

Assignment 8 - Final Project - CS545 - Machine Learning

Carlos Oliveira

December 17, 2009

Contents

1	Introduction	2
1.1	Abstract	2
1.2	Implementation	2
1.3	Boosting	3
2	Implementation and Results	4
2.1	Knowing the network model	4
2.2	Handwritten data set	4
2.3	Reduced sample nets	7
2.4	Results	9
3	Conclusions	21
4	Notes and Special Thanks	22

1 Introduction

This is the final assignment [1] of the CS545 - Machine Learning class [2] Fall 2009 section. For this assignment, I was supposed to create a project based on the areas of study learned during this class. I decided to try an approach that I proposed in my assignment 6 report to improve the predictability of handwritten digits using neural networks. Refer to the following subsections to a description of the working strategy for this assignment.

1.1 Abstract

In my assignment 6 report, I stated that neural network model on classifying handwritten digits seemed to have a preference for digits 2 and 4. My proposal is to verify whether or not another layer of predictions could improve the results. This new layer of predictions would use only the 3 digits that have the higher probability and decide among them only.

Here is the reasoning behind my suggestion:

- if the neural network finds that there is a 45% change of a digit X (considering all the other digits y have $p(y) \leq 45\%$), we can state that:

1. yes, there is a 45% probability that the digit is X
2. however, there is a 55% probability that the digit is not X, but something else.

I want to increase the likelihood of X by decreasing the searching space to the 3 most likely digits, rather than all ten and my hypothesis is that this will increase the accuracy.

1.2 Implementation

This experiment was implemented according to the steps listed below:

1. Train the neural network on the zip.data data set.
2. Test the accuracy of the model by predicting the digits on the zip.test training set.
3. Analyze the “fairness” of the model, just to make sure that there is or not a preference - this will be just a quick percentage analysis per digits

$$Accuracy = \frac{Digits\ Predicted\ Correctly}{Total\ Samples}$$

4. Write R-code to generate a data set file with handwritten digits (probably around 100 samples -10 drawing each digit) - I want a fixed set of hand written digits, so the experiment can be repeated.
5. Train a couple of more networks for the “more likely to be confused” digits (example nnet_4.8.9 would be a nnet trained to classify just the digits 4, 8 and 9). The idea here is to have a couple of these networks, such as nnet_4.8.9, nnet_2.5.6, nnet_3.8 or nnet_2.3.8. I will chose which ones I will use based on the data from my previous experiment.
6. After getting the probability of a digit X, I will also find the 2^{nd} and 3^{rd} most likely and choose what next nnet_X_Y_Z to “recalculate” the probabilities.
7. The last step would be to compare the result from nnet_X_Y_Z and from nnet with the real digit.

This will not solve problems when we have, for example $p(2)=25\%$, $p(4)=35\%$, $p(6)=15\%$, $p(8)=5\%$ when the hand written digit is 8. However, my hypothesis is that, in this example, this could help if the hand written digit is 6.

1.3 Boosting

Chapter 14 of our text book [4] talks about combining models. According to the author, “*It is often found that improved performance can be obtained by combining multiple models together in some way, instead of just using a single model in isolation.*”

The method I am using on this assignment is closely related to Boosting [5]. Boosting combines multiple ‘base’ classifiers to produce a committee that will have a better performance than any of the base classifiers would by themselves. According to the author, boosting has the base classifiers “*trained in sequence, and each base classifier is trained using a weighted form of the data set in which the weighting coefficient associated with each data point depends on the performance of the previous classifiers.*”

This is very similar to my proposal, where, after the passing the data through the first trained network, than I use a second network that was trained only for the digits that has the highest probabilities.

Digit	Test digits	Predictions
0	17.88	18.23
1	13.15	12.95
2	9.86	9.46
3	8.27	8.57
4	9.96	10.11
5	7.97	7.82
6	8.47	8.42
7	7.32	7.07
8	8.27	7.97
9	8.81	9.36

Table 1: Fairness Analysis: percentage of digits and predictions

2 Implementation and Results

2.1 Knowing the network model

I used the same approach that was done for assignment 6, reading the data files the “The Elements of Statistical Learning” website [3]. Then, with the same R-code used before, I trained the network. The network was trained with the following parameters values:

$$\begin{aligned}
 hu &= 20 \\
 \lambda &= 10^{-9} \\
 epochs &= 20,000
 \end{aligned}$$

I will report here only what was done differently from assignment 6. See on figure 1 the prediction results for the test data set.

```

Xtest <- as.matrix(zipTestdata)
Xtest <- Xtest[order(Xtest[, 1]), ]
...
mnOutTest <- useNN_LogReg_ZY(myNNnet,Xtest)

resultsTest <- mnOutTest$results - 1
...
for(i in c(1:9))
  restmp <- rbind(restmp,
    c(i,100*length(Ttest[which(Ttest==i)])/length(Ttest),
      100*length(resultsTest[which(resultsTest==i)])/length(resultsTest)))

```

The last part of this section shows us that the network seems to be fair. I used the R-code above to get the results sorted for the purpose of the fairness analysis.

On table 1 we can see that the model does not seem to have a preference for any digits as the percentage of digits predicted is close to the percentage composition of the data set. Also, it is worth to mention that the accuracy of this model using the data set was of 92.17%.

See some examples of digits drawing and their probabilities from the testing data set on figure 2.

2.2 Handwritten data set

The following R-code was used to write the digits to a file to be used later for the predictions:

