HW Assignment 4 (Due by 10:30am on Oct 15)

1 Theory (100 points)

1. Show that Logistic Regression is a special case of Softmax Regression. That is to say, if \( w_1 \) and \( w_2 \) are the parameter vectors of a Softmax Regression model for the case of two classes, then there exists a parameter \( w \) for Logistic Regression that results in the same classification as the Softmax Regression model.

2. Show that a Perceptron and a Logistic Regression model that have the same parameter vector \( w \) output the same label for an arbitrary test example \( x \) i.e., the Perceptron and the Logistic Regression model are equivalent at test time.

3. Write the Forward Propagation equations for a general neural network with 2 hidden layers, in vectorized form. Use the notation introduced in class.

4. Write down the cost function \( J \) associated with a neural network and a set of \( m \) training examples, where the cost has two components: the sum of square errors and the regularization term. Compute the gradient of \( J \) with respect to the weights of the connections between the last 2 layers.

5. Consider a dataset \( X \) containing 3 examples, as follows:

\[
X = \begin{pmatrix}
1 & -2 & 4 \\
-1 & 2 & -4
\end{pmatrix}
\]

Manually run PCA on this dataset and compute the matrix \( U \) whose columns are the principal components of \( X \). What is the minimum number of components \( k \) that are needed to preserve at least 99% of the variance? Show the projection of \( X \) on the corresponding subspace of \( k \) principal components. Make sure you show all your work.

6. **[Fine Tuning in Self-Taught Learning]** Consider a self-taught learning model with 1 hidden layer and a softmax output layer, that is trained to minimize the regularized negative log-likelihood cost function in which weight decay is applied only to the softmax parameters. Write down the equations for the forward propagation and backpropagation steps, vectorized and non-vectorized.

7. **[Tied Weights]** Write down the gradient computation for a (non-linear) sparse auto-encoder with tied weights i.e., \( W^{(2)} = (W^{(1)})^T \). Do the same for the linear auto-encoder from the previous assignment.

2 Implementation (100 points)

Download the skeleton code from http://ace.cs.ohio.edu/~razvan/courses/dl6900/hw/hw04.zip. Implement the Self-Taught learning example, as explained in the UFLDL exercise. Make sure that you organize your code in folders as shown in the table below.

Write code in the files indicated in bold. You are encouraged to reuse the code that you have written for the previous assignments.
**Bonus 1:** Fine tune the self-taught learning model, using the backpropagation equations you derived in the theory part above. Evaluate the fine tuned model and compare its performance with the original model.

**Bonus 2:** Implement the non-linear and linear auto-encoders with tied weights, using the backpropagation equations you derived in the theory part above. Compare them with the original sparse auto-encoder that you implemented for the previous assignment, especially in terms of the features they learn.

### 3 Submission

Turn in a hard copy of your homework report at the beginning of class on the due date. Electronically submit a directory that contains the required files. Make sure your code runs correctly when used in the architecture shown above. Create a ZIP archive of your directory, and upload it on Blackboard by the due date.

Please observe the following when handing in homework:

1. Structure, indent, and format your code well.

2. Use adequate comments, both block and in-line to document your code.

3. On the theory assignment, clear and complete explanations and proofs of your results are as important as getting the right answer.