ECE448 - Machine Problem 3 : Naive Bayes Classification

Shalabi, Yasser
yshalab2@illinois.edu

Luo, Chongxin
cluo5@illinois.edu

Wang, Haoran
hwang293@illinois.edu

November 2016

Contents

Abstract

Contents:

1 General MP Introduction and Overview

This MP explored the application of Bayes Theorem in the domain of pattern recognition. In part one, the objective recognizing handwritten digits based on the pixel patterns of their images. In part two, the focus was topic and sentiment classification based on features highlighting the words in the document.

The objective, for both of these problems, is the construction of models capable distinguishing the feature patterns between the different classes. For example, in part one the objective is distinguish the ten digits 0-9. This means we need to learn the distinguishing pixel patterns for the different classes.

Consider the problem of handwritten digit recognition. There are 10 possible numbers. It is reasonable to assume that images of the digits appear with equal probability $p = \frac{1}{10}$. We call this probability the *prior* probability – because it describes the probability of the event without (or before) any other evidence is considered. If we had to decide which digit was written on a piece of paper that we couldn't see a *decision rule* that always predicts one of the ten digits will correctly classify the images with a 10 percent of the time. Since we know that each digit has a visibly distinct pattern to it – we know that we can design a decision rule by learning a model based on the visual features of the hand-written digits.

This is where Bayes Theorem comes in. Bayes theorem states the following.

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

This theorem provides a means for *updating the belief* of class A in the light of evidence from class B. By incorporating how likely B is to occur with A – $\frac{P(B|A)}{P(B)}$ – we can convert the prior probability of A – $P(A)$ – into a posterior probability $P(A \mid B)$. Thus, the informal english equivalent for Bayes formula is:

$$posterior = \frac{likelihood \times prior}{evidence}$$

This is how we incorporate the knowledge embodied in the image itself into our probability scores. To do this, we need to learn the *class conditional probability* functions $P(B \mid A)$. Continuing our running example, B would be a feature from the image and A would be the digit itself. This probability reflects the likelihood of the patterns manifesting in images of specific handwritten digits. For example we would expect $P(Oval \mid Zero)$ to be higher than $P(Line \mid Zero)$. Similarly, for the number one we expect $P(Line \mid One) > P(Oval \mid One)$ This implies that if we see an image with a circular pattern, it is more likely to be a zero than a zero.

1.1 Bayesian Decision Theory

Now the question is, how do we use these likelihoods and posteriors to design a classifier? Previously, we stated that a blind classifier can be constructed by simply choosing the class with the maximum prior. The expected loss from that classifier is $1 - P(A)$ – or the probability that the sample belongs to a class other than A. Thus, to minimize the error we chose the class A with the highest probability $P(A)$.

To minimize the error when using posterior probabilities we follow the same line of reasoning. Consider posterior probability for variable X – $P(X = x \mid E)$. If we define the true value of X is x^* then we can define a unit loss function $L(x, x^*)$ which is 0 anytime $x = x^*$ and 0 otherwise. Thus the expected loss from predicting x when observing E is:

$$\sum_{x^*} L(x, x^*) \times P(X = x^* \mid E)$$

Thus to minimize the error, it is easy to see we need to pick x with the highest posterior probability $P(X = x \mid E)$ because otherwise it would contribute the largest value to the sum of errors.

This is known as the Maximum Aposteriori or MAP decision rule. The MAP decision $\hat{x}$ is:

$\hat{x} = \text{argmax}_x \text{ P(X=x} \mid E) = \frac{P(E|X=x)\, P(X=x)}{P(E)} \propto P(E \mid X = x)\, P(X = x)$

1.2 Naive Bayes

As the formula for $\hat{x}$ implies, to find the error minimizing classification we need to choose the hypothesis (i.e. classification) which maximizes the likelihoods of the observed evidence variables.

When computing the posterior probabilities, it is often the case that we have multiple pieces of evidence to consider. Thus, computing $P(C|E_1, E_2, ..., E_n)$ necessitates knowing the joint distribution of n variables – each taking one of d values. This table quickly gets large – requiring us to both store and estimate D^n probabilities.

Thus, we (must) make the "naive" assumption which is that given the hypothesis the evidence variables are independent. This greatly simplifies our computation, allowing us to compute the product of the likelihoods when computing the posteriors. Naive Bayes conditional independence assumption makes the computation of this probability tractable. Thus, to compute the likelihood of the evidence variables for the given the hypothesis digit we compute:

$$P(p_1, ..., p_n|D_j) = \prod_{i=1}^{n} P(p_i = v_i|D_j)$$

This also means that we only need to estimate the probabilities $P(p_i = v_i|D_j)$. To do this we simply count the proportion of the training samples from class D_j with pixel $p_i = v_i$. Thus, this means we can consider each frequency of $p_i = v_i$ from the training data independently. This is how we estimate the pixel estimates.

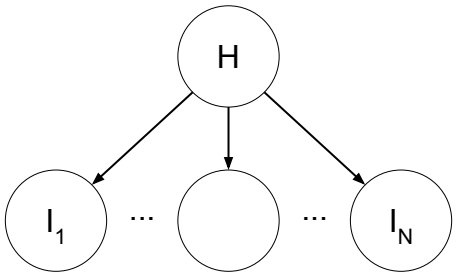


Figure 1: Modeling With Naive Bayes

This allows us to build models of the form shown in Figure 1. As shown, the information variables are completely independent of each other.

1.3 Laplace Smoothing

In general, classifiers learnt from observed data have a problem handling anomalies (and in general, infrequently ocurring events). This is because – by definition such cases will rarely occur and thus may not occur in the data-set from which the model parameters are estimated. The adage "all models are wrong, some are useful" comes to mind here. Without explicitly accounting for the possibility that the data is missing anomalies the model parameters might be defined such that these anomalies are considered impossible rather than unlikely. This is especially critical in order to ensure that the trained model generalizes well to new data.

Lets continue to focus on the the handwriting recognition problem. For the digit class D_j it could be that no images from the training sets have pixel p_i set. To say that the likelihood of pixel p_i for a digit D_j is zero would imply that p_i is never expected to be set if the digit is D_j. Thus, if p_i is set a model based on a zero likelihood for $P(p_i = 1|D_j)$ would compute a zero probability score for the posterior $P(D_j|P_i = 1)$. By the conditional independence assumption of naive bayes, no other pixel can help us recover from this.

However unlikely for this pixel to be set, anyone can make a mistake while writing a number. Thus it makes more sense to "smooth the likelihood" by initializing it to some small value.

In Section 2.1.2 we study the impact of the K factor in depth for extra credit. There we discuss how we determined the optimal value and also why some smoothing factors are better than others.

1.3.1 Smoothing Process

As alluded to earlier we estimate the likelihoods for p_i by counting the instances of $p_i = v_i$ for each possible values v_i and normalized by the number of possible pixel values for this class. We smooth the likelihoods by initializing the instance counts for $p_i = v_i$ to a smoothing factor k. The total class instance values is then increased by k*V – where V is the total number of unique values possible. Although we describe this in terms of the digit classification problem from part 1 – we use the same approach for part 2.

To calculate the optimal k we build models using different smoothing factors and test them on a subset of the training data reserved to score the quality of the smoothing factor. With evidence variables which take binary states, then to calculate the likelihood of each pixel it takes $2 \times N \times T$ operations where N is the number of classes and T is the training size. Each operation is a sum of the number of pixels with $P_i = v$ for class C.

2 Part 1: Digit Classification

2.1 Implementation Details and Discussion

2.1.1 Implementation Details

We implement our Naive Bayes training and classification code in python. We leverage NumPy, Scipy, and Pandas libraries to allow us to represent the training and classification in terms of matrix-like data structures. The following is the steps of our code: **V** : number of possible values for each pixel group (for no grouping this is 2 or 3)
N : Number of test samples **L** : Number of pixel groups, each representing one feature LL_v : $10 \times L$ sized matrix containing loglikelihoods $log(P(F_i = v|C)$ for each class (0-9) and for each feature $(F_0, F_1, ..., F_L)$. T : $N \times L$ matrix containing N samples and its L pixel group values T_v : $N \times L$ matrix of 1's and 0's. A one in $T_v(i, j)$ means for sample i pixel $F_j = v$, and a 0 means $F_j \neq v$.

1. Read and convert each training image into a vector of pixel values (binary or ternary)
2. Convert vectors into grouped equivalents according to grouping configuration
3. For each digit d, for each feature F_i, for each value v, count number of training images with $F_i = v$
4. For each class, sum these counts across all features to compute per class feature total in training set
5. For each value v, use this data to build the log-likelihood matrices LL_v.
6. Perform inference by taking the sum of the product of LL_v and T_v for each possible value v. This creates matrix $Score$ with size $N \times 10$. $Score(i, 8)$ represents the likelihood that sample i is the digit 8.
7. The Maximum-Likelihood decision is the column with the largest score
8. The Maximum a posterior decision can be computed by adding to $Score$ the vector $< Prior(0), ..., Prior(9) >^T$ and then finding the column with the largest score.

2.1.2 Extra Credit: Understanding Smoothing Factor Impact and Choosing the Best One

The first thing to understand is what is better, larger or smaller smoothing factors? With large feature sizes, the computed likelihoods will already be small. This makes it to where large smoothing factors can drown out likelihoods from our model. For example consider the digit 0 and the likelihoods shown in Figure 2 and Figure 3. We use smoothing factors .004 and .38 (respectively) to generate those plots. As we can see, the large smoothing factor downs out the probabilities. This effectively over-fits our model – making it generalize poorly.

To study the performance of different K factors we take the following approach.

1. Take training set and split it, reserving a randomly selected subset of the images to evaluate the performance of the smoothing factor

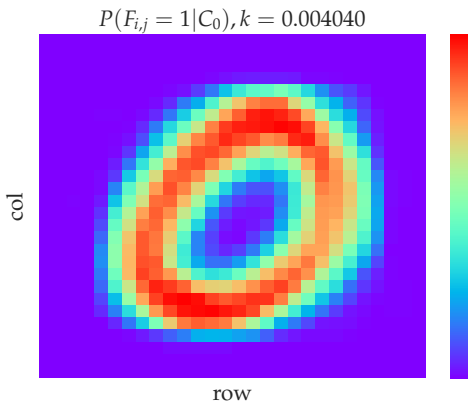


Figure 2: Impact of Small K Values on Likelihoods for 0

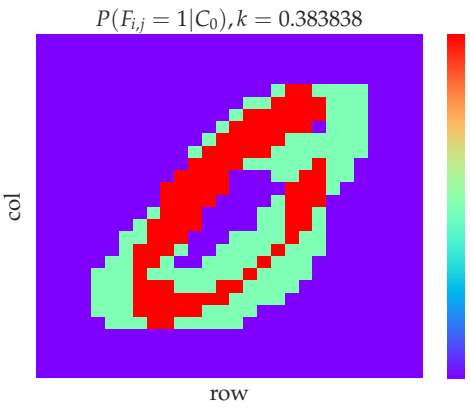


Figure 3: Impact of Large K Values on Likelihoods for 0

2. Try 100 different smoothing factors between 0 and 1 for different grouping sizes

3. Pick factor K with the highest score on the reserved training set

We see the results of those experiments in Figures 1-2. Figure 1 shows the need for smoothing, as without smoothing we can only achieve 65% accuracy. It also shows how, in general, performance is hurt as K gets bigger. Figure 2 shows the per digit smoothing impact. Some digits are not adversely impacted as K is increased – for example C_3 or C_6. But generally, the others are decreasing.

We do not run this methods for picking K for any configuration with more than 2^8 possible feature values (for example 4x4 has 2^{16} possible values for each pixel. Thats because it takes far too long to compute the impact of the smoothing. For these configurations.

Table 1 shows some of the smoothing factor. Rows with **(O)** indicating a grouped configuration. This table was generated as part of our extra-credit effort to find the best K value. Another extra credit was to try to understand the connection between the size of the training data and the performance of the model. We will discuss that in later sections.

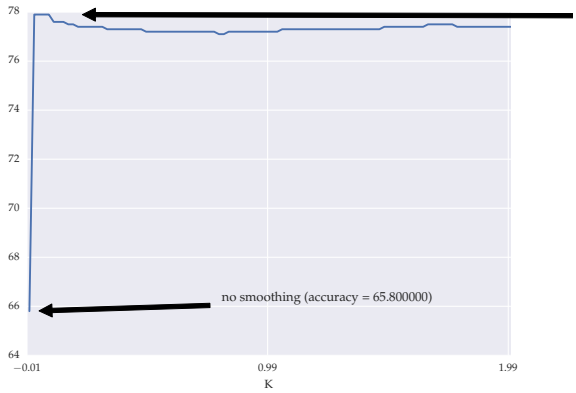


Figure 4: Impact of Smoothing factor

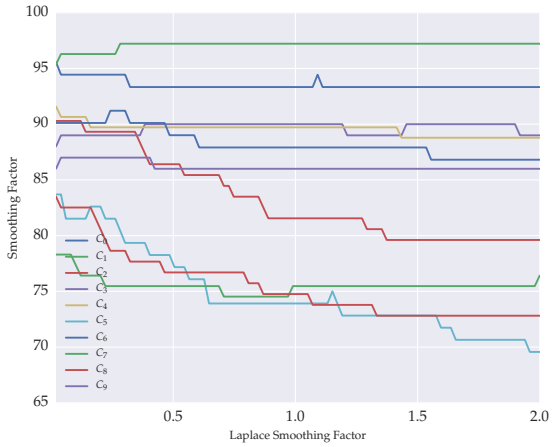


Figure 5: Per Digit Smoothing Factor Performance (2x2 Overlap)

Table 1: Smoothing Factor vs Grouping Mode and Training Size

	500	**1000**	**2000**	**3000**	**5000**
1x1	0.020202	0.028283	0.0323	0.09697	0.028283
2x2	0.00404	0.012121	0.044444	0.00404	0.00404
2x2 (O)	0.0404	0.00404	0.00404	0.00404	0.00404
3x2 (O)	0.00404	0.00404	0.004	0.004	0.004
4x2	0.028283	0.080808	0.028283	0.00404	0.008081
4x2 (O)	0.008081	0.00404	0.00404	0.00404	0.00404

2.2 Three Unit Portion

2.2.1 3Unit Results (Summary)

The detailed results are in Appendix-A.1.

In Figure 6 we show a graph visualizing the quality of a MAP based, Naive Bayes classifier. Each bar adds up to one hundred percent. On the x axis we have the true class, and the bars show the breakdown of the MAP predictions. The overall accuracy is **77.4%**. As visually evident (numerically presented in Appendix A.1.1) the classes with the lowest accuracies are 5 and 8 (with 66.3% and 60.19% accuracy respectively). As the graph and the table in Appendix A.1.1 shows, 8 was frequently confused with 3 and 9 and 5 was frequently confused with 3.

The pairs of digits most often confused are listed below. Note that we give the mi-sprediction rates in both directions. For example, $4 \rightarrow 9$ means that 17.8% of the time a sample of a handwritten 4 is predicted to be 9. We also give the percentage in the other directions. This gives us an idea of which direction the confusion stems from.

Top Confusion Rates:

1. $4 \rightarrow 9$ — 17.8%, $9 \rightarrow 49.0\%$
2. $5 \rightarrow 3$ — 16.3%, $3 \rightarrow 51.0\%$
3. $7 \rightarrow 9$ — 14.2%, $9 \rightarrow 72.0\%$
4. $8 \rightarrow 3$ — 12.6%, $3 \rightarrow 84.9\%$
5. $8 \rightarrow 9$ — 11.7%, $9 \rightarrow 82.0\%$

For the complete and exact per digit classification rates please see the table in Appendix A.1.1. For the complete and exact per digit confusion rates please see the table in Appendix A. For the test samples of the digits which have the highest posterior probabilities please see Appendix A. Feature likelihood and odds ratio plots are given and discussed in the next section.

2.2.2 Discussion of Results

There are two patterns present here – one with both directions having high/moderately high mis-classification rates and the other is where only one digit is confused for the other.

Lets begin by focusing on the (9,4) digit pair – the pair with the highest inaccuracy. A reasonable explanation for this is that these two numbers are tricky to distinguish. They both are similar in that they have some edge on the right side and some empty space enclosed by more edges. Three pieces of evidence support this hypothesis. First, the fact that the misprediction rate is high for $4 \rightarrow 9$ and moderate-to-high for $9 \rightarrow 4$ indicates to us that it is easy to mis-characterize 9's as 4's and 4's as 9's. Second, observe the odds ratio graph in Figure 9. Notice the red regions in the upper half. These are key regions for both the 4 and 9 digits – a handwritten 4 or 9 will very likely set pixels in these regions (also shown in the likelihood graphs in Figure 13 and Figure 14. However, as the odds ratio plot shows pixels set in these regions will increase the likelihood of the sample being a 9 more than that of the sample being a 4. The third and final piece of evidence is shown when we evaluate our 1000 sample testing data set.

Observe that the $5 \rightarrow 3$ mis-prediction is high while the $3 \rightarrow 5$ mis-prediction rate is low. Similarly, the $7 \rightarrow 9$ misprediction rate is high while the $9 \rightarrow 7$ misprediction rate is low. By looking at the corresponding odds ratio figures (Figures 15 we can understand why. The likelihood of five – shown in figure 11 – shows us

that fives fairly frequently encroach pixels in top right corner. The odds ratio plot paints these in the red color – which implies the likelihood of 5 is less than 3 for those regions. This is what causes the misclassifications in the $5 \rightarrow 3$ direction. On the other hand regions of the three will rarely occupy these contested regions, making their scores rarely swing higher for the five.

Similar explanations for the other pairs can be seen. The key idea is to overlay a drawing of the number being misclassified. Any handwritten digit which can be drawn in a way where it occupies more of these pixel regions which are biased for the other class can tip the posterior scores in the other directions.

High and Low Posteriors Part of our assignment was to collect the test samples with the highest and lowest posteriors according to our trained model. Essentially this shows us what is easiest to classify and what is hardest. We list these results in Appendix A. Its worth pointing out to observations that we saw in those results. Skinny numbers score low and fat ones score high. This is because the fat numbers occupy more pixels likely set for that class while the skinny ones occupy less. Examples of this can be seen in numbers 2,3,8, and 9. Another observation to make is that some numbers have alternative forms that are rare. For example, the four can be written with an open top or closed top. The number 7 can be written with a dash across the body. This is not the common form of these numbers. We see this in our data set, the number 7 has a dash across it and it scores the lowest.

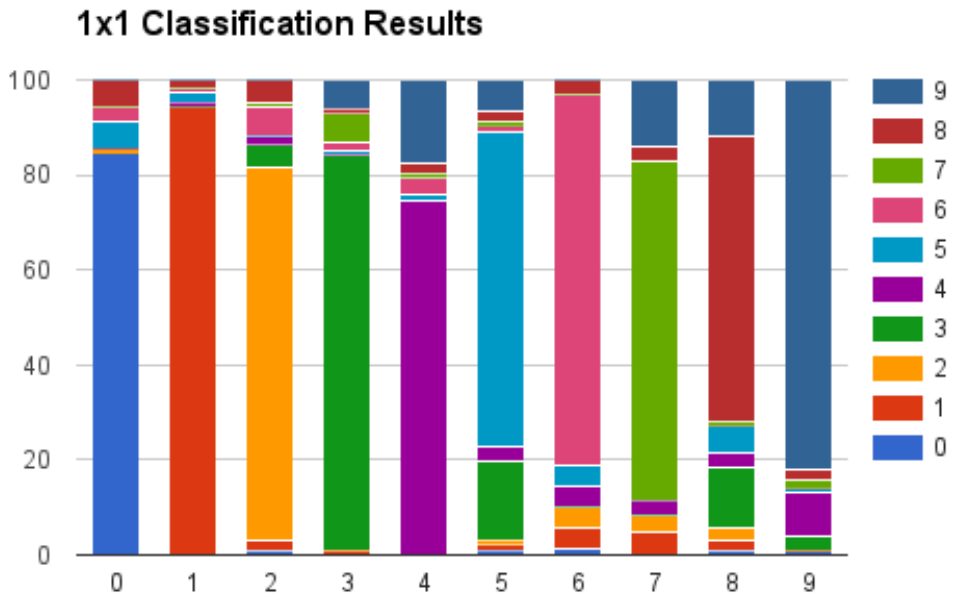


Figure 6: Classifications Results

2.3 Four Unit Portion

2.3.1 Four Unit Results (Summary)

In the four unit portion of the project we essentially explored the topic of feature design. In the 3 unit portion we essentially used the most basic and straightforward feature set. Each image has 28×28 features each corresponding to one of the pixesls from the handwritten digit. These features however, have significant problems we discussed earlier. The first is that for numbers with similar shapes – like 3 and 8 or 4 and 9 – the can frequently be mis-predicted. Things would be much better if our model was built on the shapes present in the images instead of dots.

