24-787 Homework 7

Due date: 04/03/2021 @ 11:59 pm EST

Note: In case a problem requires programming, it should be programmed in `Python`. In Programming, you should use plain `Python` language, unless otherwise stated. For example, if the intention of a Problem is familiarity with numpy library, it will be clearly noted in that problem to use numpy. Please submit your homework through Gradescope.

Submissions: There are two steps to submitting your assignment on Gradescope:

1. **HW07_Writeup**: Submit a combined pdf file containing the answers to theoretical questions as well as the pdf form of the FILE.ipynb notebooks.

- To produce a pdf of your notebooks, you can first convert each of the .ipynb files to HTML.
- To do this, simply run: `ipython nbconvert -to html FILE.ipynb` for each of the notebooks, where FILE.ipynb is the notebook you want to convert. Then you can convert the HTML files to PDFs with your favorite web browser.
- If an assignment has theoretical and mathematical derivation, scan your handwritten solution and make a PDF file.
- Then concatenate them all together in your favorite PDF viewer/editor. The file name (FILE) for naming should be saved as HW-assignmentnumber-andrew-ID.pdf. For example for assignment 1, my FILE = HW-1-lkara.pdf
- Submit this final PDF on Gradescope, and **make sure to tag the questions correctly!**

2. **HW07_Code**: Submit a ZIP folder containing the FILE.ipynb notebooks for each of the programming questions. The ZIP folder containing your iPython notebook solutions should be named as HW-assignmentnumber-andrew-ID.zip

Q1: Feed Forward and Backpropagation (50 pts)

Submission Instructions: Please submit a typed pdf file or a pdf file of a clearly handwritten solution for this Problem.

Consider the neural network with one hidden layer. x_0 is a 6×1 vector. x_1 is a 4×1 vector. x_2 is a probability distribution over 3 classes.

We also add a bias term in both x_0 & x_1 and we set them both equal to 1. Thus, $x_0(0) = x_1(0) = 1$. W_1 is the matrix of weights from the inputs to the hidden layer and W_2 is the matrix of weights from the hidden layer to the output layer. Indexing is such that $W_1(1, 2)$ is the weight from $x_0(2)$ to $x_1(1)$). W_2 is defined similarly. Similarly b_1 and b_2 represent the vectors of bias weights in the two layers.

We will use a sigmoid activation function for the hidden layer and a softmax for the output layer. We shall use is the cross entropy loss.

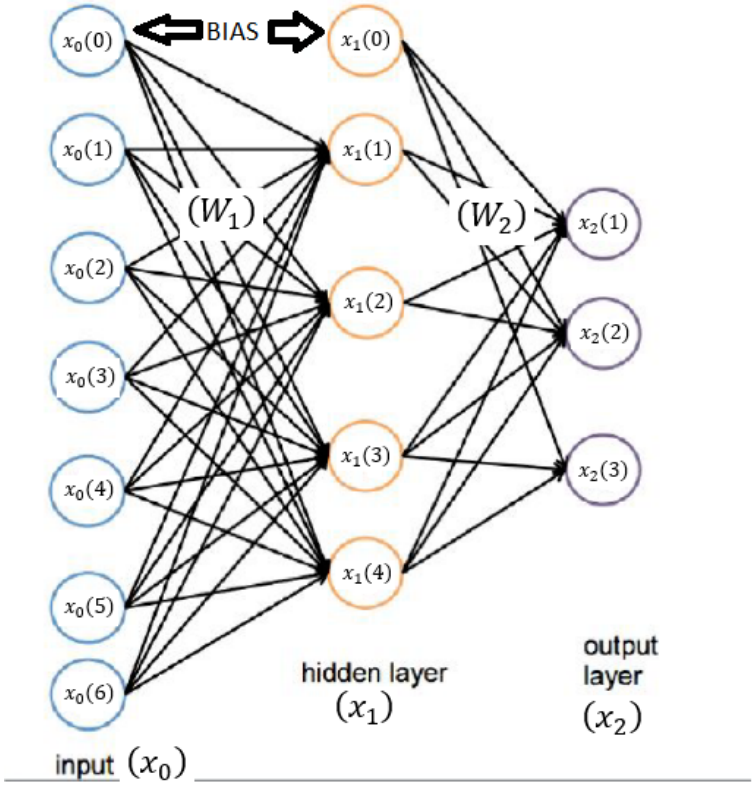

Figure 1: Neural Network with One Hidden Layer

Thus, the Neural Net follows notation follows the structure introduced in class. At the output:

- Output: $x_2 = softmax(a_2) = softmax(W_2.x_1 + b_2)$
- Cross Entropy Loss: $L = crossentropy(x_2, t) = -t^T.log(x_2)$

where x_2 is our network output and t represents ground truth, both 3×1 vectors.

The network classified label is considered to be the label for which the Softmax function has the maximum value. Thus, if output vector for a training datum is $x_2 = [0.3, 0.2, 0.5]^T$ then it would be classified as having class 3. Ground Truth vector is in the form of one-hot vector so for a datum classified as having class 3, we would have $t = [0, 0, 1]$. We want to train our network such that it gives out a value of x_2 which is as much close to t as possible i.e. not only the correct label has value nearer to 1 but all other values are near to 0. Cross entropy would help us do that as it not only considers the loss in the correct label but also in all the other incorrect ones too.

