Deep Learning: Regression and Classification using (Dense) Neural Networks

EECE-4710 IoT and Machine Learning

Cristinel Ababei
Electrical and Computer Engr., Marquette University

1. Objective

Investigate the use of (Dense) Neural Networks (NN) to do Regression in the case of the problem of Boston house price prediction. Learn about normalization of data. Learn about MAE, RMSE, and tuning of hyperparameters. Investigate the use of dense NNs to do Classification of digits 0-9 on MNIST dataset of images.

2. House Pricing Regression using Dense Neural Networks

Example 1:
In this example, we build a regression model for prediction of house prices given 13 input features from the Boston house-price data. This is a dataset taken from the StatLib library which is maintained at Carnegie Mellon University.
The dataset has 506 samples. Each datapoint or sample has 13 attributes or Features Xi (i from 0 to 12) of houses at different locations around the Boston suburbs in the late 1970s. The attributes themselves are defined in the StatLib website (as per capita crime rate in the area, number of rooms, distance from employment center, etc.).
The target (Y) is the median values of the houses at a location (in USD 1,000)

The Jupyter Notebook for this example is:
TF_Boston_Housing_Regression.ipynb

So, open Google Colab and, click on File->Upload Notebook and select from your computer the said Notebook. Once you uploaded the Notebook, do first Edit->Clear All Outputs. Then, go through all code cells and observe.

The NN model that is created in this example is a Dense NN with:

- One Input layer
- One Hidden layer
- One Output layer

[input] ==> [hidden] ==> [output]
13 ==> [20] ==> 1
The Input Layer accepts 13 input features the Output Layer has 1 output to match the target (y). The number of Neurons (or Units) on the Hidden layer is arbitrary (e.g., 20) – but, you can set to different values in your attempt to increase model performance.

Also in this example, we learn about KerasTuner - an easy-to-use and scalable hyperparameter optimization framework that automates the rather challenging hyperparameter search process. Learn more about that here:

- https://www.tensorflow.org/tutorials/keras/keras_tuner
- https://keras.io/keras_tuner/

3. Digit Classification using Dense Neural Networks

Example 2:
In this example, we use a dense NN to do digit Classification. We work with MNIST handwritten digits dataset (https://www.tensorflow.org/datasets/catalog/mnist). The MNIST database of handwritten digits includes a training set of 60,000 28x28 grayscale images of the 10 digits along a test set of 10,000 images. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

This example is presented in the Jupyter Notebook:
TF_MNIST_Classification.ipynb

So, start Colab, then, click on File->Upload Notebook and select from your computer the said Notebook. Once you uploaded the Notebook, do first Edit->Clear All Outputs. Then, work through all its code cells and observe.

4. Assignment

First take a few minutes and read the following tutorial:
"How To Split a Tensorflow Dataset into Train, Validation, and Test sets" at:
https://towardsdatascience.com/how-to-split-a-tensorflow-dataset-into-train-validation-and-test-sets-526c8dd29438

Note that the second example in the previous section splits the MNIST dataset of 70,000 images into 60,000 for training, 0 for validation, and 10,000 for testing. This is roughly an 85%-0%-15% split for training, validation and testing.

In this assignment you must modify and re-do this example four different times (each time, run the same experiment 5 times and calculate averages and std that you will report) to investigate the following splits:
60-0-40%
70-0-30%
80-0-20%
90-0-10%

In all four cases, during the "Train the model" section record the values of "Training Data Accuracy:". Also, save each of the plots loss vs. epoch in this section (one only for each of the five runs of the same experiment). You will have to include them in your report.
From the "Testing the trained model" section, record the values of "Testing Data Accuracy:".

Then, create two additional separate plots:

- Training Data accuracy Vs. Splits
- Testing Data Accuracy Vs. Splits

5. Deliverables

You must write (typed) a report and upload it as a PDF file on D2L. The report should be named "**LastName_hw3.pdf**". The report should include the following sections:
1) Title + course info + your name
2) **Summary.** Describe in one paragraph what the objective of the assignment is.

3) **Exploring Different Dataset Splits.** Describe your experiments. Include all plots that you created and present a discussion in which you compare your results. Which split give you best results?
4) **Conclusion.** Present your conclusions and describe what issues you encountered and how you solved them.
5) **References.** Include all references that you used, as a numbered list. Cite them in the report itself; do not just list them here.