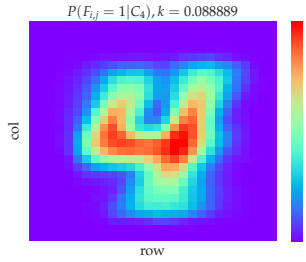


Figure 7: Log Likelihood for 4

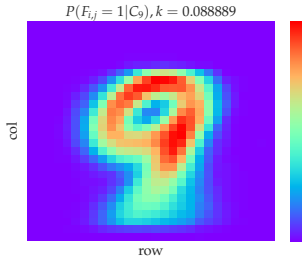


Figure 8: Log Likelihood for 9

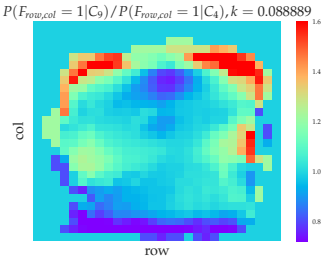


Figure 9: Odds Ratio 9 over 4

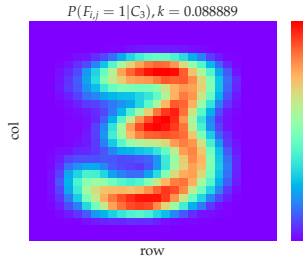


Figure 10: Log Likelihood for 3

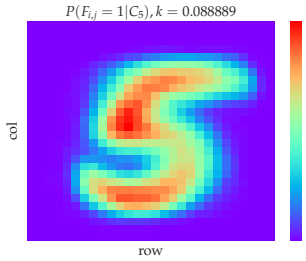


Figure 11: Log Likelihood for 5

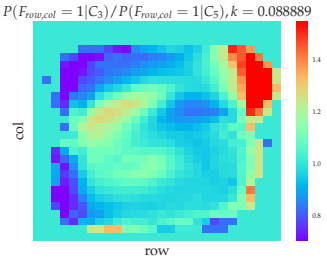


Figure 12: Odds Ratio 5 over 3

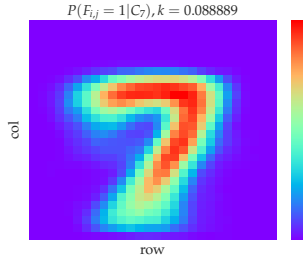


Figure 13: Log Likelihood for 7

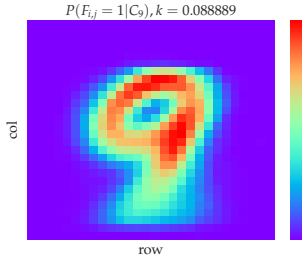


Figure 14: Log Likelihood for 9

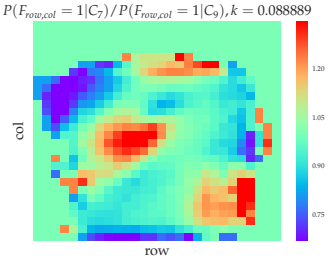


Figure 15: Odds Ratio 7 over 9

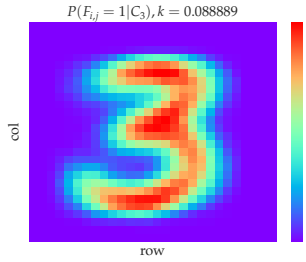


Figure 16: Log Likelihood for 3

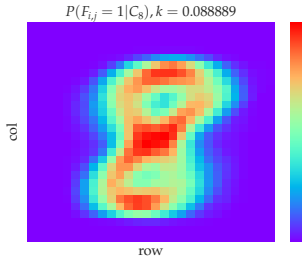


Figure 17: Log Likelihood for 8

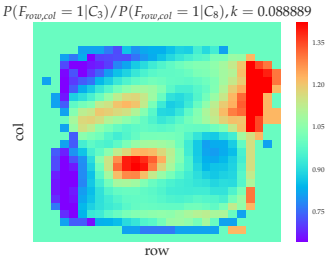


Figure 18: Odds Ratio 8 over 3

Since the conditional indepndence assumption we make with the naive bayes model is critical for efficiency, we have to find another way other than incorporating the way multiple pixels behaved. The solution is to group pixels together. For example, we can the reduce the image from an image with dimensions 28×28 to an image with dimensions of 14×7. This can be done by dividing the images into 98 rectangles each contain 2×4 pixels – four from one row and four from the row below. This way we now have 98 features, each taking one of $2 * 8$ values from 0 to 255. This also implies we need 256 likelihoods to be built for each class and for each feature. So, the number of class conditional likelihoods that had to be computed for this problem is:

$10 \times 2^{L \times W} \times \frac{28^2}{L \times W}$ – where L and W are the length and width of the rectangles used during the reduction. So, for 2x4 rectangles we had to construct $10 \times 256 \times 98 = 250,580$ likelihood. For a 4x4 rectangle we will have to construct over 32 million of these.

This represents not only a burden during training, but also a burden during inference. During inference, we compute 10 scores – one for each class. The total score will be summed using each appropriate likelihood depending on the value for the group being scored in the test image. We will present the results of our analysis of the relationship between training and inference time and the size of the feature set in Section **??**.

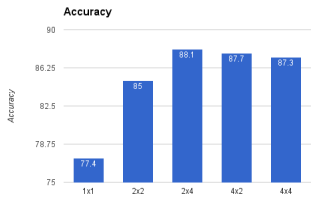


Figure 19: Overall Accuracies Of Groupings (No Overlap)

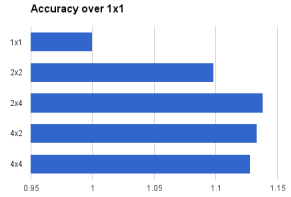


Figure 20: Performance Improvement Over No-Grouping (1x1) By Grouping (With No Overlap)

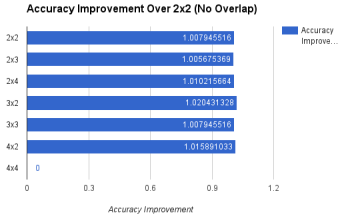


Figure 21: Performance Improvement Of Grouping With Overlap (Over no Overlap)

A graph presenting the overall accuracies is shown in Figure 19. As it shows, grouping is effective at improving the accuracy for reasons we will discuss in the following section. **Detailed results, in form of confusion matrices and accuracy tables can be found in Appendix A.2**.

2.3.2 Discussion Of Results

Discussion of trend
Disjoint Cases
The trends for different feature sets are different. For the disjoint feature sets, we have 2*2, 2*4, 4*2 and 4*4 four cases. Since the group of pixels is disjoint, there are less features constructed in the model, we have a total of14*14=196 features for 2*2 disjoint, 14*7=98 for 2*4 and 4*2 cases, and 7*7=49 features for 4*4 cases. Since the pixels are grouped by group size it follows that the uniqueness of each value for feature is also different. Since we only have 0 and 1 as value for one pixel, so we totally have 2^{m*n} possible values for each individual feature, for example, for 2*2 case, we totally have $2^{2*2} = 16$ possible values. Intuitively speaking, we can achieve better performance if the feature has more precise value representation. However, since more values to choose, less features available in the training process. So there is a trade-off between the amount of features and amount of possible values for the feature. This is why, in later sections we see that more training data is needed for more features and more feature will contribute better overall accuracy.

And as the result shows, in the 2*2 case, we get 85.8% overall accuracy, in 2*4 case, we get 88.6% overall accuracy. In 4*2 case, we get around 87.9% overall accuracy. We could see that we get similar result in 2*4 and 4*2 case, and the overall accuracy of them are both higher than that of 2*2 case. Finally, for 4*4 disjoint case, we get about 84.6% overall accuracy because we have less features to be trained although we get more choice in the values for the features. So as we discussed before, there is a trade-off between these two factors. And in 4*4 case, it is not as good as the previous cases, the overall accuracy will decrease when it override certain threshold. This is one trend we observed by using the same smoothing factor k = 0.1. For the running time, it is not too much difference between 2*2 (25.5189s), 2*4(21.45s) and 4*2(23.319s), but 4*4 will take 138.74s to finish since it has more unique values for features.

Overlap Cases

The trends for overlap case is also different from the disjoint one. We could see that the overall performance of the overlap are better than disjoint cases. Because there are more features in overlap cases compared with disjoint cases. For instance, we get 27*27= 729 features in 2*2 overlap, which is much higher

than 2*2 disjoint case. So the performance of the overall cases should be better. For k=0.1, we get 87.1% overall accuracy for 2*2 case, 89.6% for 2*4, 90.2% for 4*2, 88.8% for 2*3, 90% for 3*2, 90% for 3*3, 87.4% for 4*4. So we could see if the group size is bigger we could get better performance. However, there is also a threshold for the size of groups, e.g. 4*4 is not better than 3*3 and 2*4. And for 2*4 and 4*2 we get similar result, maybe 2*4 is better because it is more appropriate for the representation of digits in the matrix. As for the running time, as the group size increases, the running time is also increasing, we could see that we need 82.12 seconds to get the result of 2*2 overlap, 118.92 seconds for 2*4 overlap, 106.40 seconds for 4*2 overlaps, 91.68 seconds for 2*3 overlaps, 115.454s for 3*3 overlap and 2044.0577s for 4*4 overlaps. It is reasonable since we have more unique values for the features along with the size of group. Besides, in general, we will run longer in overlap cases compared with disjoint cases. Because we have more features to be trained and tested in overlap cases.

2.4 Common Discussion

2.4.1 Running Time Discussion

Inference and training is linear in the number of features and samples. That's great, but unfortunately training is exponential in the number of values those features can take. This is because we will need to store and cycle through 2^d likelihoods to be built. During inference we will only use the appropriate likelihood, so inference stays linear in the number of samples and the number of features.

We gathered empirical results that confirmed this theoretical result. In Figure 22 we see that the 2x4 and 4x2 configurations take equal time. This is to be expected because they have same number of values possible (2^8). Then, with 4x4 configuration we double the number of grouped pixels but see a 10x increase in the running time – as expected based on our exponential relationship expectation.

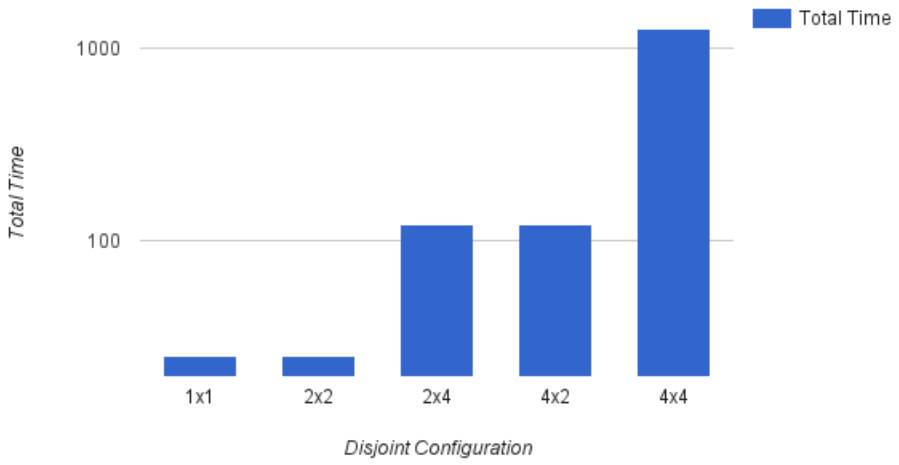


Figure 22: Running time of disjoint

In this problem we explored enhancing classification performance by expanding the features. We made the conditional independence naive bayes assumption to make things feasible to evaluate. But this caused us to fail to separate some digits – like 9 and 4 – based on structural differences – like the connected loop in 9 vs the disconnected loop in 4. This can be overcome by considering multiple pixels at the same time.

The draw back to doing this is the fact that the number of likelihoods exponentially increases with the dimensions of the grouping. This is because there are $2^(L \times W)$ possible states of a square of pixels with dimensions $L \times W$. This in turn causes dramatic increases the time for training and inference times.

Figure 22 shows this time vs feature relationship.

2.5 Extra Credit

It is difficult to conclude whether ternary values should be used or not. What we see in the data shown in Figures 23-25 is that for some configuration ternary pixel values helps for others it hurts. Furthermore, for some digits – such as 8 – it helps but for others it hurts for some feature configurations but hurts for others. For example, for the number 3. Ternary values help improve the performance of the 2x4 configuration but it hurts the 1x1 configuration. The configuration that is clearly helped is the 2x2 overlapping case. For all the other cases, it seems like the ternary posteriors need to be used on a per digit basis.

We believe the cause for this behavior is that when you look at the images, the added information applies to a small regions of the samples. The dark regions shown in the binary pixel values of Figure 27 corresponds to the bulk of the handwritten digit. Whats left can be described as "weakly shaded" regions. As we see in Figure 26 the more common pixel type is the bulk region. We think the relatively small amount of these "lightly shaded" regions makes it to where they mostly devolve to added noise. In some cases they help – in others they hurt.

2.6 Ternary Pixel Values: Discussion Of Results

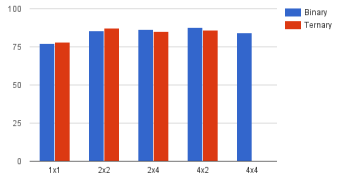


Figure 23: Overall Accuracies Of Groupings (No Overlap)

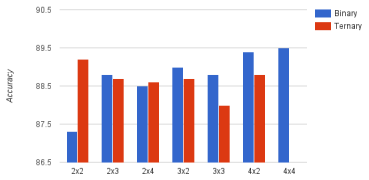


Figure 24: Performance Improvement Over No-Grouping (1x1) By Grouping (With No Overlap)

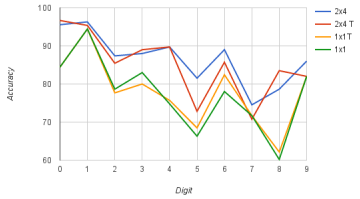


Figure 25: Performance Improvement Of Grouping With Overlap (Over no Overlap)

We wanted to run our this data set for all configurations, but unfortunately the change from a branching factor of 2 to 3 causes an expected increase in the running time of more than 650 (3^16 / 2^16) for our worst 4x4 configuration! For this reason we couldn't finish the 4x4 (overlap and disjoint) or 3x3 overlap case.

2.7 Extra Credit: Face Data Set

For extra credit we created models for the face data set. This problem is different in that it is a binary classification problem. As we see from Figure 28 the pixel likelihoods for the face class shows the characteristics of what we can consider to be the average face. On the other hand, the non-face class pixel likelihoods (Figure 29 resembles the pattern of white noise. Exactly what we would expect from a class that contain so many different types of figures.

For this data set, we find extremely good results. As we see in Table **??** many configurations two can achieve perfect recall at a modest 1.37% false positive rate. Whats even more impressive, these are achieved without needing any grouping.

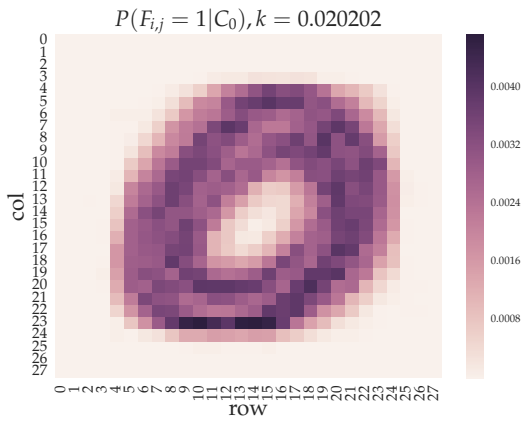


Figure 26: Zero, Ternary Pixel Values

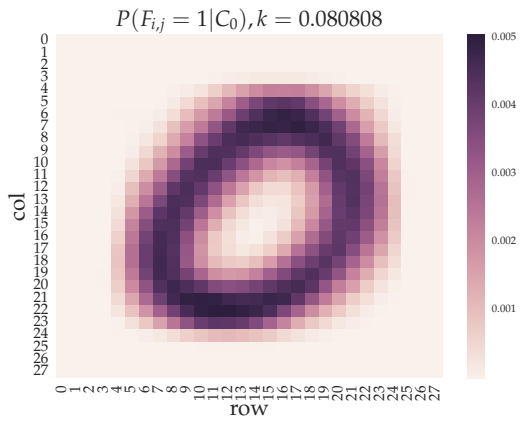


Figure 27: Zero, Binary Pixel Values

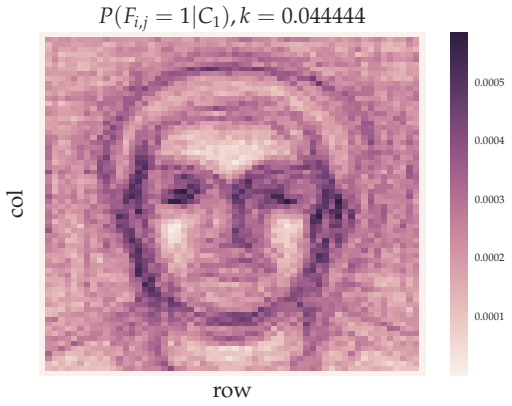


Figure 28: Pixel Likelihoods for Face Class

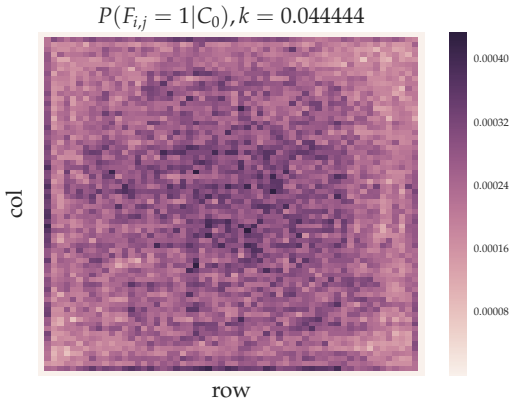


Figure 29: Pixel Likelihood for Non-Face Class

Table 2: Face Classifiers

	Recall	**FPR**
1x1	92.1	16.44
2x2	100	1.37
2x4	100	1.37
4x2	96.1	0
4x4	94.81	0
2x2 (O)	97.4	0
2x3 (O)	98.7	1.37
2x4 (O)	98.7	1.37
3x2 (O)	97.4	0
3x3 (O)	98.7	1.37
4x2 (O)	97.1	0
4x4 (O)	96.1	0

Unfortunately we couldn't finish running the face dataset on all our datasets – this is because the size of the grid is 5 times that of the digits. This made it very difficult to run the overlapping cases.

2.8 Extra Credit: Studying Naive Bayes Data Requirements

We tried to also understand the impact of the training data size on the performance of the Naive Bayes technique for classification. Different classification and learning techniques have varying data requirements.

The first experiments we ran were looking at the impact of the training size on the accuracy of the resultant models. What we found is that there is a impact. Like most things, at some point the law of decaying returns kicked in. The trend our graph shows is that as the number of features expands the need for more data to achieve optimal performance grows. This is shown in particular by the 5x5 configuration. By reducing the training set to 500 samples, it looses over 16% of potential accuracy. In most cases, 3000 samples was enough. And in two cases training with 3000 samples actually produced better results.

