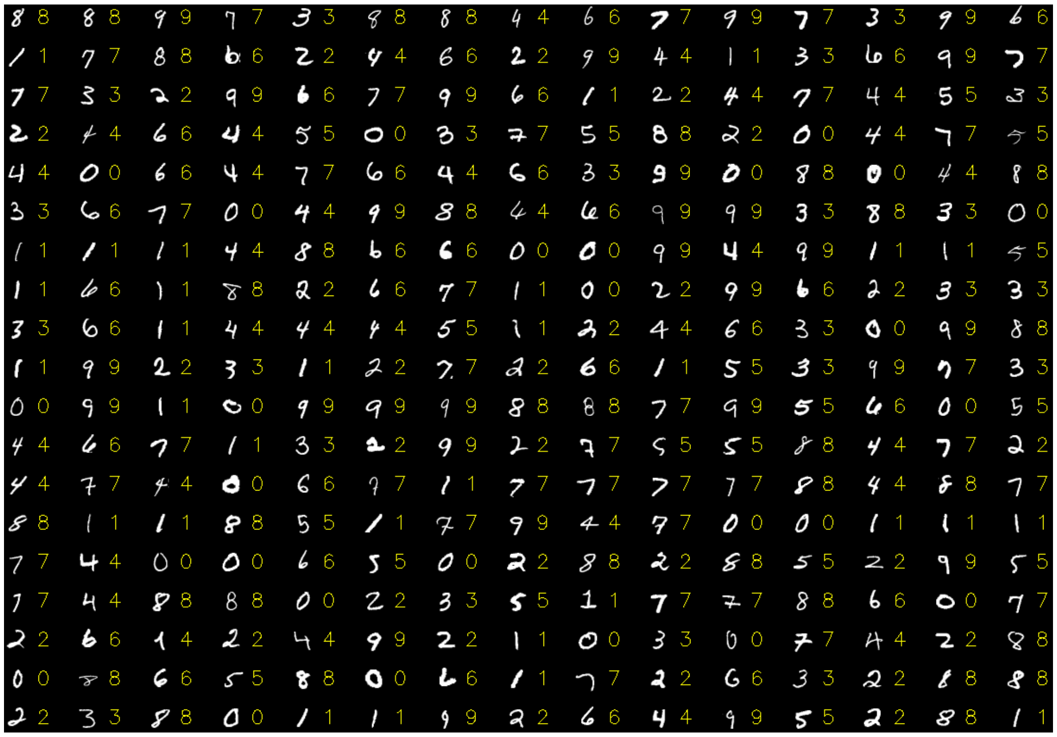
Joshua Levy
MATH 34 LMC
11/30/2017

Classifier Comparison- Neural Network vs. Nearest Neighbor Classifiers in the Task of Classifying Handwritten Digits



Research Question

For the practice of using supervised machine learning to classify handwritten digits into numbers 0 through 9, is a neural network classification model able to more accurately classify than a nearest neighbors model? After the nearest neighbors classification model and neural network models are trained on 600 handwritten digits, can the neural network model, on average, more accurately classify 100 handwritten digits?

The explanatory variable is which machine learning model is employed, whereas the response variable is the accuracy (number of successful classifications) of the specific model for classifying 100 digits at a time after being trained on 600 digits. We will use a matched pair t-test because each model will be acting on the same data.

Hypotheses

H_0: There is no difference in the mean classification accuracy on 100 digits between the nearest neighbors and neural network classifiers. ($\mu = 0$)

H_a: The neural network classifier is able to more accurately classify 100 digits on average than the nearest neighbors model. The nearest neighbors model is able to less accurately classify on average than the neural network model. ($\mu < 0$)

Where μ is the true matched pair difference in classification accuracy for handwritten digits (nearest neighbor's accuracy minus neural network model accuracy).

Population in Question

The population that we are looking at are handwritten digits written 0 through 9. We have chosen to sample 70000 digits from the MNIST data set and split them into 100 runs of 700 digits to determine the mean accuracy score differences for 100 digits between the two models.

Sampling and Random Assignment Methods

To conduct this experiment, we want to use two different machine learning models to look at images handwritten digits and decide which number the written digit is. To do this, we will be accessing a data set that includes 70000 images of handwritten digits, the well-known MNIST data set.

