

Semeion Digit Classification

Written by: Gnanadeep Chinnabathini & Yingjie Chen



April 30, 2021

University at Buffalo

# Table of Contents

1. Background Information About the Dataset………………………………………….2
2. Hypothesis Generation…………………………………………………………………3
3. Loading Packages and Data…………………………………………………………..3
4. Exploratory Data Analysis……………………………………………………………...4
5. Further Improvement……………………………………………………………………7
6. Summary……………………………………………………………………………..…10
7. Work Allocation………………………………………………………………………...11

# 1. Background Information About the Dataset

The dataset we were given was called Semeion. It is based on 1593 handwritten digits from around 80 people. Each people wrote the digits from 0 to 9 twice, first time in an accurate way and a fast way with no guarantee of accuracy for the second time.

The dataset consists of 1593 rows which correspond to the 1593 written digits and 256 attributes which correspond to the scanned image of each digit based on the grey scale. After we examined the dataset, we found out there was 266 columns in each row. The first 256 columns are Booleans that represent the 256 attributes and last 10 columns represent the actual digit written. For example, 1000000000 represents number 0 and 0100000000 represents number 1, so the index (starting from 0) of the column where the ‘1’ is situated indicates the value of the written digit.

To better understand the dataset, the next step we took was trying to recreate the image using the 256 attributes given. We first split the 256 attributes evenly into 16 rows and connected all the ‘1’s to create the image. One example is shown below.

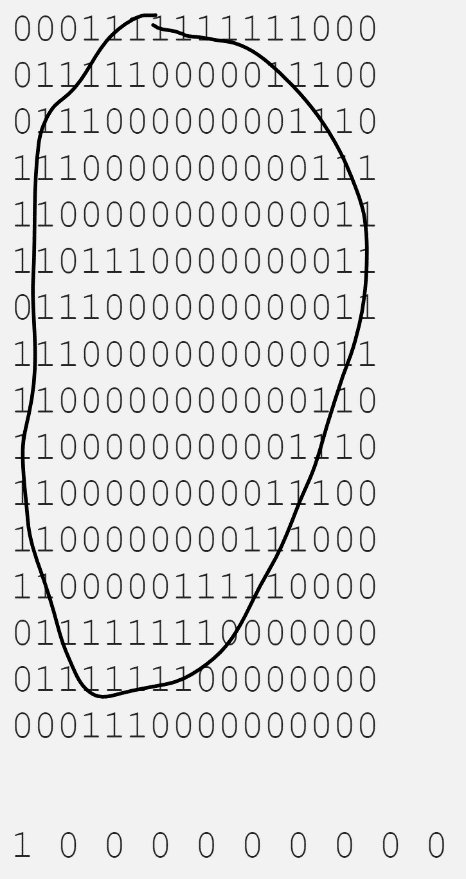


Figure 1: Recreation of the scanned image using the 256 attributes

# 2.Hypothesis Generation

The aim of this project is to develop and train a classification method to successfully identify the written digit using the Semeion data.

Based on the trial we have done to recreate the image from the dataset, we have gained some fundamental knowledge of the dataset. We then started to screen all the methods we have been taught so far to deal with the associated tasks of this dataset. Since we have illustrated that the dataset was transformed from scanned images, it means each row should be treated as a whole, with the first 256 column as our input arguments and the last 10 being our expected output.

And we have instructed to randomly select 5 samples out of 1593 as our test set, so we won’t have to do a traditional 80/20 split or k-fold validation, but they could be useful when we develop and train our algorithm. Also, our test set is much smaller compared to the training set, so it should not result in overfitting the dataset.

Our initial thought is to divide the dataset into 10 classes based on number 0 to 9. In this way we could apply classification methods, such as KNN and SVM. Another approach was using perceptron, so we wouldn’t have to pre-classify our dataset and leave all the work to the algorithm.

# 3.Loading Packages and Data

The original Semeion data has 266 columns and we separate it into two parts, the first 256 columns and the last 10 columns. We then translate the last 10 columns into numbers ranging from 0 to 9 for simplicity reasons.

The packages we used for this project are:

1. e1071:

This package is installed to perform SVM and tuning.

1. class:

This package is installed to perform KNN.

1. neuralnet:

This package is installed to construct a neural net.

1. caret

This package is installed to train and tune our predicative models

**R code used for processing data:**

original\_data <- semeion[1:256]

original\_data$V257 <- 0\*semeion$V257+1\*semeion$V258+2\*semeion$V259+3\*semeion$V260+4\*semeion$V261+5\*semeion$V262+6\*semeion$V263+7\*semeion$V264+8\*semeion$V265+9\*semeion$V266

# 4.Exploratory Data Analysis and Model Building

In the early stage of our project, we hand-picked 5 samples that represented number 1,4,2,6,0 as our test data and used the rest as our training data. The purpose of this part is to firstly develop a well-rounded knowledge of the packages and functions that we will be using for the whole course of this project. It also serves as a basis for us to improve our methods in the future.