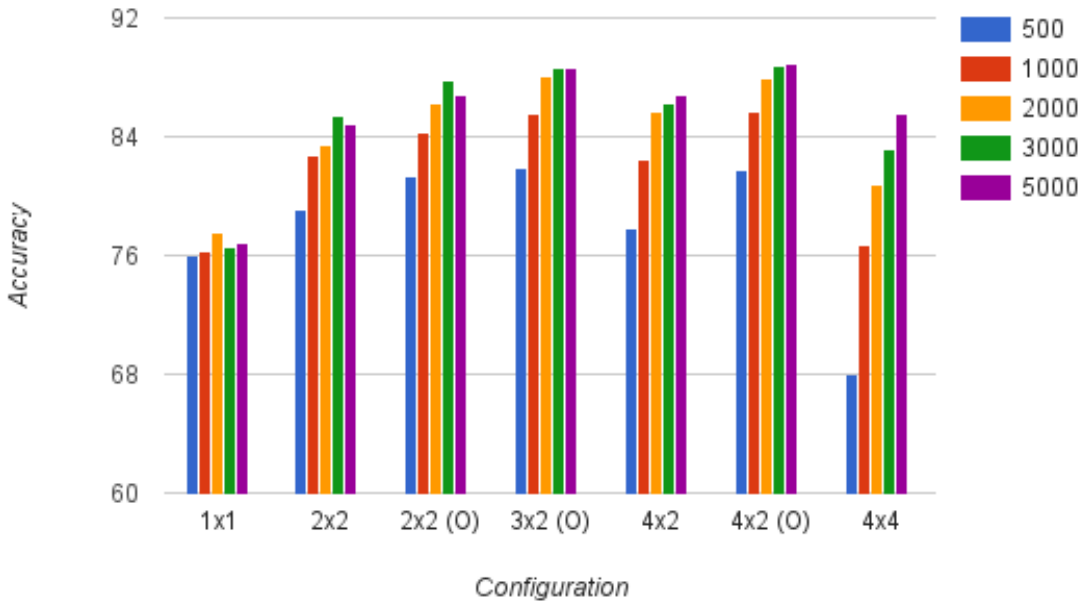


Figure 30: Impact Of Training Data Size On Accuracy

3 Part 2: Text Document Classification

3.1 Overview

The goal of the text document classification aims to build a Navie Bayes classifier to solve a binary classification problems in natural language processing. In part 2.1, we have two datasets to classify, one is the sentiment analysis of movie reviews and the other one is conversation topic identification. And the objective for these two datasets is that we need to achieve a binary classification based on the corpora. So in the implementation of Navie Bayes event models, we have three choices: Gaussian navie bayes, Multinomial navie bayes and Bernoulli navie bayes. As the basic model is based on bag of words model, so the data we are dealing with is discrete data instead of continuous data, so we choose Multinomial and Bernoulli navie bayes to implement this task. After the implementation of the model, we trained and tested the two models with our pre-processed datasets. And we got the overall accuracy and confusion matrix for each model.

For the part 2.2, the dataset is larger and it is not binary classification any more. The basic data is similar with the conversation topic dataset as described in the part 2.1. But there are 40 topics to classify and the corpora of words is larger than that of 2 topic case. We also trained and tested data by using the previous two models and get an overall accuracy and confusion matrix for each topic. Besides, we also find out the most likely confused topic for each individual topic.

To simplify the description of different data in these dataset, we use document to represent each train or test case in different corpora. And the document is comprised of different words. And before the implementation of the model, we first implemented functions to read the training data and test data, and we also design a set to collect all unique words in the training data. This set is acturally our vocabulary for the model.

3.2 Multinomial Naive Bayes Implementation

With the multinomial model for bags of words case, the feature vector represents the frequencies with which certain events have been generated by a multinomial $(p_1, p_2, ..., p_n)$. p_n is the probability that the word i occurs in the document. The document of length N contains words: $\{w_1, w_2, w_3, ..., w_n\}$, where w_i is the i-th word in the document. And the value for the variable is actually the occurrence of that word in a single document. As we have discussed at the very beginning of the Navie Bayes Theorem, we need to estimate the likelihoods $P(w_i = k \mid class)$. We could use the frequency count of words to estimate likelihoods since we have the occurrence of words in each class. Besides, considering some words we may not have seen in our training data, we also need to add smoothing factor in the calculation.

The likelihood is calculated as:

$$P(X = word_i \mid class_j) = \frac{\#\ word_i\ in\ class_j + k}{\#\ total\ words\ appeared\ in\ class_j + k * V} \tag{1}$$

where the total number of words appeared in one class is actually the sum of the length of document in this class. And k is the smoothing factor, V is the size of vocabulary.

The prior of $class_j$ is:

$$P(class_j) = \frac{\#\ docs\ in\ class_j}{\#\ total\ trained\ docs} \tag{2}$$

To implement the mathematical expression of likelihood, we used the **dictionary** data structure in Python to store the entire math expression. The *key* in the dictionary is the unique word we have seen from our vocabulary V. And the *value* is the count of the word in the document. And another dictionary is nested in the previous dictionary in order to store the count to different class category. We call the *key* "label" in the nested dictionary. So the representation of the counting process is:

$$count = likelihood[word][label] \tag{3}$$

where

$$word \in V, label \in \{-1, 1\} \tag{4}$$

After we apply the equation of calculating likelihood, we need to convert the probability into logarithm expression (log) in order to calculate it conveniently later. And it is easy to implement this by using *math.log()* function. At this stage, the training process is finished. Now we have the likelihood for the words and the prior probability for each class. In the test process, our objective is to calculate the posterior as:

$$P\left(Y = class_j | X_1, X_2, \cdots, X_{n'}\right) \approx \prod_{i=1}^{n'} P\left(X_i \mid Y = class_j\right) P(Y = class_j) \tag{5}$$

where $Y \in \mathcal{L} = \{class_1, class_2, \cdots, class_k\}$, $X_i \in \mathcal{V} = \{word_1, word_2, \cdots, word_m\}$

To implement the calculation, we need to make a judgement whether one word in the test document has been seen in the vocabulary. The pseudo code is:

```
For label IN all_labels:
    posterior[label] += prior[label]
for wordk count IN testData:
    IF word IN keys of likelihood:
        temp_log = likelihood[word][label] * count
        posterior[label += temp_log
```

As the pseudo code indicates, if the answer for whether we have seen this word is no, we could just **ignore** the word as we already take the missing word into consideration by adding smoothing factor in the likelihood calculation. If the answer is yes, then we also need to consider how many times this word appears at the test document. For instance, if the test document is:

$$document = \{red : m,\ blue : n\} \tag{6}$$

The posterior could be calculated as:

$$P\left(Y = 1 | X_1, X_2, \cdots, X_9\right) \approx \prod_{i=1}^{m} P\left(X_i = red \mid Y = 1\right) \cdot \prod_{j=1}^{n} P\left(X_j = blue \mid Y = 1\right) P\left(Y = 1\right) \tag{7}$$

Then we could get totally posterior, where N is the number of labels (i.e. class) in the corpora. And we could get the maximum posterior according these N values. The label of this maximum posterior is the prediction label for this test data. And it is also the output of our trained model.

3.3 Bernoulli Naive Bayes Implementation

In the Bernoulli event model, feature vector is easier to represent compared with multinomial. The feature is the Boolean of whether the word has appeared in this class. And we dont care the word frequency in Bernoulli event model. The idea is that if the word has appeared at the document belongs to one class, then the feature value is 1, otherwise it is 0. We could estimate the likelihood by counting how many documents contain the word in the set of our training documents. The math expression is:

$$P(word_i = 1 \mid class_j) = \frac{\#\ document\ contains\ word_i\ in\ class_j + k}{\#\ total\ \ documents\ \ in\ class_j + k * V} \tag{8}$$

where k is the smoothing factor and V is the size of possible values for features, in this case, $V = 2$ because the value for the word could only be 1 or 0. The count could be stored as:

$$count = likelihood[word][1][label] \tag{9}$$

where the label "1" indicates the Boolean value for this word is 1. And we also need to consider when the probability that the word has not appeared in a particular class. Intuitively speaking, the probability of the word has not appeared in a label is 1 minus the probability that it appears at this class. So the math expression is:

$$likelihood[word][0][label] = 1 - likelihood[word][1][label] \tag{10}$$

The prior is calculated the same as Multinomial case, so we omit it here. So after the calculation of all likelihood for each word in each label, we also need to convert the result into logarithm expression as what we did at multinomial case. Then the training process is finished. As for the test process, the situation is different compared with the multinomial case. First, our pseudo code is:

```
FOR label IN all_labels:
    posterior[label] += prior[label]
FOR word IN vocabulary:
    IF word IN testData:
        temp_log = likelihood[word][1][label]
    ELSE:
        temp_log = likelihood[word][0][label]
    posterior[label] += temp_log
```

So we could notice that we need to traverse every word in the vocabulary for test one document. And it is the same in the training process. So the running time for Bernoulli model should be longer than multinomial case. And in the testing process, we need to calculate the product of each variable X_i that whether it appears at the test document, which means we need to traverse the entire vocabulary to get the estimated posterior. The math expression is:

$$P\left(Y = class_j | X_1, X_2, \cdots, X_m\right) \approx \prod_{i=1}^{m} P\left(X_i \mid Y = class_j\right) P(Y = class_j) \tag{11}$$

Then we could get totally N posterior, where N is the number of labels (i.e. class) in the dataset. And we could get the maximum posterior according these N values. The label of this maximum posterior is the prediction label for this test data. And it is also the output of our trained Bernoulli model.

3.4 Full 40-Topic Corpus (Four Units)

In this part, our dataset is changed to a 40 topic corpus, which focuses on the classification for 40 different topics. The implementation of the Multinomial and Bernoulli model is the same as what we have discussed in the part2.1.As the training set becomes larger, we need more time to build our data structure in word dictionary. Specifically speaking, after finding all unique word in the training set, the size of vocabulary is 52993. It is almost five times bigger than that of vocabulary in part2.1. Besides, there are totally 40 labels in our model. So in our data structure by using dictionary of Python, we need totally $V = 40 * 52993$ index in the dictionary. So the training process is longer than part2.1.

Besides,it takes longer time to get the result for Bernoulli than multinomial in part2.2. Since in the testing process of Bernoulli model, we need to traverse each word in the vocabulary for each label, which means for each test case, we need to traverse 52993 individuals in the likelihood dictionary in order to calculate the posterior. So, the testing of Bernoulli model will take us longer time, sometimes maybe over one hour to get the final result.

3.5 Results

For detailed results, please see Appendix-B. An important thing to clarity in our result. There are two labels: 0 and 1, 0 is mapping with -1 in the original dataset. For the convenience of representation in our implementation, we use 0 to replace -1 here.

3.6 Discussion

3.6.1 Part2.1

Analysis of Movie Review Result

In the dataset of movie review, we notice that the the overall accuracy of both multinomial and Bernoulli model are similar:

Overall accuracy
Multinomial: 0.761
Bernoulli: 0.755

```
HINT: row index is actual value, column index is predicted value
-------------------------------------------------------------
Confusion Matrix
 [0,           1]
0[0.756, 0.244]
1[0.234, 0.766]

-------------------------------------------------------------
Classification rate for each digit:
The class 0: 0.756
The class 1: 0.766
-------------------------------------------------------------------------------------------------------------
Top 10 words with highest likelihood for class 0:   Top 10 words with highest likelihood for class 1:
movie: -4.6759392568762985                          film: -4.704509111705017
film: -4.91991689509335                             movie: -5.12405895969492
like: -5.249396096223593                            --: -5.440519996696744
one: -5.379449224471791                             one: -5.642002051229775
--: -5.604330395684541                              like: -5.755330736536778
bad: -5.8719257095695845                            story: -5.806624030924328
story: -5.894915227794283                           good: -5.917849666034552
much: -5.918445725204478                            comedy: -5.929684123681556
time: -6.01852918376146                             way: -5.96605176785243
even: -6.086582647006476                            even: -6.016695500671185

-------------------------------------------------------------------------------------------------------------

Top 10 words with highest odds ratio for class 0:   Top 10 words with highest odds ratio for class 1
flat: 2.719288599579288                             disturbing: 2.696811802625132
stale: 2.6502957280923365                           refreshingly: 2.386656874321293
dull: 2.6139280839214614                            haunting: 2.386656874321293
tired: 2.4961450482650784                           engrossing: 2.386656874321293
plain: 2.4091336712754483                           grief: 2.386656874321293
mediocre: 2.4091336712754483                        refreshing: 2.2913466945169674
unfunny: 2.3138234914711244                         gripping: 2.2913466945169674
forced: 2.3138234914711244                          inventive: 2.2913466945169674
```

```
poorly: 2.3138234914711244                    polished: 2.2913466945169674
bore: 2.3138234914711244                      gem: 2.2913466945169674

-------------------------------------------------------------------------------------------------------
The accuracy is:  0.761
Execution time is: 0.373941s
```

First, for multinomial model, we notice that the top 10 words with highest likelihood for different class is totally different from the top 10 words with highest odds ratio. It is easy to see from the result that the top10 highest likelihood words are some general words, e.g. "film", "like", "one", "bad", "good", "much", "time", "event", "story". These are the words that usually used to describe a movie generally. They are more frequently used when we want to express some ideas to a specific topic of movie. So, it is reasonable to see that for both of classes, we have the similar top 10 highest likelihood words. As for the top 10 words with highest odds ratio. Our calculation of the odds ratio is as follows:

$$odds\ ratio\ = \frac{P(word_i \mid label = 0)}{P(word_i \mid label = 1)} \tag{12}$$

So the odds ratio could show which word is used more frequently in a specific class compared with another one, in other words, it is topic-specific instead of general appearance like top 10 words highest likelihood. Therefore, the top 10 words with highest odds ratio could reveal the characteristics of the class, for example, the word "flat appears in the top 10 words with highest odds ratio of negative class(i.e. class 0), which means people would prefer the word "flat in negative review while people rarely use the word "flat in a positive review. And "disturbing, "haunting are in the top 10 words for positive class, which means people would like to use these words to characterize a positive review while people wont use these words in a negative review.

Therefore, we could conclude that the top 10 words with highest odds ratio could characterize the class (i.e. positive or negative). But top 10 words with highest likelihood could just show that these words are more frequently used in one movie review, it doesnt matter whether the review is positive or negative. So these words are general words to express some ideas about the topic of movie, rather than describing features of the opinion in the review.

```
HINT: row index is actual value, column index is predicted value
-----------------------------------------------------------------
Confusion Matrix
 [0,      1]
0[0.768, 0.232]
1[0.258, 0.742]

-----------------------------------------------------------------
Classification rate for each digit:
The class 0: 0.768
The class 1: 0.742
-------------------------------------------------------------------------------------------------------
Top 10 words with highest likelihood for class 0:   Top 10 words with highest likelihood for class 1:
movie: -1.9529277217139598                          film: -1.9599948889370522
film: -2.1902559080201263                           movie: -2.4089451089849554
like: -2.545656154526858                            one: -2.9014215940827497
one: -2.674648274717961                             like: -3.006782109740576
story: -3.1592507033848496                          --: -3.016934481204594
much: -3.183061352078568                            story: -3.0910424533583156
--: -3.207452805202727                              comedy: -3.195182712610913
bad: -3.258096538021482                             way: -3.2324541074081443
time: -3.311442518726775                            even: -3.258096538021482
```

```
even: -3.353406717825807                                    good: -3.297836866670996

-------------------------------------------------------------------------------------------------------

Top 10 words with highest odds ratio for class 0:   Top 10 words with highest odds ratio for class 1:
flat: 2.7080502011022096                            disturbing: 2.639057329615259
stale: 2.639057329615259                            refreshingly: 2.3978952727983707
dull: 2.5649493574615363                            grief: 2.3978952727983707
tired: 2.484906649788                               engrossing: 2.3978952727983707
mediocre: 2.3978952727983707                        haunting: 2.3978952727983707
plain: 2.3978952727983707                           refreshing: 2.302585092994046
poorly: 2.302585092994046                           gripping: 2.302585092994046
forced: 2.302585092994046                           polished: 2.302585092994046
unfunny: 2.302585092994046                          affecting: 2.302585092994046
bore: 2.302585092994046                             inventive: 2.302585092994046

-------------------------------------------------------------------------------------------------------
The accuracy is:  0.755
Execution time is: 54.626312s
```

For the result of Bernoulli model, we notice that the top 10 highest likelihood words and top10 highest odds ratio words are similar with the multinomial model. And the reason why we get such result is the identical with the multinomial case.

As we can see from the above result, there is not big change compared with the top 10 words with highest likelihood we get from multinomial model. These words are the words that we usually use in the review for movies. So we could see that the top 10 words with highest likelihood are almost same for the class 0 and class 1. Just one word is "good" for class 1, which is positive review, while "bad" for class 0, which is negative review. And it makes sense for the frequently used words we get from the top 10 highest likelihood in the result. This is one of the trends we find in the results. In addition, we find that the top 10 words with highest odds ratio is also with that of multinomial model. For example, we notice people will often use "dull","tired","mediocre","poorly" to describe negative experience in a movie review. While using "engrossing","polished","gripping" to describe positive experience in a movie review. So these words could reflect the most sharping experience people feel for a group of movies(i.e.negative or positive in this case). And that's the knowledge we could get from the top 10 highest odds ratio words. That's the biggest difference between top 10 highest likelihood words and top 10 highest odds ratio words.

Analysis of Binary Topic Result

As for another case: conversation binary topic classification. The idea is similar with the movie review case. After several test on different smoothing factors k, we found out that the multinomial model performs a little bit better than Bernoulli model. And they can both get over 90% accuracy. And if we tune the smoothing factor k, we could get different accuracy, which indicates the impact of smoothing factor on the models. For the trend we find, we think in a small range, for example, $(0.01, 1)$, $k = 0.01$ will give us a better result, and if we tune k to smaller, the overall accuracy will decrease. So, we think there is a threshold value of k that will give us a better accuracy. But it is a little tricky because the default value we use in text document classification is 1. So this is a phenomenon we find in the testing process.[**Bonus Point**]

Binary topic multinomial model result:

```
HINT: row index is actual value, column index is predicted value
-------------------------------------------------------------
  [0, 1]
0[0.9387755102040817, 0.061224489795918366]
```

```
1[0.10204081632653061, 0.8979591836734694]

------------------------------------------------------------
Classification rate for each digit:
The class 0: 0.9387755102040817
The class 1: 0.8979591836734694
------------------------------------------------------------------------------------------------------------
Top 10 words with highest likelihood for class 0:   Top 10 words with highest likelihood for class 1:
know: -2.910374098291088                            know: -2.9656753972801257
yeah: -3.09338333161108                             yeah: -3.0920674144329356
uh: -3.495268685128181                              like: -3.5442710676555786
like: -3.5192182486052146                           uh: -3.8203013626144338
um: -3.772754801261674                              um: -3.932062959614287
right: -3.946199695203724                           right: -3.9908900394304148
just: -4.018611645280864                            don: -4.055063301882509
think: -4.03495890770624                            think: -4.070541403616782
oh: -4.120772339973308                              just: -4.104863993157005
don: -4.141399503703598                             oh: -4.21200027495845

------------------------------------------------------------------------------------------------------------

Top 10 words with highest odds ratio for class 0:   Top 10 words with highest odds ratio for class 1:
relationship: 5.288602653961092                     wage: 5.013675699632688
compatibility: 4.785951946878692                    minimum: 5.012695787746339
communication: 4.7590444939587675                   welfare: 4.857760486458663
marriage: 4.719641946092739                         wages: 4.724752243205215
partner: 4.687729819710658                          inflation: 4.505784063301485
relationships: 4.56575390579424                     waitresses: 4.348155119097903
friendship: 4.558373798496616                       tax: 4.335885026506088
attracted: 4.547200497898491                        waitress: 4.23189531298204
dating: 4.489380927009665                           salary: 4.175542376430909
compatible: 4.415272954855943                       increase: 4.06857025687874

------------------------------------------------------------------------------------------------------------
The accuracy is:  0.9183673469387755
Execution time is: 0.573207s
```

From the above result, we notice that top 10 highest likelihood words are the words we usually used in the daily conversation, for instance, "yeah", "like", "oh", etc. And it is reasonable why these words appear frequently in the train data. Because we use them much more frequently compared with others. And we could also notice that the top 10 highest likelihood words are similar for class 0 and class 1, which proves our intuitive idea that these are the words we daily used. So no matter which class it belongs to, the most frequently used words are similar in the top 10 list.

