Assignment 1

David Nagy

February 12, 2012

1 Bernoulli ML

The observed variables x_i are 64 dimension vectors where $x_{ij} \in \{0, 1\}$, and each x_{ij} is assumed to be from a Bernoulli distribution for which the only parameter is the p_j probability of success (of the bernoulli trial) for each pixel. Here success means that $x_{ij} = 1$.

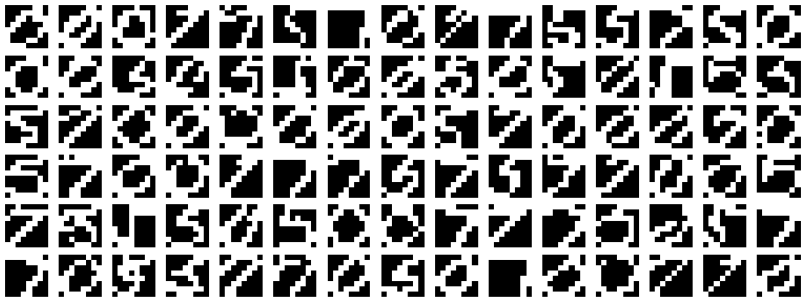

Figure 1: Training data

Since it follows from the law of large numbers that the relative frequency of a Bernoulli trial's outcome approaches the probability of that outcome as the number of trials approach infinity (if the trials are independent), we can determine the value of the p_j parameters by taking the average number of successes for each pixel, and these will approach the true success probability ('true' meaning that this would be the actual value of p_j, if the pixels indeed came from a bernoulli process, which in this case is clearly not true. This limits the usefulness of this model).

It can be shown that these are the same parameters that maximize the likelihood function: We can consider the multiple Bernoulli trials as a binomial distribution for which the likelihood function is

$$L(p) = f(y|p) = \begin{pmatrix} n \\ y \end{pmatrix} p^y (1-p)^{n-y}$$

where n is the number of trials and y is the number of successes. We have to maximize this function:

$$\frac{\partial}{\partial p}\left(\left(\begin{array}{c} n \\ y \end{array}\right) p^y (1-p)^{n-y}\right) = 0$$

from which

$$p^{y-1}(1-p)^{n-y-1}(y-np) = 0$$

thus the non-trivial solution is

$$p = \frac{y}{n}$$

Hence, the bernoulli parameter matrix is the average of all the training data, shown here:

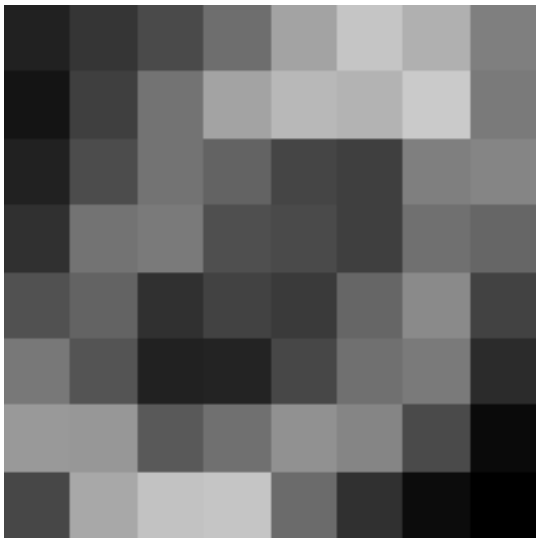

Figure 2: Bernoulli parameters

2 Boltzmann distribution

The exact calculation of the maximum likelihood parameters for a Boltzmann distribution is unfeasible and for this reason we were instructed to approximate them using Monte Carlo methods, specifically Gibbs sampling. This is implemented by the dynamics of the Boltzmann machine.[1]

[1] From the slides of lecture 3, page 60.

2.1 Time evolution of the state of the BM

At each step the state of one dimension of the state vector (which in this case is a pixel) can be updated. This means that at each step a Bernoulli trial is performed where the probability of the new state being 1 is

$$P\left(s_i = 1\right) = \frac{1}{1 + e^{-z_i}}$$

where

$$z_i = b_i + \sum s_j w_{ij}$$

Using this updating rule, the network eventually reaches equilibrium, where the probability of a state is determined by the relative energy of that state to other states and the network is sampling from a Boltzmann distribution.

2.2 Learning rule for the parameters of the BM

To create a model of the environment from which the training data is coming from, the network has to fit the parameters of the Boltzmann distribution it is sampling from. This is achieved by using the following learning rules:[2]

$$\Delta w_{ij}(t) = \epsilon \left(r_i r_j - s_i s_j\right)$$

for the weights and

$$\Delta b_i(t) = \epsilon \left(r_i - s_i\right)$$

for the bias.

These mean that in each cycle of learning, a data point from the training set is compared with a fantasy image generated by the network. I have used the same learning parameter $\epsilon = 0.01$ for both learning rules.

Outline of the learning cycle

1. Choose a training data point.
2. Generate an image by running the machine with the current parameters.
3. Update the weight matrix by comparing the training data with the fantasy data, using the learning rule above.
4. Update the bias vector.
5. Repeat.

After repeating this cycle for the whole dataset 20 times, the weight matrix and the bias vector is shown on Fig. 3 and 4 (black is -0.7 white is +0.7).

[2] From the slides of lecture 3, page 70.