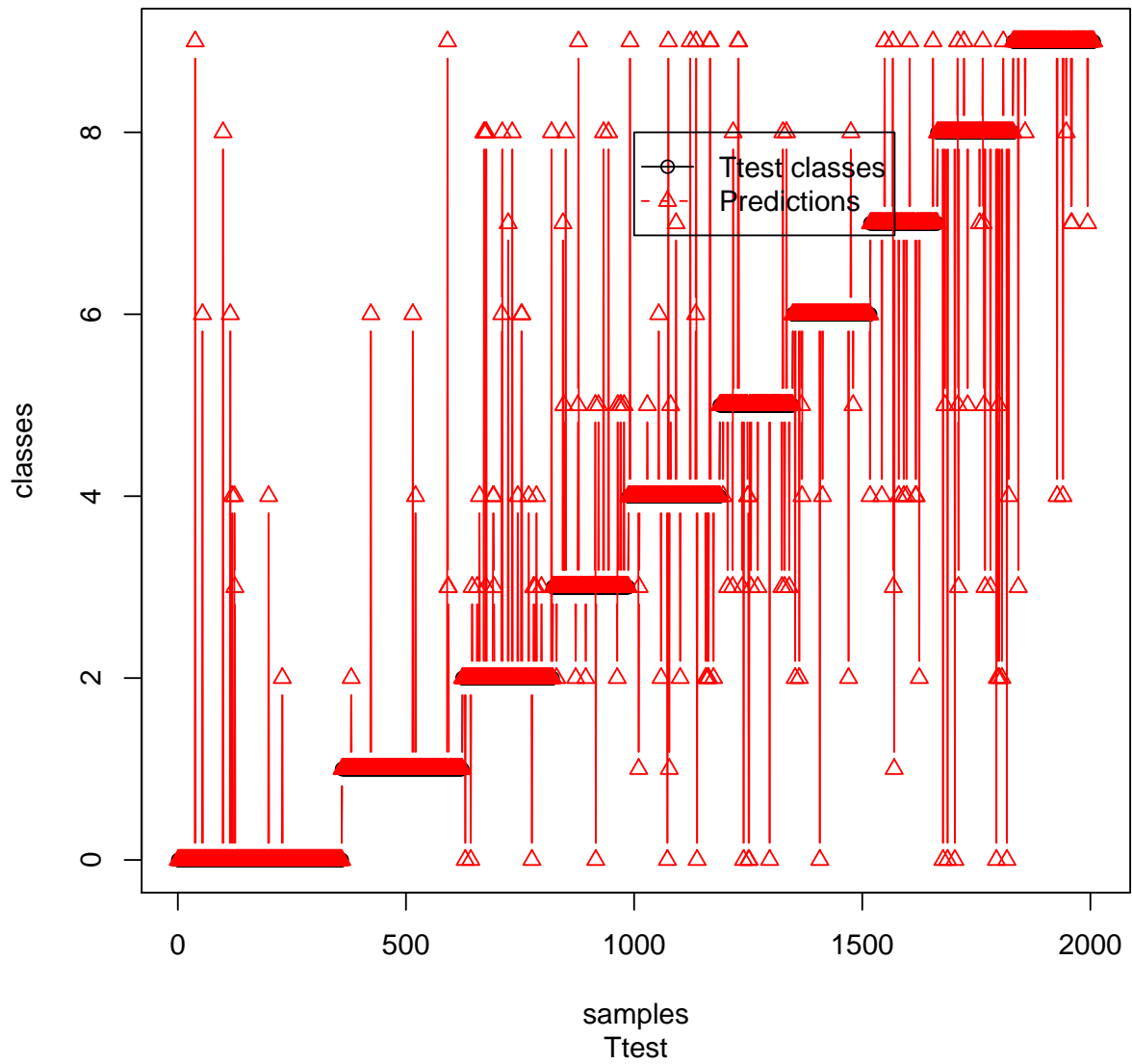


Figure 1: Predictions for the test data set

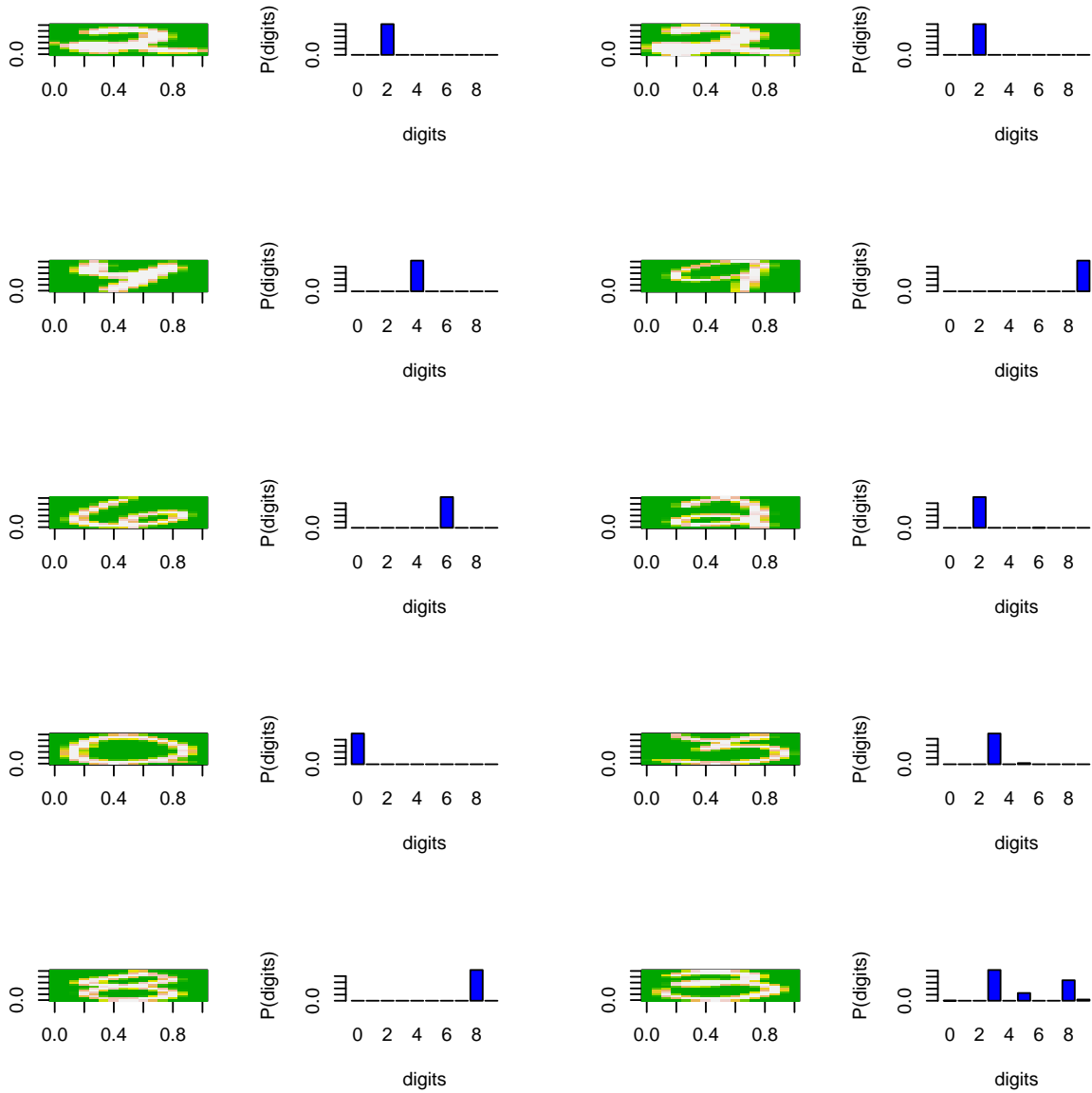


Figure 2: Digits have too few significant data points

Digits Prediction	1	2	3	4	5	6	7	8	9	0
Digit 1	3	4	–	3	–	–	–	–	–	–
Digit 2	–	7	–	3	–	–	–	–	–	–
Digit 3	–	–	1	6	–	–	2	–	1	–
Digit 4	–	–	–	10	–	–	–	–	–	–
Digit 5	–	2	–	4	4	–	–	–	–	–
Digit 6	–	3	–	6	–	1	–	–	–	–
Digit 7	–	–	–	1	–	–	9	–	–	–
Digit 8	–	1	–	8	–	–	1	–	–	–
Digit 9	–	1	–	7	–	–	1	–	1	–
Digit 0	–	3	–	4	–	1	2	–	–	–
Total Predictions	3	21	1	52	4	2	15	–	2	–

Table 2: Frequency table, hits on predictions

```
entryDigitInTable <- function(filename, digit) {
  dev.set(which=dev.next())
  x <- t(matrix(drawImage()))
  x <- cbind(x,digit)
  print(x)
  write.csv(x,filename,append=TRUE)
  dev.set(which=dev.next())
}
```

On my first attempt, the digits that were generated were too thin and not centered and the predictions were a disaster. See an example on figure 3.

2.3 Reduced sample nets

The table 2 is from assignment 6 and shows us how some digits were not being identified correctly. For example, notice that digit 1 was classified as 1, 2 and 4; while digit 8 was classified as 1, 4 and 7.

This could be the base to choose what new neural networks could be created to address these problems.

I have chosen to create new networks for the following groups of values: { 1 2 4}, { 1 2 7}, { 0 6 8}, { 5 6 8 }, { 3 8 9}, { 4 7 9}, { 1 2 4}, { 2 4 8}, { 3 4 7}, { 2 4 5}, { 2 4 7} and { 2 4 9}.