As for top 10 highest odds ratio words, we notice that for class 0("Life Partners"), the words "relationship", "partner", "dating" ranks highly as shown in our result, which is a good reflect of the topic of this class: Life Partners. And for class 1("Minimum Wage"), we could see that the words "wage", "minimum","tax","salary" appears more times in the documents belongs to class 1. So these words are good characteristic words for that class, which means they could give people the most sharping information about what is about in this topic. In other words, words in top 10 highest odds ratio list could characterize the class ("Life Partners" or "Minimum Wage") it belongs to. We could call these words "feature words" of this class.

Binary topic Bernoulli model result:

```
HINT: row index is actual value, column index is predicted value
------------------------------------------------------------
Confusion Matrix
[0,                                    1]
0[0.9183673469387755, 0.08163265306122448]
1[0.10204081632653061, 0.8979591836734694]

------------------------------------------------------------
Classification rate for each digit:
The class 0: 0.9183673469387755
The class 1: 0.8979591836734694
-------------------------------------------------------------------------------------------------------------
Top 10 words with highest likelihood for class 0:   Top 10 words with highest likelihood for class 1:
know: -3.6370465104852876                           know: -3.6326209953369784
like: -3.6370465104852876                           um: -3.6326209953369784
just: -3.6393270131840127                           think: -3.6326209953369784
think: -3.6416127284648687                          don: -3.6326209953369784
don: -3.6416127284648687                            just: -3.6326209953369784
yeah: -3.6416127284648687                           like: -3.6326209953369784
um: -3.6531203353163484                             people: -3.6348911438715175
right: -3.6554378334567112                          yeah: -3.6348911438715175
oh: -3.657760714872851                              oh: -3.6394469604073785
really: -3.662422727978662                          really: -3.6417326756882344

-------------------------------------------------------------------------------------------------------------

Top 10 words with highest odds ratio for class 0:   Top 10 words with highest odds ratio for class 1:
attracted: 4.1590030305830386                       wage: 4.343685474630318
compatibility: 4.127254332268458                    wages: 4.233986557373894
relationship: 4.0431712150579155                    inflation: 4.127014437821726
relationships: 3.8782414009758295                   waitresses: 4.007213238009106
attraction: 3.8713209581312578                      minimum: 4.007213238009106
friendship: 3.8713209581312578                      waitress: 3.9701719663287562
communication: 3.8178322731802714                   welfare: 3.93170568550096
marriage: 3.8178322731802705                        salary: 3.817592378733539
dating: 3.791856786777009                           retail: 3.6887595068905705
qualities: 3.737789565506735                        increase: 3.663441698906281

------------------------------------------------
The accuracy is:  0.9081632653061225
Execution time is: 9.274343s
```

From the above result, we could see that the top 10 words with highest likelihood is just similar as what we get in multinomial model. And for each class("Life Partners" or "Minimum Wage"), the top 10 words with highest likelihood are also similar. So this phenomenon also indicates that these words are the most frequently used words in our daily conversation. We couldn't get the extra information about what content or what topic it belongs to. So it could only reflect the words we daily used, not the sharping feature of the class. In the contrary, we could get this information from the top 10 highest odds ratio words. And another interesting thing we find is that the top 10 highest odds ratio words are similar for each class in the multinomial model and Bernoulli model. Just some of them in different orders. So the basic idea is the same as we have discussed, the top 10 words with highest odds ratio could characterize the class it belongs to, which means the words like "friendship", "dating" couldn't appear at the class 1("Minimum Wage") while "wage", "inflation" could certainly appear more times at the class 1("Minimum Wage").

Finally, we also notice a phenomenon that there are some words with the same likelihood or odds ratio,

but in each training process, the order they stored in the dictionary data structure may be different, so the final top 10 words might get changed due to this reason. Therefore, we notice that we may get different result on the top 10 words list at each independent execution of our program.

3.6.2 Part2.2

The more accurate model we get:

```
-------------------------------------------------------------------------------------------
Classification rate for each digit:                    Most likely confused topic:
The class 0: 0.8444444444444444                        topic 0:23
The class 1: 0.8928571428571429                        topic 1:25
The class 2: 0.8775510204081632                        topic 2:39
The class 3: 0.8163265306122449                        topic 3:17
The class 4: 0.9545454545454546                        topic 4:29
The class 5: 1.0                                       topic 5:39
The class 6: 0.9166666666666666                        topic 6:8
The class 7: 0.8571428571428572                        topic 7:33
The class 8: 0.8823529411764706                        topic 8:27
The class 9: 0.8823529411764706                        topic 9:29
The class 10: 0.9714285714285714                       topic 10:28
The class 11: 0.8333333333333334                       topic 11:26
The class 12: 1.0                                      topic 12:39
The class 13: 1.0                                      topic 13:39
The class 14: 0.84                                     topic 14:30
The class 15: 0.8928571428571429                       topic 15:31
The class 16: 1.0                                      topic 16:39
The class 17: 0.5625                                   topic 17:23
The class 18: 0.9285714285714286                       topic 18:15
The class 19: 1.0                                      topic 19:39
The class 20: 0.8571428571428571                       topic 20:15
The class 21: 1.0                                      topic 21:39
The class 22: 0.96875                                  topic 22:31
The class 23: 0.9                                      topic 23:37
The class 24: 0.8518518518518519                       topic 24:31
The class 25: 0.782608695652174                        topic 25:23
The class 26: 0.7142857142857143                       topic 26:37
The class 27: 0.8620689655172413                       topic 27:37
The class 28: 1.0                                      topic 28:39
The class 29: 0.9444444444444444                       topic 29:8
The class 30: 0.88                                     topic 30:27
The class 31: 0.8846153846153846                       topic 31:12
The class 32: 0.875                                    topic 32:31
The class 33: 0.96875                                  topic 33:2
The class 34: 0.9705882352941176                       topic 34:7
The class 35: 0.8461538461538461                       topic 35:36
The class 36: 1.0                                      topic 36:39
The class 37: 0.9285714285714286                       topic 37:27
The class 38: 0.9615384615384616                       topic 38:1
The class 39: 0.6521739130434783                       topic 39:2
-------------------------------------------------------------------------------------------
Smoothing k = 0.1
The accuracy is:  0.8994800693240901
Execution time is: 27.553080s
```

The confusion matrix for 40 topic two models:

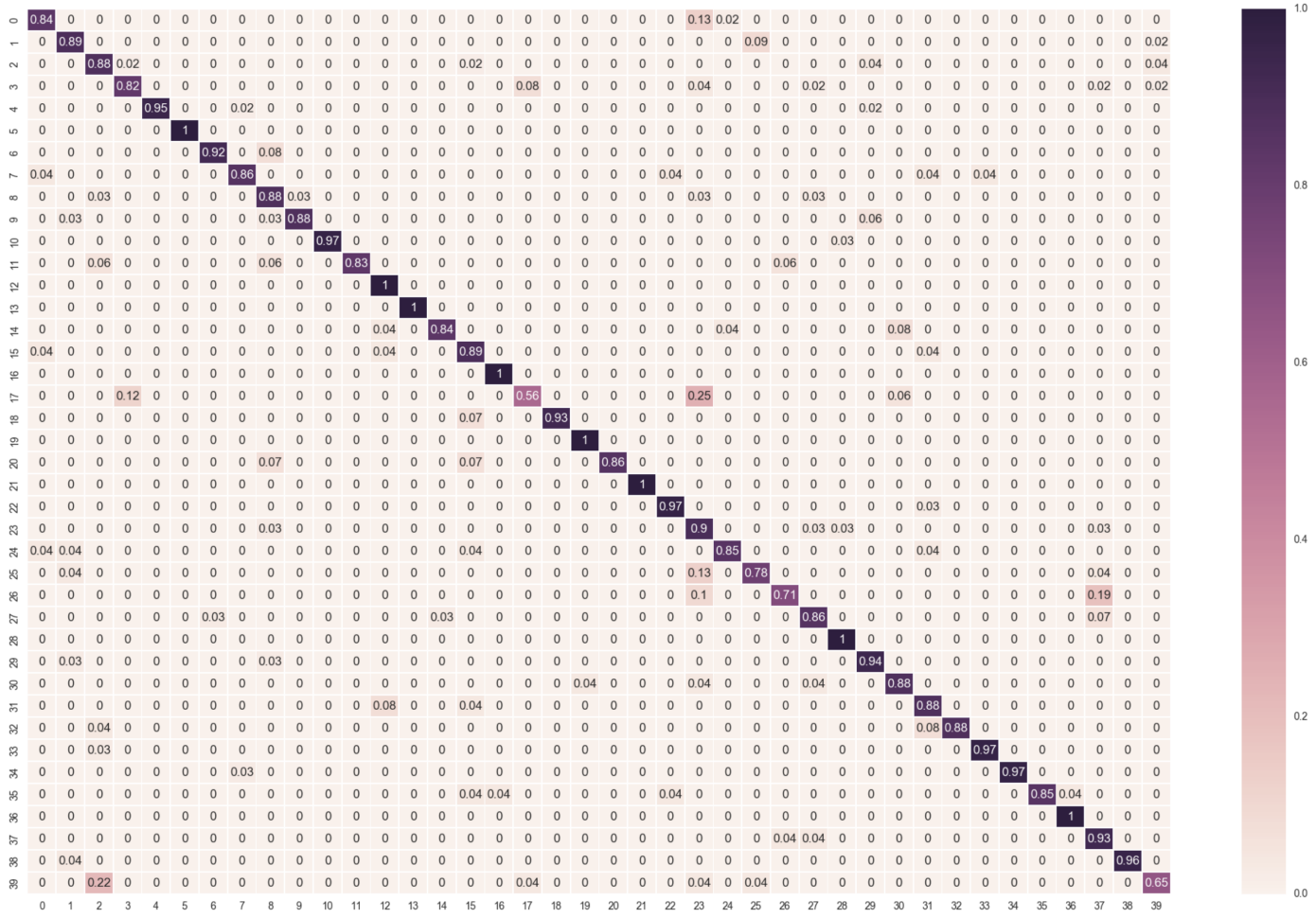


Figure 31: Confusion matrix of 40 topics multinomial model

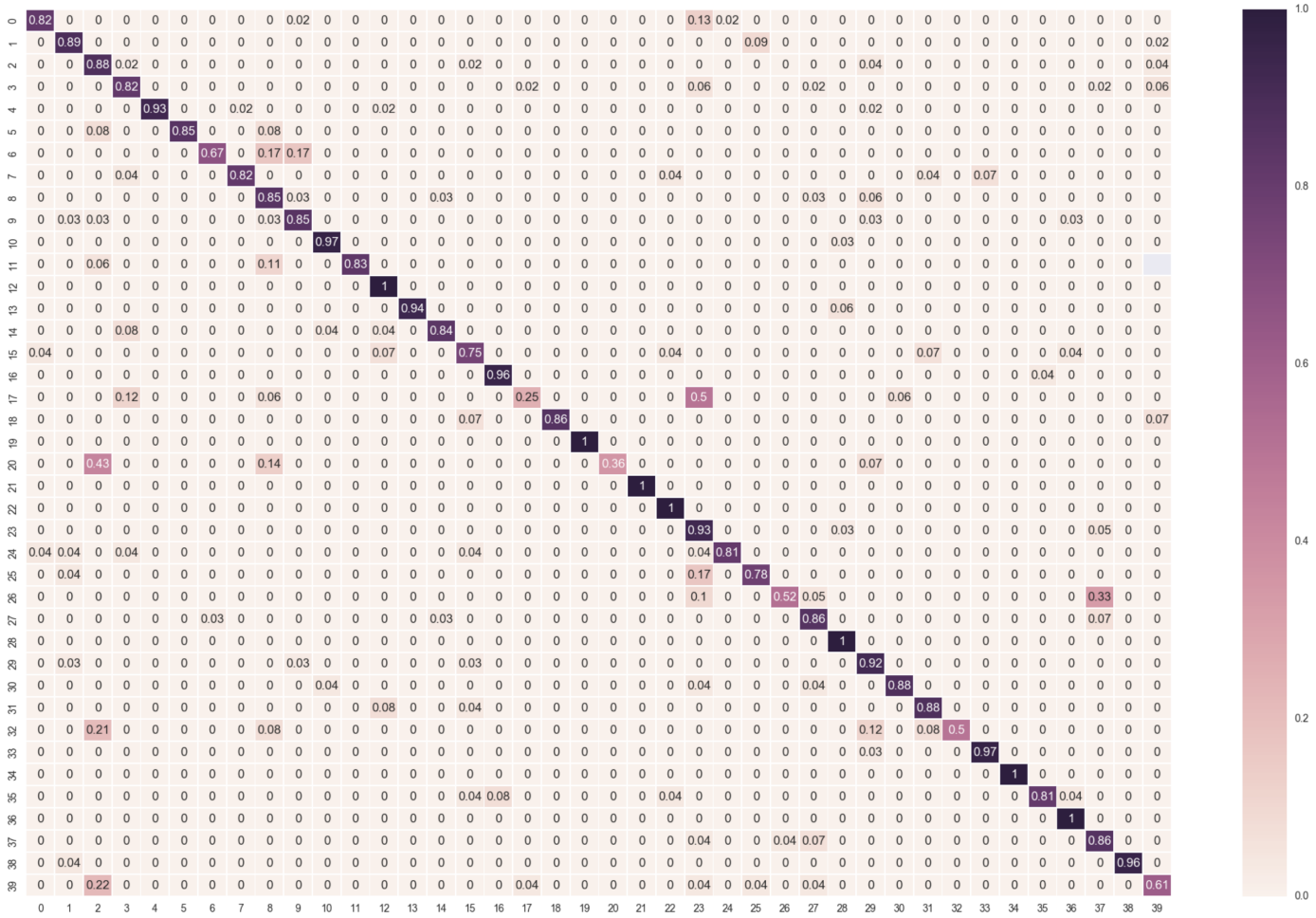


Figure 32: Confusion matrix 40 topics Bernoulli model

Analysis of Result

As the result show, multinomial model performs better than bernoulli model. There are several reasons about this result. First, the variance of the document length. Bernoulli model could work well with short documents as it only takes the Boolean appearance of the word as the features, the shorter the document, the better the model for training the likelihood. However, multinomial model could do better in long document because it takes the frequency of the word into the consideration. The multinomial model makes an assumption of positional independence while Bernoulli model ignores positions in documents altogether because it will traverse all words in the training vocabulary. Besides, Bernoulli model could work better with fewer features while multinomial model could handle more features. (e.g. for different unique words in the vocabulary in bags of word model). So generally speaking. multinomial would perform better when there are more features in the document.

Second, the **feature filtering** is also the reason why multinomial model performs better in this larger corpus. As the goal is not binary classification, we have totally 40 topics now. And for Bernoulli model, it is important to do the feature filtering in this multi-classification situation. Because Bernoulli model doesnt consider the multi-appearance of the word, so some trivial, no-meaning word would influence the result of the likelihood. For example, some words in our normal conversation like “um, “hh,“eh,“en... dont help as a feature for the Bernoulli model. Besides, some high-frequency words are also not helpful as features for the model. For example, we notice word like know appears around 130 times in each label, but it doesnt

help for classification at all in this case. Because once you have the word "know" with the same likelihood for each label, this word means nothing in the contribution to our classification. So we may need to remove this kind of word before training. So under this idea, we will reduce the size of our vocabulary, which will improve the accuracy as well as the training and testing time of our model.

Third, as we have discussed, the **frequency of the word** is also the reason why multinomial performs better. Because Bernoulli does not consider the frequency of a term in a document. If the word is really important for the classification, we will lose this feature in Bernoulli model. For example, if the word "wage" appears 30 times in a single document that talks about salary. But we will treat the word "wage" the same as other general words like "good", "story","okay" in our Bernoulli model. So this is unreasonable criteria in this circumstance.

Finally, according to the heatmap of the confunsion matrix, it is easier to see which topics are likely to be confused with the other topic based on the colors. For example, in confusion matrix of multinomial model, we could see the color of row 23 which represents topic 23, and column 17, which represents topic 17, are easily to be confused since the color of col 17 is deeply compared with other cells, and the probability is 0.25, which is also higher than others. The accurate classification probability for topic is 0.9. So the color of the heatmap could reflect the importance of the cell in a matrix.

Tune on K

As we know, in the construction of the model, we use the smoothing factor $k = 1$ as default. However, the result of Bernoulli model in part2.2 indicates the accuracy is not ideal, it is only around 60% accuracy. And if we set the smoothing factor $k = 0.1$ in this case, we could see the accuracy is improved to 86% now. Besides, the accuracy of multinomial is also improved from 83% to 89%. Here is analysis of these two results.

First, the smoothing factor k is used because we want to avoid zero probability problem in the training process. Because some counting of the word will be zero for a specific label, so we need to add a numeric value k to avoid this problem. Second, the smoothing factor actually represent the belief that we put on the words we have already seen. The impact of the smoothing factor is that it would give some probability for the rarely appeared word and the word we have never seen. So the value of k reflects our belief about the amount of the words we have never seen. If we think we have seen almost all the words in the training process(i.e. vocabulary), we could set the k smaller than default. Back to part2.2, because we have 10244 documents trained, so we could be confident to say that we have seen almost all of the words(i.e. we have totally 52993 words in vocabulary) So we adjust the smoothing factor k to smaller in order to reflect this confidence. We tested several cases and found out that the overall accuracy of Bernoulli model is 86% when $k = 0.1$, 78.9% when $k = 0.20$. And it is much higher than 60% when $k = 1$. So it proves that our analysis on the smoothing factor is correct. So we use the smoothing factor $k = 0.1$ finally. [**Bonus Point**]

