% of the inputs are missing.
  - First fill in the missing inputs randomly
  - Then use the back-propagated energy derivatives to slowly change the filled-in values until they fit in with the learned constraints.
- Why don’t the corrupted inputs interfere with the learning of the constraints?
  - The energy function has a small slope when the constraint is violated by a lot.
  - So when a constraint is violated by a lot it does not adapt.
    - Don’t learn when things don’t make sense.

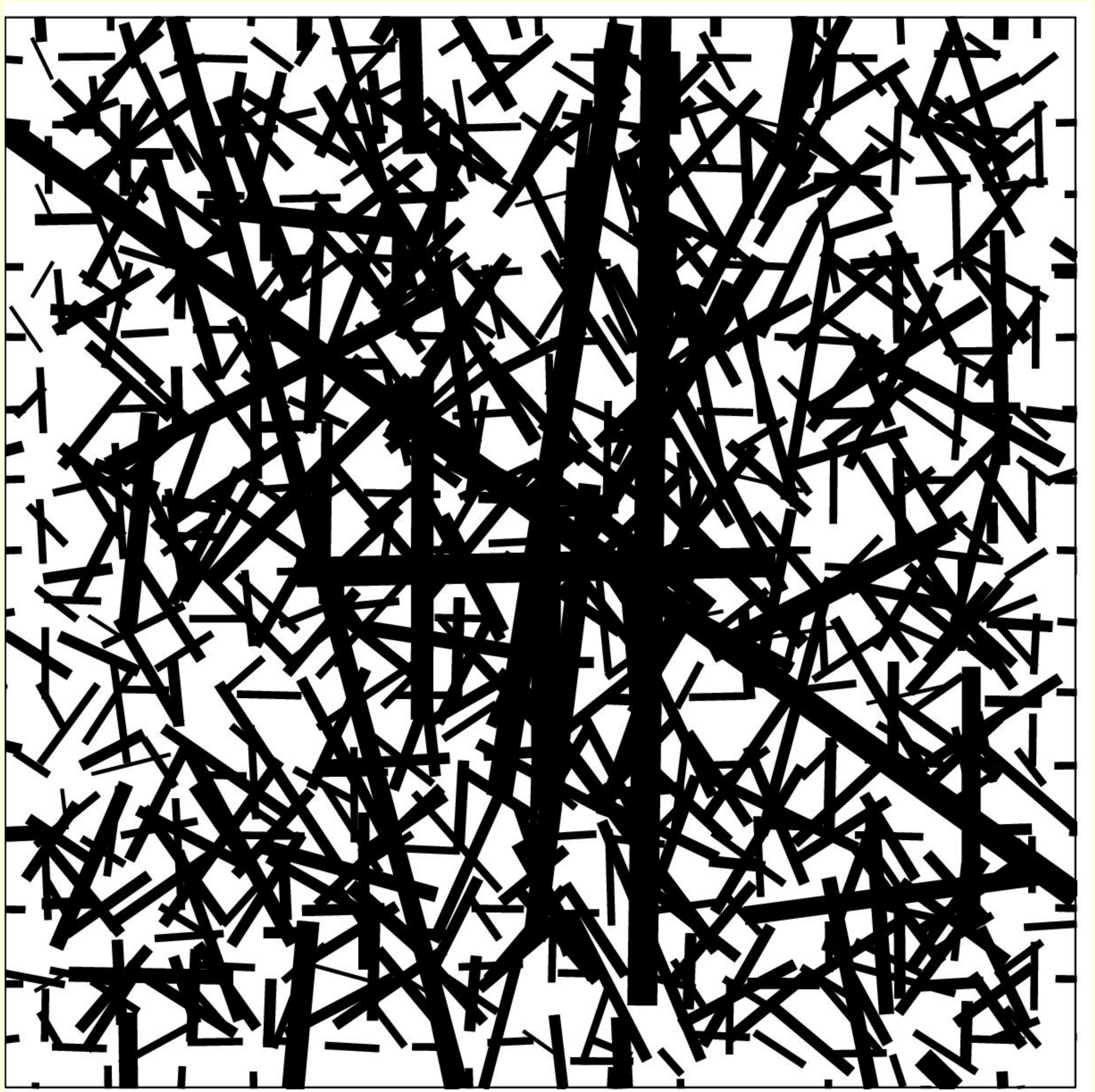
Learning constraints from natural images (Yee-Whye Teh)

- We used 16x16 image patches and a single layer of 768 hidden units (3 x over-complete).
- Confabulations are produced from data by adding random momentum once and simulating dynamics for 30 steps.
- Weights are updated every 100 examples.
- A small amount of weight decay helps.