We now initialize the weights as follows:

$$W_1 = \begin{bmatrix} 1 & 2 & -3 & 0 & 1 & -3 \\ 3 & 1 & 2 & 1 & 0 & 2 \\ 2 & 2 & 2 & 2 & 2 & 1 \\ 1 & 0 & 2 & 1 & -2 & 2 \end{bmatrix} \quad W_2 = \begin{bmatrix} 1 & 2 & -2 & 1 \\ 1 & -1 & 1 & 2 \\ 3 & 1 & -1 & 1 \end{bmatrix}$$

Weights on the bias terms in both layers is initialized to 1. You are given a training example $x_0 = [1, 1, 0, 0, 1, 1]^T$ with label class 2 (i.e. $t = [0, 1, 0]^T$).

Forward Propagation: Using initial weights and example, run the feed forward of the network to answer the following questions:
(Round your answers to first 4 decimal places.

1a) Develop expressions for $f_1'(.)$, $f_2(.)'$ (activation functions in the hidden and output layers).

1b) Develop an expression for $\frac{\partial L}{\partial x_2}$

1c) What is a_1, x_1 and δ_1 vectors corresponding to the single instance x_0?

1d) What is a_2, x_2, and δ_2 vectors corresponding to the single instance x_0?

1e) Which class would we predict for x_0?

1f) What is the loss for x_0?

Back Propagation: Now use the results of the previous question to run backpropagation over the network and update the weights. Use a learning rate = 0.5. Do your backpropagation calculations without rounding, and then in your responses, round to four decimal places.

2a) What is the updated W_2?

2b) What is the updated b_2 ?

2c) What is the updated W_1?

2d) What is the updated b_1 ?

2e) After we update all our weights and we run the feed forward over the same example again, which class would we predict?

Submit your calculations and box answers to every questions stated above.

Q2: Classification Toolbox (50 pts)

Over the past two months, we have covered multiple models that can be used for classification. For instance: Decision Tree, K-nearest Neighbor, logistic regression, SVM and most recently Neural Network. In this problem, you will be working on a real world dataset of red wines `winequality-red.csv`. The input variables are as follows (each one corresponding to one column in the dataset):

- fixed acidity
- volatile acidity
- citric acid
- residual sugar
- chlorides
- free sulfur dioxide
- total sulfur dioxide
- density
- pH
- sulphates
- alcohol

The output data, based on sensory data is **quality**, which is scored between 0 and 10. In this problem, any wine sample with quality higher than or equal to 6 are classified as **quality wine**, otherwise, they are classified as **bad wine**. In the dataset, the first 1000 rows (excluding header) are used as training set, the rest are used as test set.

(a) Model tuning (30 pts) Now, you are supposed to build classifiers using `sklearn` with the following models and train them with the training set, then report the accuracy of trained models on the training and test sets. For each kind of model, you need to vary the listed parameters of each model, reporting the training and test accuracy under different parameters. Finally, report the best parameters of each model evaluated by the test accuracy. The parameters listed below match the `sklearn` documentation. Please refer to the corresponding documentation if you are not sure of the meanings of certain parameters.

- **Decision tree**: critetrion: {"gini","entropy"}; max_depth: choose 3 values that you find work best
- **K-nearest Neighbor**: n_neighbors: choose 3 values that you find work bests
- **Logistic Regression**: penalty: {'l1', 'l2', 'elasticnet'}

- **SVM**: kernel:{'linear', 'poly', 'rbf'}. For 'linear' and 'rbf' kernels, use the following regularization parameters: C:{0.01,10,1000}

- **Neural Networks** (`sklearn.neural_network.MLPClassifier`): hidden_layer_sizes: choose 3 cases each with 3 hidden layers that you find work best, e.g. (10,20,10) stands for 3 a hidden layers case, you don't need more than 5 layers; activation: 'logistic', 'tanh', 'relu'

  Submit your code; training and test accuracy for different models and different parameters, as well as the best parameters combination you choose for each type of model

  (b) Model evaluation (20 pts) There are quite a lot of criterion for evaluating the performance of a classifier, and depends on the goals the classifier would like to achieve or the different angles you want to study the classifier, you would want to use different criterion. Firstly, please study the definition of the following criterion: **precision, recall, F1 and confusion matrix**. Explain the definition for each criterion in 1-2 sentences. (You can simply use the markup cell within Jupyter notebook to write down the definitions)
  Then with the best parameters for each type of models you get from (a), calculating the **precision, recall, F1 and confusion matrix** of each type of model with its best parameters using `sklearn.metrics` module on the test set.
  Case 1: based on the numerical results, among the 5 models, if you want to sell your trained model to customer A that don't want your model to miss any **good wine**, which model would you sell to A. Explain why.
  Case 2: in another case, customer B also don't want to miss too many **good wines** when buying wine with your model, but at the same time, she also doesn't like it when lots of wines classified as **good wine** actually taste bad. In this case, based on the numerical results, which model would you sell. Explain why.

  (Link for `sklearn.metrics`: https://scikit-learn.org/stable/modules/model_evaluation.html)

  Submit your explanation for precision, recall, F1 and confusion matrix; submit your code; submit precision, recall, F1 and confusion matrix for the 5 models on test set; submit your answers and explanation for the case 1 and case 2