A learning algorithm for cortex

Geoffrey Hinton
Canada Research Chair in Machine Learning
University of Toronto

How does the brain learn its internal representations?

Theory1: Supervised learning (wrong for 2 reasons)

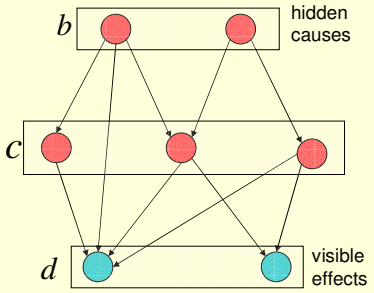
- Develop representations by finding combinations of features that can be used to predict the correct output.

Theory 2: Unsupervised learning

- Develop representations by building a model of the world that explains why those particular patterns of sensory input occurred.
  - What kinds of model are there?

Causal generative models

- It is easy to generate an unbiased example at the leaf nodes.
- It is hard to compute the posterior distribution over all possible configurations of hidden causes (except in special cases)
- Given samples from the posterior, it is easy to learn the local interactions



$$p(d) = \sum_{b,c} p(b)\,p(c \mid b)\,p(d \mid b,c)$$

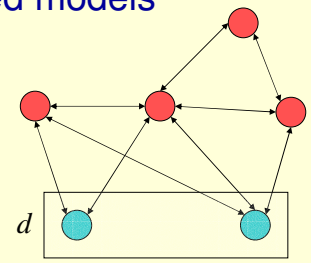
Approximate inference

- What if we use an approximation to the posterior distribution over hidden configurations?
  - e.g. assume the posterior factorizes into a product of distributions for each separate hidden cause.
- If we use the approximation for learning, there is no guarantee that learning will increase the probability that the model would generate the observed data.
- But there is a different function, called "variational free energy" that is guaranteed to improve:

$$\log p(data) - Divergence(approximate\ posterior,\ true\ posterior)$$

Energy-based models

- Use a neural network to assign an "energy" to each possible data-vector. Then define probabilities in terms of energies.
  - Learning is done by manipulating energies.



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

It is easy to lower the energy of d

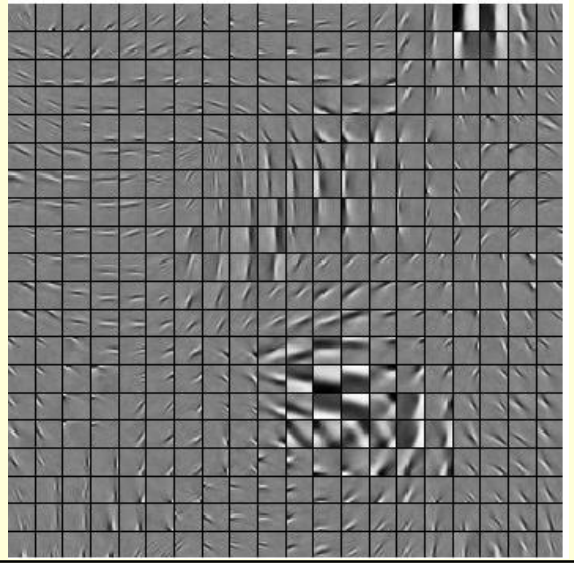
We need to find the serious rivals to d and raise their energy. This seems hard.

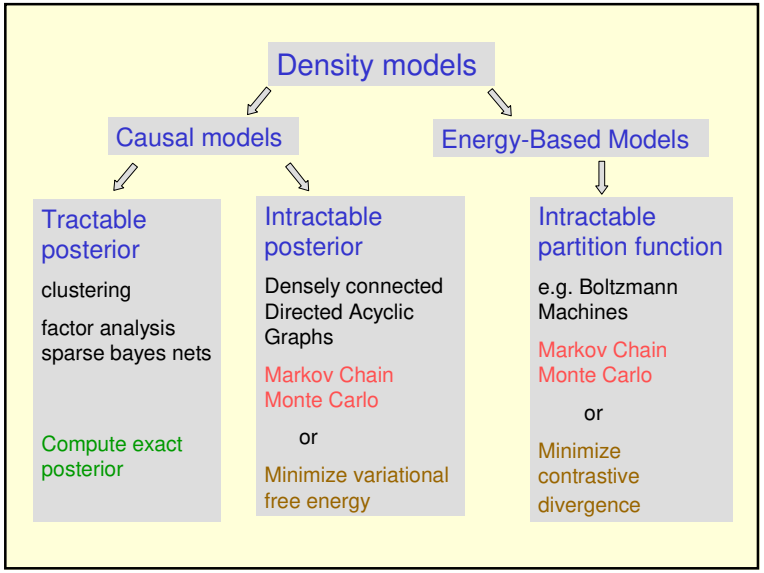
The standard way to find the serious rivals

- Use Markov Chain Monte Carlo
  - This means: Fiddle with the states of both the visible and the hidden neurons in a stochastic way that tends to lower the energy of the overall configuration of neural activities.
- The stochastic fiddling eventually generates rivals with a probability that depends on their energy in just the right way.
  - But it is very, very slow.