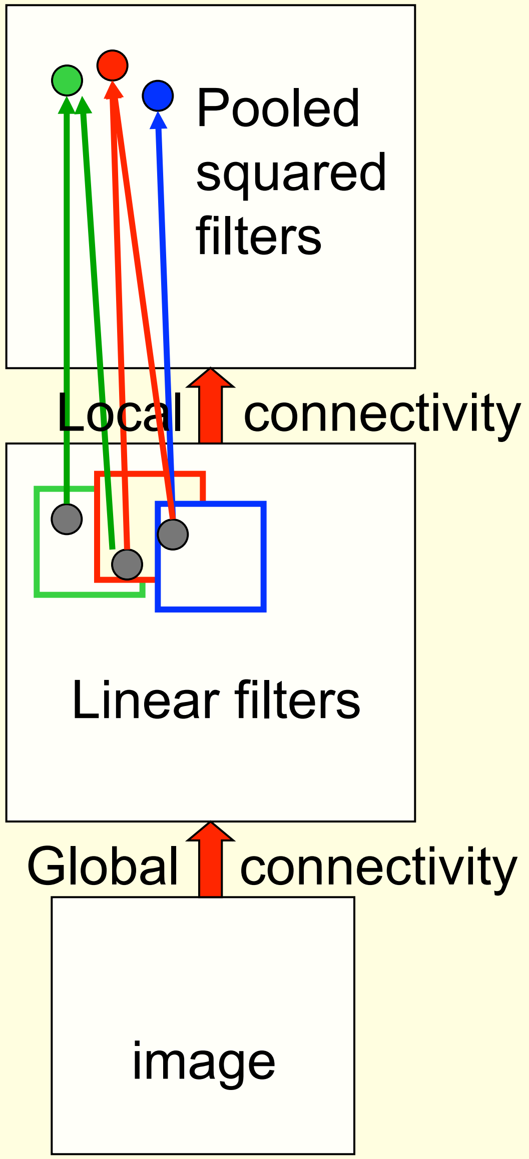
A random subset of 768 basis functions



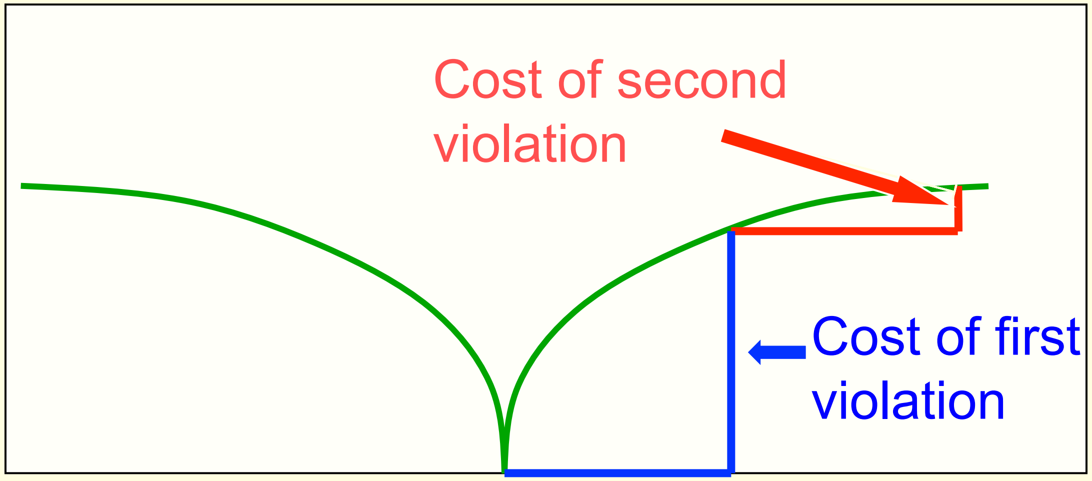
The distribution of all 768 learned basis functions