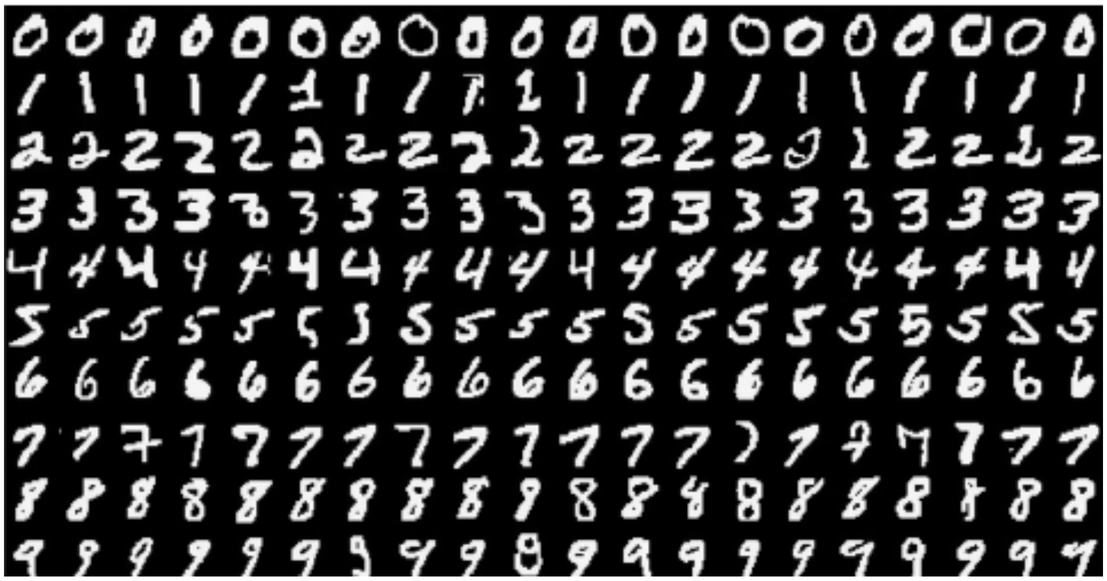


Figure 1: Handwritten digits from the MNIST dataset. Can the machine learning model correctly classify each handwritten digit?

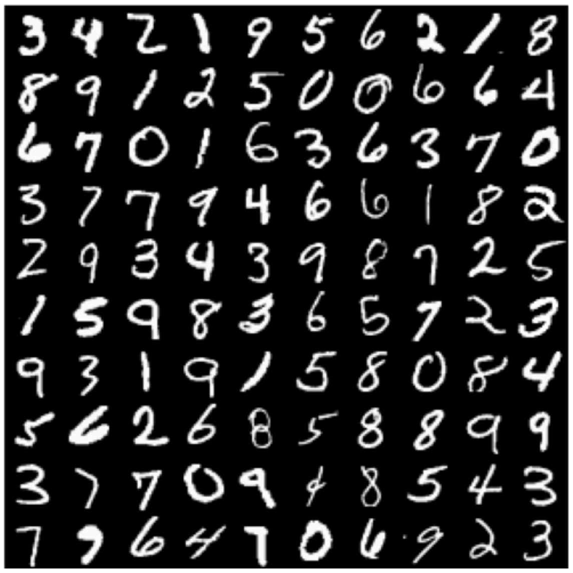


Figure 2: Example of the 100 testing handwritten digits for a single run of the classification models. After the models are fit to classify 600 other random handwritten digits, they will attempt to recognize these 100 digits, and for a run, an accuracy score is the number of successful classifications of the test digits. We see that the numbers chosen are random.

For each picture included, there is a corresponding correct number associated with it, numbered 0-9. A data point (or run) in this context means that we will train the machine learning model on 600 random digits, and then test the models on the next 100 digits, and output the number of digits each model was able to correctly classify (accuracy measure). Then we will subtract the two accuracy scores of the nearest neighbors classifier by the neural network classifier's accuracy.

Note that a single run contains 700 randomly selected handwritten digits, where 600 digits are used for training the machine learning model, and 100 digits are used for testing, and the accuracy score is computed from the classifications of 100 testing digits. Random samples were selected by using a computer algorithm to randomize the digits of the MNIST data set, and then splitting the randomized set up into 700 digits per run, and splitting each of those runs into training and testing datasets correspondingly. Our sample size is thus 100 runs. A combination of R and Python programming languages will be employed to handle the computations.

This is an experiment, but there is no control group that exhibits standard classification tendencies to compare against.

