Vinayak Ravichandran

Github: @vravich01

Email: vinayak1ravichandran@gmail.com

Image Classification & Model Training

Introduction:

In this analysis, handwritten digits (either “4” or “6”) were classified into their respective groups using a model which was trained by a known value data set. In *Section 1*, the training data was used to build a linear regression model via Matlab programming. In *Section 2*, the model was used to classify testing data.

Section 1: Dimension Reduction

1.1) Methods & Figures

1.1.abc)

Initially, four data sets were prepared beforehand for this analysis: $trainX$, $trainY$, $testX$, $testY$. The first contains the handwritten digit data whose actual digits are known. The second contains classification data which stratifies the known digits into the “4” cluster or the “6” cluster (by using indexing which corresponds to the digit data). The last two sets contain the same type of data, but the model will not be given these data sets.

The data sets were converted from unsigned 8-bit integers into floating point double precision format. Then, the digit data was centered by subtracting the respective column mean from each value; this allows the data to be better visualized.

1.1.de)

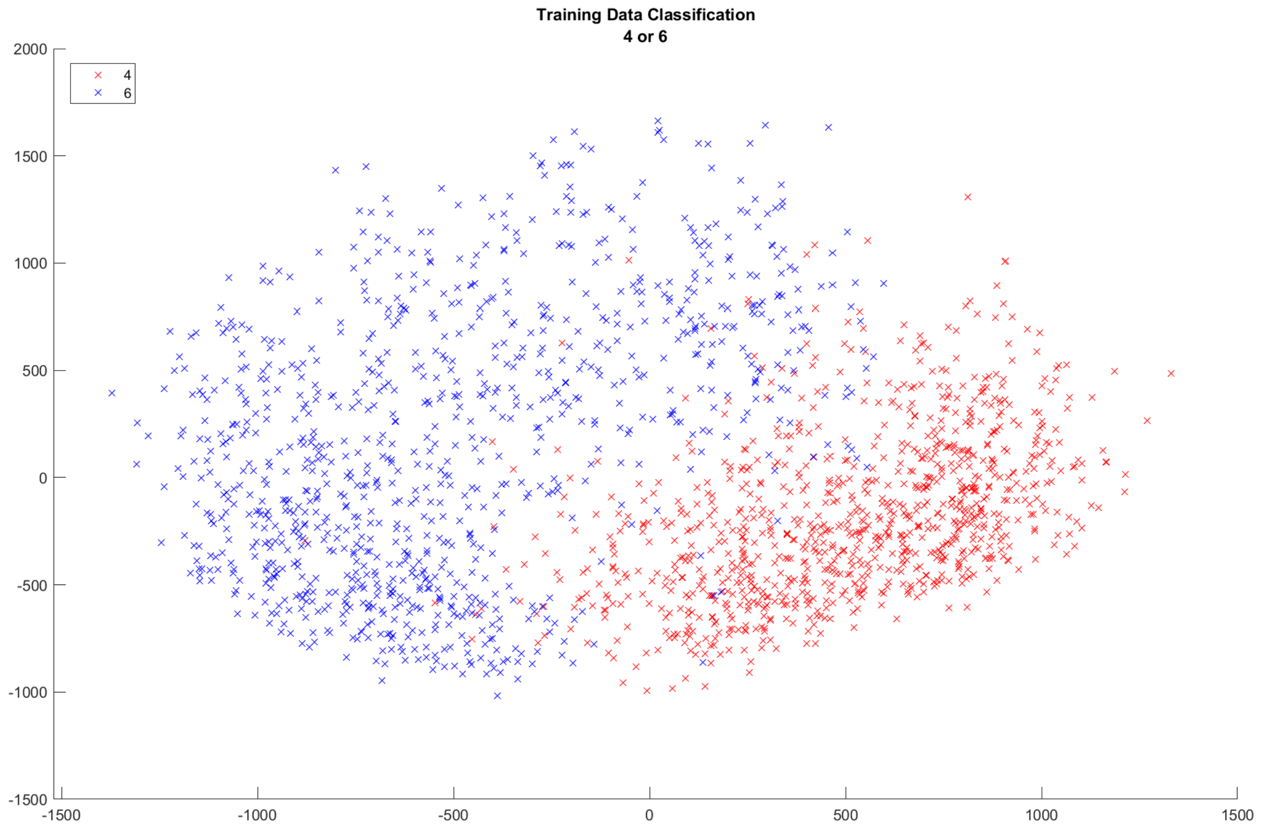
Then, the digit data was truncated into two dimensions using SVD ($k = 2$). This certainly obfuscates the information stored in the digit data, but it is a necessary evil since it allows us to visualize the model on a standard Cartesian plane.