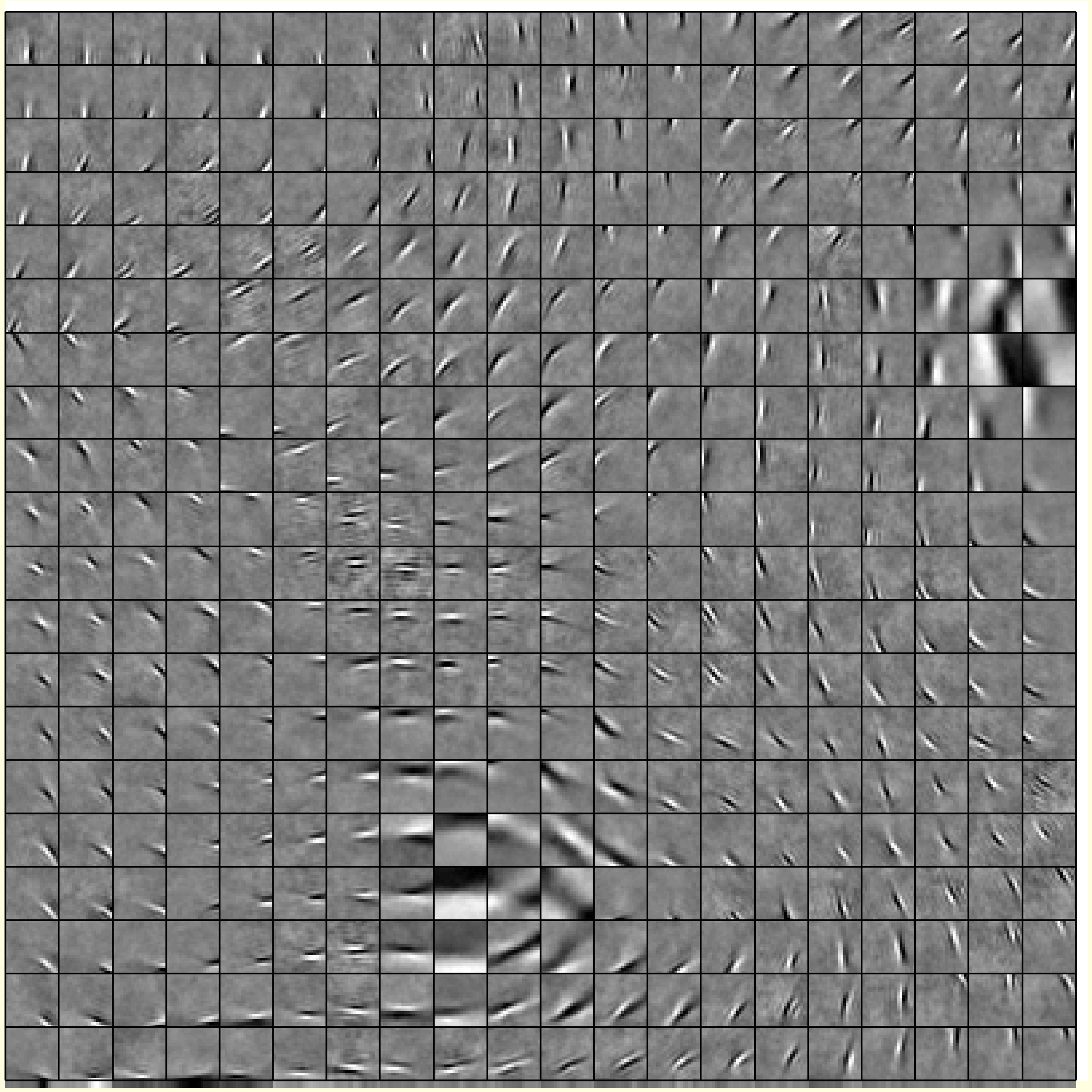
How to learn a topographic map



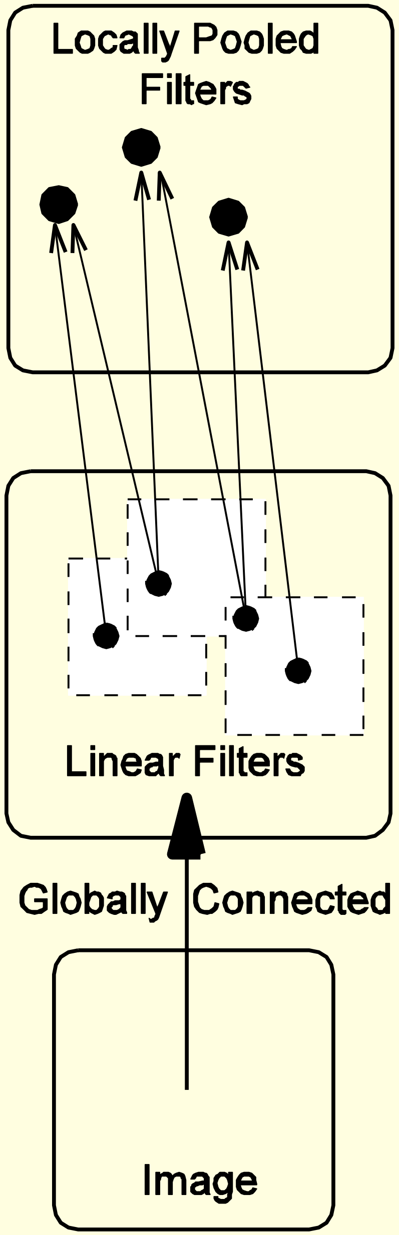
The outputs of the linear filters are squared and locally pooled. This makes it cheaper to put filters that are violated at the same time next to each other.

