# Machine Learning - CS 7641 Assignment 1

**Anthony Menninger**
Georgia Tech OMSCS Program
tmenninger@gatech.edu

**Abstract**

FILL IN?

## Introduction

## Data Sets

The first data set is from 1994 and 1995 current population surveys conducted by the U.S. Census Bureau [1] from the UC Irvine Machine learning site. The classification task is to determine if the user type for each instance has an income above $50,000 or below ($94,000 in today's dollars). The 1994 median annual salary was $16,000 (2020 median salary was $34,600)

There are 40 usable features, with both continuous features, such as age, and discrete, such as married status or sex. The training set has 199,523 instances and the test set has 99,762 instances. Because many of the features in the file are string based, I created transformed instances using the sklearn preprocessing LabelEncoder to translate discrete string values into consistent numeric values. This was necessary for algorithms that only used numeric features. I chose this data set because it seemed relatively large and I expect somewhat noisy with a smallish number of features.

The second data set is The MNIST Database of Handwritten Digits [2]. This consists of instances of 28 X 28 greyscale images of the digits 0-9. The only transformation needed was that each of these images was flattened, creating 784 features, one for each pixel. There are 60,000 training instances and 10,000 test instances. I chose this because it is a good comparison to the first data set, with a very different type of data (images), which leads to significantly more features (784). In addition, each of the features can be thought of as related to each other, as they are all positional in a two dimensional grid, while the first data set features do not have any necessary relation between themselves ie: age is not related to sex.

ADD DISCUSION ON BALANCE OF LABELS - CENSUS CAN GET GOOD SCORE WITH JUST 0

## Decision Trees

Decision Trees. For the decision tree, you should implement or steal a decision tree algorithm (and by "implement or steal" I mean "steal"). Be sure to use some form of pruning. You are not required to use information gain (for example, there is something called the GINI index that is sometimes used) to split attributes, but you should describe whatever it is that you do use.

I used the DecisionTreeClassifier algorithm from the sklearn library. By default, this uses a GINI index to split the data. It was possible to use an information gain entropy for splitting, but this did not make a meaningful difference in results for the datasets so the GINI index was used. A key aspect of sklearn is that it only accepts numeric data for features and treats all features as continuous. Because of this, a feature can appear in multiple nodes within the same path, with different thresholds. It also means this is a binary tree, with each node having only two edges. Other decision tree implementations might allow for discreet features, meaning a feature could only appear once in any tree path and the tree might not be binary.

sklearn also uses a randomizing element, which means that given the same set of data, it may produce different results each run. In order to produce consistent, repeatable results, the random state setting was set to a fixed value to produce the same results each time.

I also used the sklearn Cross Validation module for estimating the best solution without using the test data. This was then confirmed by reviewing the test data.

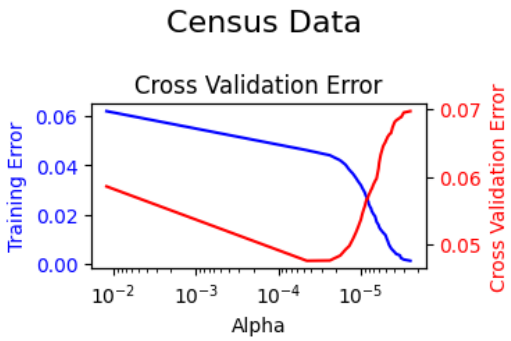


Figure 1: Census Data Cross Validation. The cross validation error starts to rise as the decision tree starts to over fit. The best cross validation $\alpha$ value of 0.0000153 had a test accuracy of 95.24%, which was very close to the actual best testing score of 95.30%.