See the R-code below for an example on how I selected the data and created the network:

```
# 1 2 7
Xtrain127 <- zipcodedata[myDigits==1,]
Xtrain127 <- rbind(Xtrain127,zipcodedata[myDigits==2,])
Xtrain127 <- rbind(Xtrain127,zipcodedata[myDigits==7,])

Ttrain127 <- as.matrix(Xtrain127[,1])
Xtrain127 <- Xtrain127[,-1]

myNNet127 <- makeNN_LogReg(Xtrain127,Ttrain127,numHiddenUnits=hu,
                           nIterations=epochs,lambda=lambda,ftracep=FALSE)

# 0 6 8
Xtrain068 <- zipcodedata[myDigits==0,]
Xtrain068 <- rbind(Xtrain068,zipcodedata[myDigits==6,])
Xtrain068 <- rbind(Xtrain068,zipcodedata[myDigits==8,])

Ttrain068 <- as.matrix(Xtrain068[,1])
Xtrain068 <- Xtrain068[,-1]
```

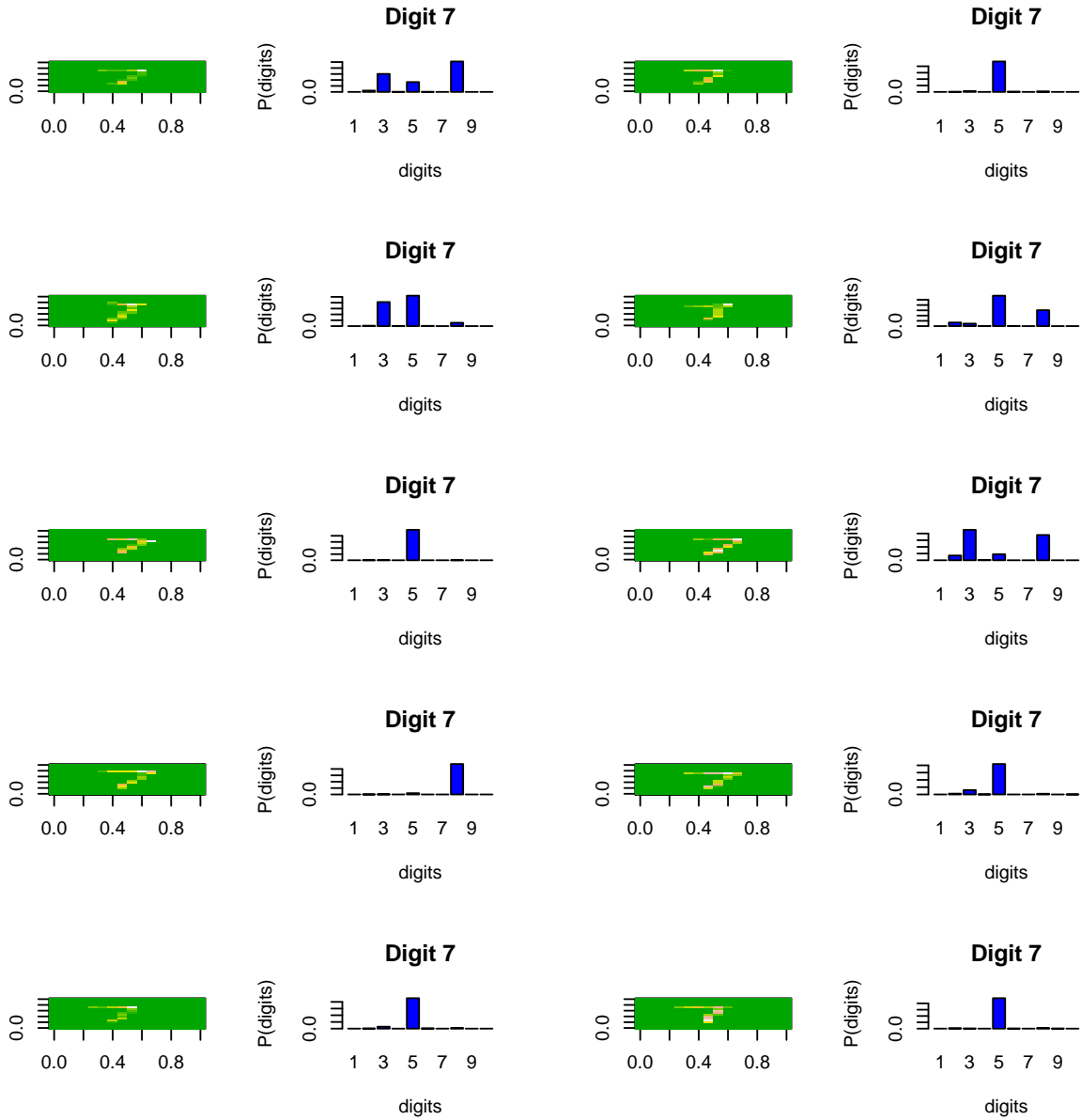


Figure 3: Digits have too few significant data points

```
myNNnet068 <- makeNN_LogReg(Xtrain068,Ttrain068,numHiddenUnits=hu,
                           nIterations=epochs,lambda=lambda,ftracep=FALSE)
```

Then after finding the 3 highest probabilities for each prediction, I selected which second network to used based on 2 out of the 3 digits. The R-code below is extracting the 3 highest probabilities and storing them on prob3highest.

```
subSetXtrain <- matrix(Xtrain[i,],nrow=1)
...
nnOutTrain <- useNN_LogRegProbDig(myNNnet,subSetXtrain)
tmpProbs <- nnOutTrain$probabilities

prob3highest <- c()

# picking up the best 3 candidates
for (i_counter in c(1,2,3))
{
  prob3highest <- c(prob3highest,which(tmpProbs==max(tmpProbs))-1)
  tmpProbs[which(tmpProbs==max(tmpProbs))] = 0
}
```

Depending on the digits that are in the prob3highest variable I will pick one of my pre-trained networks if the network has at least 2 digits that are in prob3highest.

```
if(sum(as.numeric(c(1,2,4) \%in\% prob3highest))>1)
{
  nnOutTrain <- useNN_LogRegProbDig(myNNnet124,subSetXtrain)
  correction <- paste("Pred",nnOutTrain$retDig-1,"using 1,2,4.")
  cat(correction,"\n")
}
if(sum(as.numeric(c(1,2,7) \%in\% prob3highest))>1)
{
  nnOutTrain <- useNN_LogRegProbDig(myNNnet127,subSetXtrain)
  correction <- paste("Pred",nnOutTrain$retDig-1,"using 1,2,7.")
  cat(correction,"\n")
}
```

The final step here will be to compare the new nnOutTrain\$probabilities with the previous calculated one.

2.4 Results

I have around 3 to 4 pages of figures per digit to add to this report. However I am adding just one page of figures per digit as the information on the other pages would not add value to the conclusions. Besides, this is an ecologically correct report.

To understand figure 4, the first column has an image of the digit to be classified; the second column has the probabilities barplot using the general¹ network and the third column shows the probabilities using the retrained network and also what set of numbers was used to define which network to apply. For example, on the top of the 3 rows, notice that the subnet for {2,4,5} is used as the original predictions have 2 and 5 among the 3 highest.

On figure 5 we see that there is not improvement on the predictions.

When predicting the digits 2, we can see on figure 6 that the network using {2,4,7} helped to fix the prediction of the first row.

