EE6530 Neural Networks

Homework #6: Restricted Boltzmann Machine (RBM) and Contrastive Divergence Learning, due 5/28/2015 in class.

In this homework, you are asked to build an RBM that performs unsupervised learning. The RBM shall consist of a visible layer and a hidden layer. You shall test your machine on the MNIST hand-written digits and tune the machine so it learns to detect features.

My code will be provided a week from now (on May 21). To implement it on your own, I recommend you to refer to Hinton Lecture 12c and d. When updating the weights using the contrastive divergence rule

$$\Delta W_{ij} = \varepsilon\left(\langle v_i h_j\rangle^0 - \langle v_i h_j\rangle^1\right),$$

make sure that you estimate the expected values $\langle v_i h_j\rangle^0$ and $\langle v_i h_j\rangle^1$ prudently, for example, using mini-batches or otherwise repeating experiments. In my own implementation, I managed to get it to work by hand-calculating $\langle v_i h_j\rangle^0$ but not $\langle v_i h_j\rangle^1$; I intended to obtain $\langle v_i h_j\rangle^1$ by averaging across several repeats. However, in practice I found simply set $\langle v_i h_j\rangle^1 \simeq v_i^1 h_j^1$ works quite well, and I am still puzzled. I haven't tried mini-batches, fantasy particles, or mean-field approximations at all.

Other details that matter include (1) how to initialize your weight, and (2) how the learning rate ε affects the results. I'd be glad to hear your comments in the report.

Some typical results of W_{ij} are shown below.

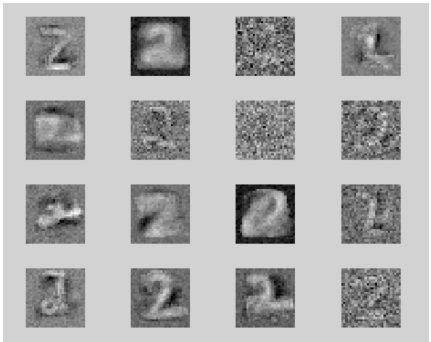

Fig. 1: Learning to see '2'

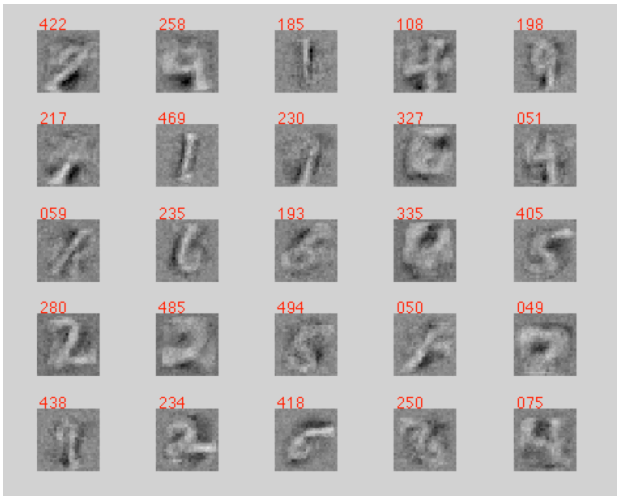

Fig. 2: learning to see all the digits.

In Fig. 1, I trained the network to 'see' the digit 2. Each panel shows the final weights for a different hidden node. Note that some of the neurons do not learn successfully so their weights remain random.

In Fig. 2, I trained the network to see all the digits. The training process involved feeding the network with examples in a random order. This particular figure shows the weights for 25 out of 500 nodes. The red number on top of each panel is the serial number of the node. I selected the "most active" nodes to monitor based on the maximum weights they have. It is puzzling that my results look very different from Hinton's results on the last slide of Lecture 12d. I'd be curious what you get.

In your report, you may also want to demonstrate that your machine has the capability to regenerate images that look similar to what it has seen. I will talk about this in class.

先這樣。Enjoy teaching your net!

Additional remarks:

A. To conduct a Bernoulli trial, you can do the follows,

```
p = 1/(1+exp(-DeltaE)); % probability(h==1)
tmp = rand(); % a random number subject to uniform distribution in [0,1]
h = (tmp < p);
```

B. Images are two dimensional, so $E = -\Sigma v_i w_{ij} h_j$ really is $E = -\sum v_{ij} W_{ijk}\, h_k$ here. Listed below are a few functions you may find helpful:

- squeeze(): to eliminate "singleton" dimensions for 3D or higher dimensional arrays.
- reshape(): to convert a vector to a matrix or a tensor, and vice versa.

C. Please use the MNIST digits (same as in HW4 and HW5) as your test materials.