Korea Advanced Institute of Science and Technology
School of Electrical Engineering
EE531 Statistical Learning Theory , Fall 2020

Issued: Dec. 18, 2020
Due: Dec. 28, 2020

Assignment III

Policy

Group study is encouraged; however, assignment that you hand-in must be of your own work. Anyone suspected of copying others will be penalized. The homework will take considerable amount of time so start early.

1. **Regression**
   Here we will look at a very popular dataset referred to as `Boston housing dataset` from the `CMU StatLib Library` that lists the prices of houses in Boston suburbs. A data sample consists of 13 attribute values (indicating parameters like crime rate, accessibility to major highways etc.) and the median value of housing in thousands that we wish to predict. The data are in the file `Boston housing.txt`, the description of the data is in the file `Boston housing description.txt` on the KLMS.

   (i) Part 1. Data analysis. Examine the dataset `Boston housing.txt` using Matlab. Answer the following questions.

   a. How many binary attributes are in the data set? List the attributes.
   b. Calculate and report correlations in among the first 13 attributes (columns) and the target attribute (column 14). What are the attribute names with the highest positive and negative correlations to the target attribute?
   c. Note that the correlation is a linear measure of similarity. Examine scatter plots for attributes and the target attribute using the function you wrote in problem set 1. Which scatter plot looks the most linear, and which looks the most nonlinear? Plot these scatter plots and briefly (in 1-2 sentences) explain your choice.
   d. Calculate all correlations between the 14 columns (using the corrcoef function). Which two attributes have the largest mutual correlation in the dataset?

   (ii) Part 2. Linear Regression.
   Our goal is to predict the median value of housing based on the values of 13 attributes. Please divide the dataset into two datasets: (1) a training dataset `Boston housing train.txt` you should use in the learning phase, and (2) a testing dataset housing test.txt to be used for testing. Please use 90% for training and 10% for test. Assume that we choose a linear regression model to predict the target attribute. Using Matlab:

   a. Write a function LR solve that takes X and y components of the data (X is a matrix of inputs where rows correspond to examples) and returns a vector of coefficients θ with the minimal mean square fit. (Hint: you can use backslash operator '/' to do the least squares regression directly; check Matlab's help).
   b. Write a function LR predict that takes input components of the test data (X) and a fixed set of weights (w), and computes vector of linear predictions y.

c. Write and submit the program `main submit part2.m` that loads the train and test set, learns the weights for the training set, and computes the mean squared error of your predictor on both the training and testing data set. Please submit the `main submit part2.m` on the KLMS

d. In your report please list the resulting weights, and both mean square errors. Compare the errors for the training and testing set. Which one is better?

(iii) Part 3. Gradient descent

The linear regression model can be also learned using the gradient descent. One concern when using the gradient descent method is that it may become unstable when fed with un-normalized data.

a. Implement and submit an online gradient descent procedure for finding the regression coefficients θ. Your program should:

- start with zero weights (all weights set to 0 at the beginning);
- update weights using the learning rate $\frac{0.05}{t}$, where t denotes the t-th update step. Thus, for the first data point the learning rate is 0.05, for the second it is 0.025, for the 3-rd is 0.05/3 and so on;
- repeat the update procedure for 1000 steps reusing the examples in the training data if necessary (hint: the index of the i-th example in the training set of size n can be obtained by (i mod n) operation);
- return the final set of weights.

b. Write and submit a program `main submit part3.m` that runs the gradient procedure on the data and at the end prints the mean test and train errors. Please note that your program should normalize the x (input) part of the data before running the method. This is performed by first calculating the means and stds of all input attributes on the train data and by applying these to normalize both the train and test inputs. Use compute norm parameters and normalize functions for this purpose. Run your program and report the results. Give the mean errors for both the training and test set. Is the result better or worse than the one obtained by solving the regression problem exactly.

c. Try to run your `main submit part3.m` program from above without data normalization, that in on the original un-normalized data. Please report on what you observed?

d. Modify main submit part3.m from above such that it lets you to progressively observe changes in the mean train and test errors during the execution of the gradient descend procedure.

e. Experiment with the gradient descent procedure. Try to use: fixed learning rate (say 0.05, 0.01), or different number of update steps (say 500 and 3000). You may want to change the learning rate schedule as well. Try for example $\frac{2}{\sqrt{n}}$. Report your results and any interesting behaviors you observe.

2. **Neural Network Programming**

The problem is to implement a classifier that can classify cats and dogs with your neural network model. Download the baseline code and dataset in 'student.zip'. The baseline code 'main.py' is for helping to implement your classifier with neural network. The dataset is composed of 20 cat and dog images with size of 300×300 pixels for training in 'data' folder and 5 cat and dog images for test in 'test' folder. Please read sub-problem 1,2 and 3 in the code and perform the given problems. (*Python 3 is recommended.)

(i) Problem 1: Neural Network

Implement and explain your network in the code for classifying the cats and dogs
(Please show your code here if it is not too long.)

(ii) Problem 2: Train your model
Train your model with given train dataset in 'data' folder. You may divide the dataset into training and validation for developing your model.
(Please show your code here if it is not too long.)

(iii) Problem 3: Test your trained model on test dataset in 'test' folder.
Check the results on test dataset and write your accuracy.

(iv) Write three ways to improve the performance on test dataset.

3. **Sequential Neural Network Model** This problem considered a text classification problem. Given a surname, the model need to identify the language corresponding to the surname. The dataset has the surname from 18 different languages. See the example below.