¹By general I mean the network originally trained with all digits

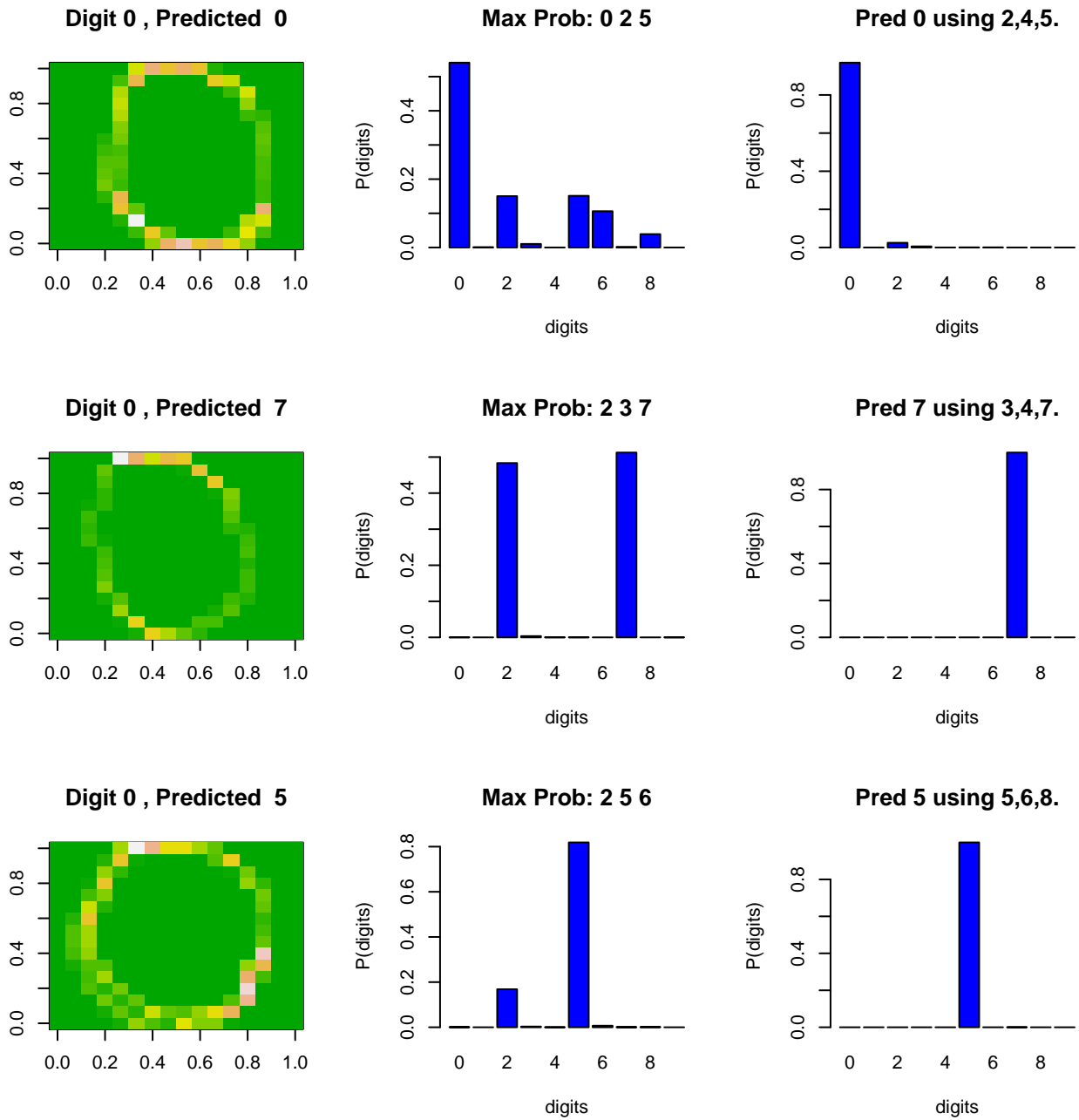


Figure 4: Digit 0: Retrained nets confirm the mistake

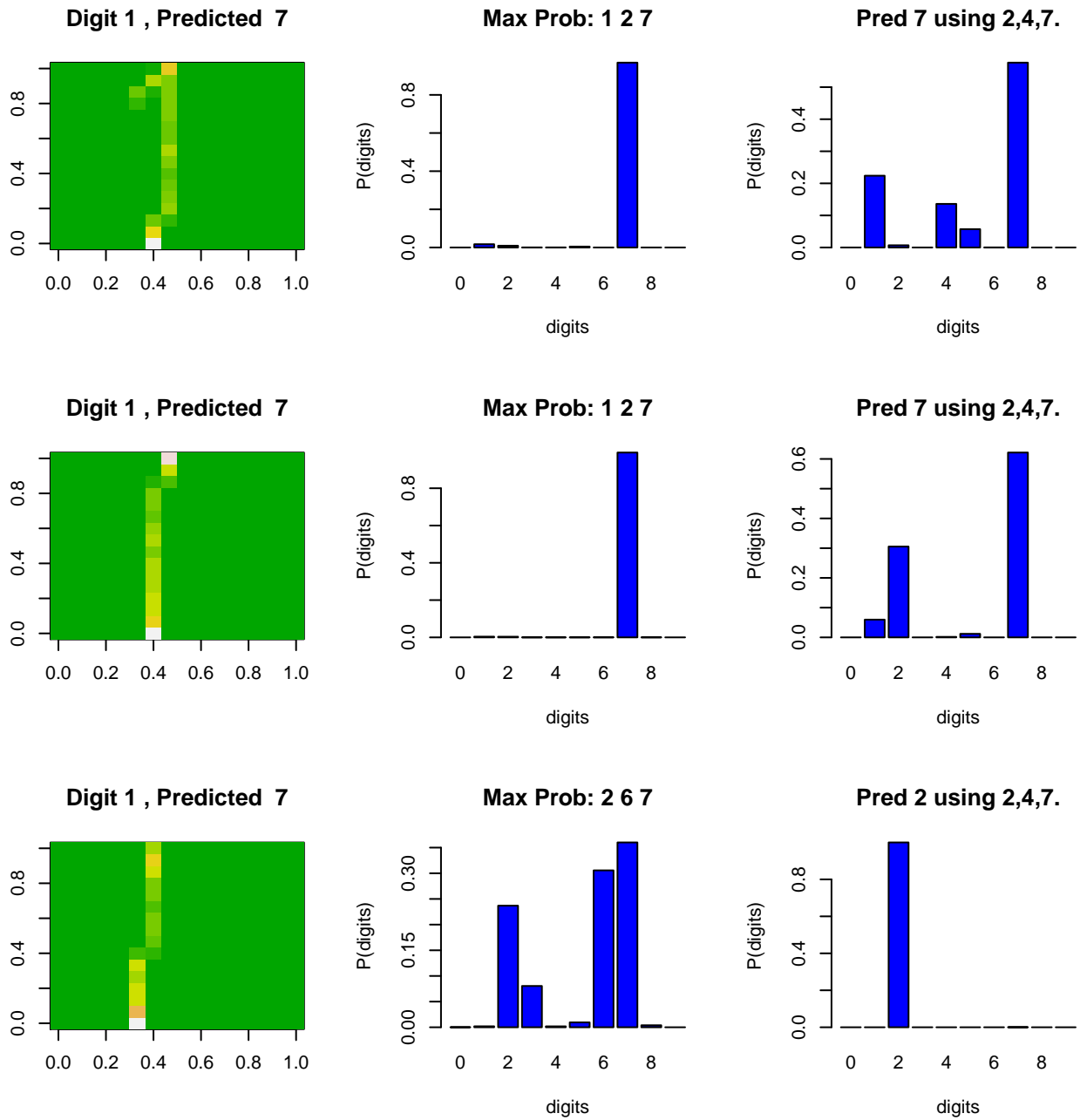


Figure 5: Digit 1: Retrained network decides on different values, but still wrong.

