HW Assignment 4 (Due by 9:00am on Mar 16)

1 Theory (100 points)

1. [Self-Taught Learning, 50 points] 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. Derive the equations for the forward propagation steps and the backpropagation steps needed to compute the gradients, in vectorized form.

2. [Tied Weights, 50 points] 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 (50 + 50 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 below, using (1) SciPy and (2) TensorFlow. Make sure that you organize your code in folders as shown in the table below.

dl6890/
  hw04/
    scipy/
      stlExercise.py
      softmax.py
      sparseAutoencoder.py
      feedForwardAutoencoder.py
      computeNumericalGradient.py
      trainSoftmax.py
      trainAutoencoder.py
      output.txt
      weights_digits.jpg
      display_network.py

tensorflow/
  stlExercise.py
  softmax.py
  sparseAutoencoder.py
  feedForwardAutoencoder.py
  trainSoftmax.py
  trainAutoencoder.py
  output.txt
  weights_digits.jpg
  display_network.py

mnist/
Write code only in the files indicated in bold. For `softmax.py` and `sparseAutoencoder.py`, you are encouraged to reuse the code from previous assignments. For Scipy, these can be exact copies of the previous ones. For TensorFlow, the skeleton code contains additional statements for saving and restoring the model parameters, through a `Saver` object. The visualization of the hidden units should be saved in the `weights_digits.jpg` files listed above, whereas the trace from running `python3 stlExercise.py` should be saved in `output.txt`.

**Step 1: Generate the unlabeled and labeled datasets:** In this exercise, our goal is to distinguish between the digits from 0 to 4. We will use the digits 5 to 9 as our "unlabeled" dataset which to learn the features; we will then use a labeled dataset with the digits 0 to 4 with which to train the softmax classifier. The corresponding datasets are created for you by code in `stlExercise.py`.

**Step 2: Train the sparse autoencoder:** Next, use the unlabeled data to train a sparse autoencoder. This is done by calling the function `run_training()` from the `trainAutoencoder.py` module. For this to work, you need to write code in `sparseAutoencoder.py`, for which you can reuse the code written for second assignment. When training is complete, you should get a visualization of pen strokes like the image shown below. This step will also save the trained encoder parameters on disk.

![Figure 1: Features learned from digits 5 to 9, using the sparse autoencoder.](image)

**Step 3: Computing features:** After the sparse autoencoder is trained, you will use it to extract features from the handwritten digit images, where the feature parameters will be read from the disk. Complete `feedForwardAutoencoder.py` to produce a matrix whose columns correspond to activations of the hidden layer for each example, i.e., the vector \( a^{(2)} \) corresponding to activation of layer 2. After completing this step, calling `feedForwardAutoencoder()` in `stlExercise.py` should convert the raw image data into the hidden unit activations \( a^{(2)} \).

**Step 4: Training and testing the softmax regression model:** Train a softmax classifier using the training set features `train_features` and labels `train_labels`. This is done by calling the function `run_training()` from the `trainSoftmax.py` module.
Step 5: Evaluating the model on the test set: Finally, complete the code to make predictions on the test set \texttt{test\_features} and see how your learned features perform. If you’ve done all the steps correctly, you should get an accuracy of about 98% percent. As a comparison, when raw pixels are used (instead of the learned features), we obtained a test accuracy of only around 96% (for the same train and test sets).

3 Bonus (50 + 25 points)

Bonus 1: Fine tune the self-taught learning model in \texttt{SciPy} and \texttt{TensorFlow}. For \texttt{SciPy} use 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 in \texttt{SciPy}, 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.

4 Submission

Turn in a hard copy of your homework report at the beginning of class on the due date. Electronically submit on Blackboard a hw04.zip file that contains the hw04 folder in which you change code \textbf{only in the required files}. The screen output produced when running the \texttt{stlExercise.py} code should be redirected to (saved into) the \texttt{output.txt} files.

On a Linux system, creating the archive can be done using the command:

\begin{verbatim}
> zip -r hw04.zip hw04
\end{verbatim}

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.
4. Make sure your code runs correctly when used in the directory structure shown above.