3.7 Extra Credit

3.7.1 Bag-of-Words visualization world cloud map

Figure 33: Word of Bag for movie class 0

Figure 34: Word of Bag for movie class 1

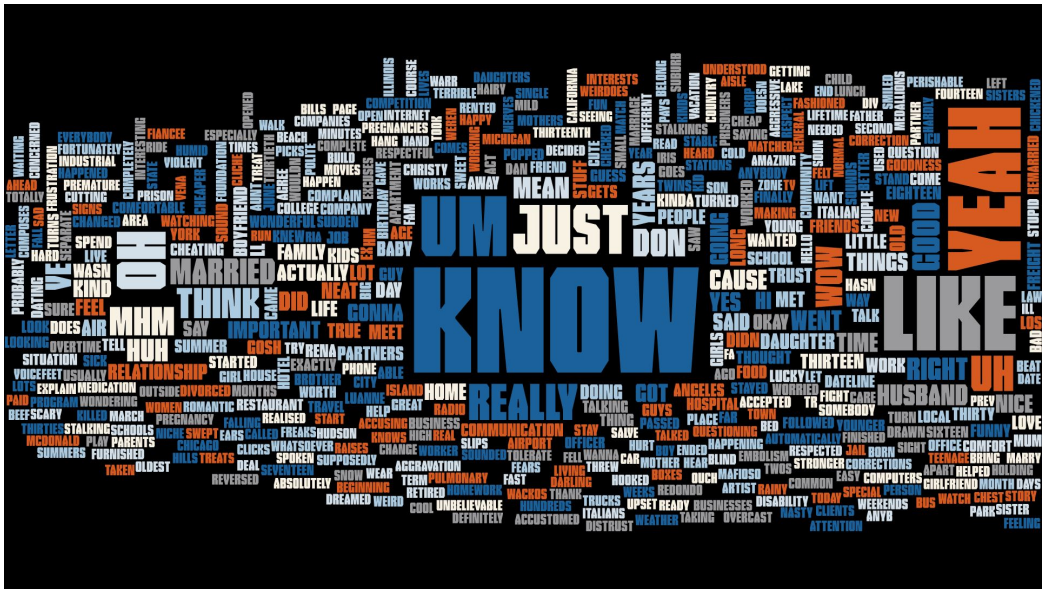


Figure 35: Word of Bag for two topics class 0

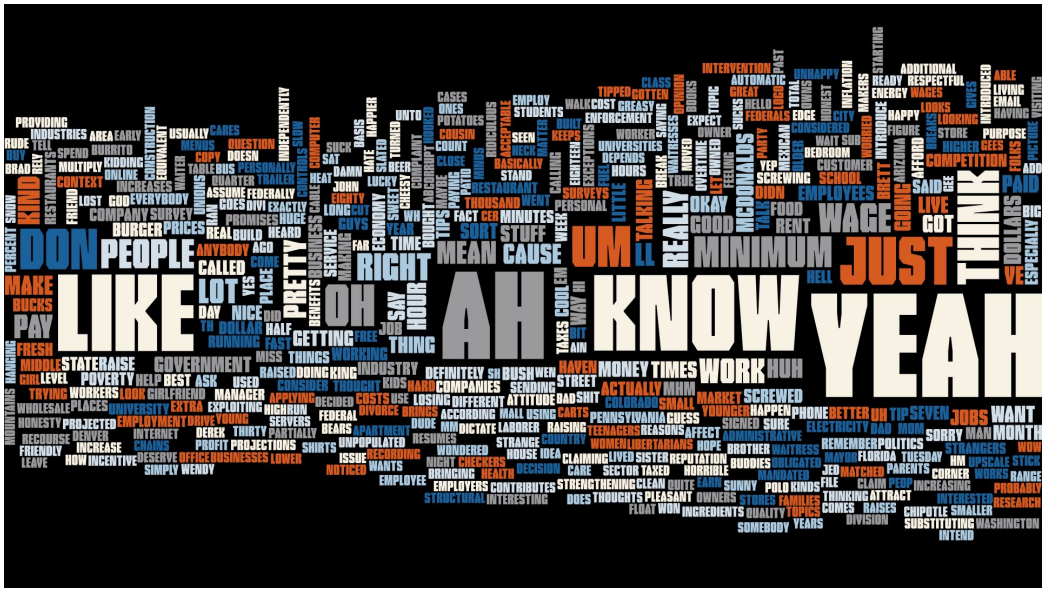


Figure 36: Word of Bag for for two topics class 1

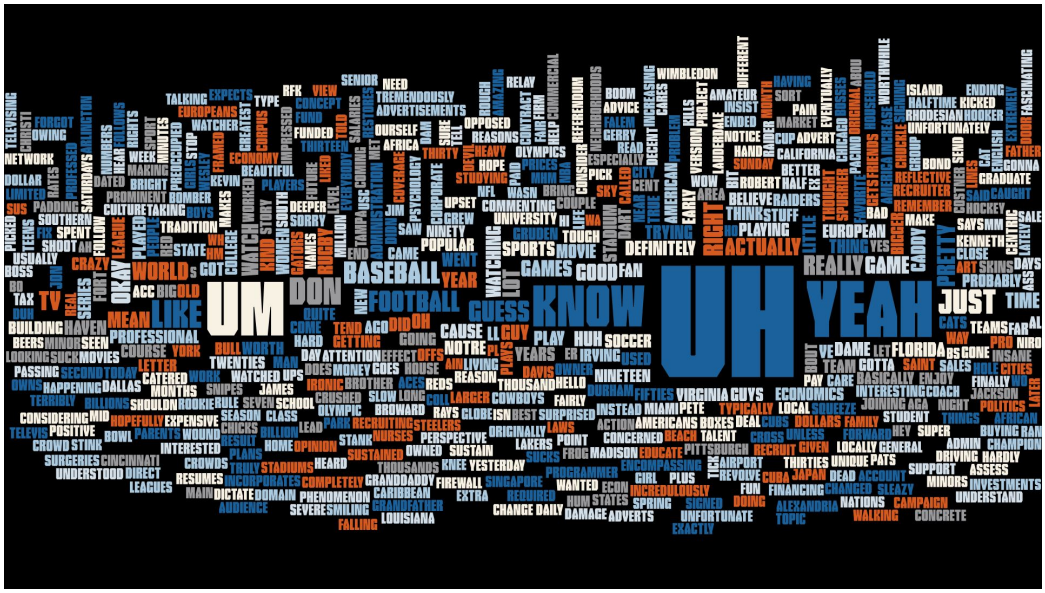


Figure 37: Word of Bag for 40 topics class 0

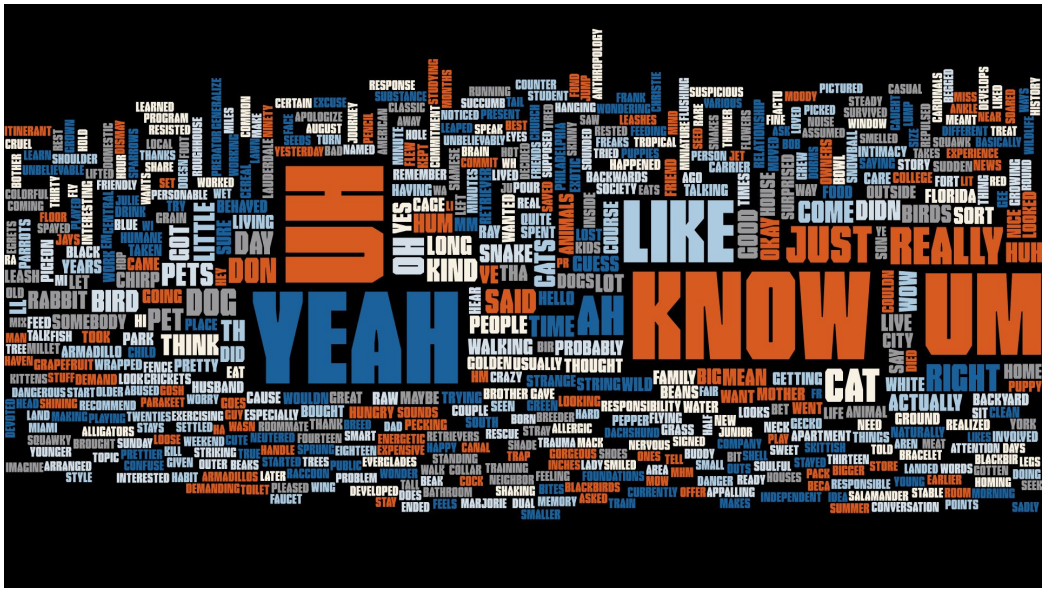


Figure 38: Word of Bag for 40 topics class 1

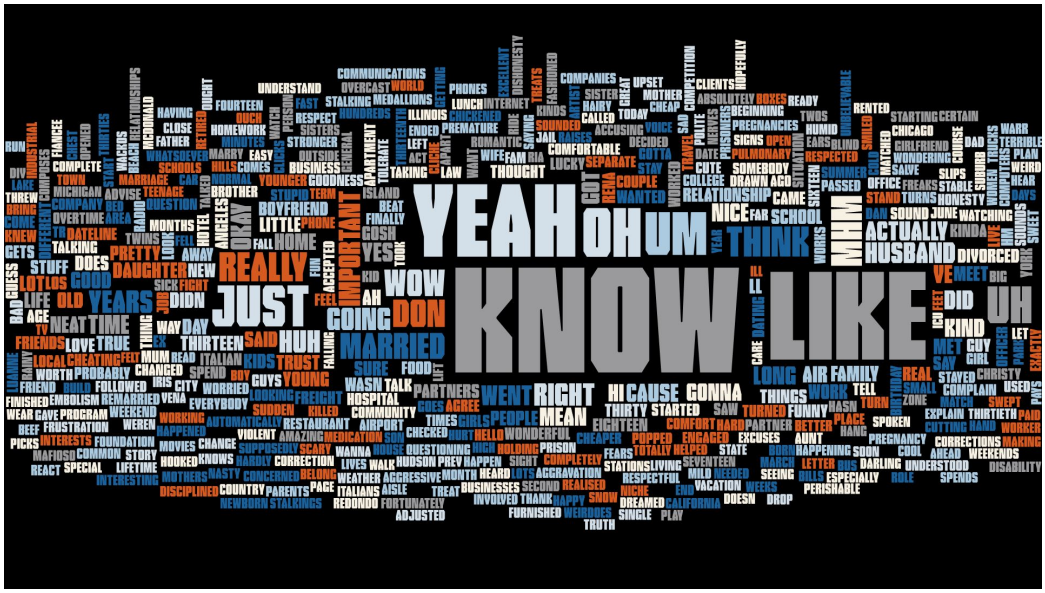


Figure 39: Word of Bag for 40 topics class 2

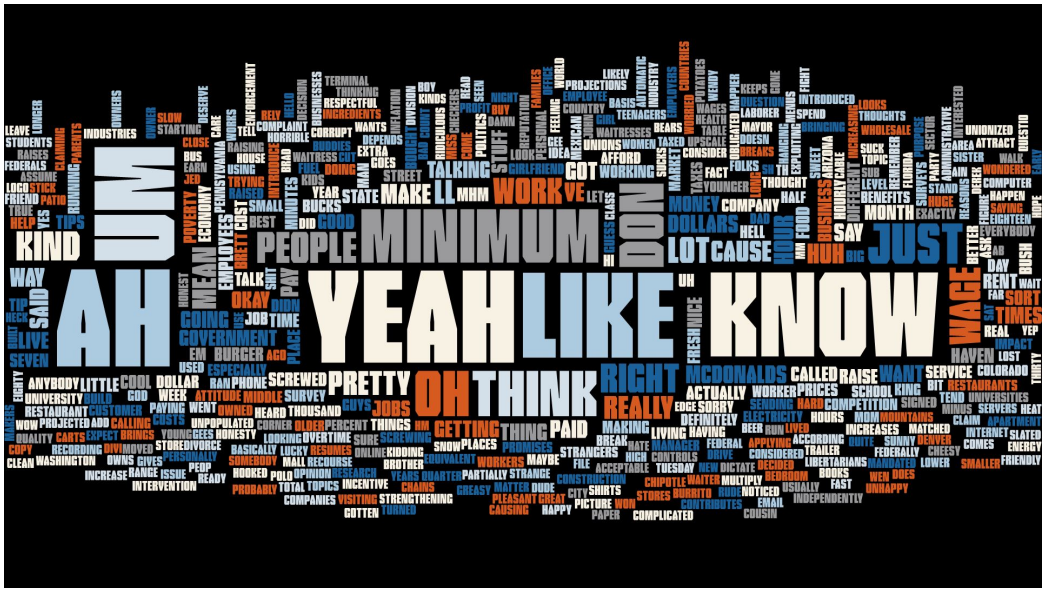


Figure 40: Word of Bag for 40 topics class 3

Figures above shows the vitalized word of bag for movie, two topics, and forty topics corpus. The word clouds are generated with maximum of 2000 words, with highest frequency shown in largest font size. This means that the word clouds show the top 2000 most used words in the file, with the font size represents the frequency of words inside the word of bags. From the word clouds shown above, we can clearly see the most frequently used words in each topics for each class. For part 2.2 40 topics results, only class 0 to class 3 wordcloud are shown in the figure above.

3.7.2 Confusion matrix visualization heatmaps

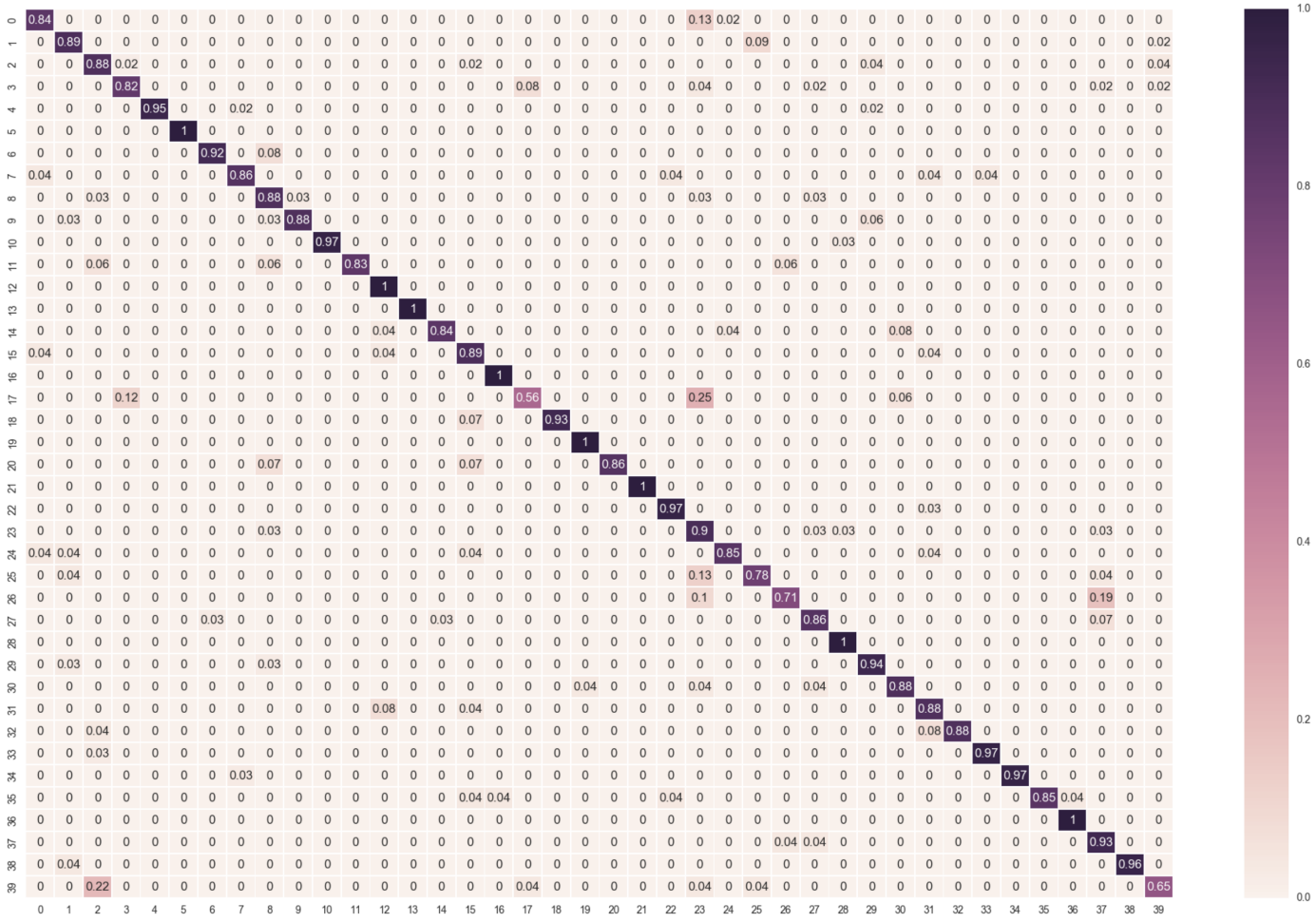


Figure 41: Confusion matrix of 40 topics multinomial model

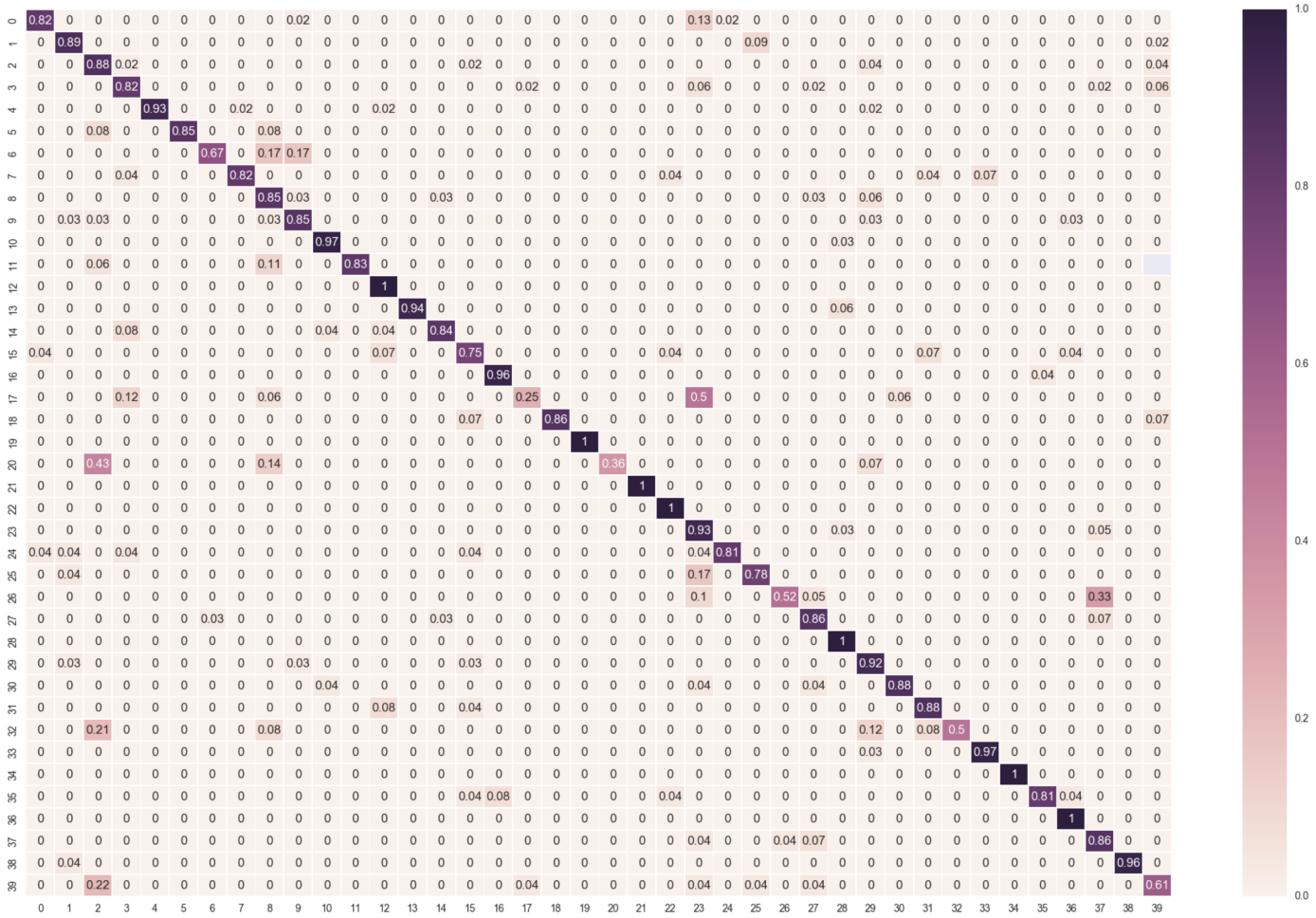


Figure 42: Confusion matrix of 40 topics Bernoulli model

We used the package seaborn in python to achieve the heatmap visualization of our confusion matrix. Heatmap is a kind of graphical representation of data where the individual values contained in a matrix are represented as colors. So it is more clear to see which cell has more weights according to its color in the cell. It is more intuitive to view the confusion between different topics in heatmap.

3.7.3 Lemmatization for the vocabulary

The corpora of 40 topic is not pre-processed since we notice that some words appear in our vocabulary like "um", "eh", "yeah", etc. Intuitively speaking, some words are not helpful in the classification of the topics. There are three categories for these words: 1) First, some words are frequently used in our daily conversation almost for both topics, no matter what are the people talking about. For example, "like", "am", "oh". We often use these basic words in our daily conversation. These words are called stop words in Natural Language Processing (NLP), so we should remove these words in our vocabulary first. 2) Second, some words are rarely used in the conversation, maybe some of them only appear once in the whole training data but we still need to add them into our vocabulary. Therefore, these words are not helpful for our classification, either. 3) Third, some words may have the same meaning but in different forms. For example, "car's", "cars", "cars'" they all express the meaning of "car", so we could classify these words into one group and use the lemmatization to achieve the target.