Quick Overview of Machine Learning Classification Methods

The Data Set

Each handwritten digit image contains 728 pixels, in grayscale. We can turn each of these digits into a vector, with each element in the vector being the relative darkness of the pixel, with 1 being completely black pixel, and 0 being a white pixel.

The label assigned to a digit is what is used to train the machine learning model. For the 728-element vector, a correct number is assigned to it from the MNIST data set. A set of 600 training handwritten digits can be stacked on each other, forming a 600 by 728 element matrix, where the rows correspond to the particular training digit and the columns are the pixel darkness intensities. A model is trained through a few different methods that will be described. The final model is a weights matrix and bias vector that operate on the digit **x**, which yields some 10-dimensional vector. The index of the resulting 10-dimensional vector that contains the maximum of the vector is the classification digit.

Training data set: $\begin{bmatrix} .45 & \cdots & .68 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & .11 \end{bmatrix}$ Testing data set: $\begin{bmatrix} .33 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ .144 & \cdots & .782 \end{bmatrix}$ Predicted Labels: $\begin{bmatrix} 7 \\ \vdots \\ 3 \end{bmatrix}$

Training Labels: $\begin{bmatrix} 9 \\ \vdots \\ 3 \end{bmatrix}$ Actual Labels/Digits to be Compared against Predicted Labels: $\begin{bmatrix} 9 \\ \vdots \\ 3 \end{bmatrix}$

$$[.45 \;\; \ldots \;\; .68]^{\mathrm{T}}\, \mathbf{W} + \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \rightarrow 9$$

A 10-dimensional vector output to the machine learning algorithm converted into a digit.

Nearest Neighbors Classifier

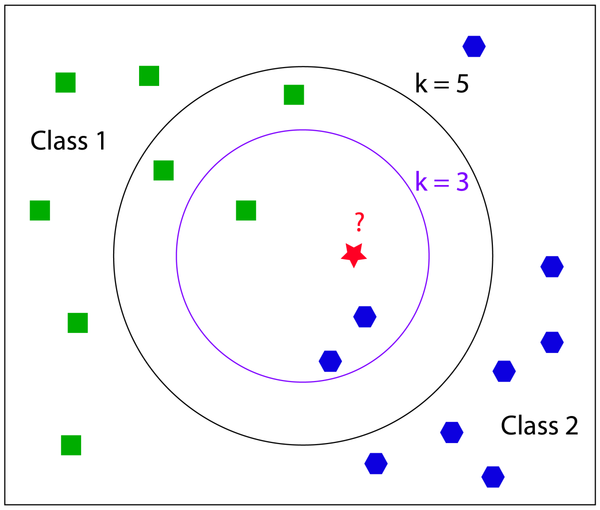


Figure 3: Nearest Neighbor Classifier

Imagine that each digit is a data point somewhere in 2-dimensional space (the points are actually in 728-dimensional space, beyond the scope of our discussion). This classification model first finds the k (number specified by the user) closest training data points to a particular test data point using some measure of distance. Each of these k-closest data points have a particular correct digit label attached to them, because they are training data points. The algorithm tallies up the number of training points belonging to a particular class, and the new test data point or handwritten digit is assigned a digit based on which class of the k-closest training points is the most prevalent. In the above example, for the 3 closest points to the red point, there are more blue points than green points, so the red point can be classified as blue. When k = 5, this changes to green. In the algorithm employed in our experiment, k=5.

Neural Network Classifier

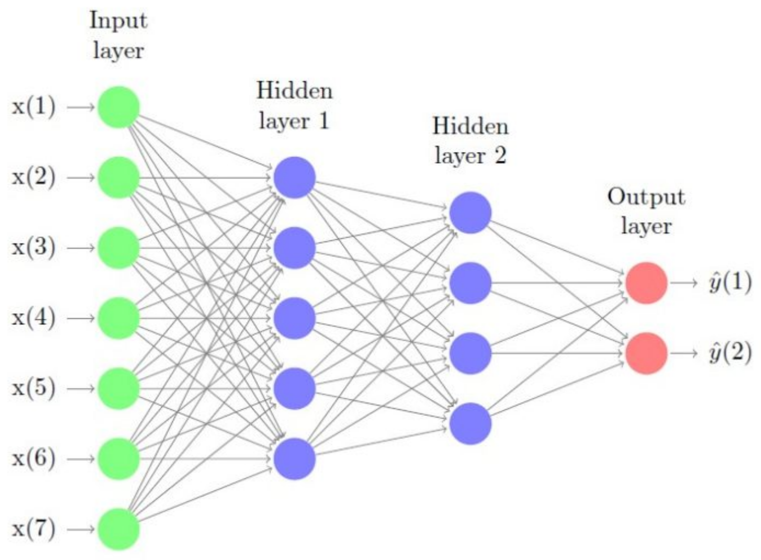


Figure 4: Neural Network Model

Weights **W** and bias **b** and collective activation functions F, are applied to vector **x** to produce output layer **y**: (oversimplified)