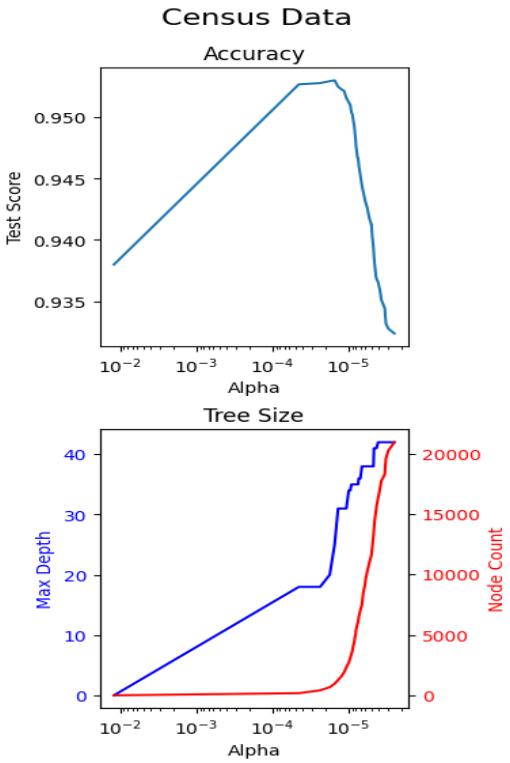


Figure 2: Census Data Decision Tree: The right side of the upper chart shows overfitting as the pruning is reduced and the number of nodes and tree depth increases. An unusual aspect is the left side of the chart shows very high accuracy with very few nodes. This stems from one feature being highly correlated to the labeled output.

To prune the tree, sklearn uses a minimal cost-complexity setting. Each node in a tree has an $\alpha$ value which measures the node misclassifications minus leaves misclassifications divided by the number of leaf nodes. The DecisionTreeClassifier then takes an $\alpha$ minimum setting, pruning all subtrees with a lower $\alpha$. For both datasets, an accuracy curve was created by fitting the tree with different $\alpha$ values.

For both data sets, I also performed cross validation to use the training set only to determine the best $\alpha$ parameter, with a 5 fold setting (80% of training data used for training, 20% used for validation). I would then use the test data at different $\alpha$ values to confirm the finding.

For the Census Data, Figure 1 shows the use of cross validation to estimate the best solution, in this case setting for $\alpha$, without using the test data. The best cross validation $\alpha$ value of 0.0000153 had a test accuracy of 95.24%, which was very close to the actual best testing score of 95.30%. Cross Validation provides a powerful tool for evaluation a given training set without reference to test data when available.

The Census Data, shown in Figure 2, mostly shows the expected curve. Accuracy improves as nodes are added, to a point, then accuracy declines due to over fitting as more nodes are added. Alpha, the pruning constant, leaves more nodes in the tree as it is decreased. The Census Data also showed a unique characteristic. One of its features (Capital Losses), showed a very high correlation with the labeled output, Income above $50k. Thus, a one node tree already showed 94% accuracy. Figure 2 shows this, with the the left side of the chart with few nodes already having a very high accuracy.

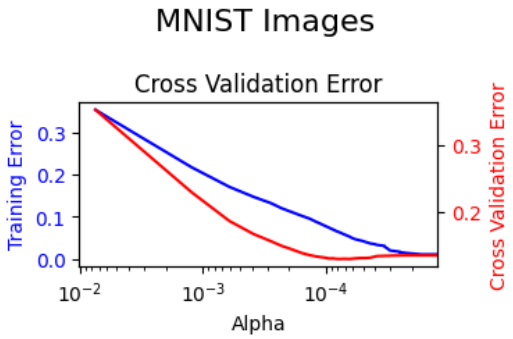


Figure 3: MNIST Images Cross Validation Error: The chart shows continual reduction in training error, while the Cross Validation error reaches a minimum at an $\alpha$ of 0.0000542 and then starts to rise. This was very close to the best $\alpha$ found using the testing data.

Figure 4 shows the cross validation error curves for the MNIST image data. The best cross validation $\alpha$ value of 0.0000762 had a test accuracy of 87.18%, while the actual best testing score of 88.54%. This was a larger gap than found in the Census data, but still relatively close.

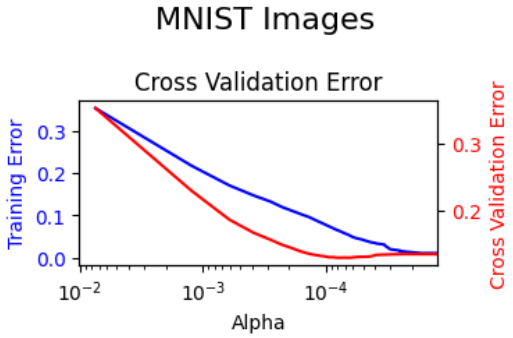


Figure 4: MNIST Images Cross Validation Error: The chart shows continual reduction in training error, while the Cross Validation error reaches a minimum at an $\alpha$ of 0.0000542 and then starts to rise. This was very close to the best $\alpha$ found using the testing data.