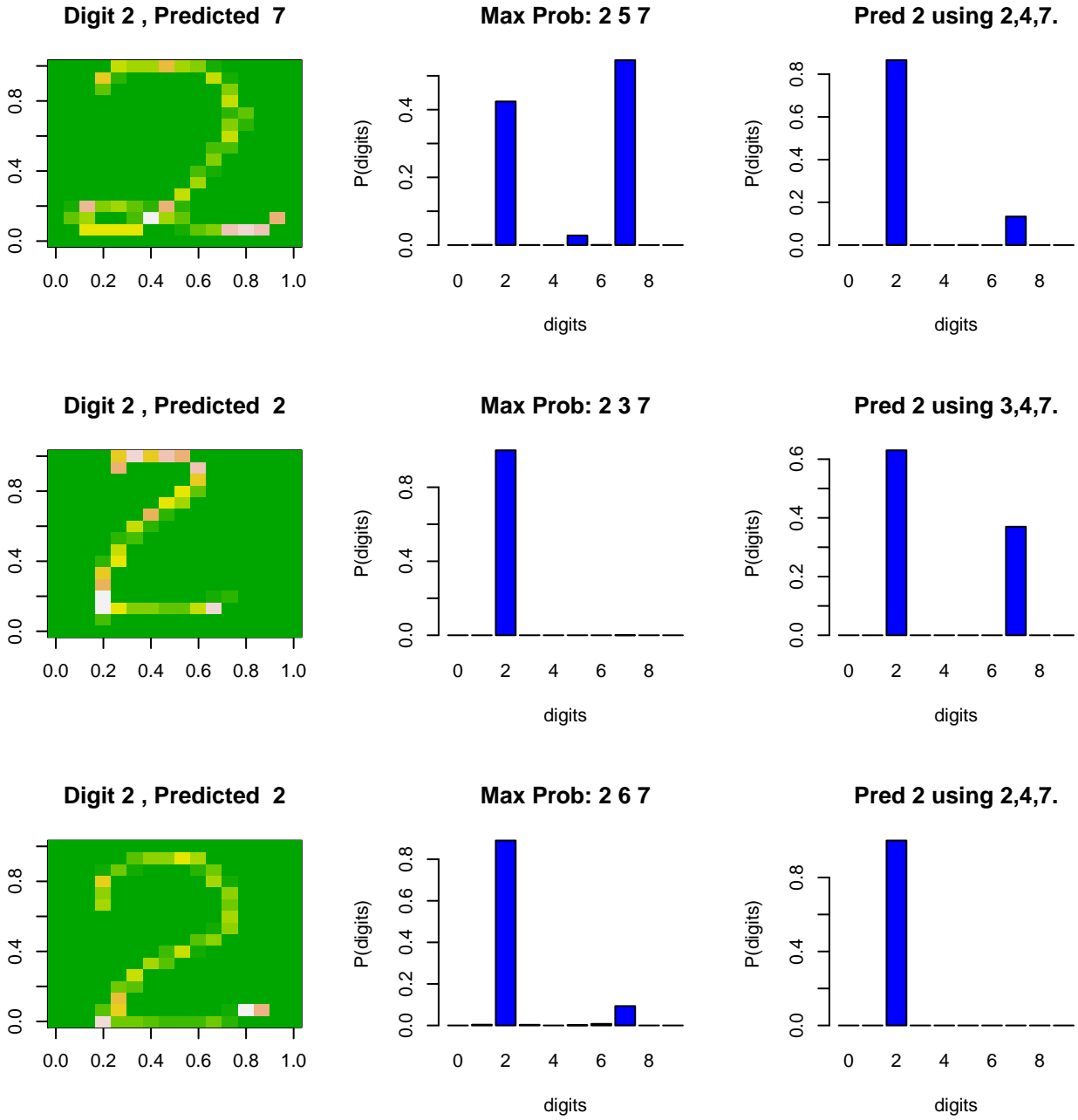


Figure 6: Digit 2: Ah-ha! Got all right this time.

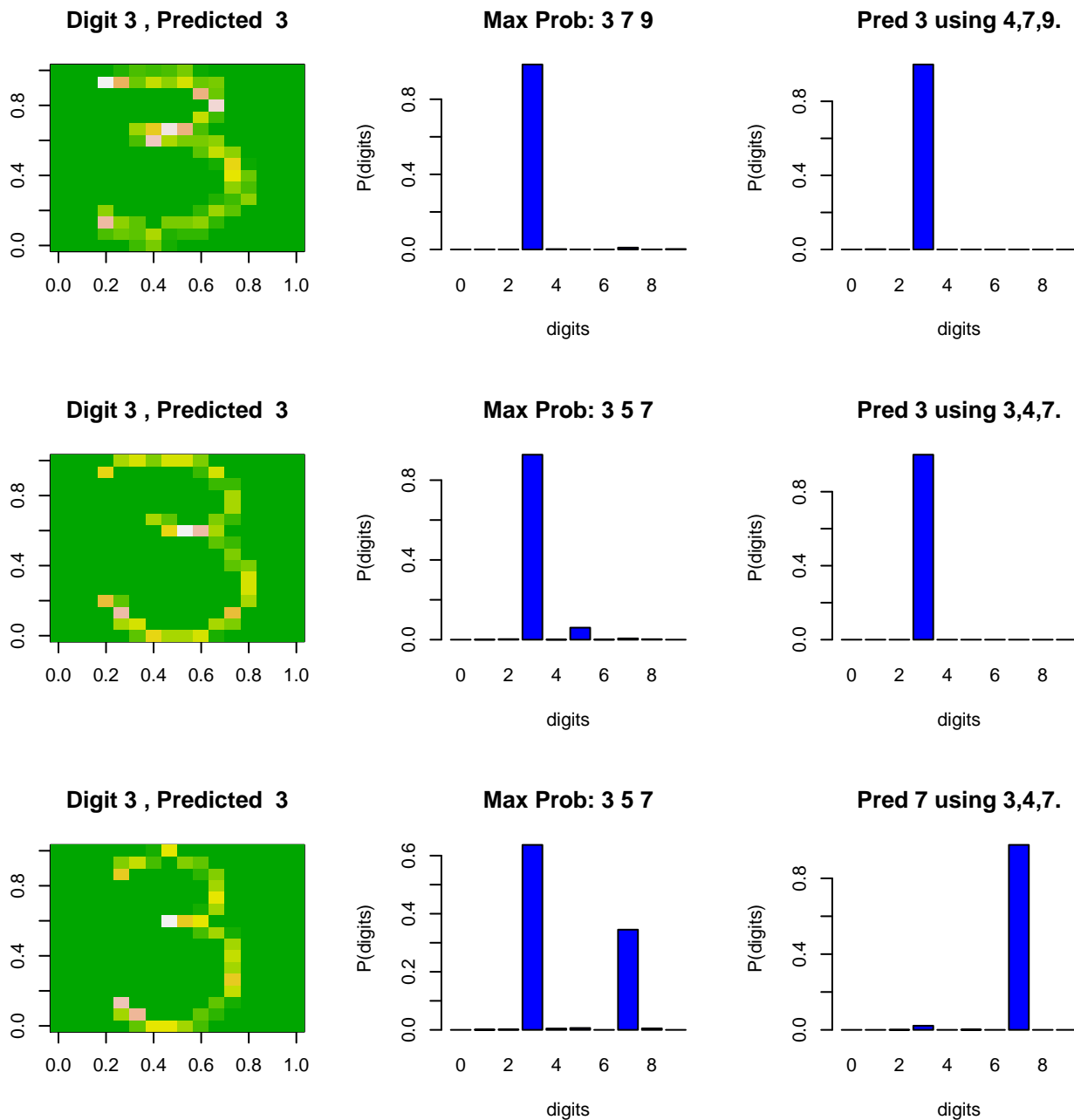


Figure 7: Digit 3: It was too good to be true.