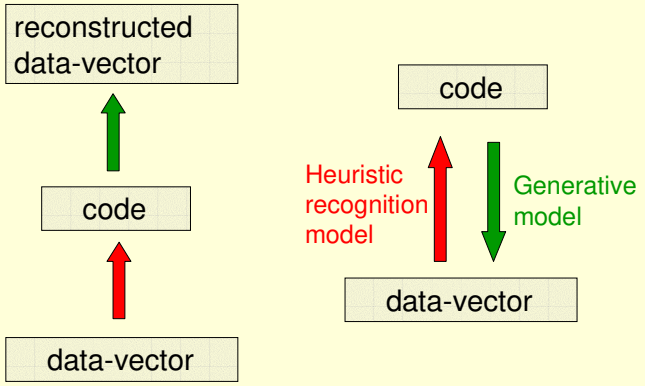
A very surprising shortcut

- Instead of getting the correct probability distribution over rivals by running a Markov chain until it reaches the equilibrium distribution, use slight corruptions of the data-vectors as the rivals (Bartlett's experiments)
- Learn by modifying the weights so as to lower the energy of the true data and raise the energy of the slightly corrupted data.
- The learning stops when the model's prejudices do not corrupt the data in any systematic way.
  - Its OK if unlikely data-vectors are usually corrupted into more likely ones.
  - This is exactly balanced by corrupting a small fraction of the likely data-vectors into unlikely ones.





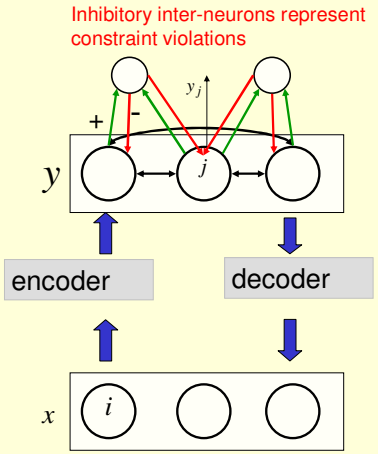
Autoencoders as generative models



Ways to restrict the code

- Only use a few hidden units: PCA
- Only allow one hidden unit to be on: Clustering
- Impose a cost for each active hidden unit: ICA
- Use lots of code units but squeeze the redundancy out of the code-space.
  - Give each code an energy and learn an energy-based model that makes almost all of the possible codes highly improbable.

A learning module



$$p(y) = \frac{e^{-E(y)}}{\sum_u e^{-E(u)}}$$

$$p(x \mid y) = e^{-\|x - \hat{x}(y)\|^2}$$

$$p(x) > p(y) \quad p(x \mid y)$$

Prior probability of code that x selects for itself

probability of data given code

Bottom-up and top-down weights can be learned without considering the partition function

$$p(x) > \frac{e^{-E(y)}}{\sum_u e^{-E(u)}} \quad e^{-\|x-\hat{x}\|^2}$$

$$\log p(x) > -E(y) - \log Z \; - \| x - \hat{x} \|^2$$

The partition function of the hidden units is not affected by the bottom-up and top-down weights. It depends only on the lateral interactions between hidden units.

A picture of the learning procedure

1. The bottom-up input causes an initial state of the code units.
2. This initial state settles to a state that takes the code energies into account.
3. Then we ignore the bottom-up input and do a little more settling to get the corruption.

To learn the feed-forward connections:

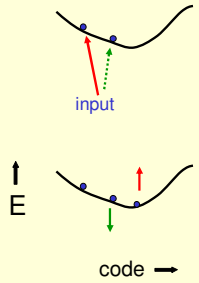
the settled state is the target output

the initial state is the actual output.

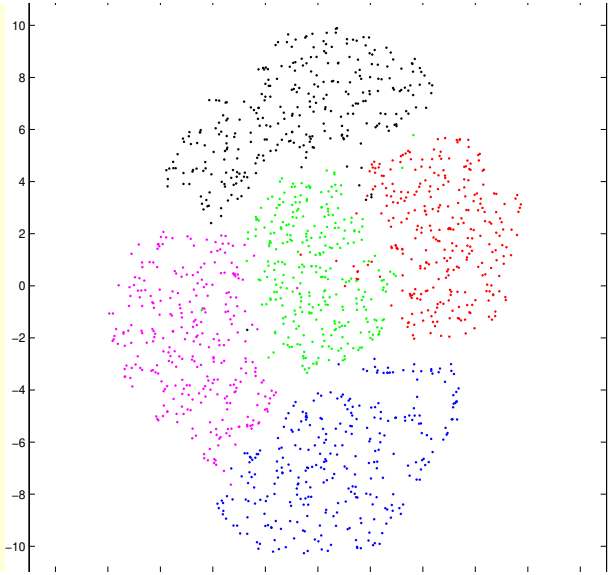
To learn the lateral connections:

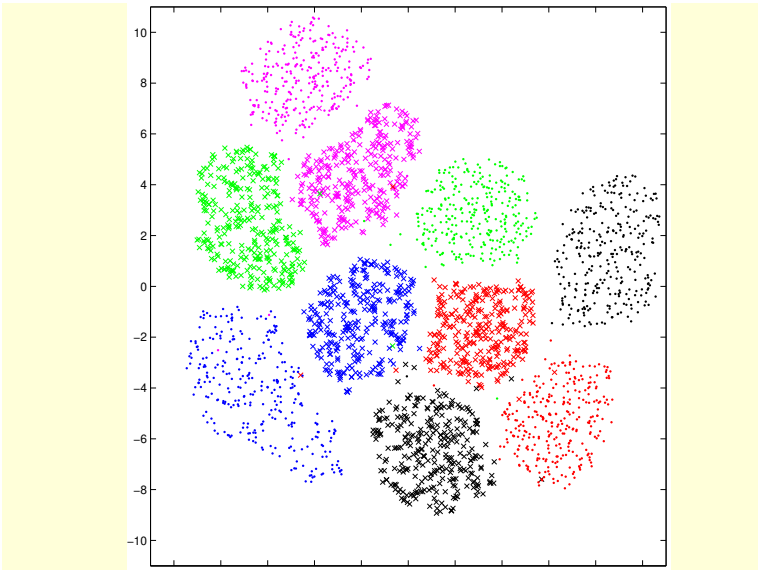
the energy of the settled state is lowered

the energy of the final state is raised.



THE END




10
8
6
4
2
0
−2
−4
−6
−8
−10