$F(\mathbf{x}^T \mathbf{W} + \mathbf{b}) = \mathbf{y}$

Essentially, a neural network model is a type of regression model that takes a digit represented by the 728 element vector **x**, and transforms it via activation functions F, weights **W** and biases **b** into a 10-dimensional element vector **y**, where each element of the output matrix corresponds to a particular output digit. It is a neural network model because each element of **x** is fed into elements of hidden layers. These elements of hidden layers function as neurons of a brain, and transform each x-value via an activation function $f_i(\mathbf{W_j x} + \mathbf{b})$, where f_i is some arbitrary function that is beyond the description of this project. There are numerous activation functions in multiple hidden layers, which introduce large complexities to this nonlinear model. Each layer of the algorithm is represented by some vector of a specified dimensionality, and weights are assigned between nodes (representing a vector element) in each layer. The output of this model is a 10-

dimensional vector, and if you take the maximum entry of this vector, you get a predicted digit value. To train it, the error is found between the output found from the initial model, with functions F, weights **W** and biases **b**, and the actual training labels/digits found, and modify the model parameters F, **W**, and **b**, as such to minimize the total error or cost found by comparing all predicted digits to the actual digits (gradient descent). This model can be thought of as a nonlinear regression model with the independent variable having 728-dimensional representation and the dependent variable having 10-dimensional representation. The model used in our algorithm also includes three hidden layers, of vector dimensionality 50, 100, and 50 respectively.

These two machine learning models are supervised, meaning they have to be trained by data that have known outputs in order to make predictions on test data. The algorithms used in this project are not optimized for performance.

Data Description

I have chosen to not directly include the image data as aforementioned, as there are 70000 images with 728 elements each. Instead, I used Principal Component Analysis (an unsupervised machine learning method to capture the axes that contain most of the variation in the data) to project and display the data in 3 dimensions. Those runs have been included as html files appended to this report. Each run contains 100 digits, projected onto 3-D space. If you hover over a point with your cursor, you can see what the actual handwritten digit was. The points are colored according to what each of the two machine learning models found, and you can change the coloring of the points by dragging the slider on the bottom of the page to the right. Note that the coloring changes as you drag the slider between coloring that describes the actual handwritten digits to coloring plots that describe the results of each model. This means that the models incorrectly classified some of the numbers, and thus, the accuracy of the model is the total number of points of which their colors do not change between actual and model predicted.

