Homework Assignment 4

Due: Friday, March 5, 11:59pm, 2021

This assignment includes two problem sets.

PART I: Neural Network

Overview

In this mini project, you will need to apply the feedforward neural network (FNN) on the California housing dataset (regression). The dataset includes 20640 samples and each sample is a California district. Each district is represented by 8 features:

- MedInc median income in block
- HouseAge median house age in block
- AveRooms average number of rooms
- AveBedrms average number of bedrooms
- Population block population
- AveOccup average house occupancy
- Latitude house block latitude
- Longitude house block longitude

The target variable is the median house value for California districts. This dataset was derived from the 1990 U.S. census, using one row per census block group. A block group is the smallest geographical unit for which the U.S. Census Bureau publishes sample data (a block group typically has a population of 600 to 3,000 people). It can be downloaded/loaded using the *sklearn.datasets.fetch_california_housing* function. Check the following sample codes for how to download the dataset

```
#load datasets
from sklearn.datasets import fetch_california_housing

ca_house_db=fetch_california_housing()
print(ca_house_db.data.shape)
print(ca_house_db.target.shape)
print(ca_house_db.feature_names)
print(ca_house_db.DESCR)
print(ca_house_db)
```

In this project, you will use the field .data (20640 by 8) as the feature matrix and the field .target (20640 by 1) as the target values to regress.

Method

To define a FNN model, you will need to specify the following hyper-parameters:

i) number of hidden layers and number of neural units for each layer;
ii) learning rate for the gradient descent method (optimizer), and
iii) activation function (relu, sigmoid, softmax,tanh,elu etc.) for hidden neurons. You might find the activation functions by Tensorflow: Layer activation functions (keras.io)

There is no need for you to code the neural network algorithm from scratch. You can use either third-party libraries or the Tensorflow implementations provided in Lecture 05. Please develop a .py file to include the following major steps.

Step 1. Data splitting. Please randomly split the 20640 training samples (and their labels) into two halves for training models and validation purposes, respectively. Please further randomly split the 10320 training samples into two subsets: a subset of 2064 samples for validation purpose and the rest 8256 samples for training. Therefore, you will get three subsets: 8256 training samples, 2064 validation samples, and 10320 testing samples. You could sue any third-party functions for this splitting operation.

Step 2. Validation for model selection. Please specify at least three sets of hyper-parameters (see the above). For each set, call the third-party functions or tensorflow to train an FNN model on the training samples, and apply the learned model over the validation set (8256 samples). For each model and its results, please compute its R^2 (determination coefficient). Report the model that achieves the top R^2.

Question 3. Testing. According to the above-mentioned metric R^2, select the top ranked model and apply it over the testing samples (10320 samples). Calculate and report the R^2 over testing samples.

Question 4. Analysis. Analyze the testing results of your top-ranked model. Please first calculate for each testing sample the absolute error between the model's prediction and ground-truth value (both are real-valued). Then, report TEN testing samples which received the largest absolute errors. Try to analyze the reasons of these failure cases (e.g., if any samples include unexpected features).

PART II

The purpose of this problem set is to practice the classification model: Support Vector Machine (SVM).

Overview

In this project, you will need to train an SVM classifier that can identify the gender of a crab from its physical measurements. For every crab, there are six physical features: species, front-allip, rear- width, length, width and depth. You will need to train a binary SVM model from a set of training samples, and apply this model over the testing samples to evaluate its performance.

The dataset is provided in the file 'crab.csv', including 200 samples with gender labels (1: male, -1: female). The starter codes are provided in the file 'main_svm.py'.

The starter codes include five major steps. To implement the training & testing, you might use the Scikit-learn module as instructed in the lectures.

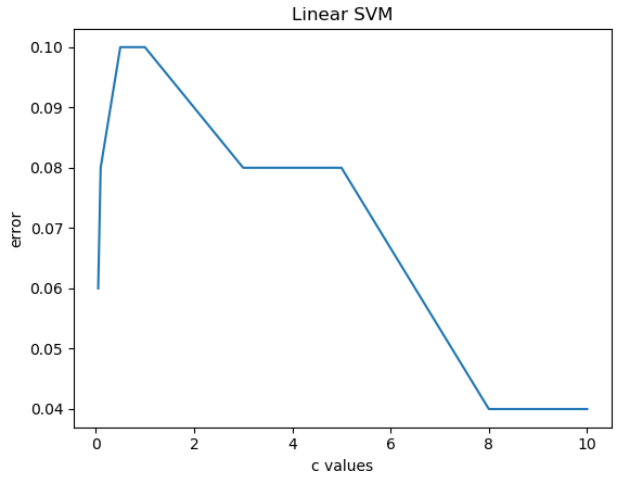
Step-1: Load data from 'crab.csv' to get feature matrix X and label vector Y. X is of 6 by 200 dimension, where each column represents a crab sample. The matrix Y is of 1 by 200 dimension, including the related gender labels (1 or -1). The starter codes will randomly split these 200 samples into two even subsets: use one for training & validation and another for testing. You don't need to change the codes in this step.

Step-2: Randomly divide the training set into two EVEN subsets: use one for training your model, and another for validation. You will need to implement your codes to do such random splitting. This step includes the "placeholder 1: training/validation".

Note that, you will need to use the same training/validation/testing splitting (steps 1 and 2) while evaluating different models in the later steps – you can either store and load the splitting data, or make sure all model selection processes are performed with the exactly same training/validation.

Step-3: Select the optimal model parameters using validation samples. You will need to consider the parameter C (the weighting parameters), kernel types (linear or others) and kernel parameters (if applicable). Please try as many parameters as you can to get the best result. In particular, you will need to generate two figures (in placeholders 2 and 3)

Figure 1: Selection of Cs. Please use different C (e.g., 2, 4, 6, 10) to train the SVM classifier (with other hyper-parameter fixed). For each classifier, calculate its validation error rate (number of misclassified samples over the total number of validation samples). With these results, please generate the following figure:



where the horizontal direction represents different values of C, and the vertical direction represents validation errors. Use at least 3 different values for C.

Figure 2: selection of kernels. Plot the validation errors while using linear, RBF kernel, or Polynomial kernel (with other hyper-parameters fixed);

Step-4: Select the best hyper-parameters (C, kernel types, etc.) and apply them over the testing subset. You may write a script to do the selection, or manually pick up the hyper-parameters based on your results. To do the latter, you might run steps 1 to 3 while temporarily commenting step-4 and step-5.

This step includes the "placeholder 4: testing".

Step-5: evaluate your results using confusion matrix and other metrics, including accuracy, precision and recall rates. Analyze your results through visualizing both success and failure examples. Include 5 success examples and 5 failure examples in your report.

This step includes the "placeholder 5: metrics" and "placeholder 6: success and failure examples".

What to Submit

- For Part I: a single .py file that implements the FNN model in responses to the four questions
- For Part II: the modified main_svm.py file with all completed placeholders.
- A PDF to include your responses to each of the above questions or instructions.

Submission instructions:

- Prepare a Single PDF file to describe your decision process while completing the above questions. Include all figures, tables, intermediate results, or other results that might help understand your efforts.
- No need to include full paragraphs of your source codes in the report. Coding snippet is allowed.
- Submit your PDF file and source codes through the UCLA system.
- No HARD COPY IS REQURIED.