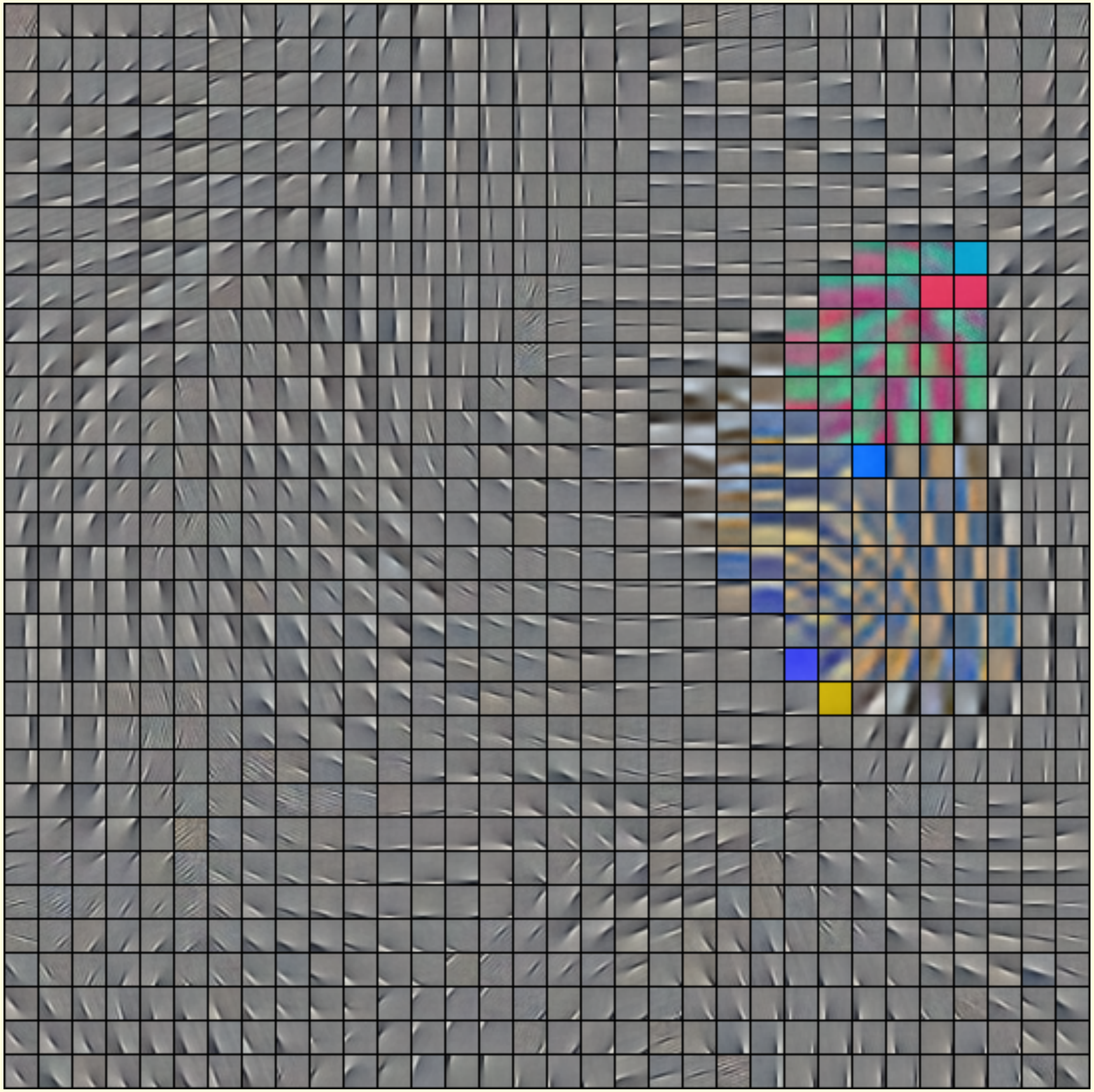


A



B


Locally Pooled Filters
Linear Filters
Globally Connected
Image



Feature detectors learned by a neural net exposed to patches of color images.

The only built in structure is some local connectivity between hidden layers.