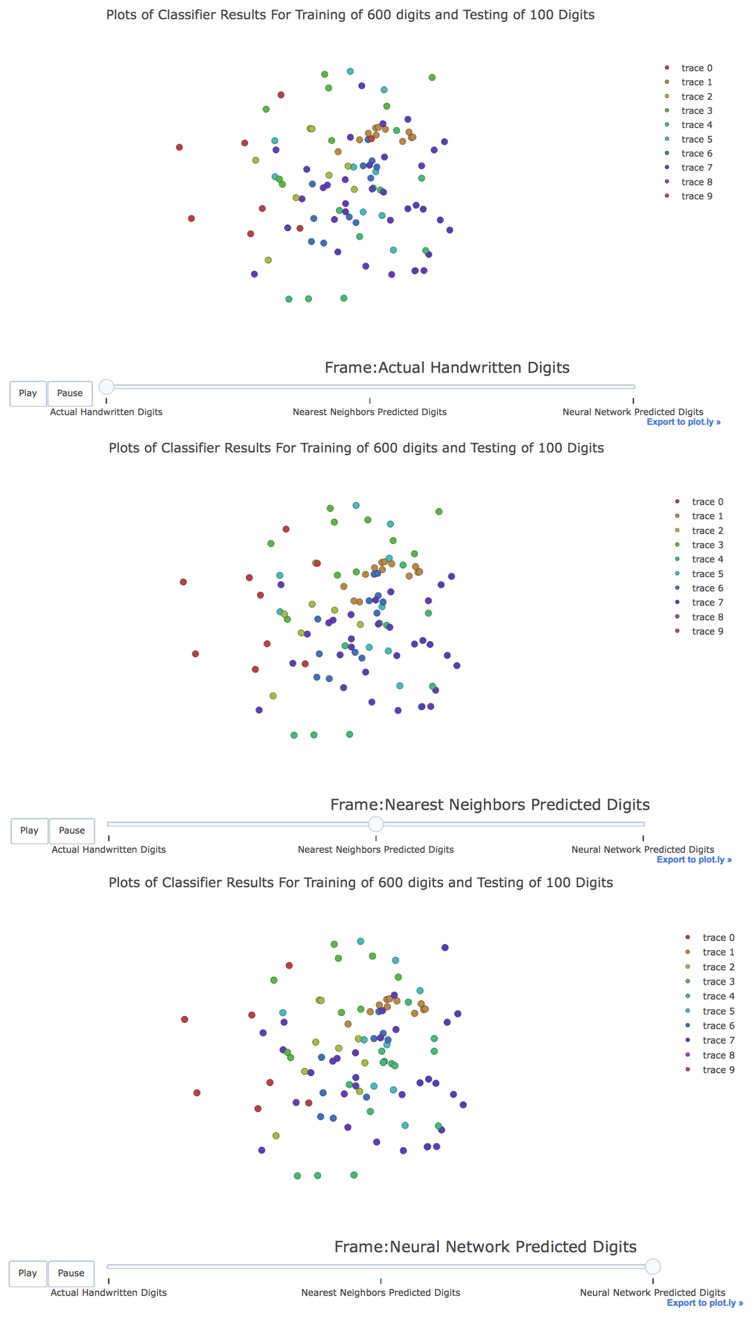
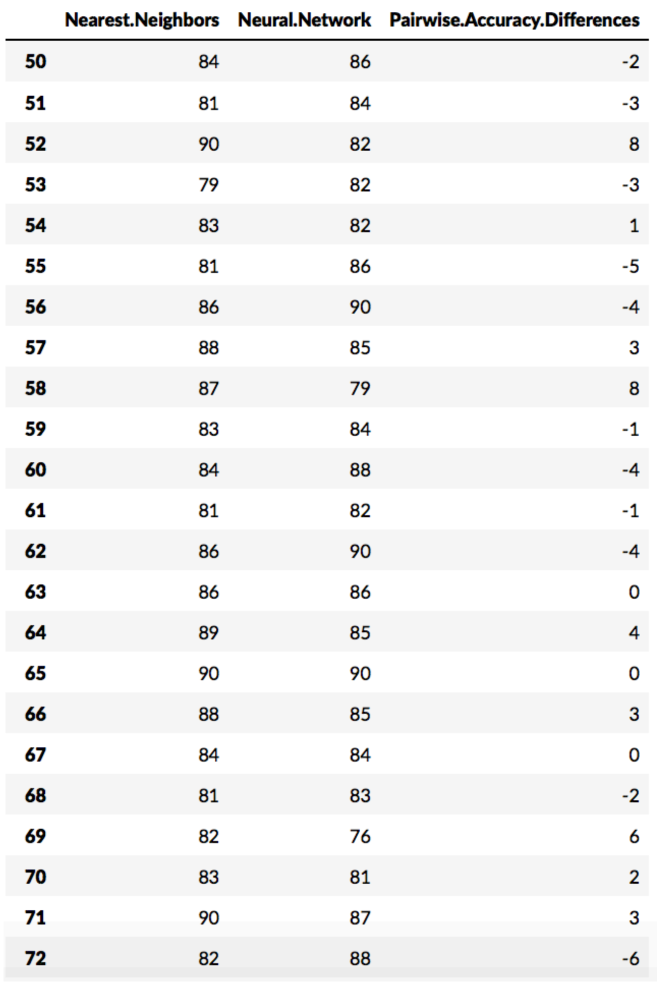


Figure 5: 3D Plot of Run 1's 100 test data handwritten digits with classifier outputs

From this, I was able to find the accuracy values for the Neural Network and Nearest Neighbor classifiers, and subtracted the neural network classifier accuracy from the nearest neighbor classifier accuracy for each of the 100 runs of 100 test digits to find pairwise differences in classifier accuracy.

The output of this data can be found in the attached csv file.