In this part, we were testing a wide variety of methods and screening the best solution for us to work on during the rest of this project.

## 4.1 k-Nearest Neighbors (kNN)

Since the Semeion dataset is consisted of only 0s and 1s, there will be no need to normalize the data. We then performed our first kNN using our hand-picked samples as the test data, the result is plotted below:

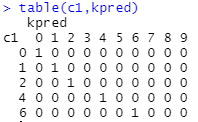


Figure 2: KNN on hand-picked samples

In Figure 2, c1 is the factor of true classifications of the training set and kpred is our result. It can be seen from the table that kNN achieved an accuracy of 100% out of 5 test samples.

**R code used for kNN:**

kpred <-knn(original\_data[-c(6,26,46,86,126),1:256],test\_data[,1:256], original\_data[-c(6,26,46,86,126),257], k=5)

cl <- original\_data[c(6,26,46,86,126),257]

table(c1,kpred)

## 4.2 Support Vector Machines (SVM)

The second method we applied is SVM. Even though our outcome variable are numbers, we cannot apply the regression formulation of SVM here considering the outcome can only be integers from 0 to 9. Therefore, our SVM model was based on the classification formulation and we still used the same test samples as we did with kNN. Besides the raw classification, we also looked at the probabilities to see how the model is fitted.

Our SVM model successfully classified all 5 test samples. But when we dive deeper into the possibilities, the results started to vary a little bit. For numbers like 0 and 2, the probability can reach over 99%, but for the number 1, the probability is only 93%.

|  |  |
| --- | --- |
|  | probability |
| 0 | 0.993876088 |
| 1 | 0.938311083 |
| 2 | 0.998307026 |
| 4 | 0.9843912 |
| 6 | 0.9752526 |

Table 1: SVM probability results

Despite slight difference in probability when identifying some of the numbers, the SVM model with an average probability of around 97% still gave a very accurate prediction overall.

**R code used for SVM:**

svm1 <- svm(as.factor(V257)~. ,data=original\_data[-c(6,26,46,86,126),], probability=TRUE)

result <- predict(svm1,original\_data[c(6,26,46,86,126),],probability = TRUE)

## 4.3 Perceptron/Neural Networks

Artificial neural networks are most often created to mimic some functionality of an actual biological neural networks, in other words, they are built to process information in the same way as humans. By definition, neural networks are a perfect choice for what we are dealing with in this project.

As our problem statement is classification we use **cross entropy** as error function and the linear output is set to **False**. For our first case,we used 2 **hidden layers**: first layer has 16 neurons and the second layer has 8 neurons.

The result is shown below and the accuracy for each digit is marked with a red circle.

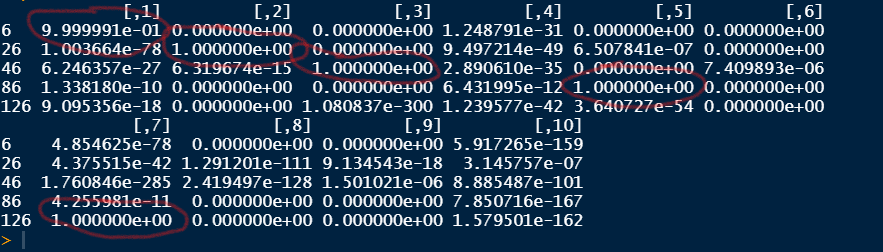


Figure 3: Result of a (8,4) neural network

We then increased the size of our neural network by increasing the number of neurons in each layer and the total number of layers.

Case 2: 5 layers with a (64,32,16,8,1) structure

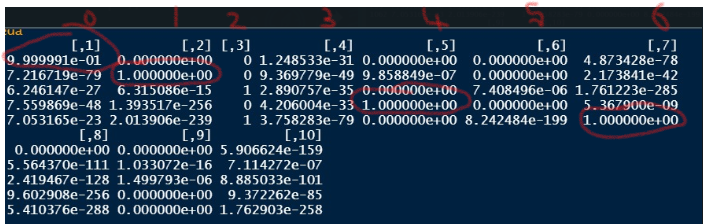


Figure 4: Result of a 5-layer neural network

Both case 1 and 2 yielded almost identical results, so there is no need to increase the size of the network, as the accuracy was already over 99.99%

**R code used for neural network:**

nn<- neuralnet( v257 ~., data = original\_data[-c(6,26,46,86,126),], linear.output = FALSE ,hidden = c(8,4),err.fct = "ce")

result <- predict(nn,original\_data[c(6,26,46,86,126),1:256])

# 5.Further Improvement

## 5.1 Cross-Validation kNN

We started with a cross-validation on our previous kNN function and extended the test range to the whole dataset. To better analyze the outcome, our cross-validation result was printed in a confusion matrix.