We use the package nltk in Python to help us to do the lemmatization and find the rarely words and stop words. In nltk, we could import the corpus and functions as follows:

```
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk import pos_tag
from nltk.corpus import wordnet
```

And these are the stops words defined in English field of nltk:
{'ourselves', 'hers', 'between', 'yourself', 'but', 'again', 'there', 'about', 'once', 'during', 'out', 'very', 'having', 'with', 'they', 'own', 'an', 'be', 'some', 'for', 'do', 'its', 'yours', 'such', 'into', 'of', 'most', 'itself', 'other', 'off', 'is', 's', 'am', 'or', 'who', 'as', 'from', 'him', 'each', 'the', 'themselves', 'until', 'below', 'are', 'we', 'these', 'your', 'his', 'through', 'don', 'nor', 'me', 'were', 'her', 'more', 'himself', 'this', 'down', 'should', 'our', 'their', 'while', 'above', 'both', 'up', 'to', 'ours', 'had', 'she', 'all', 'no', 'when', 'at', 'any', 'before', 'them', 'same', 'and', 'been', 'have', 'in', 'will', 'on', 'does', 'yourselves', 'then', 'that', 'because', 'what', 'over', 'why', 'so', 'can', 'did', 'not', 'now', 'under', 'he', 'you', 'herself', 'has', 'just', 'where', 'too', 'only', 'myself', 'which', 'those', 'i', 'after', 'few', 'whom', 't', 'being', 'if', 'theirs', 'my', 'against', 'a', 'by', 'doing', 'it', 'how', 'further', 'was', 'here', 'than'}

So we based on these stop words to remove the daily used words. Then for the rare words, we remove them according to their frequency in the entire documents. Third, for the lemmatization, we re-write a function to achieve it. We first use *pos_tag*() function to find the correct form of the word, like "VERB", "NOUN", "ADJ", etc. Then we call the lemmatize() to lemmatize the words to its basic form. e.g. "operating" will be converted to "operate" after calling this function. Therefore, this is lemmatization process we used in for our raw vocabulary. For the outcome, the size of raw vocabulary is 52993. After going through the mentioned processing, we get around 40800 words. Although maybe it is still a larger vocabulary, it is more efficient in the training. And our time spent on training is also reduced.

We use the multinomial case as an example to show the impact of the lemmatization:

```
Before lemmatization: k = 1
                      Overall accuracy: 0.838
After lemmatization:  k = 1
                      Overall accuracy: 0.844
```

Therefore, lemmatiation could improve the accuracy of our model in this case.

3.7.4 tf-idf weight

tf-idf weight was implemented in order to increase the performance of both classifiers. tf-idf is the product of two statistics, term frequency and inverse document frequency. For Term frequency, logarithmically scaled frequency was being used in our implementation. The calcuating equation is as following:

$$tf(t,d) = 1 + logf_t, d, orzerooff_t, diszero \tag{13}$$

The term frequency is simply the logarithmically scaled of raw frequency of a term in a document.
Secondlly, the inverse document frequency is a measure of how much information the word provides. In another words, it shows if a term is common or rear across all documents. The calculation of inverse document frequency is as following:

$$id(t,D) = log(N/|1 + \{d \in D : t \in d)\}|) \tag{14}$$

with
N : total number of documents in the corpus $N = |D|$
$|\{d \in D : t \in d\}|$ indicate the number of documents where the term t appears. 1 is added to this to avoid

divide by zero situation.
Lastly, the tf-id frequency is being calculated as following:

$$tfidf(t, d, D) = tf(t, d) * idf(t, D) \tag{15}$$

The tf-id frequency is calculated before classifier training process, and being used as weight for all vocabulariy-class paris during classifier testing process. However, after applying the tf-id frequency to both movie and two topic corpus, the result classification accuracy dropped slightly with 1 to 2 percentages. The multinomial naive bayes model with two topics accuracy drops from 91% to 89%. The multinomial naive bayes model with movies accuracy drops from 90% to 88.6%. This might due to the common words are being eliminated from the original input training files, and the inverse document frequency in the tf-id frequency calculation leads to a drop in the performance. If the input training documents contains common vocabularies such as "like", "the", the addition of tf-id frequency could increase the performance of both classifiers.

4 Work Distribution

4.1 Joint Efforts

- MP report

4.2 Yasser Shalabi

- Part 1 implementation
- Part 1 Extra Credit

4.3 Chongxin Luo

- Part 2 Extra Credit
- Strong mental support to teammates

4.4 Haoran Wang

- Part 2 implementation

A
Part One Results

A.1 Three Unit Results

A.1.1 Accuracy and Confusion Matrices

```
1x1 Test:                    binary_digits_1x1_OLap=False
    Best Smoothing Factor:   0.020202
    Overall Accuracy:        77.800000
```

Recall Matrix

Digit	0	1	2	3	4	5	6	7	8	9
Accuracy	83.33	94.44	78.64	83.0	75.7	72.83	79.12	70.75	57.28	83.0

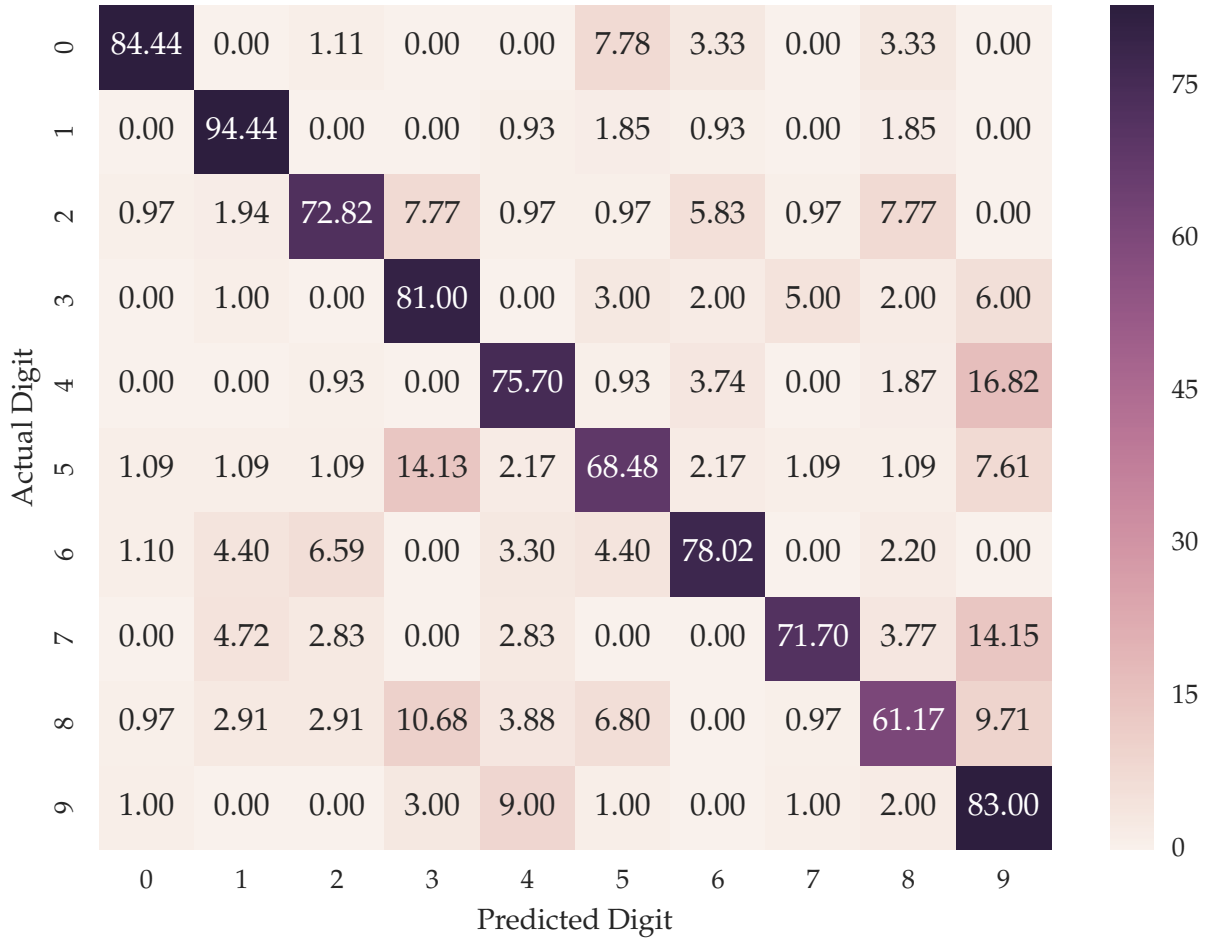
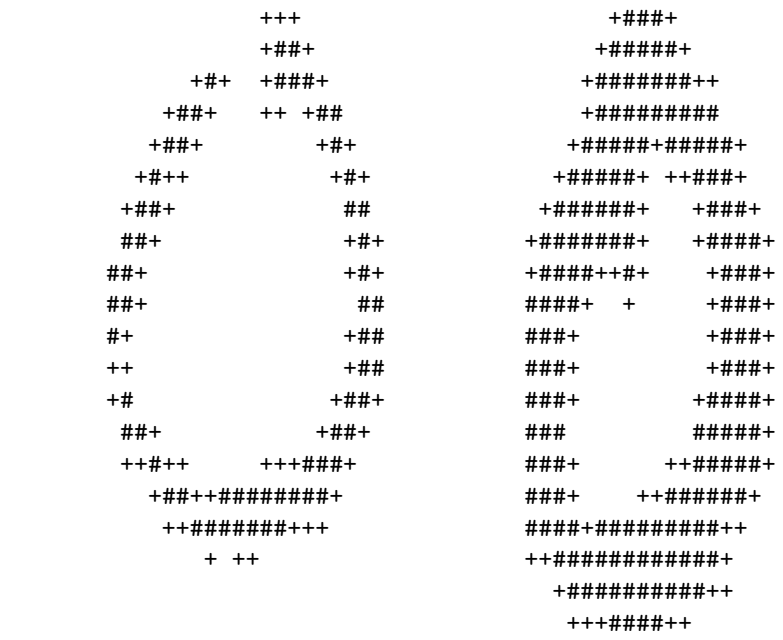


Figure 43: 3-Unit 1x1 Confusion Matrix

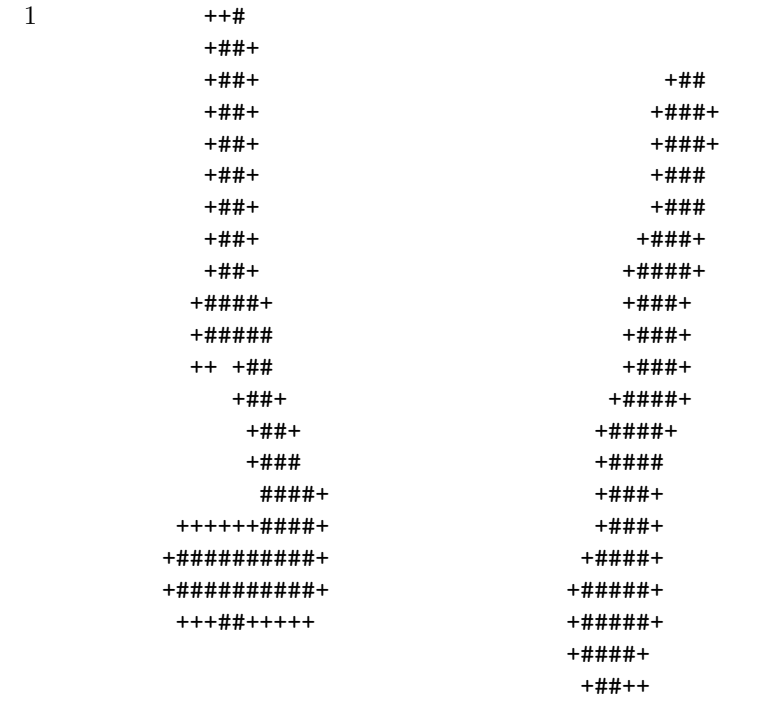
A.1.2 Samples With Best/Worst Posterior Probabilities

Per class samples worst (left side) and best (right side) posterior probabilities