Here is a screenshot of some of that data:



	Nearest.Neighbors	Neural.Network	Pairwise.Accuracy.Differences
50	84	86	-2
51	81	84	-3
52	90	82	8
53	79	82	-3
54	83	82	1
55	81	86	-5
56	86	90	-4
57	88	85	3
58	87	79	8
59	83	84	-1
60	84	88	-4
61	81	82	-1
62	86	90	-4
63	86	86	0
64	89	85	4
65	90	90	0
66	88	85	3
67	84	84	0
68	81	83	-2
69	82	76	6
70	83	81	2
71	90	87	3
72	82	88	-6

Figure 6: Classifier Accuracy Comparison

Summaries of Data

I plotted a side-by-side comparison of the boxplots of both classifier accuracies over the 100 runs, and a boxplot and histogram of their pairwise differences, and the 5-number summary for the pairwise accuracy differences are posted below:

5-number summary of pairwise differences in classifier accuracy

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  -9.00   -3.00    0.00   -0.28    2.00   12.00 
```

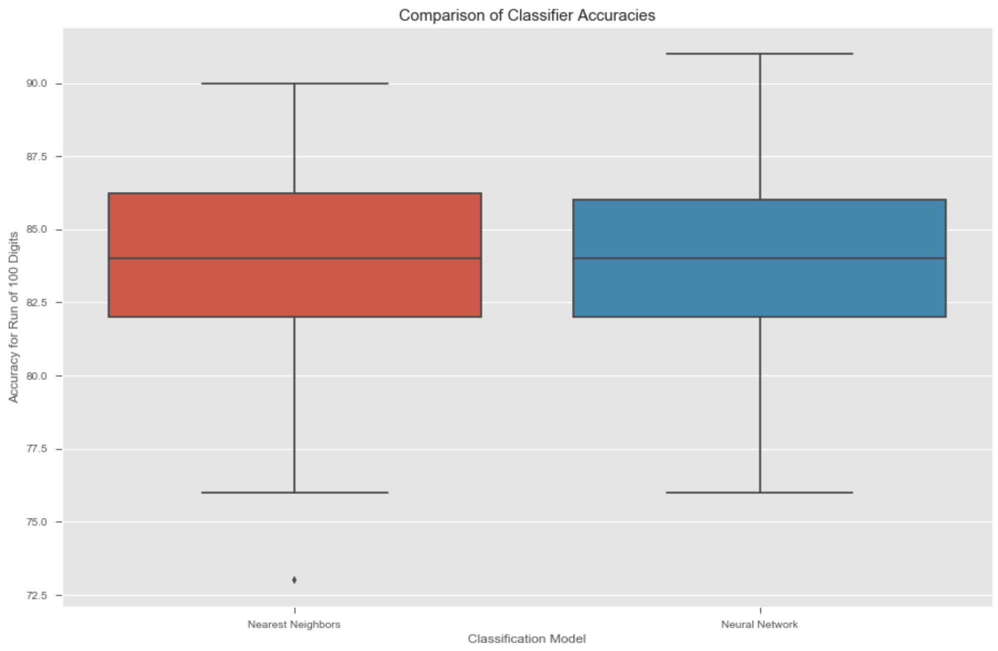
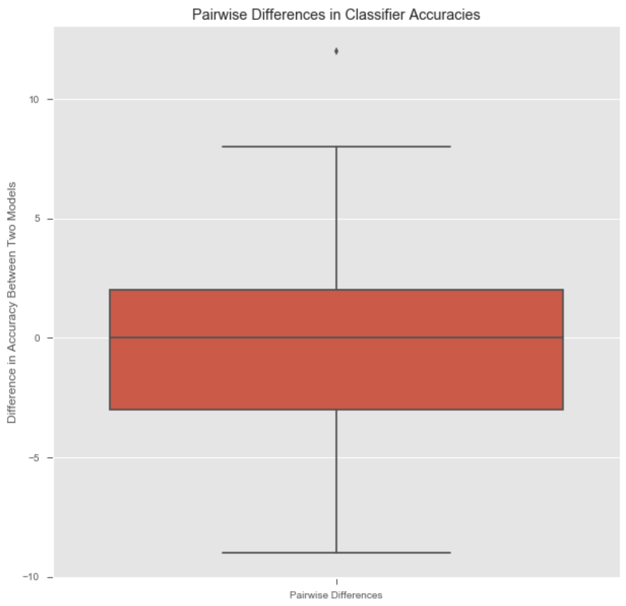


Figure 7: Side-by-Side Boxplot Comparison of Classification Accuracies



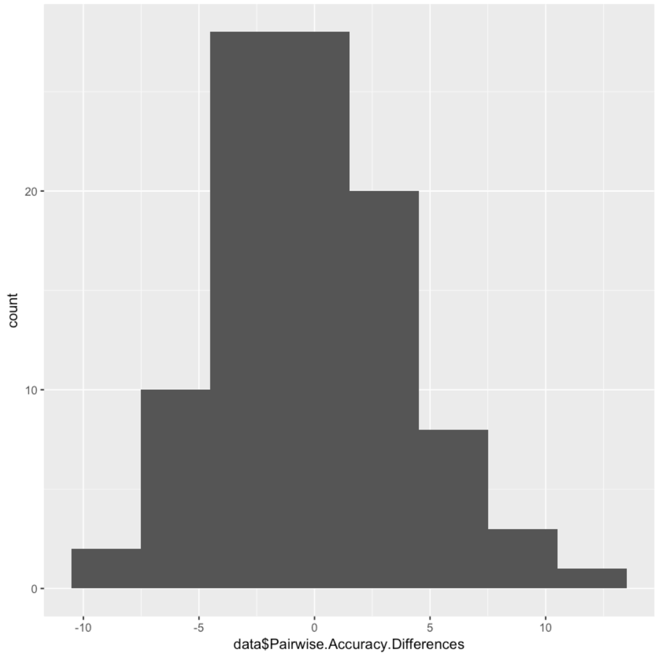


Figure 9: Histogram of Pairwise Differences in Classifier Accuracy

Figure 8: Boxplot of Pairwise Accuracy Differences

