CSC321: Introduction to Neural Networks and Machine Learning

Lecture 17: Boltzmann Machines as Probabilistic Models

Geoffrey Hinton
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.

\[ p(\text{Model } i \mid \text{data}) = \frac{p(\text{data} \mid \text{Model } i)}{\sum_j p(\text{data} \mid \text{Model } j)} \]
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:

$$p(s^\alpha) = \prod_j \left( s_j^\alpha p_j + (1 - s_j^\alpha)(1 - p_j) \right)$$

- 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.

\[
p(s^\alpha) = \sum_{m \in \text{Models}} \prod_{j} \left( s_j^\alpha p_j^m + (1 - s_j^\alpha)(1 - p_j^m) \right)
\]
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 it 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

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

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.

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

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

\[
p(v^\alpha) = \frac{\sum_{\beta} e^{-E^{\alpha\beta}}}{\sum_{\gamma\delta} e^{-E^{\gamma\delta}}}
\]
An example of how weights define a distribution

<table>
<thead>
<tr>
<th>v</th>
<th>h</th>
<th>-E</th>
<th>$e^{-E}$</th>
<th>$p(v, h)$</th>
<th>$p(v)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>7.39</td>
<td>.186</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>7.39</td>
<td>.186</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2.72</td>
<td>.069</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>.025</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>7.39</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2.72</td>
<td>.069</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>.025</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2.72</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>0.37</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Total** = 39.70

Diagram:
- **h1** connected to **h2** with weight 0.466
- **v1** connected to **v2** with weight 0.305
- **v1** connected to **h2** with weight 0.144
- **v1** connected to **v2** with weight 0.084

**Note:** The weights are calculated using the formula $p(v, h) = e^{-E}$, where $E$ is the energy term.
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.