CSC2515 Assignment 3
Due: Nov 20 2007, 11am at START of class

November 7, 2007

This is the final version of the programming part of assignment 3

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 **Prof. Zemel's** *office door before 11.00am the next day.*

1 PCA and Factor Analysis

(a). How do the factors found by running EM to optimize a probabilistic PCA model relate to the sample covariance matrix of the data? Support your answer. (2 points)

(b). Derive the updates for the parameters in the M step when using EM to optimize a factor analysis model. You can use the expression for the expected complete-data log likelihood from the lecture notes for this derivation. (2 points)

2 Using unlabeled data and greedy learning to improve discrimination

There are many datasets in which only a small fraction of the cases have labels. We would like to use the unlabeled cases to help us learn good features. Then the labeled cases only need to provide enough information to fine-tune the features we already learned (see lecture 8). You will be comparing the performance of three different training methods. For each training method you will be trying **both** 100 and 200 hidden units. To compare performance you should use **both** the number of errors and the negative log probability of the correct answer (i.e. the cross-entropy error that is normally used with a softmax (see lecture notes).

You will use the same data as in assignment 1, but the dataset has been divided up differently. There are 1000 labeled training cases (as in assignment 1), 5000 unlabeled training

cases, and 5000 (labeled) test cases. All of these subsets have equal numbers of each of the ten classes. To load the training and test sets, invoke matlab and load the file
`http://www.cs.toronto.edu/~hinton/csc2515/matlab/assign3v6.mat`
This should work for all versions of matlab from version 6 up.

The code you will need for training RBM's is at:
`http://www.cs.toronto.edu/~hinton/csc2515/matlab/rbm.m`
The code assumes your training data has been divided into minibatches. You could do this, or you could modify the code. If you do use minibatches, make sure that each minibatch has equal numbers of each class.

1. First train a restricted Boltzmann machine (once with 100 and once with 200 hidden units) on the 6000 labeled + unlabeled training cases (but without using the labels). Code is provided for this. Do not change the weightcost parameter. Then use multinomial logistic regression on the 1000 labeled training cases, with the inputs being the real-valued activation probabilities of the hidden units of the RBM. (Hint: multinomial logistic regression is just backprop with softmax output units but no hidden layer, so you just need a simplified version of the code you wrote for assignment 1.)

2. Starting with the weights and biases found by method 1, use standard backpropagation with one hidden layer to fine-tune all the weights learned in part 1. The backpropagation only uses the labeled training cases.

3. Use backpropagation in two networks with exactly the same architectures as in part 2, but start with small random weights and zero biases. So, for this method, you do not make use of the unlabeled training data.

To reduce the amount of fiddling around required, you are **not allowed** to use weight-decay in any of the methods (except for the weight-decay that comes with the code provided when you are training the RBM).

You should submit:

- (4 points) One page or less containing a clear description of the results you obtained for each of the three methods with each of the two numbers of hidden units. The one page should include the results for all 3×2 results.

- (8 points) One page or less discussing what you think these results show and describing **exactly** four other experiments you would do to make sure that your conclusions were correct.