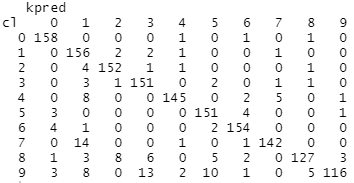


Figure 5: Cross-Validation kNN on Semeion dataset

The cross-validation yielded a general accuracy of around 90% on the whole dataset, as the correct results appear on the diagonal line and false results fall out of the diagonal line. It is worse than our previous case and it could be caused by potentially over fitting the data.

**R code used for cross-validation kNN:**

kpred1 <- knn.cv(original\_data[,1:256], original\_data[,257], k=11)

cl <- original\_data$V257

table (cl,kpred1)

## 5.2 Cross-Validation SVM

We also performed cross-validation analysis on the SVM model using the tune() function in R, where we performed a 10-fold validation on our model using the same 5 hand-picked test samples. We selected the best SVM using the best.model command and tested it on our test sample. The cross-validated SVM model still gave a 100% accuracy on the test sample this time.

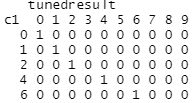


Figure 6: Resulted confusion matrix of cross-validated SVM model

Since the tune() function did not allow us to train on probability, we weren’t able to collect the probability result of our best model and it limited our ability to analyze the impact of performing a cross-validation on our model.

**R code used for cross-validation SVM:**

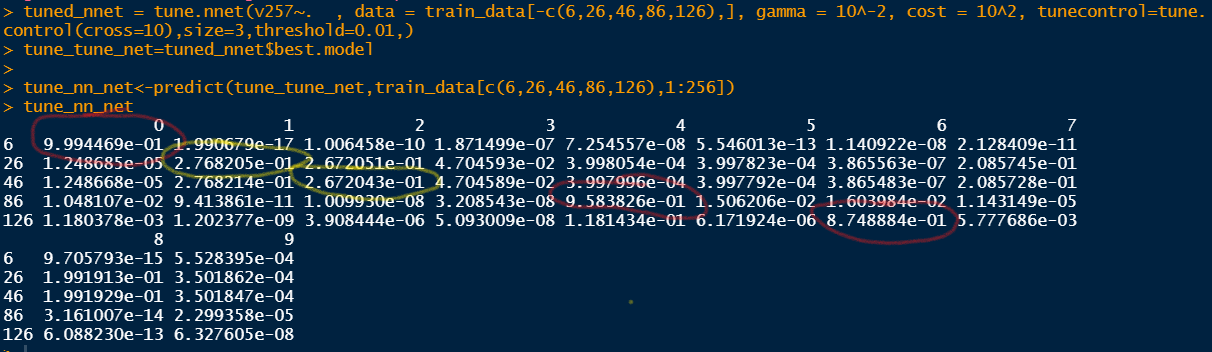
tuned = tune.svm(V257~. ,data=nn\_data[-c(6,26,46,86,126),],gamma=10^-2,cost = 10^2,tuneControl = tune.control(cross=10))

svmfit = tuned$best.model

tunedresult = predict(svmfit,original\_data[c(6,26,46,86,126),1:256])

## 5.3 Cross-Validation Neural Network

The cross-validation on neural network was performed using the tune() function in R. The cross-validation used the same setup as we did to our SVM model, and the neural network was trained and test with same data sets.

  
Figure 7: Result of cross-validated neural network

As it can be seen from Figure 7, while some digits were still predicted accurately, some received an accuracy less than 50%. The poor performance of a cross-validated neural network could be a result of overfitting.

# 6.Summary

During this project, we applied three different classification methods on the dataset. For the most part, we were able to produce the desired result at the end but we still encounter some difficulties along the way. Some of our classification methods like SVM and Neural Networks can also work as regression models, we made some early mistakes and ended up getting some float value from our prediction models. We also had some difficulties plotting our result, since every method works in a different way. For example, we were able to not only get the classification result from SVM and Neural Networks but also the accuracy related to each result. Combing the result and probability offers us more information about our prediction models and helps us better understand our models. kNN, on the other hand, is often described as a ‘lazy learner’ that it does not produce a predictive model. So we have perform the kNN function every time we need it, but it generates results much faster compared to SVM and Neural Networks.

We have also randomly selected other test samples that contain digits 0,1,2,4,6. The results were close to our first test set, the hand-picked one. Therefore, we did not include those results in this report for consistency and simplicity reasons.

Comparing the three methods we used for this project, SVM gave the best result overall, kNN and neural networks worked well initially but suffered a loss in accuracy during cross-validation. Since cross-validation always comes with the possibility to over-fit the data and we have already achieved high accuracy without it, we didn’t continue to pursue in this direction.

# 7.Work Allocation

The team was initially assigned with three members. However, both Gnanadeep Chinnabathini (GC) and Yingjie Chen (YC) weren’t able to establish a consistent communication with the third group member Hao Wang (HW). Therefore, the only two contributors to this report are GC and YC. YC was in charge of the general formatting, introduction and summary of this report. YC and GC contributed equally to perform kNN, SVM and cross-validation analysis on the data. The neural network was solely prepared by GC.