The MNIST image data, shown in Figure **??**, showed a more normal accuracy curve than the Census Data, with accuracy improving dramatically with added nodes until a maximum accuracy was reached, then accuracy declines due to over fitting. The maximum accuracy achieved was 88.7%, which was significantly below the census data. A decision tree is looking at one pixel at a time, so small variations in position or size would be very challenging for this algo-

rithm. I think other classifiers, especially the neural network to better handle image data.

## Neural Networks

For the neural network, I used PyTorch with two hidden layers, the Adam optimizer and Linear layers. I then adjusted the size of the layer, the learning rate, and epoch cycles to find the optimal settings.

In just the initial setup for the Census data, I quickly determined I would need to scale the data as the solver would often ”blow up” with really large values. I used the sklearn StandardScaler module, which removes the mean from each feature then applies a sigmoid function to create values from -1 to 1. This is somewhat suspect for the categorical features, as there is not really a positional relationship, but it was still effective. I also switched to using a 4 Fold Cross Validation model as I became more worried about the smaller data.

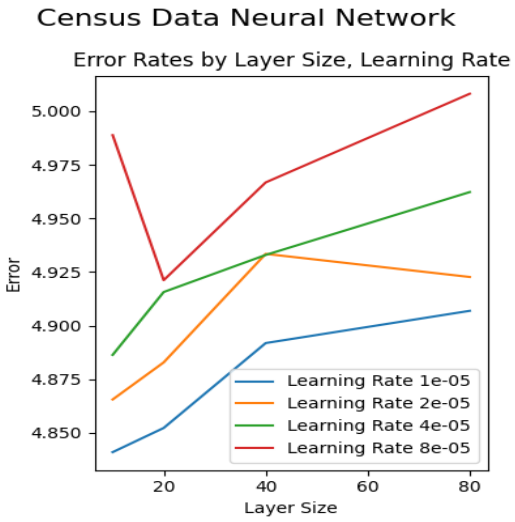


Figure 5: Census Data Neural Network: Review started to look for optimizing the neural network. The lowest error is clearly shown for the smaller learning rate with the smaller network.

Figure 5 clearly shows the preference for the smaller layer size and smaller learning rate, although the differences in absolute accuracy are quite small. This is a victory still due to the very skewed dataset, with only 6% of the data being labeled positively. The smaller layer size performance is straight out of the class lecture with regard to preferring simpler solutions. I had expected larger layer sizes to be more effective. The charts indicates that further exploration at the lower end was worthwhile.

Figure 6 shows one of the best performing settings from the initial review. It shows that the cross validation error was very close to the real test data error and a good guide for decision making. The chart also doesn't clearly have a movement up of the cross validation error, indicating more epoch training may be beneficial

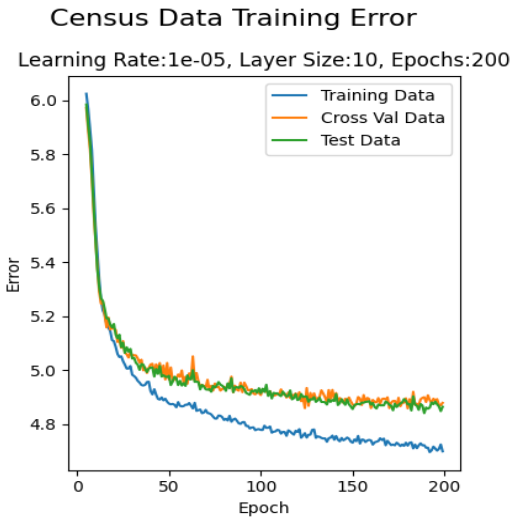


Figure 6: Census Data Neural Network Training Error: It shows that the cross validation error was very close to the real test data error and a good guide for decision making.

## References


1 Census Income (KDD). url: https://archive-beta.ics.uci.edu/ml/datasets/census+income+kdd.

2 The MNIST Database of Handwritten Digits. url: http://yann.lecun.com/exdb/mnist/.