**CSCE 823: Machine Learning**

**Summer 2020**

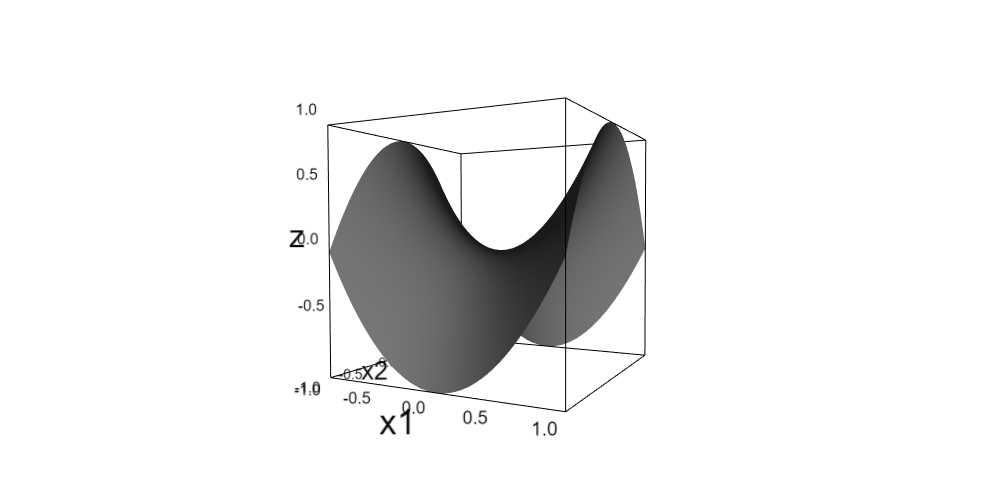
**HW2**

The homework will be composed of an integrated code and report product using Jupyter Notebook. Use python to perform calculations or mathematical transformations or generate graphs and figures or other evidence that explain how you determined the answer. Each step listed below should correspond to code and/or text in your file. Use the step identifiers (for example: “**Step 1:**”) clearly identified in both your code and your notebook markdown cells.

This assignment uses Keras (on top of Tensor\_Flow) to build, train, and evaluate networks quickly. Be sure to read through the Keras documentation and understand the different options from which you choose. Time spent understanding Keras now will help in future assignments and your project. You will not need the GPU backend working for this assignment, however larger models (and larger batchsizes during training) will run significantly faster if using a GPU.

**Artificial Neural Net for 2-input nonlinear regression on the “Saddle” dataset**

You will train an ANN on a noisy dataset with two features to estimate a real number. The data is generated from a saddle function: , where *x*1 and *x*2 are the input features. Visually it looks like a saddle:



In this assignment, you will use 2-or-more-layer ANNs, where each layer will have exactly one specific activation function (but there may be different activation functions for each layer).