Finally, a scatterplot of the training digit data was made; the training classification data was used to color the points depending on whether they are a “4” or a “6.” The scatterplot is shown below.

Vinayak Ravichandran

Github: @vravich01

Email: vinayak1ravichandran@gmail.com



1.2) Conclusion

An important thing to note from *Section 1* is the added benefit of centering the data. Originally, the values were scattered without centering (accidentally) and the clusters were not identifiable (without color) as easily as they are with centering. It is a clever way to transform the data to improve access to visual analysis without altering the information embedded within the data.

Another thing worth noting is the necessity for truncation. By the nature of this analysis, there is an imposed restriction that disallows the use of every dimension provided In the training data: we want a visual, specifically one which sits nicely on a computer screen or piece of paper, both of which are two-dimensional. It would have been simple to extend this analysis to three dimensions since the Cartesian plane can be expanded into the familiar three-dimensional Euclidean space $\mathbb{R}^3$ but we must forgo higher dimensional visuality. It is possible to deconstruct a higher dimensional graph into many subgraphs of a limited number of dimensions, but that option sacrifices visual identification of clusters since any one graph would omit at least one dimension. However, it is certainly possible to perform this analysis in higher dimensions to obtain several numerical results such as accurate, false positive, and false negative rates.

Vinayak Ravichandran

Github: @vravich01

Email: vinayak1ravichandran@gmail.com

Section 2: Classification

2.1) Methods & Figures

2.1.abd)

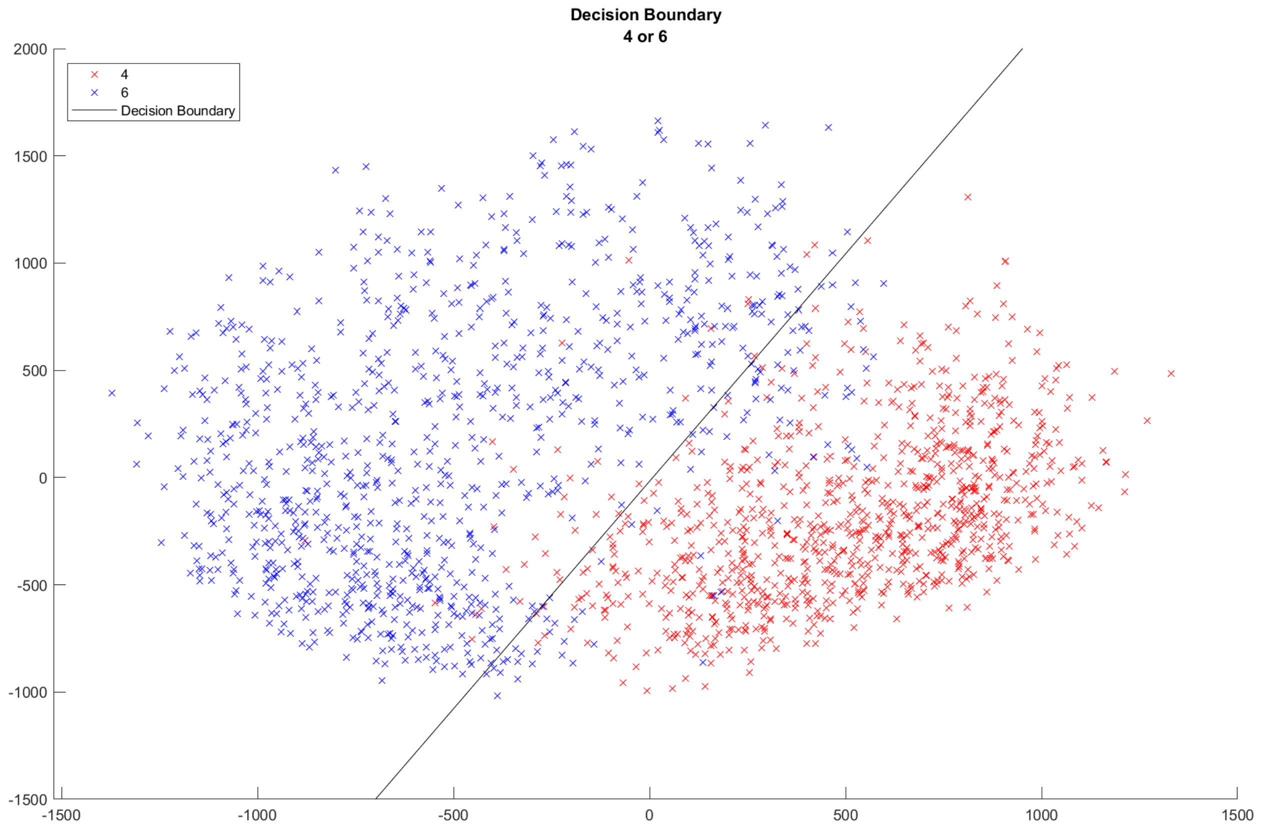
Then, a numeric variable (signed, normalized/unit scalar) for the classification of the training data was created so that a linear regression could be performed on the training data. Without this, the method used to generate a decision boundary will be inaccurate since 4 and 6 are non-normalized values; more clever equations would be needed.

