Neural Networks

Lecture 17
Boltzmann Machines as Probabilistic Models

Modeling binary data

- Given a training set of binary vectors, fit a model that will assign a probability to other binary vectors.
  - Useful for deciding if other binary vectors come from the same distribution.
  - This can be used for monitoring complex systems to detect unusual behavior.
  - If we have models of several different distributions it can be used to compute the posterior probability that a particular distribution produced the observed data.

$$p(Model\ i \mid data) = \frac{p(data \mid Model\ i)}{\sum_j p(data \mid Model\ j)}$$

A naïve model for binary data

For each component, j, compute its probability, p_j, of being on in the training set. Model the probability of test vector alpha as the product of the probabilities of each of its components:

$$p(\mathbf{s}^{\alpha}) = \prod_{j} \left(s_j^{\alpha} p_j + (1 - s_j^{\alpha})(1 - p_j) \right)$$

Binary vector alpha

If component j of vector alpha is on

If component j of vector alpha is off

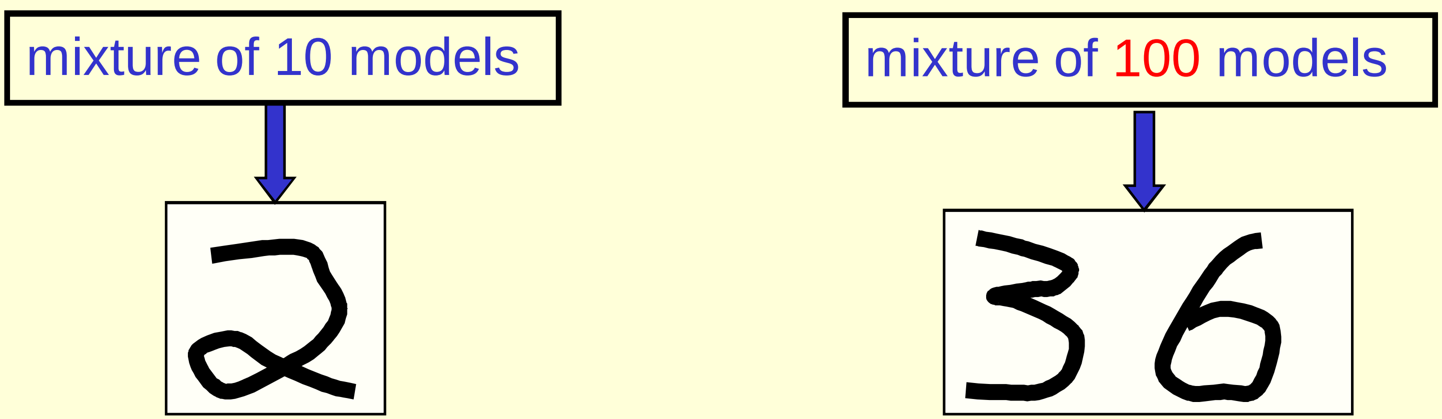
A mixture of naïve models

- Assume that the data was generated by first picking a particular naïve model and then generating a binary vector from this naïve model.
  - This is just like the mixture of Gaussians, but for binary data.

$$p(\mathbf{s}^{\alpha}) = \sum_{m \in Models} \pi_m \prod_j \left(s_j^{\alpha} p_j^m + (1 - s_j^{\alpha})(1 - p_j^m) \right)$$

Limitations of mixture models

- Mixture models assume that the whole of each data vector was generated by exactly one of the models in the mixture.
  - This makes is easy to compute the posterior distribution over models when given a data vector.
  - But it cannot deal with data in which there are several things going on at once.



Dealing with compositional structure

- Consider a dataset in which each image contains N different things:
  - A distributed representation requires a number of neurons that is linear in N.
  - A localist representation (i.e. a mixture model) requires a number of neurons that is exponential in N.
    - Mixtures require one model for each possible combination.
- Distributed representations are generally much harder to fit to data, but they are the only reasonable solution.
  - Boltzmann machines use distributed representations to model binary data.

How a Boltzmann Machine models data

- It is not a causal generative model (like a mixture model) in which we first pick the hidden states and then pick the visible states given the hidden ones.

- Instead, everything is defined in terms of energies of joint configurations of the visible and hidden units.

The Energy of a joint configuration

binary state of unit i in joint configuration alpha, beta

$$E^{\alpha\beta} = -\sum_{i \in units} s_i^{\alpha\beta} b_i - \sum_{i<j} s_i^{\alpha\beta} s_j^{\alpha\beta} w_{ij}$$

Energy with configuration alpha on the visible units and beta on the hidden units

bias of unit i

weight between units i and j

indexes every non-identical pair of i and j once

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.

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

$$p(\mathbf{v}^{\alpha}, \mathbf{h}^{\beta}) = \frac{e^{-E^{\alpha\beta}}}{\sum_{\gamma\delta} e^{-E^{\gamma\delta}}}$$

partition function

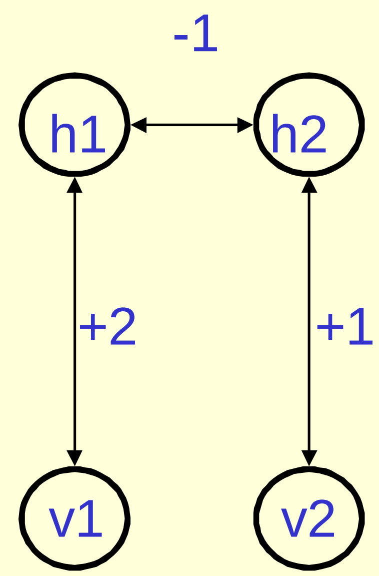
configuration alpha on the visible units

$$p(\mathbf{v}^{\alpha}) = \frac{\sum_{\beta} e^{-E^{\alpha\beta}}}{\sum_{\gamma\delta} e^{-E^{\gamma\delta}}}$$

An example of how weights define a distribution

$\mathbf{v}$	$\mathbf{h}$	$-E$	e^{-E}	$p(\mathbf{v},\mathbf{h})$	$p(\mathbf{v})$
1 1	1 1	2	7.39	.186	
1 1	1 0	2	7.39	.186	0.466
1 1	0 1	1	2.72	.069	
1 1	0 0	0	1	.025	
1 0	1 1	1	2.72	.069	
1 0	1 0	2	7.39	.186	0.305
1 0	0 1	0	1	.025	
1 0	0 0	0	1	.025	
0 1	1 1	0	1	.025	
0 1	1 0	0	1	.025	0.144
0 1	0 1	1	2.72	.069	
0 1	0 0	0	1	.025	
0 0	1 1	-1	0.37	.009	
0 0	1 0	0	1	.025	0.084
0 0	0 1	0	1	.025	
0 0	0 0	0	1	.025	

total =39.70



Getting a sample from the model

- If there are more than a few hidden units, we cannot compute the normalizing term (the partition function) because it has exponentially many terms.
- So use Markov Chain Monte Carlo to get samples from the model:
  - Start at a random global configuration
  - Keep picking units at random and allowing them to stochastically update their states based on their energy gaps.
  - Use simulated annealing to reduce the time required to approach thermal equilibrium.
- At thermal equilibrium, the probability of a global configuration is given by the Boltzmann distribution.

Getting a sample from the posterior distribution over distributed representations for a given data vector

- The number of possible hidden configurations is exponential so we need MCMC to sample from the posterior.
  - It is just the same as getting a sample from the model, except that we keep the visible units clamped to the given data vector.
    - Only the hidden units are allowed to change states
- Samples from the posterior are required for learning the weights.