1. **Obtain and load the two datasets for the regression problem** (CSCE823\_HW2\_regression\_non\_testdata.csv) (CSCE823\_HW2\_regression\_testdata.csv). The first two columns of the files are the *x*1 and *x*2 features. The last column is the target *y* value. Note that the non-test data is a noisy estimation of the saddle function at 900 points while the test data contains exact values of the saddle function at 2500 points.
2. Data Exploration:
   1. **Build a function to return the value of z from the value of the mathematical saddle function** described above. This function should be designed to work directly on numpy arrays without using a for loop to compute the answers.
   2. **Build a function to display a 3d representation of the saddle** like the one above (surface or mesh) on the input range   
      [-1,1] in both *x*1 and *x*2. Consider using the packages “mpl\_toolkits.mplot3d”, “pylab”. The package “ipyvolume” would show a nice interactive plot of the data in 3d. You may have to set up your jupyter notebook (or jupyter lab) to use widgets. See <https://ipywidgets.readthedocs.io/en/latest/user_install.html> for more details. Alternately you can use matplotlib’s plot\_trisurf and scatter3D for this task.
   3. **Visually Explore the non-test data using both 2d representations (such as histograms) and 3d representations**. (Do not view or explore the test data.) For 3d representations, build 3d scatterplots overlayed on the 3d surface.
   4. **Determine the raw errors from the non-test set, with respect to values returned from the saddle function, display these errors using a histogram with 50 bins and discuss the histogram shape**. Errors are the numerical difference from each saddle value to data point y value. These errors represent “noise” in the measured datapoints. In your description of the histogram, what is its shape? Is it skewed? What can you say about the noise on these datapoints and how it will affect your model.
   5. **Provide a scatterplot of the true saddle values versus raw errors and discuss**.
   6. **Determine and report the MSE, RMSE and mean absolute error (MAE) on the raw errors**. These will be your baseline values to beat when you fit an ANN model to the data. The goal is for your model to generalize the saddle surface from the noisy datapoints such that each of the computed errors is lower than the same errors on the raw datapoints in the non-test set
3. **Prepare Data for Training/Validation - split the non-test data into two sets: train and validation**. While you could perform cross-validation, the interpreting ANN models through crossvalidation is complicated and time-consuming so you will not do it in this assignment.
4. **Build a function which accepts hyper-parameter configurations and returns a (compiled, but untrained) Keras model**. The output layer should use a linear activation function, and your loss function should be chosen appropriately (for example, MSE, or MAE). The model-building function should accept a configuration which you define to include:
   * + Number of hidden layers
     + Number of nodes and activation functions for each hidden layer (can different be per-layer)
     + Optimizer
     + Learning rate
5. **Define configurations for multi-layer ANNs and include rationale for your decisions**. Given what you saw in your exploration of the data make decisions for hyperparameters. Note that this step defines the configurations, but doesn’t pass them to your model compiling function. Your ANN will have at least one hidden layer. You will need to select possible choices for hyperparameters. The minimum number of configurations you must evaluate is 2x2x2x2 = 16, but you can evaluate more. Beware of making too many configurations! Hyperparameters used in these configurations should include:
   * + Number of hidden layers (choose at least 2 options)
     + Activation function(s) for the hidden layer(s) (choose at least 2 options)
     + Widths for the hidden layers (choose at least 2 options).
     + Optimizer & learning rate (at least 2 options)
6. Note: this cell should only run when the constant RUN\_CONFIGURATION\_LOOP is set to True (and set this global variable to False before submitting). **Using the training set, train the model on each of the configurations you developed in the previous step**. Note: You will need to manually select number of (maximum) epochs to train required based on your choices above (this will be a fixed parameter that you might need to select via exploration)
   1. **Train your model in the validation loop** to determine the best setup (according to final validation loss on the trained models).
   2. **Report the best configuration and discuss** why you think this setup worked the best in terms of its configuration.
7. **Select the best configuration** from the previous step and **manually hard-code the configuration** at the beginning of this cell. Then use your model configuration builder to **obtain a compiled model with this configuration**.
8. **Retrain your model** using all the non-test data.
9. **Determine if your model trained on all the non-test data produces saddle-function predictions better than the non-test data y-values themselves** (in other words, determine if your model acts as a de-noising function on values in the non-test CSV):
   1. **Run predictions on all the non-test data** to obtain prediction values of your final model.
   2. **Determine the prediction errors on the non-test set**, with respect to values returned from the saddle function (this is the numerical difference from each saddle value to value predicted by your model (y\_hat)). These errors represent “noise” in the measured datapoints. Display both the prediction errors and the raw errors (that you found during data exploration) using a histogram with 50 bins and discuss the overlayed histograms. For example, are the prediction errors better (closer to zero) than the noise in the original data?
   3. **Provide a scatterplot of the true saddle values versus prediction errors and discuss**. Are there y values for which your prediction is worse than others?
   4. **Determine and report the MSE, RMSE and mean absolute error (MAE) on the non-test-set prediction errors**. Did these predicted error rates beat the same performance measures on raw errors caused by data noise (that you found during data exploration)?
10. Evaluate the model fit on the test data. Note that the test set *y* values should exactly match the saddle function values for each (*x*1,*x*2) point in the test set – produced from a 50 x 50 grid over the area of interest. Complete the following tasks:
    1. **Build a 3d scatterplot of the test dataset predicted (*y\_hat*) values overlayed on the 3d surface of the saddle function. Discuss where your model performed well and where it struggled**.
    2. **Determine, display, and discuss the histogram of prediction *errors* from the predictions on the test set.**  Display these errors using a histogram with 50 bins and discuss the histogram shape. For example, what is its shape? Is it skewed? What can you say about the prediction errors on these datapoints?
    3. **Display & discuss residuals:** Provide a 2d scatterplot of the test set y values (on the plot’s X axis) versus prediction errors (on plot’s Y axis) and discuss.
    4. **Determine and report the error measures on the test set *predictions*.** Compute MSE, RMSE and mean absolute error (MAE) on the test set prediction errors. Do the test set prediction rates beat the baseline rates from on the raw noise error rates in the non-test data?