Lab 8: neural networks and Scikit-learn

Overall Deliverables & Rubric:

- Report for exercise 1, and complete set of code and figures (one notebook per exercise) concatenated as 1 PDF file, uploaded to CANVAS.
- Total Points 50

0. Computer Programming Environment

The first exercise involves a website-based graphical user interface for tinkering with a neural network on 'toy' data.

The second exercise involves Python and can be done from your web browser using Google's Colab.

1. Exercise 1: multi-layer neural network for 2D regression

The toy data features are the coordinates of in a square with sides of length 12 centered at the origin. The sample consists of a uniform distribution over the square. The target value has a pattern of three hills and three valleys of elevation 1 and -1 units, respectively, arranged on as three by two grid, with hills centered at (-4,3), (0,-3), and (4,3) and valleys at (-4,-3), (0,3), and (4,-3).

The performance metric is mean squared error of the elevation prediction.

The data is divided such that 20% of the points are used for training and 80% for testing. This testing data should be considered validation data as we will use it for selecting the best model/hyper-parameters to train a model.

The possible models are neural networks with different numbers of neurons per layer and the number of layers can vary too. Each neuron uses the tanh activation function.

The weights are updated with mini-batch stochastic gradient descent with batch size of 10, learning rate of 0.03, weight decay (L2 regularization) can be used, and the regularization rate is a hyper-parameter.

Go to this link to see the interface:
https://playground.tensorflow.org/#activation=tanh&regularization=L2&batchSize=10&dataset=circle&regDataset=reg-gauss&learningRate=0.03&regularizationRate=0&noise=0&networkShape=2,2&seed=0.04730&showTestData=false&discretize=false&percTrainData=20&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=regression&initZero=false&hideText=false

Adjust the architecture (as a hyper-parameter):

With a regularization rate of 0, adjust the number of hidden layers and the number of neurons per layer. Press the play button at the top left to start training. You will notice the learning curves for train and test on the top right. Repeat this step for multiple hidden layers and number of neurons. Continue tinkering with it until you achieve a test loss less than 0.020.

Take a screen shot of the architecture/model and copy the URL (settings and hyper-parameters are stored there) that achieves this loss. You will include it in a report document.

Adjust the regularization rate:
Make a table in your report document of the performance of the model (test loss) for your best-performing architecture across 4 different regularization rates (0, 0.01,1,10).

Discussion
Q1. What insights did you gain from this exercise?
Q2. For this problem what is more important the number of layers, and number of neurons in a neural network?
Q3. For the regularization rate part, look at the color (predictions) of the output for training with different rates. Are the magnitude of the elevations value predictions correlated with the regularization rate? If so what is the relationship?

Deliverable & Rubric:
Make this Exercise 1 in word processing document and print it to a PDF.
5 points for the screenshot; 5 points for the table; 5 points for each question (25 points in total).

2. Exercise 2: geospatial regression
Upload the GDS_lab8_exercise_2.ipynb to Google Colab or use your own machine. It will install the necessary packages.

Figure 1:
Make a bar graph and plot the feature importance variable from the random forest regressor. Adjust the x-tick label to have the feature names. (You will have to search the internet on how to do this for Matplotlib.)

regr.feature_importances_

Figure 2:
Make a bar graph and plot the standardized features coefficients for the ElasticNet model. Adjust the x-tick label to have the feature names, including the binary features for each grocery store region.

Figure 3:
Make a figure that has both the true contour of the satisfied index as well as the scatter plot of the validation set points colored by the predictions from the RandomForest model. You may need to add the predictions as a column into the GeoDataFrame.

Code Block 1:

Compare Gaussian Process modeling across 4 sets of features with and without scaling (8 models in total). The performance comparison uses the R^2 the coefficient of determination.

The feature sets consist of the following:

1. x and y coordinates
2. dist_park and dist_grocery
3. x and y coordinates, household size, and population
4. dist_park and dist_grocery, household size, and population

Organize the performance results into a 2 by 4 data frame with a labeled row for whether scaling is used or not.

Code Block 2:
Use scikit learns's nearest neighbor regression https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html
The input features should just be the x and y coordinates.
Train and score a model for different choices of $k \in \{1,2,3,\ldots,15\}$ the number of nearest neighbors. Store the results in an array.

Figure 4:
Make a plot of the performance across the different ks. Make sure to label the axes.

Deliverable & Rubric:
Make this Exercise 2 in one standalone Jupyter Notebook and print it.
4 points for each figure; 5 points for code block 1; and 4 points for code block 2 (25 points in total).