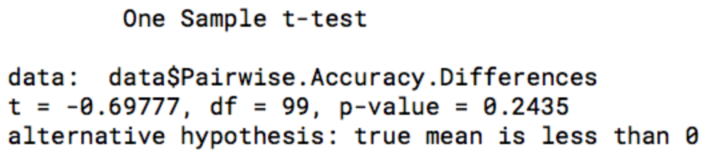
We see that the pairwise different plots appear normally distributed, with typical accuracy differences found between -3 and 2. The distribution is centered around 0, and one outlier was identified at a difference of 12 accuracy points.

Since the distribution of the pairwise accuracy differences appear to look normal, and the sample size is greater than 30 (100 runs), we have satisfied the conditions necessary to run a matched pair t-test on the pairwise differences in classifier accuracy, as this satisfies the central limit theorem. I calculated the mean and standard deviation of the sample to be -0.28 and 4.013 respectively, with 99 degrees of freedom.

Hypothesis Test

Our null hypothesis is that the population mean of the pairwise differences in accuracies of the two models is equal to 0. Our alternative hypothesis is that this population mean is less than 0, which would indicate the neural network model has higher classifier accuracy than the nearest neighbors model.

Using R, I conducted a matched pairs t-test to find the following results for the pairwise differences in classification accuracy:

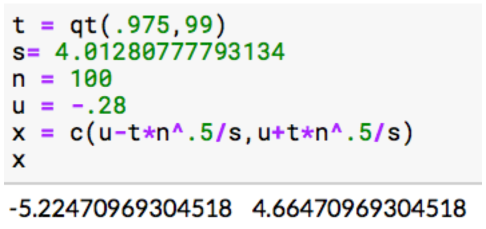