Instead, we perform a linear regression in the form $z_i \approx a_0 + a_1 x_1 + a_2 x_2$ where z_i is the numeric variable and (x_1, x_2) is the ith training data point. This presents itself as a least squares problem $min_1(||Ar - b||_2)$ where A is the truncated matrix of training data points (x_1, x_2) and b is the column vector containing numeric variable classifications for the data points. Note that the truncated matrix must be padded with a leading column of zeroes to allow the matrix operations to be carried out such that the solution to the least squares problem, r, is a 3-element vector. Essentially, each (x_1, x_2) is mapped to ± 1, and the vector r is the best solution to the least squares problem which defines the mapping. The decision boundary (linear regression) is shown below, plotted over the training data scatterplot.

Vinayak Ravichandran

Github: @vravich01

Email: vinayak1ravichandran@gmail.com



2.1.ce)

The accuracy rate of the model (linear regression) was computed using the training data. Note that this accuracy rate is not 100% since the essence of a least squares problem is to find the solution which minimizes error for a problem which has no exact solution.

Then, the model was used to classify the testing digit data. An accuracy rate for this was also computed since the classification data for the testing digit data was provided.

Accuracy rates were computed using the entries of the associated confusion matrix for the respect data set.

Vinayak Ravichandran

Github: @vravich01

Email: vinayak1ravichandran@gmail.com

2.2) Results

2.2.d)

Constant:	a_0	a_1	a_2
Value:	-0.01020408163265	0.001238251315099	-0.000583868133289

2.2.ce)

Training Data		Actual:	
		"4"	"6"
Experimental:	"4"	907	83
	"6"	34	936
	Accuracy Rate: 94.03%	6 mistaken as 4: 4.23%	4 mistaken as 6: 1.73%

Testing Data		Actual:	
		"4"	"6"
Experimental:	"4"	146	17
	"6"	10	151
	Accuracy Rate: 91.67%	6 mistaken as 4: 5.25%	4 mistaken as 6: 3.09%

2.3) Conclusion

A good takeaway is to notice that the model, which was built upon the training data, failed in perfectly classifying the training data. This is expected, as previously explained.

Additionally, we can observe that the model performed well in classifying the testing data in comparison to the training data: both are above 90% accuracy.

It would be somewhat reasonable to hypothesize that both accuracy rates would increase if more dimensions are used, but, more interestingly, we could hypothesize about the benefits of integrating the data from each testing program execution. The decision boundary would adapt each time the model was used to classify data, presumably getting more accurate each time. I suppose this is a rudimentary approach to the basics of machine learning.