CSCE 636: Neural Networks (Fall 2018)
Assignment #1
Due 09/17/2018

---

1. You need to submit a report in hard-copy before lecture and your code to Blackboard.
2. Hard-copy is due in class before lecture and electronic copy is due 11:30AM on Blackboard on the due date.
3. Unlimited number of submissions are allowed on Blackboard and the latest one will be graded.
4. Please read and follow submission instructions. No exception will be made to accommodate incorrectly submitted files/reports.
5. Only write your code between the following lines. Do not modify other parts.

   ### YOUR CODE HERE

   ### END YOUR CODE

Linear Models for Handwritten Digits Classification between 1 and 5: In this assignment, you will implement the perceptron and logistic regression model on a partial dataset from MNIST. In this classification task, the model will take a 16×16 image of handwritten digits as inputs and classfy the image into 1 and 5. The "data" fold contains the dataset which has already been split into a training set and a testing set. In either file, it contains a matrix where each row corresponds to a data sample. For each data sample, the first entry is the label (1 or 5) and the remaining 256 values between $[-1, 1]$ are grayscale pixel values for a 16×16 image. The "code" fold provides the starting code. You must implement the models using the starting code.

1. Data Preprocessing [20 points]: In this problem, you need to finish "code/DataReader.py".
   (a) Explain what the function *train_valid_split* does and why we need this step.
   (b) Before testing, is it correct to re-train the model on the whole training set? Explain your answer.
   (c) In this assignment, we use two hand-crafted features:
   The first feature is a measure of symmetry. For a 16×16 image x, it is defined as

$$F_{symmetry} = -\frac{\sum_{pixel} |x - flip(x)|}{256},$$

   where 256 is the number of pixels and $flip(\cdot)$ means left and right flipping.
   The second feature is a measure of intensity. For a 16×16 image x, it is defined as

$$F_{intensity} = \frac{\sum_{pixel} x}{256},$$

   which is simply the average of pixel values.
   Implement them in the function *prepare_X*.
   (d) In the function *prepare_X*, there is a third feature which is always 1. Explain why we need it.

(e) Implement the function *prepare_y*.

(f) Test your code in "code/main.py" and visualize the training data by implementing the function *visualize_features*. The visualization should not include the third feature. Therefore it is a 2-D scatter plot. Include the figure in your submission.

2. Perceptron [20 points]: In this problem, you need to finish "code/Perceptron.py".

(a) Implement the training process of perceptron model in the function *fit*.

(b) Implement the functions *predict* and *score* for prediction and evaluation, respectively.

(c) Test your code in "code/main.py" and visualize the results after training by implementing the function *visualize_results*. Include the figure in your submission.

(d) Implement the testing process and report the test accuracy of your best perceptron model.

3. Cross-entropy loss [30 points]: In logistic regression, we use the cross-entropy loss.

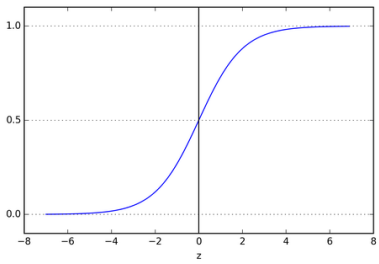
(a) Write the loss function $E(w)$ for one training data sample (x, y). Note that the binaray labels are 1 and -1.

(b) Compute the gradient $\nabla E(w)$. Please provide the intermediate steps.

(c) Once the optimal w is obtained, it can be used to make predictions as follows:

$$\text{Predicted class of } x = \begin{cases} 1 & \text{if } \theta(w^T x) \geq 0.5 \\ -1 & \text{if } \theta(w^T x) < 0.5 \end{cases}$$

where the function $\theta(z) = \frac{1}{1+e^{-z}}$ looks like



However, this is not the most efficient way since the decision boundary is linear. Why? Expalin it. When will we need to use the sigmoid function in prediction?

(d) Is the decision boundary still linear if the prediction rule is changed to the following? Justify briefly.

$$\text{Predicted label of } x = \begin{cases} 1 & \text{if } \theta(w^T x) \geq 0.9 \\ -1 & \text{if } \theta(w^T x) < 0.9 \end{cases}$$

(e) In light of your answers to the above two questions, what is the essential property of logistic regression that results in the linear decision boundary?

4. Logistic regression [30 points]: In this problem, you need to finish "code/LogisticRegression.py".

(a) Based on (b) in the last problem, implement the function *_gradient*.

(b) There are different ways to train a logistic regression model. In this assignment, you need to implement gradient descent, stochastic gradient descent and batch gradient descent in the functions fit_GD, fit_SGD and fit_BGD, respectively. Note that GD and SDG are actually special cases of BGD.

(c) Implement the functions *predict* and *score* for prediction and evaluation, respectively. Additionally, please implement the function *predict_proba* which outputs the probabilities of both classes.

(d) Test your code in "code/main.py" and visualize the results after training by using the function $visualize\_results$. Include the figure in your submission.

(e) Implement the testing process and report the test accuracy of your best logistic regression model.