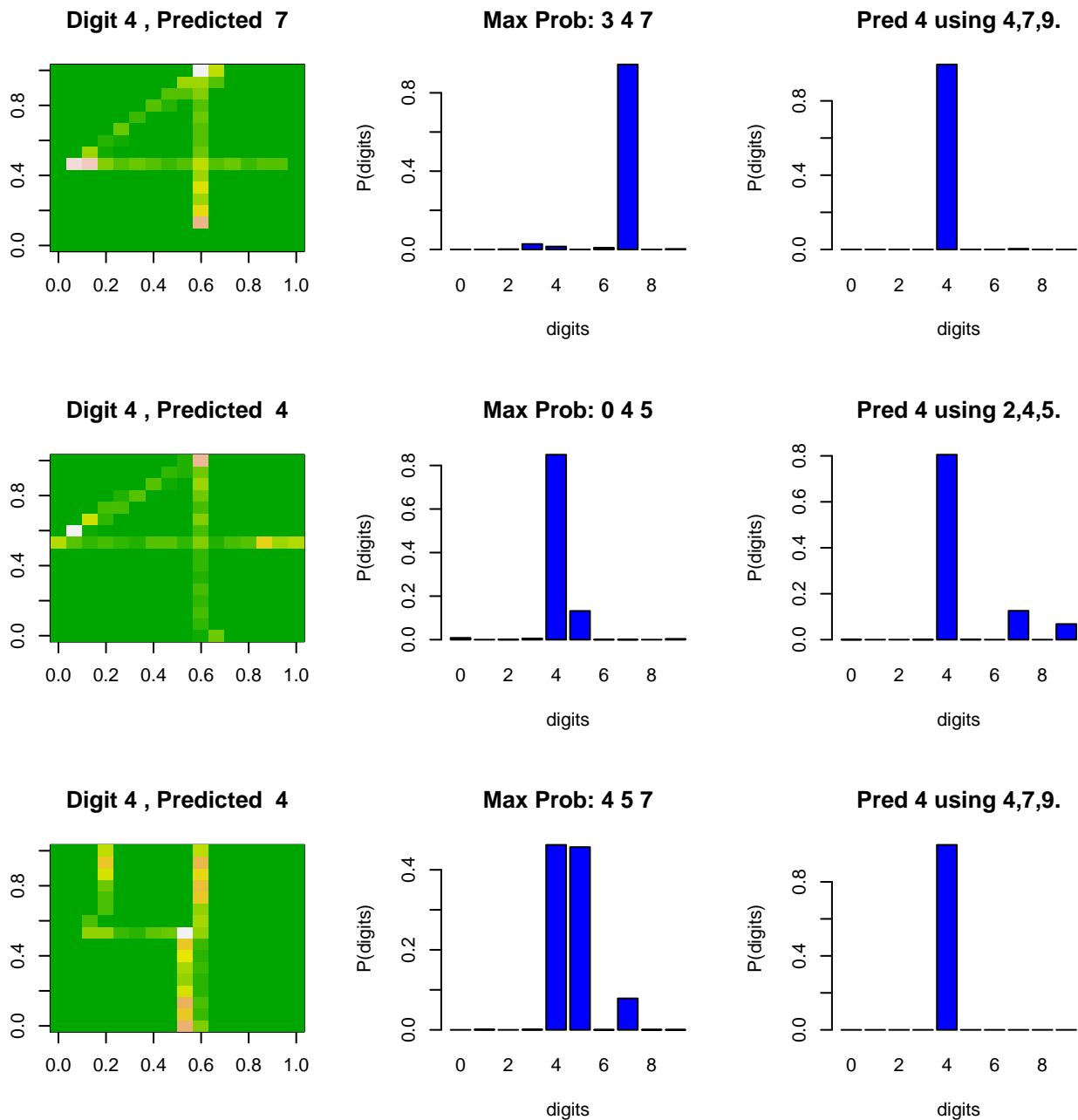


Figure 8: Digit 4: Score again.

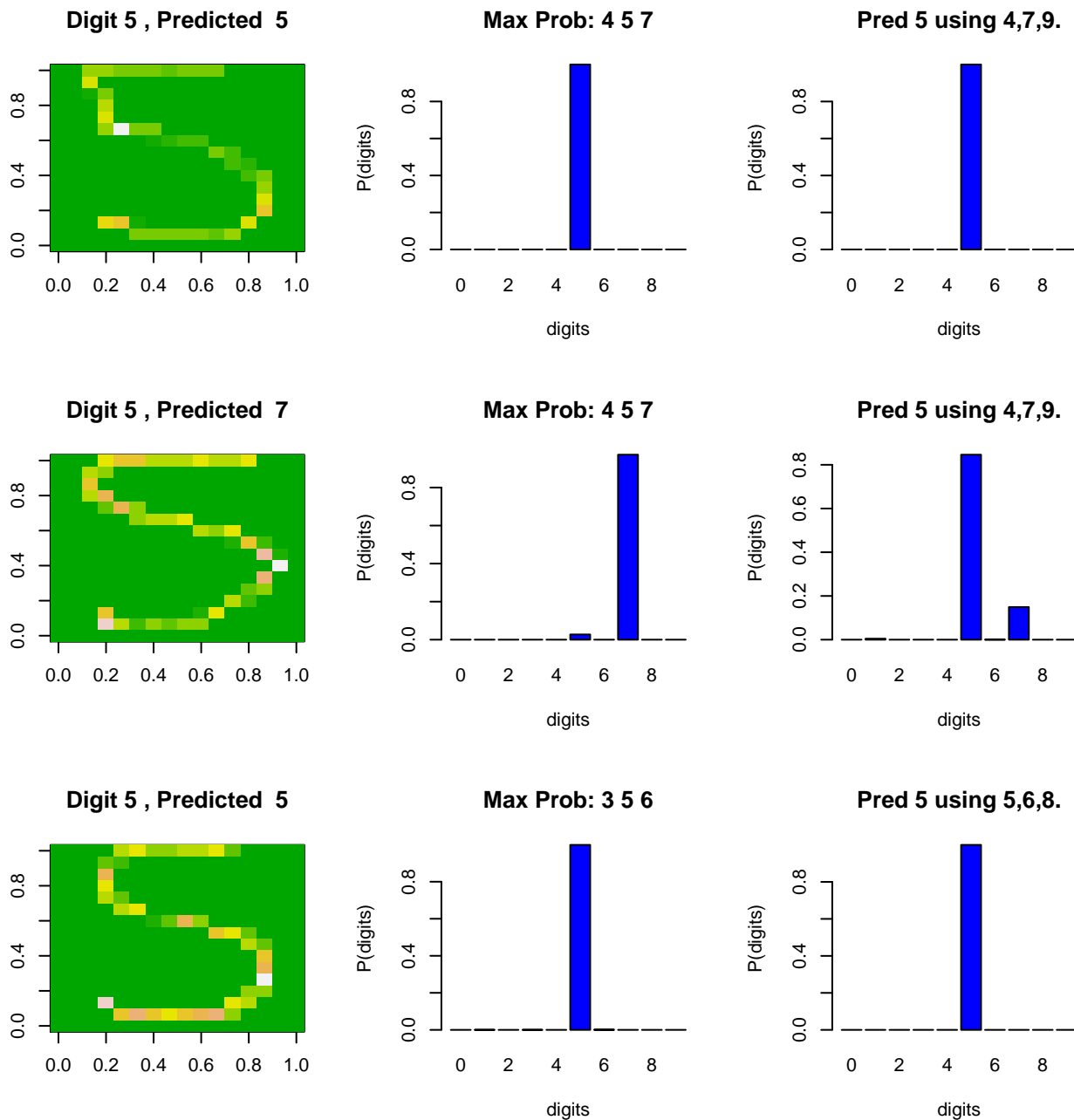


Figure 9: Digit 5: ... and again.

We are making some progress here. Figures 2.4 and 2.4 show that the retrained nets are helping to boost the predictions. On the examples on these figures we got 2 mistakes fixed.

Figures 2.4, 2.4 and 2.4 show digits that were hard to predict and the retrained networks did not help much. Figure 2.4, although is an easy one to predict, did not show the retrained networks doing a good job either.

The overall performance of the networks before and after was approximately 42% accuracy when not using the second layer of predictors and 35% when using it. I still believe that I could have better results but I am not sure if the cost of this approach would be too high in terms of what to implement. I will address this on the conclusion section.