Name	Bélanger	Adlersflügel	Yeon	Nguyen
Language	French	German	Korean	Vietnamese

The skeleton code is given in `rnn.zip`. The code is implemented in `python` and `Pytorch` framework. It can be run on either Windows or Linux. You need to install pytorch (CPU version) following https://pytorch.org/. If you are not familiar with Pytorch, it is recommended to try some basic tutorials at https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html.

This is many-to-one case of RNN. Each name have multiple letters and each letter is in turn going to the model to get a single output. See example below:

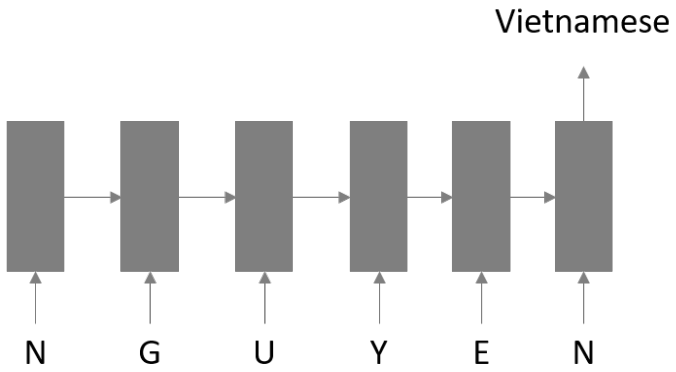

Figure 1: The many-to-one RNN model. Each letter goes to an RNN cell, the signal is fed back and combine with next letter. The RNN cell produce output at final input letter.

The heart of the model is an RNN cell, as shown in Figure 2. You are going to implement the RNN cell given the skeleton code.

(i) In `model.py`, implement the RNN cell. Then train the model by `python train.py`. After training the model. You should see the loss curve similar as Figure 3.

(ii) The trained model is saved at `char-rnn-classification.pt`. Evaluate the trained model by `python evaluate.py`. You should a confusion matrix similar to Figure 4. What can we say about model accuracy based on this confusion matrix?

(iii) Try with your own `SURNAME` by `python predict.py SURNAME`. You can try with your own one (if your language is covered in the dataset. Also try 2 more arbitrary ones.

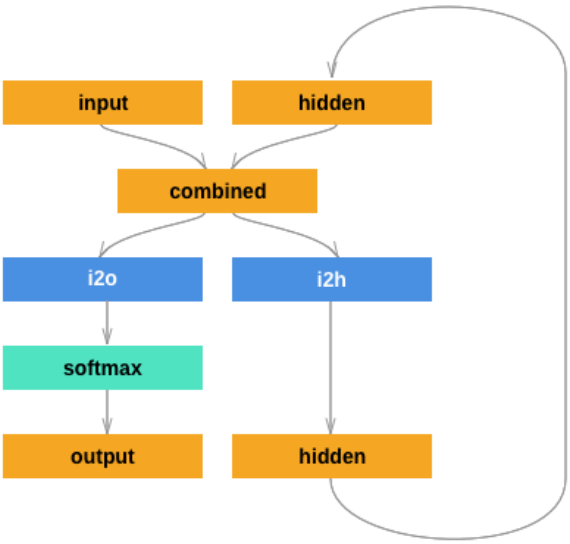

Figure 2: RNN cell. The input is combined (tensor concatenation) with previous hidden state. Then the combine tensor is fed into two branch. The `i2h` branch produce next hidden state. the `i2o` produces the output. Only the output at final stage is used as the output of the RNN model.

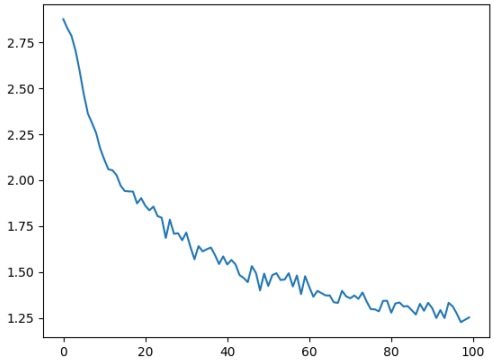

Figure 3: Loss over epochs

4. **Sampling** It is assuming a distribution of a random variable with its density function as follow:

$$f(x) = 1 - |x|, x \in [-1, 1]$$

   (i) Please provide a rejection sampling algorithm for simulating a random variable X that follows the distribution described above.

   (ii) Please derive the probability that samples are accepted.

   (iii) Please provide a method to simulate the above distribution using the inverse CDF method.

5. **Gradient Descent for Linear Regression.**

   Given truth function for data generation $f(x) = 2x + 1$. Complete following tasks using Matlab or Python.

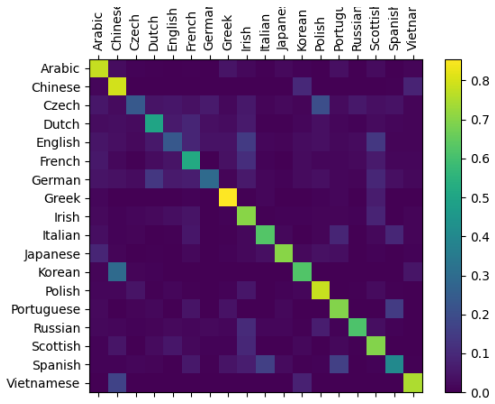

Figure 4: Confusion matrix

(i) Generate 100 samples drawn from:

$$y = f(x) + \mathcal{N}(0, 0.5).$$

(ii) We are going to find the predicted data generation $\hat{f}(x) = \theta_1 x + \theta_0$. Write gradient descent algorithm to estimate θ_0 and θ_1. Assuming that we are using MSE loss for this linear regression problem.

(iii) Compare the predicted function with the truth function? Why are they slightly different?