HW Assignment 1 (Due by 10:00am on October 9)

1 Implementation (100 points for softmax.py, additional 20 points as bonus for computeNumericalGradient.py)

Implement the softmax regression model in Python, using NumPy, and evaluate them on two 2D non-linear classification tasks: flower and spiral. Starter code and functions for generating the datasets are available in hw1.zip on Cougar Course. The provided code also displays and saves images of the datasets and the trained model's decision boundaries. Make sure that you organize your code in folders as shown in the table below. Write code only in the Python files indicated in bold.

hw1/
- **softmax.py**
- **computeNumericalGradient.py**
- softmaxExercise.py
- utils.py
- flower-data.jpg
- flower-boundary.jpg
- spiral-data.jpg
- spiral-boundary.jpg

NumPy Implementation (100 points for softmax.py, additional 20 points as bonus for computeNumericalGradient.py)

Coding effort: my implementation has 11 lines of code in softmax.py and 8 lines of code in computeNumericalGradient.py.

1. **Cost & Gradient:** You will need to write code for two functions in sofmax.py:
   (a) The softmaxCost() function, which computes the cost and the gradient.
   (b) The softmaxPredict() function, which computes the softmax predictions on the input data.

The cost and gradient should be computed according to the formulas shown on the slides in Lecture 4, modified however to represent explicitly the bias and its gradient. Thus, if there are D features and K classes, the softmax model will be comprised of two types of parameters: **W** will be a K x D matrix of the feature weights, whereas **b** will be a K x 1 vector of bias terms.

2. **Vectorization:** It is important to vectorize your code so that it runs quickly.

3. **Ground truth:** The groundTruth is a matrix M such that M[c, n] = 1 if sample n has label c, and 0 otherwise. This can be done quickly, without a loop, using the SciPy function sparse.coo matrix(). Specifically, coo matrix((data, (i, j))) constructs a matrix A such that A[i[k], j[k]] = data[k], where the shape is inferred from the index arrays. Sample code for computing the ground truth matrix has been provided in Lecture 1.

4. **Overflow:** Make sure that you prevent overflow when computing the softmax probabilities, as shown on the slides in Lecture 4.

5. **Numerical gradient:** Once you implemented the cost and the gradient in softmaxCost, implement code for computing the gradient numerically in computeNumericalGradient.py, as shown on the slides in Lecture 4.

6. **Gradient checking:** Use computeNumericalGradient.py to make sure that your softmaxCost.py is computing gradients correctly. This is done by running the main program in Debug mode, i.e. python3 softmaxExercise.py --debug.

   In general, whenever implementing a learning algorithm, you should always check your gradients numerically before proceeding to train the model. The norm of the difference between the numerical gradient and your analytical gradient should be small, on the order of 10^{-9}.

7. **Training:** Training your softmax regression is done using gradient descent for 200 epochs.

8. **Testing:** Now that you've trained your model, you will test it against the training set to determine how well the softmax can fit this dataset. To do so, you will first need to complete the function softmaxPredict() in softmax.py, a function which generates predictions for input data under a trained softmax model. Once that is done, you will be able to compute the accuracy of your model using the code provided.

2 Submission

Electronically submit on Cougar Course a hw1.zip file that contains the hw1 folder in which you write code **only in the required files**.

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

```
> zip -r hw1.zip hw1.
```

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