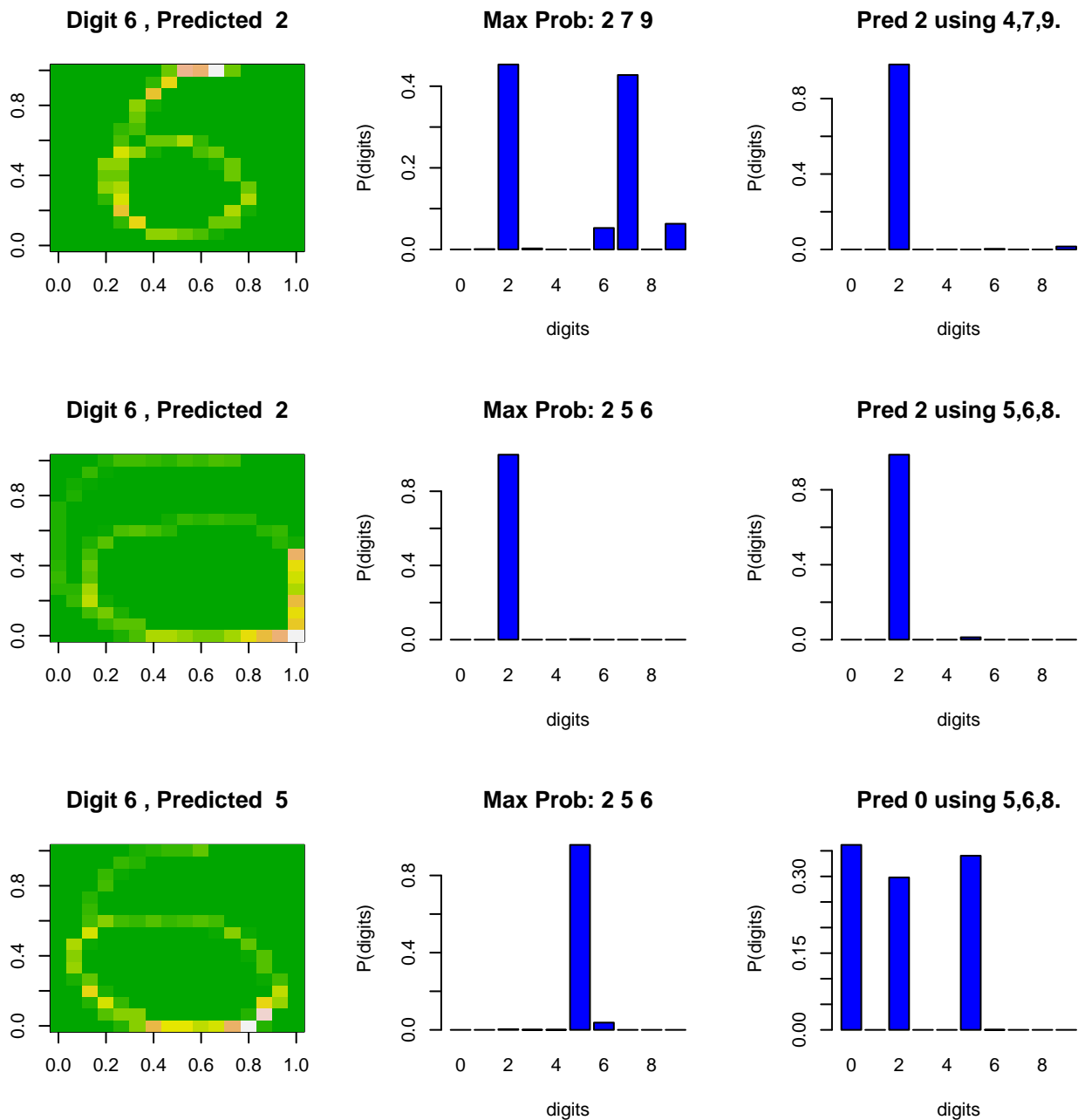


Figure 10: Digit 6: Hard to predict.

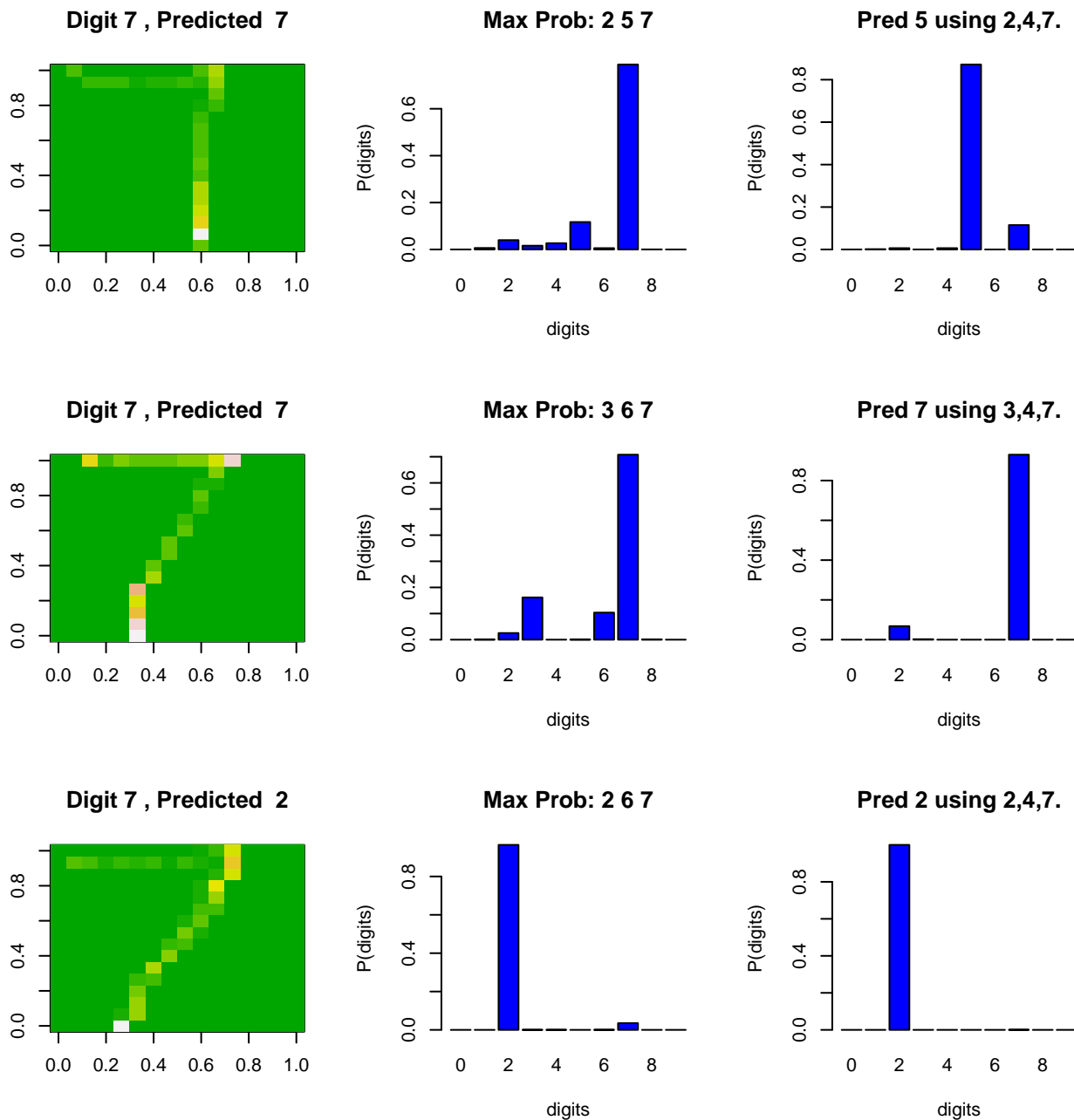


Figure 11: Digit 7: Not hard to predict but retrained networks are not doing well.