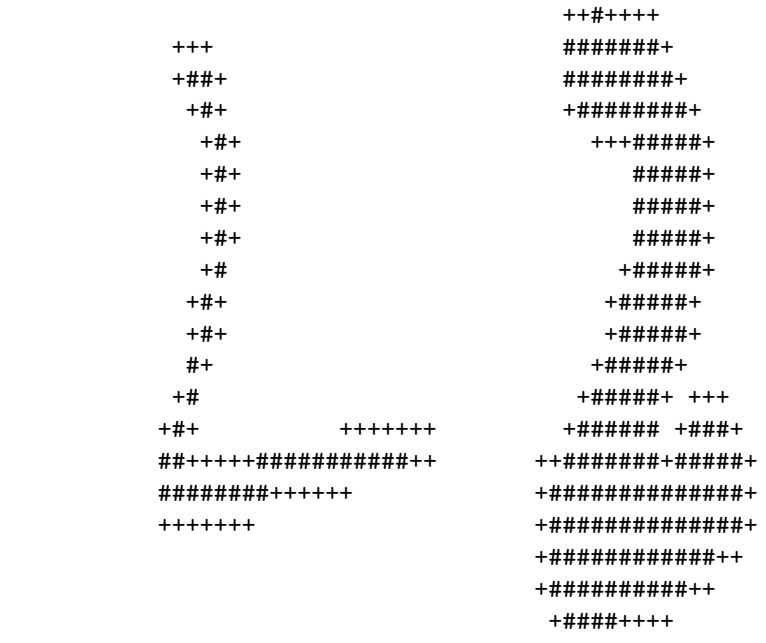
0



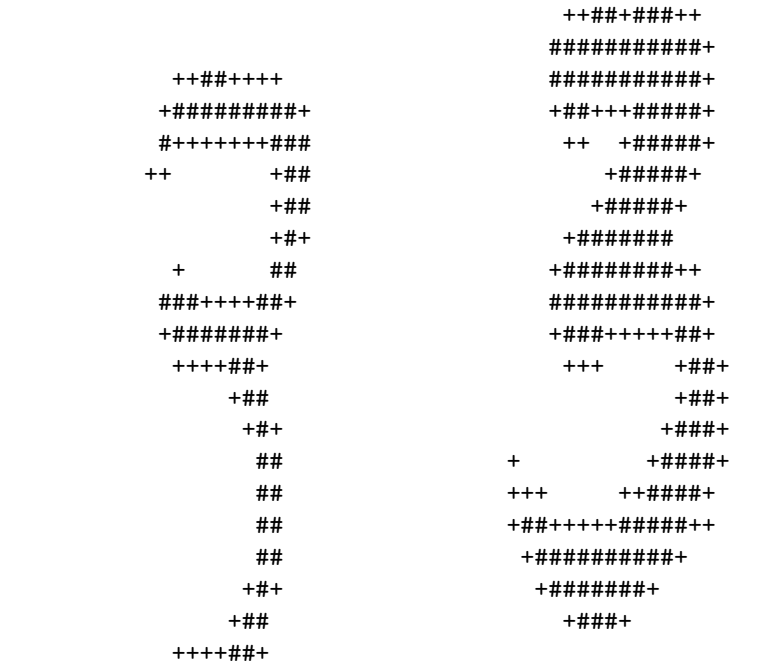
1



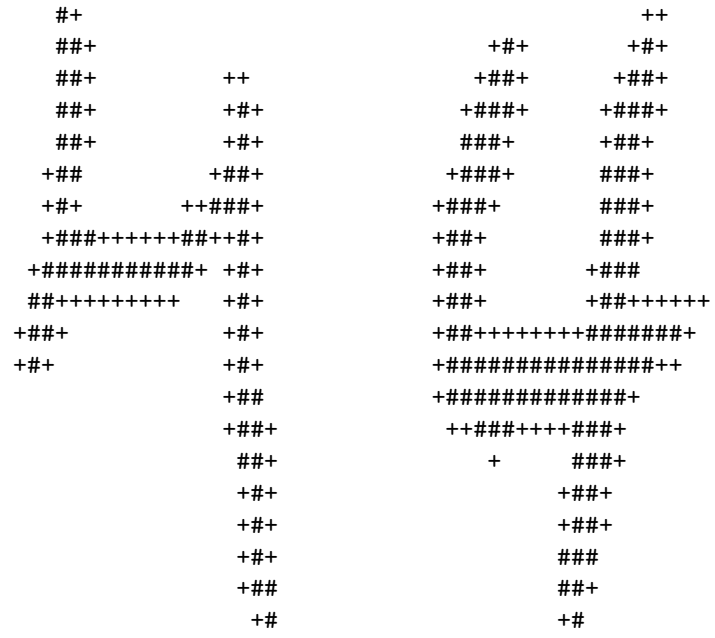
2



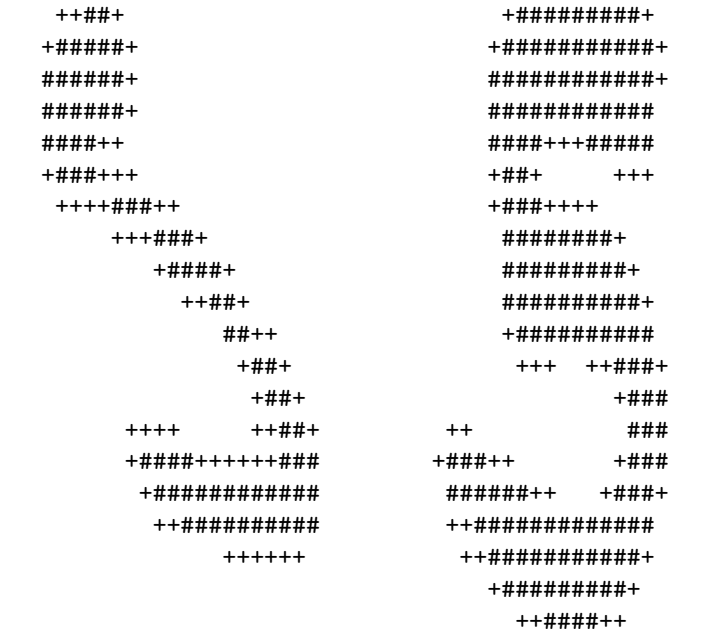
3



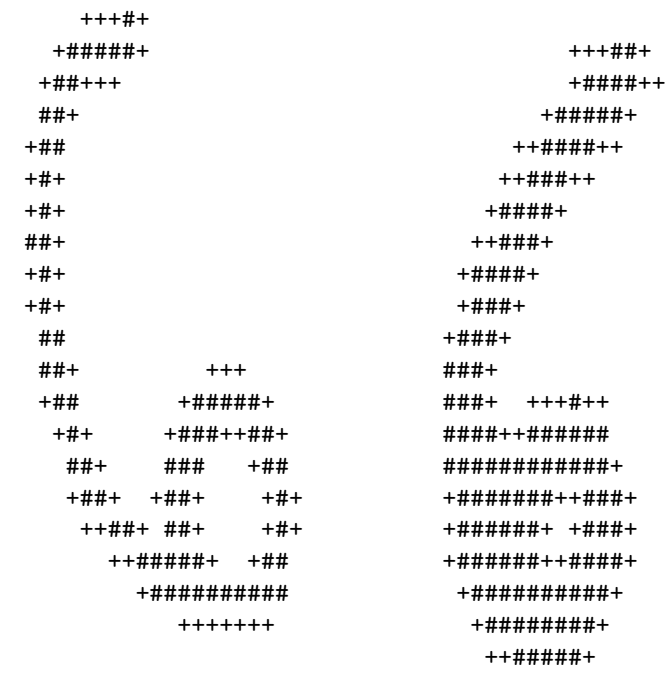
4



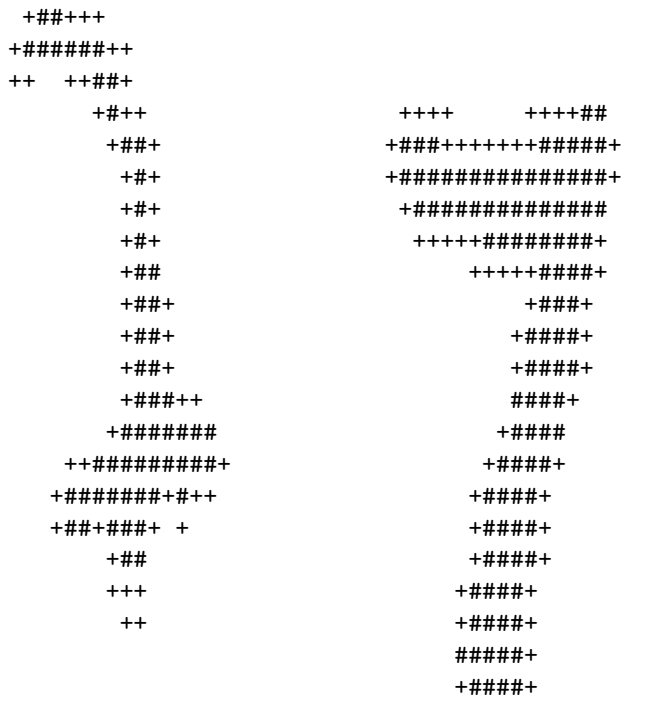
5



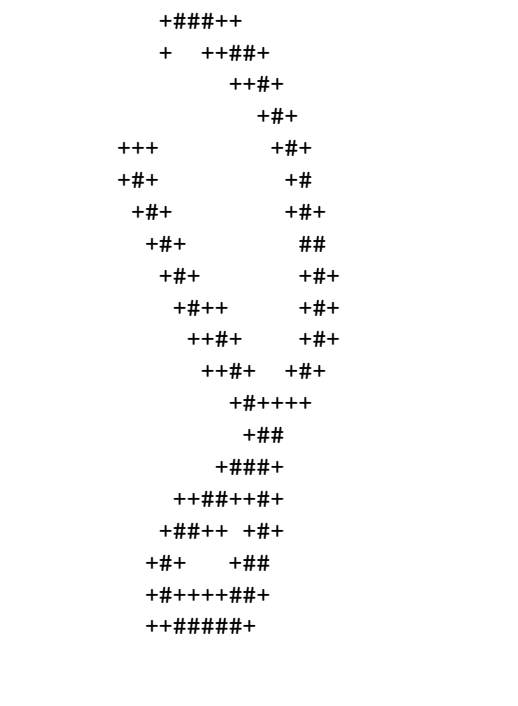
6



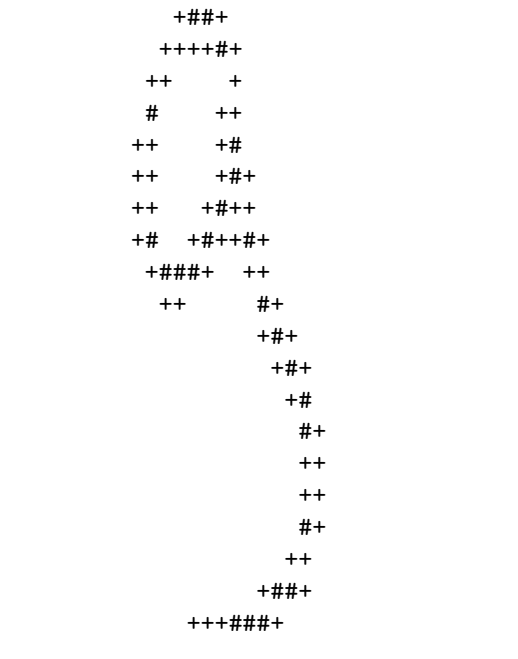
7



8



9



A.2 Four Unit Results

A.2.1 Accuracies (Non-Overlapping Groupings)

```
2x2 Test:                    binary_digits_2x2_OLap=False
    Best Smoothing Factor:   0.020202
    Overall Accuracy:        85.500000
    total times
    End-End Time:            0.434761 seconds
    Train  Time:             0.000000 seconds
    Inference  Time:         0.000000 seconds

                          Recall Matrix
    Digit          0      1      2     3      4      5      6      7     8     9
    Accuracy   94.44  95.37  87.38  88.0  85.05  80.43  90.11  77.36  76.7  81.0
```

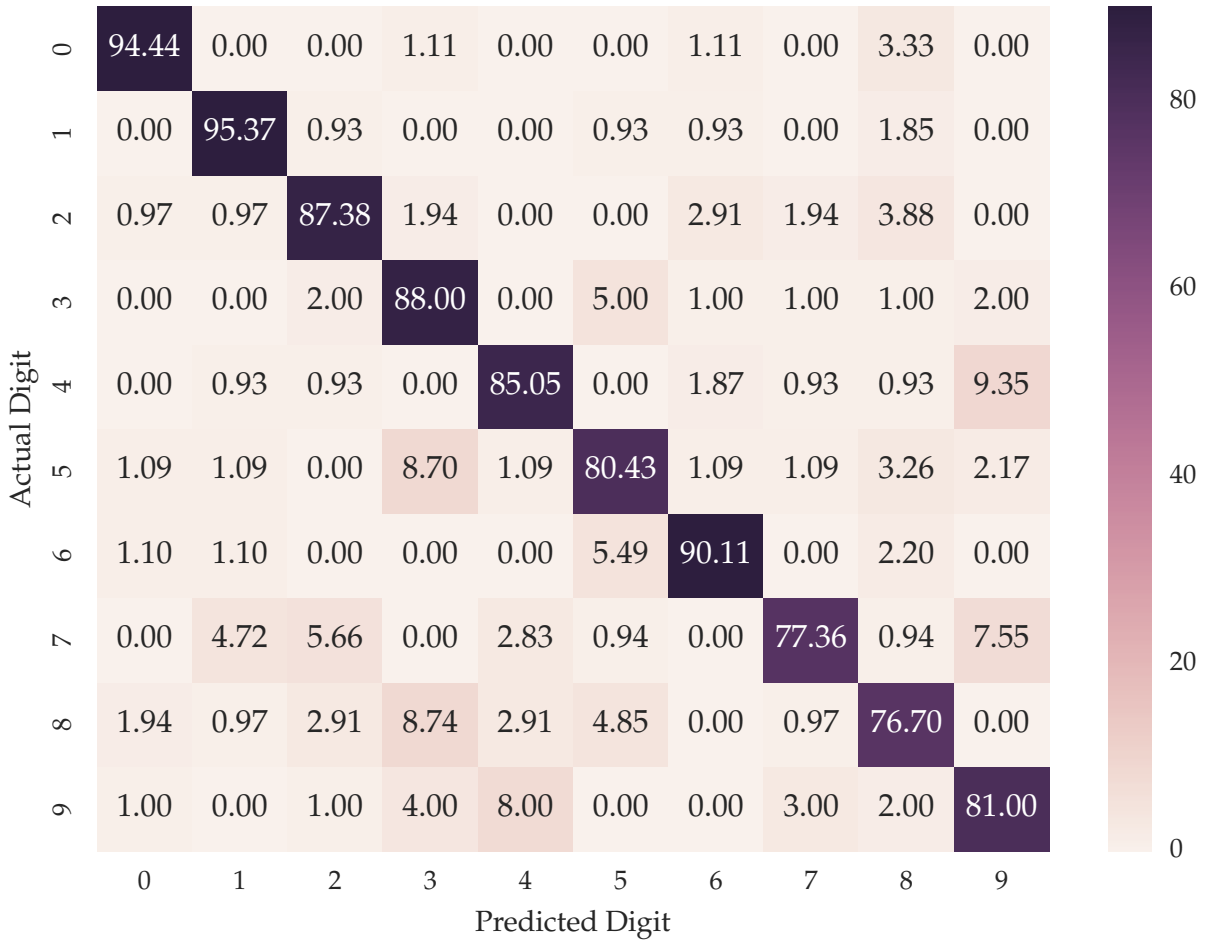


Figure 44: 4-Unit 2x2 Confusion Matrix

```
2x4 Test:                    binary_digits_2x4_OLap=False
    Best Smoothing Factor:   0.020202
    Overall Accuracy:        86.600000

                             Recall Matrix
    Digit          0     1      2     3      4      5      6      7      8     9
    Accuracy   95.56  96.3  87.38  88.0  89.72  81.52  89.01  74.53  78.64  86.0
```

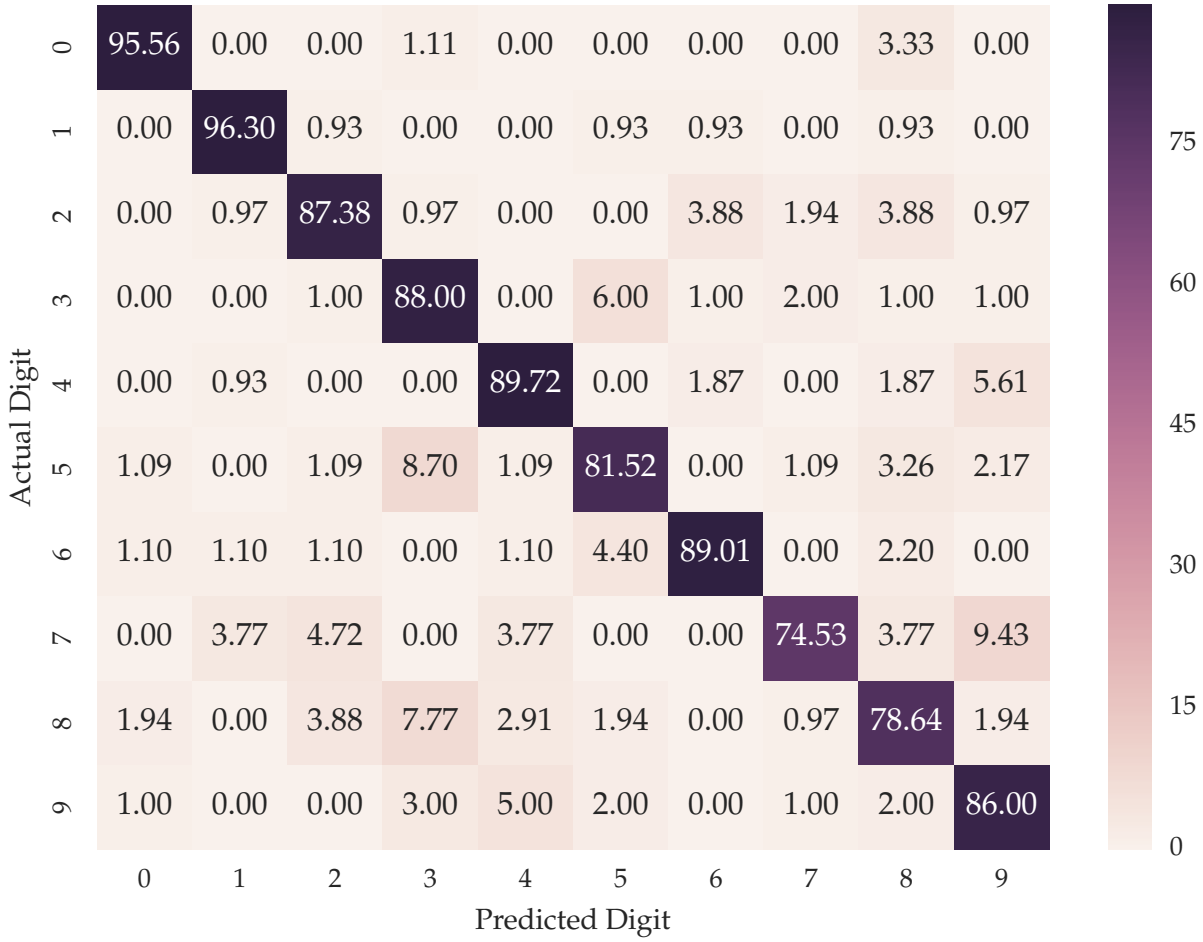


Figure 45: 4-Unit 2x4 Confusion Matrix

```
4x2 Test:                    binary_digits_4x2_OLap=False
    Best Smoothing Factor:   0.020202
    Overall Accuracy:        87.600000
    total times
    End-End Time:            4.584643 seconds

                             Recall Matrix
    Digit          0     1      2     3      4      5      6      7      8     9
```

Accuracy	96.67	96.3	92.23	89.0	89.72	78.26	87.91	76.42	85.44	84.0

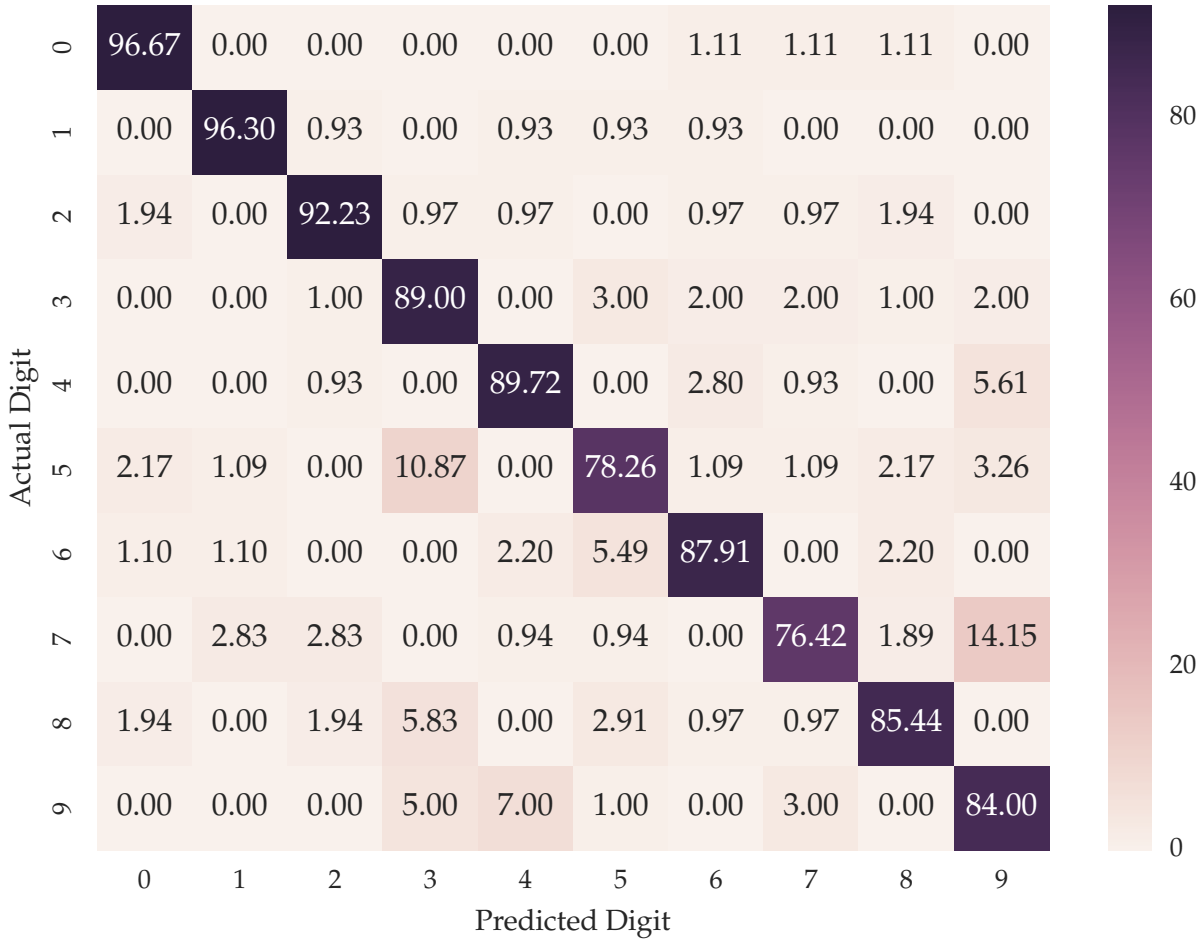


Figure 46: 4-Unit 2x4 Confusion Matrix

4x4 Test: binary_digits_4x4_OLap=False
Best Smoothing Factor: 0.020202
Overall Accuracy: 84.300000
total times
End-End Time: 1326.841396 seconds
Train Time: 1199.995430 seconds
Inference Time: 126.647667 seconds

Recall Matrix

Digit	0	1	2	3	4	5	6	7	8	9
Accuracy	97.78	98.15	86.41	92.0	95.33	64.13	89.01	74.53	67.96	77.0

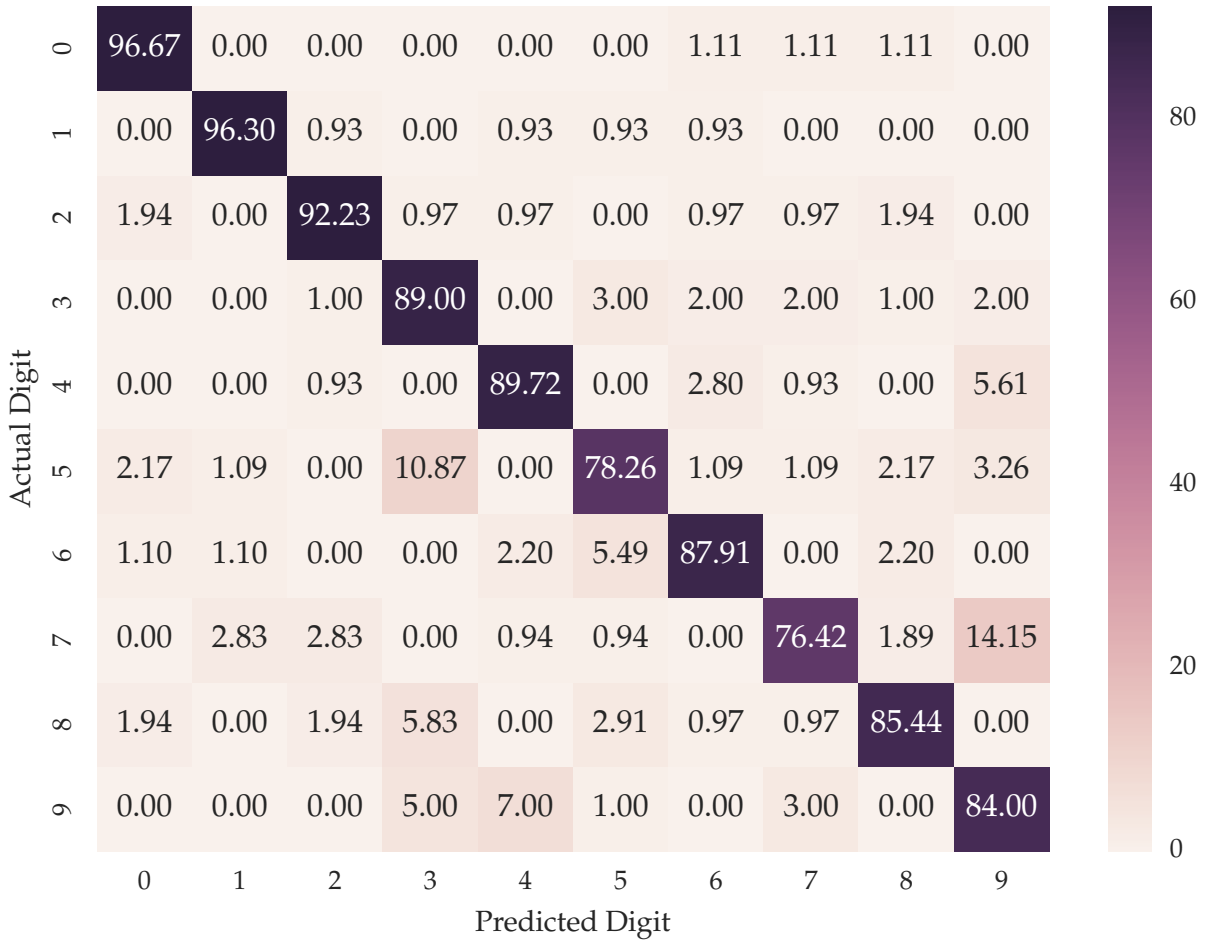


Figure 47: 4-Unit 4x4 Confusion Matrix

A.2.2 Accuracies (Overlapping Groupings)

```
2x2 Test:                    2x2_OLap=True
    Best Smoothing Factor:   0.020202
    Overall Accuracy:        87.300000
    Evaluation Time:         93.5830481052 seconds

                             Recall Matrix
    Digit          0     1      2     3      4      5      6      7      8     9
    Accuracy   94.44  96.3  89.32  93.0  87.85  82.61  86.81  77.36  80.58  85.0
```