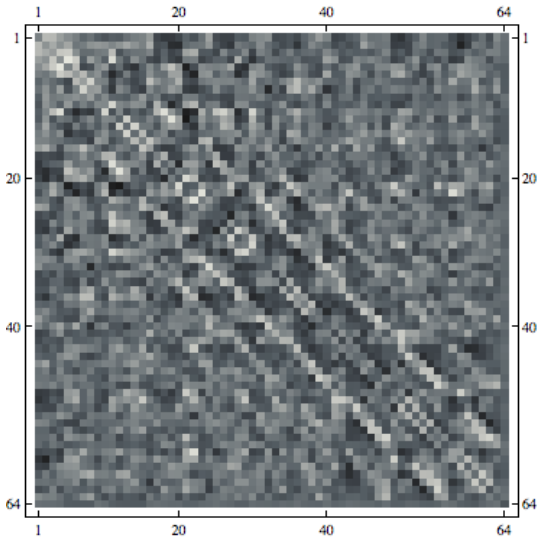

Figure 3: Weight matrix

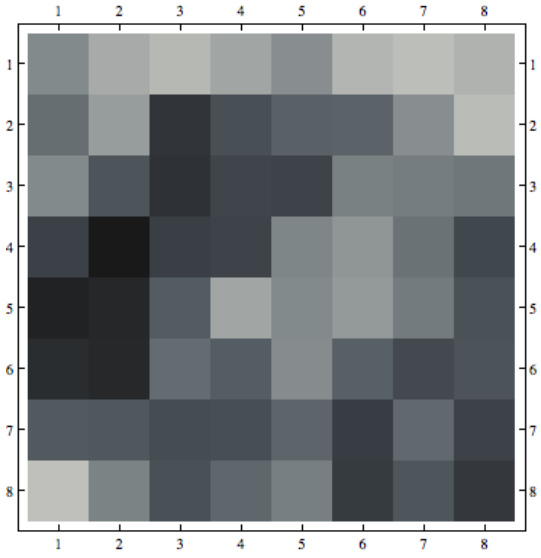

Figure 4: Bias vector

3 Fantasies

These are fantasy data generated by running the BM according to the time evolution described in the previous section. There is at least 200 steps between each image[3] to guarantee their independence.

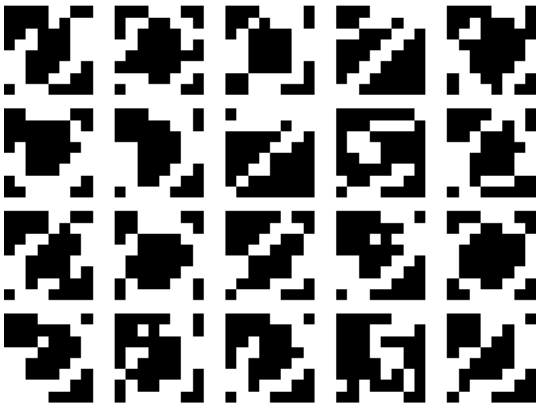

Figure 5: Fantasies

4 Pattern completion

There were three kinds of qualitatively different data in the training set, namely images of 0-s, 5-s and 7-s. I have chosen one of each and deleted half of the image, then ran the machine to see if it could complete the image with the known half fixed. The data points I have chosen were the 3rd, 4th and 14th for the 0-s, 7-s and 5-s respectively. I have set new initial conditions before each sample then let the machine 'dream' for 500 steps. The results are below:

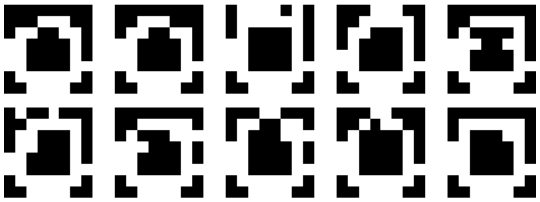

Figure 6: List of 0-s

[3] All of the pixels are updated within a step.

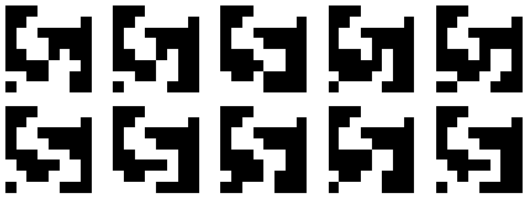

Figure 7: List of 5-s

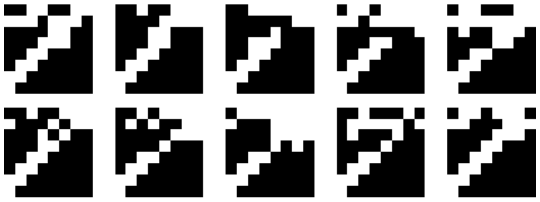

Figure 8: List of 7-s

5 Discussion of differences

Pattern completion is not possible in the Bernoulli model, because since all the pixels are independent, the clamping of some of them have no effect whatsoever on the other pixels. In the Boltzmann machine, the weight structure allows one pixel to influence another.

The 'fantasies' of the Bernoulli model will just be noisy versions of the parameter matrix, whereas the Boltzmann machine is able to learn multiple energy minimums into which it can settle. This means that it can learn qualitatively different kinds of data, and its fantasies usually fall in one of these categories.

I'm sorry, we didn't have time for the last one because this was our ski break.