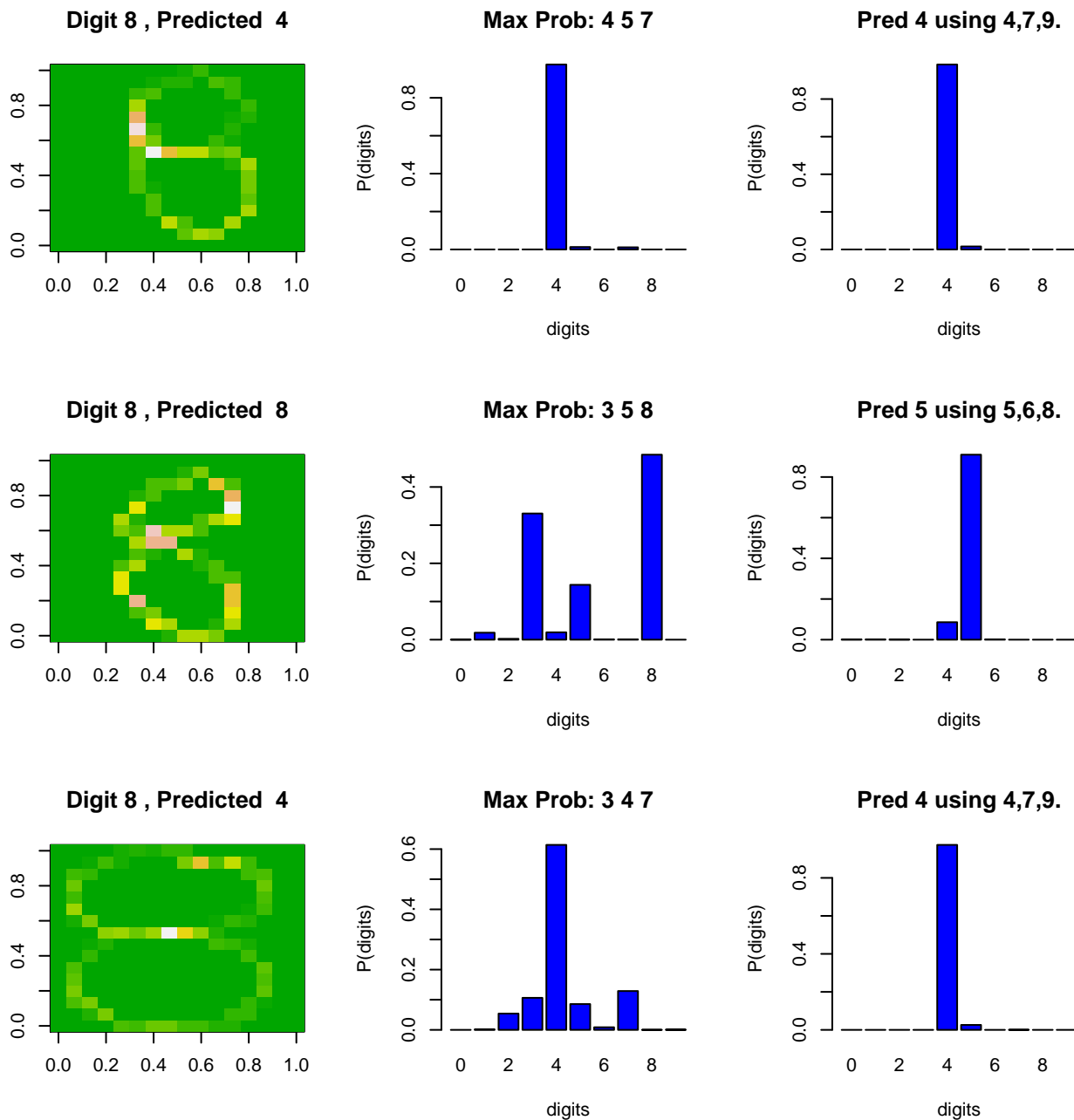


Figure 12: Digit 8: Another one hard to predict.

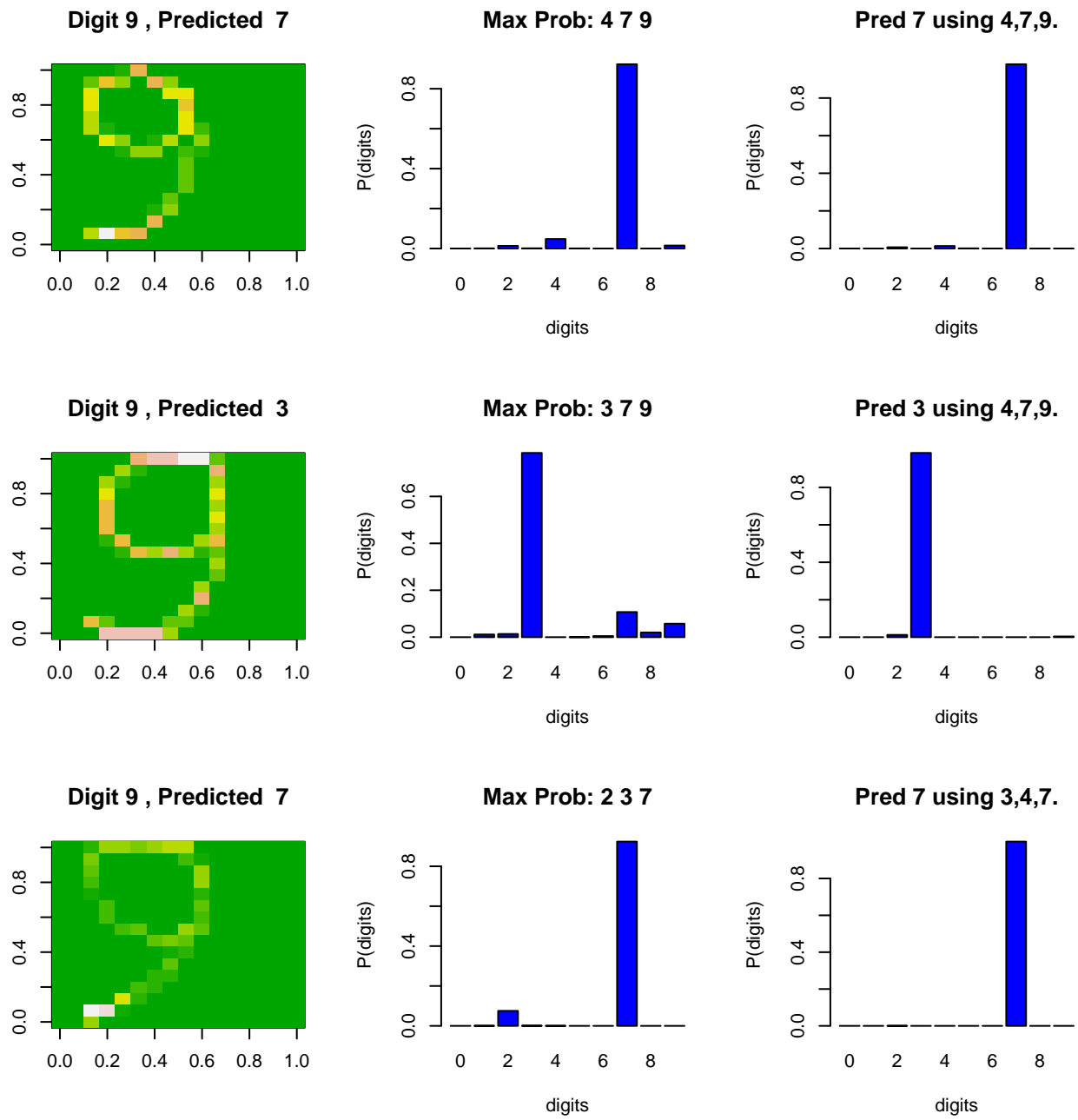


Figure 13: Digit 9: And yet another hard one.

3 Conclusions

Although the results were not what I was expecting, I have some theory on why that could be and what to do to fix it.

1. The second layer nets were approximations of the most probable digits choices. I mean, if the prediction had the highest probabilities for 1,2,5; my code was picking the network trained for 1,2 and 4 digits.
2. It is possible that we need to tune the code further to eliminate the other digits completely. For example the last row on figure 2.4 shows the highest retrained possibilities for 0, 2 and 5, although the network that was used was trained only for 5, 6 and 8. We need some work here to fix the returning values.
3. We would either have to create networks for all combinations of 3 digits $\{(1,2,3),(1,2,4), \dots, (7,8,9)\}$, which I believe is prohibitive, or we would rather big statistical sample to make sure what digits are more likely to be mistaken by others; e.g., digits 2 and 7 are good candidates as well as digits 6 and 8.
4. The handwritten digits are still not normalized. This impacts the performance of the predictions no matter how many layers we are using here.

4 Notes and Special Thanks

- Dr. Anderson, Jason and wiki users and colleagues: Thanks! Machine learning is really fun! (A lot of work, but fun.)

References

- [1] Anderson, Charles, *Nonlinear Regression with Neural Networks*, <http://www.cs.colostate.edu/~anderson/cs545/assignments/assignment8.html>, Fall 2009.
- [2] Computer Science Department, *CS545 - Machine Learning*, <http://www.cs.colostate.edu/~anderson/cs545/overview.html>, Fall 2009.
- [3] Hastie, Tibshirani and Friedman, *The Elements of Statistical Learning*, <http://www-stat.stanford.edu/~tibs/ElemStatLearn/index.html>, <http://www-stat.stanford.edu/~tibs/ElemStatLearn/index.html>
- [4] Bishop, Christopher M. “Combining Models” *Pattern Recognition and Machine Learning*, pp. 653, 2007.
- [5] Bishop, Christopher M. “Boosting” *Pattern Recognition and Machine Learning*, pp. 657, 2007.