CSC2515 Assignment 2
Due Nov 5 2008, 1.00pm at START of class

Andriy Mnih and Geoffrey Hinton

October 29, 2008

This assignment will be graded out of 30, so each point is worth 0.5% of your final grade.
Late assignments will have 25% subtracted from the total out of which they are graded for each day or part of a day that they are late. They will be more than one day late if they are not slipped under my office door (LP 290G) before 1.00pm the next day.

1. (2 points)

a) What form does the posterior distribution over the sources have in the square, noiseless version of the Independent Component Analysis model? Justify your answer.

b) Now consider the noisy square version of ICA, where noise is added to each of the linear combinations of the sources before it is observed as data. What form does the posterior over the sources have in this model? Justify your answer.

2. (3 points)

a) Show that replacing the spherical Gaussian distribution over the latent values in the Factor Analysis model with a full-covariance Gaussian does not make the model any more powerful.

b) How does this modification affect the expressive power of the PPCA model?

c) Does allowing the distribution of noise in the PPCA model to have a full covariance matrix make sense from the modelling standpoint? Explain.

3. (5 points)

a) In a restricted Boltmann machine, the "free energy" of a visible vector, v, can be defined in two apparently different ways:

$$e^{-F(v)} = \sum_h e^{-E(v,h)} \tag{1}$$

$$F(v) = \sum_h p(h|v)E(v,h) - \sum_h p(h|v) log \frac{1}{p(h|v)} \tag{2}$$

Show that these two definitions are equivalent.

b) Based on the second way of defining $F(v)$ write a short matlab program that computes F(v) in a time that is linear in the number of hidden units. Do not forget the biases. Hand in the matlab code.

4. (20 points)
This is a new assignment. Check the assignment webpage on thursday and friday to see if people found bugs that required the assignment to be modified.

Download the code for training a standard RBM from

`http://www.cs.toronto.edu/~hinton/csc2515/matlab/simple_rbm.m`

Modify the code so that the visible vector consists of some pixels plus a single “softmax” variable that can have one of five discrete states. Its easiest to introduce a new variable called “lab” with its own “labbiases” and “labhid” weights. When you reconstruct the state of this variable you could reconstruct a sampled discrete state (e.g [0,1,0,0,0] for a two) or you could cheat the same way we do with the pixels by setting each of the five labels to its probability instead of setting a single label to 1 (e.g. [0.05, 0.91, 0.00, 0.00, 0.04] if its probably a two but possibly a one or a five). To compute the label probabilities, first compute $\exp(x_k)$ for each label k and then set the probability $\exp(x_k)/\sum_i \exp(x_i)$, where x_k is the bias of label k plus the total input to k from the hidden units.

Use the modified code to train a joint density model of images plus labels. The batchdata for training and validation and final testing can be downloaded from

`http://www.cs.toronto.edu/~hinton/csc2515/matlab/assign3data.mat`

Each batch consists of 50 images that have been conveniently divided into 10 cases each of the digits 1,2,3,4,5. The corresponding labels are in separate variables called batchlabels etc. After loading the .mat file type *whos* to see what got loaded.

After training the joint density model, use the validation data to see how useful your model is for classification. To classify an image, m, plug in each of the five possible labels in turn and compute the free energy of that combination of label and image. Then assign a probability to the label k that is proportional to $\exp(-F(k, m))$.

Using the validation data, search for a number of hidden units and a number of epochs of training that gives the best results and then report the results on the test set for that many hidden units trained for that many epochs. You do not need to do a very careful search for the best number of hidden units or training time, but it would be good to show that a big increase or decrease in either number makes the performance worse on the validation set.

Do not change the training, validation or test sets. You can obviously get better results by using more of the data for training, but for an assignment its better to have less training data and more testing data so that it overfits quite quickly without requiring too many hidden units and you can measure how well it does quite accurately. The learning rate, momentum, weightcost and initial weight values in the code you downloaded should be OK and there is no need to experiment with these.

What to hand in:

1. Hand in less than a page that briefly describes what you did and anything interesting you noticed. This should include the number of hidden units and training epochs you used for the final test run and also what test error rate you got. Do not spend a long time trying to slightly reduce the error rate.
2. As the RBM is being trained, get it to plot a graph showing its error rate on the validation set every few epochs. Hand in these graphs for all of the RBM's that you trained (there may be quite a few!). Make sure the graphs are clearly labelled. It would be good to make the title of the graph include your name and the number of hidden units. Do *help title* to find out how to make titles and *help plot* to find out how to make plots.