```
            One Sample t-test

data:  data$Pairwise.Accuracy.Differences
t = -0.69777, df = 99, p-value = 0.2435
alternative hypothesis: true mean is less than 0
```

Confidence Interval

The 95% confidence interval was calculated using R. The t-statistic for a 95% confidence interval is qt(.975,99) = 1.98. With a mean of -0.28, standard deviation of 4.0128, and sample size of 100, we use R to calculate the following, where t is the t-score statistic corresponding to a 95% confidence interval, s is the sample standard deviation, u is the sample mean, and n is the sample size:



```
t = qt(.975,99)
s= 4.01280777793134
n = 100
u = -.28
x = c(u-t*n^.5/s,u+t*n^.5/s)
x
```

```
-5.22470969304518  4.66470969304518
```

We are 95% confident that the true mean difference in classifier accuracy between using the nearest neighbor's model versus the neural network model is between -5.22 and 4.66.

Results and Conclusion

The goal of our experiment was to determine if the neural network classification model could more accurately classify digits than the nearest neighbors algorithm, given the same set of handwritten digits. The true mean of the population of pairwise differences in classification accuracies (nearest neighbor accuracy minus neural network accuracy) for runs of 100 handwritten digits is μ.

Our hypotheses were as follows:
H_0: $\mu = 0$, Both classifiers have equal classifier accuracies.
H_a: $\mu < 0$, The neural network algorithm has a higher average classification accuracy than the nearest neighbors classifier.

We employed both machine learning models and trained them using the same 600 handwritten digits and made predictions of handwritten digits using 100 testing handwritten digits, and we repeated this process for 100 runs of these training/testing data splits. The accuracy of each model for each of the 100 runs was found by comparing the predicted 100 digits to the actual digits, and the neural network classification accuracy was subtracted from the nearest neighbors classification accuracy for each run to find our pairwise differences. We plotted the distribution of these pairwise differences in classifier accuracy and found that the distribution was normal, with one outlier, and had a sample size of 100. These observations passed the central limit theorem and satisfied the conditions necessary for using a t-test for the matched pair experimental design.

We are 95% confident that the true mean difference in classification accuracy, μ, for a run of 100 test digits is between -5.22 and 4.66. Zero is contained inside this confidence interval, so it may be possible that there are no statistically significant differences in classification accuracy between the two classifiers.

With a t-score of -0.698, 99 degrees of freedom, and a corresponding p-value of .2435, for an alpha significance level of 0.05, we do not have enough evidence to conclude that the neural network algorithm had a higher average classification accuracy than the nearest neighbors classifier, and thus we fail to reject the null hypothesis that both classifiers have equal classifier accuracies. There are no statistically significant differences in classification accuracy between the two classifiers when it comes to classifying 100 test digits.

Discussion

We were able to analyze some pretty powerful supervised machine learning algorithms today and test to see whether one algorithm performs better than another. However, there are many limitations to this study, some of which I will discuss. First, the data points fed into the neural network model had to first undergo a procedure called standard scaling, which means that I did some preprocessing in order to boost the classification accuracy. When I first ran and trained the model without standard scaling, it performed much worse than the nearest neighbors algorithm. The nearest neighbor algorithm also performs better after being transformed to a lower dimension because that makes the distance calculations between data points more meaningful. Furthermore, the parameters used in the algorithm, the number of nearest neighbors, number of hidden layers and their sizes, distance metrics, etc. are not fine-tuned to produce a well-fit algorithm that most accurately makes predictions given the classifier type. Perhaps with the right model parameters, one classifier outperforms the other. There is also a problem of overfitting. If the fit of the model is too good, then this no longer becomes a regression/classification problem, as the new testing data points may have a higher chance of lying outside of certain classified regions. What we need in the future is the best fit for each model to make a better comparison between the models. In the future, I may experiment with hyperparameter optimization to determine the proper values of the parameters of each model to make the best fits and the best comparison. I may also experiment with changing the sample sizes and the number of digits included on a run. We used 70000 handwritten digits to make predictions and comparisons surrounding an unlimited number of handwritten digits out there. If we include less digits per run, then our sample size may be higher (less digits per run means we can have more runs), but with less points, the models may have a harder time fitting to the digits for each run. If we include more digits, classification accuracy may improve, but we would not have enough total number of handwritten digits from which to experiment with. Finally, another limitation is that this study is limited to handwritten digits that are positioned in the same place on the image, and the conclusions only generalize to handwritten digits that have 728 pixels per image. In a further study, more than 70000 handwritten digits should be used and we should experiment with the size and quality of the images as well as with doing away with preprocessing to eliminate additional sources of confounding factors.

Despite the limitations, this experiment shows how comparable the performance and accuracies of two great and powerful machine learning algorithms (neural networks and nearest neighbors), and I think the matched pair design was a good choice for analyzing the results. Thank you for the opportunity to take this class and I hope that you have enjoyed this report!

Code and Additional Resources

Data tables and 3-D projections of all 100 runs of 100 test digits can be found in the csv and html files attached to this doc.

The code used to run the calculations on the data can be found on the following two webpages (These may be an outdated pages, I had to fix some broken code and uploaded it to a different location, which you can view upon request, p-value is different because of small modifications to machine learning algorithms, may not reflect current state of project; I recently uploaded the info and believe that these are the correct scripts, particularly the last two reflect new changes):

1. https://github.com/jlevy44/JupyterTests/blob/master/Stat_LMC_Final_Project_Python.ipynb
2. https://github.com/jlevy44/JupyterTests/blob/master/fixed_LMC_Stats_Final.ipynb
3. https://github.com/jlevy44/JupyterTests/blob/master/R_Math34_Final_Project.ipynb