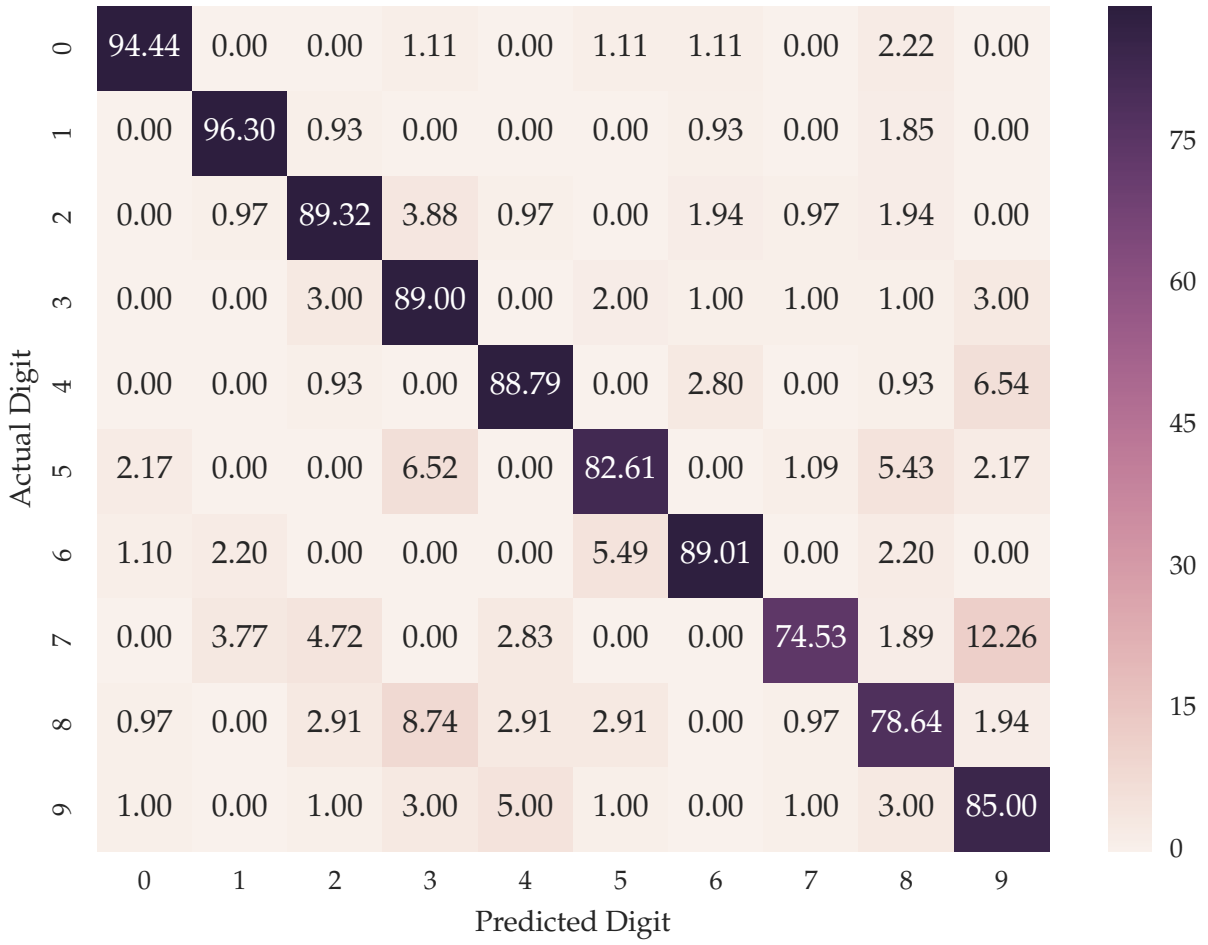


Figure 48: 4-Unit 2x2 Confusion Matrix (With Overlap)

```
2x3 Test:                    2x3_OLap=True
    Best Smoothing Factor:   0.020202
    Overall Accuracy:        88.800000
    Evaluation Time:         310.724170923 seconds

                          Recall Matrix
    Digit          0     1      2     3      4      5      6      7      8     9
    Accuracy   95.56  96.3  89.32  90.0  92.52  86.96  91.21  77.36  79.61  86.0
```

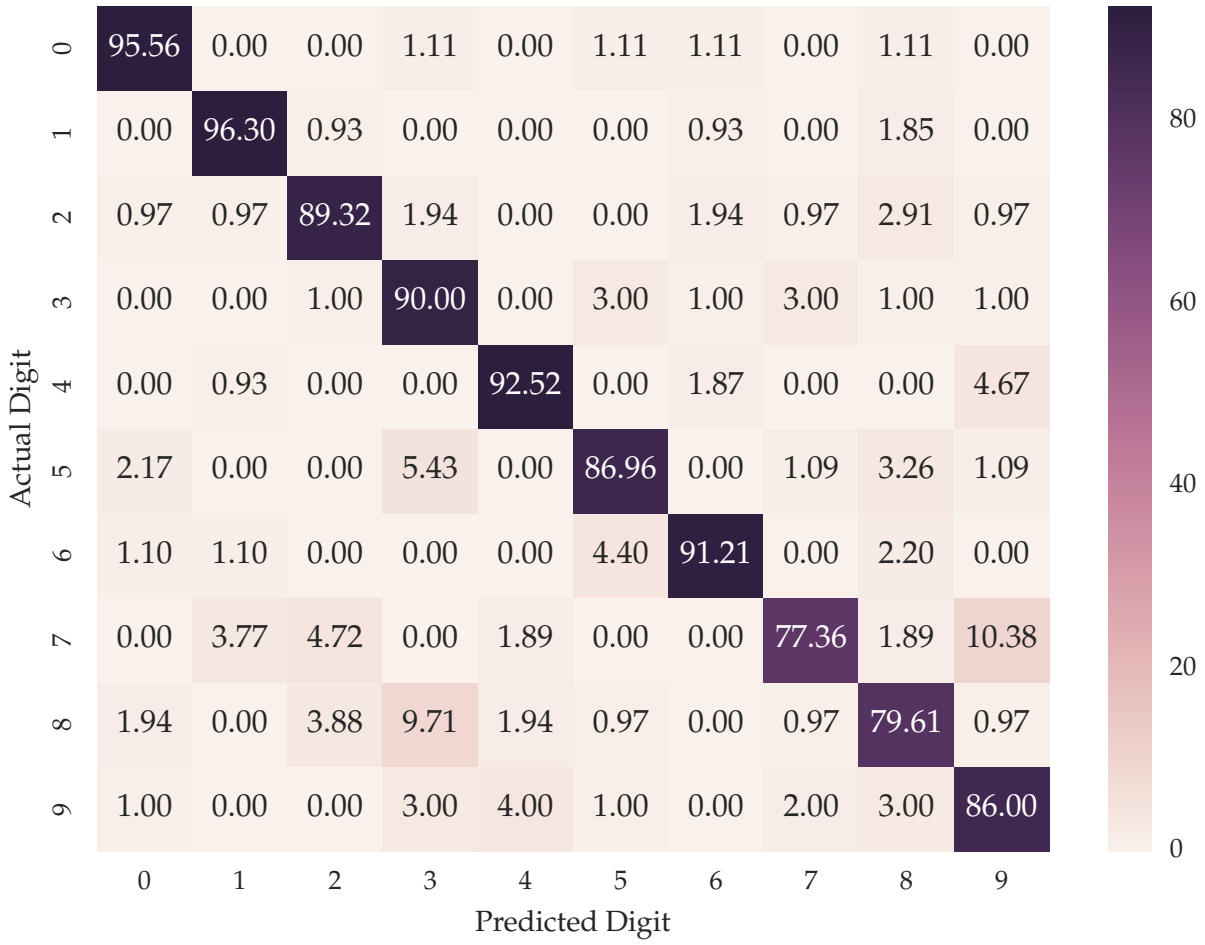


Figure 49: 4-Unit 2x3 Confusion Matrix (With Overlap)

```
2x4 Test:                    binary_digits_2x4_OLap=True
    Best Smoothing Factor:   0.020202
    Overall Accuracy:        88.500000
    total times
    End-End Time:            16.818888 seconds

                          Recall Matrix
    Digit          0     1      2     3      4      5      6      7      8     9
    Accuracy   95.56  96.3  87.38  90.0  93.46  86.96  91.21  76.42  81.55  87.0
```

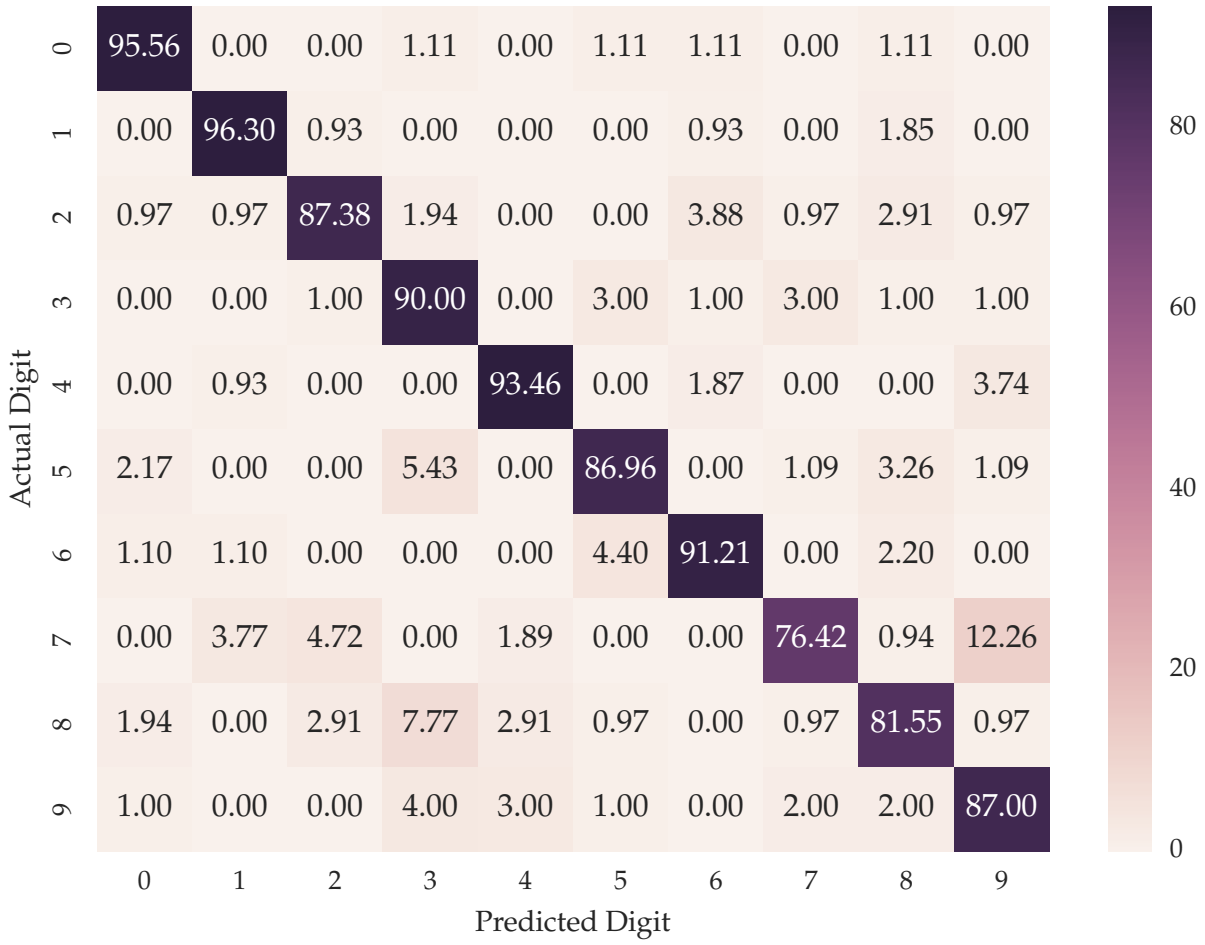


Figure 50: 4-Unit 2x4 Confusion Matrix (With Overlap)

```
3x2 Test:                    3x2_OLap=True
    Best Smoothing Factor:   0.020202
    Overall Accuracy:        89.000000
    Evaluation Time:         464.943305016 seconds

                             Recall Matrix
    Digit          0      1      2     3      4      5      6     7      8     9
    Accuracy   97.78  97.22  90.29  93.0  93.46  81.52  89.01  78.3  82.52  87.0
```

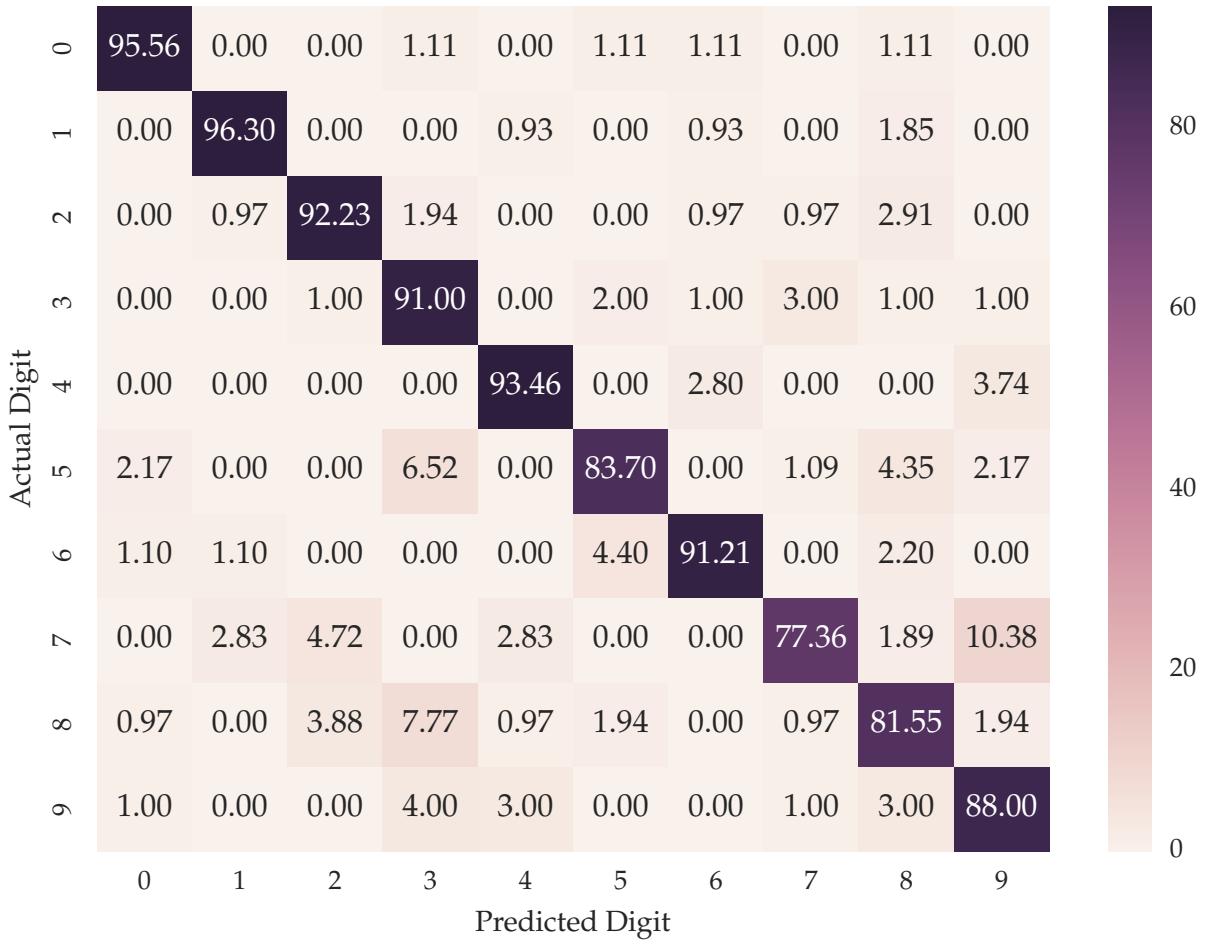


Figure 51: 4-Unit 3x3 Confusion Matrix (With Overlap)

```
3x3 Test:                    binary_digits_3x3_OLap=True
    Best Smoothing Factor:   0.020202
    Overall Accuracy:        88.800000
    total times
    End-End Time:            76.617355 seconds
    Train  Time:             69.657620 seconds
    Inference  Time:         6.923704 seconds

                             Recall Matrix
    Digit          0      1      2     3      4      5      6     7      8     9
    Accuracy   97.78  97.22  92.23  90.0  94.39  81.52  90.11  78.3  79.61  87.0
```

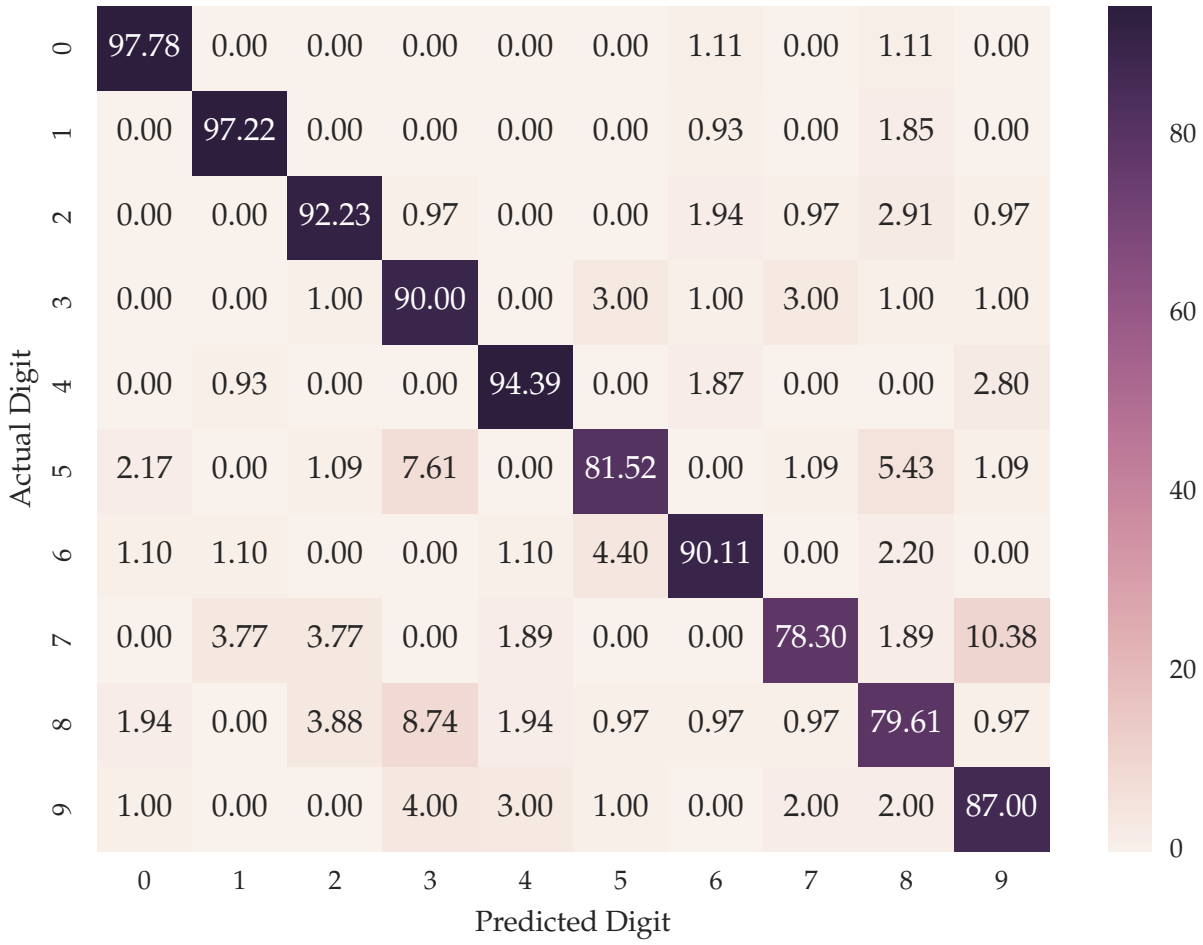


Figure 52: 4-Unit 3x3 Confusion Matrix (With Overlap)

```
4x2 Test:                    binary_digits_4x2_OLap=True
    Best Smoothing Factor:   0.020202
    Overall Accuracy:        89.400000
    total times
    End-End Time:            15.935308 seconds
    Train  Time:             0.000000 seconds
    Inference  Time:         0.000000 seconds

                          Recall Matrix
    Digit         0      1      2     3      4      5      6      7      8     9
    Accuracy  96.67  97.22  92.23  91.0  93.46  82.61  91.21  80.19  82.52  87.0
```

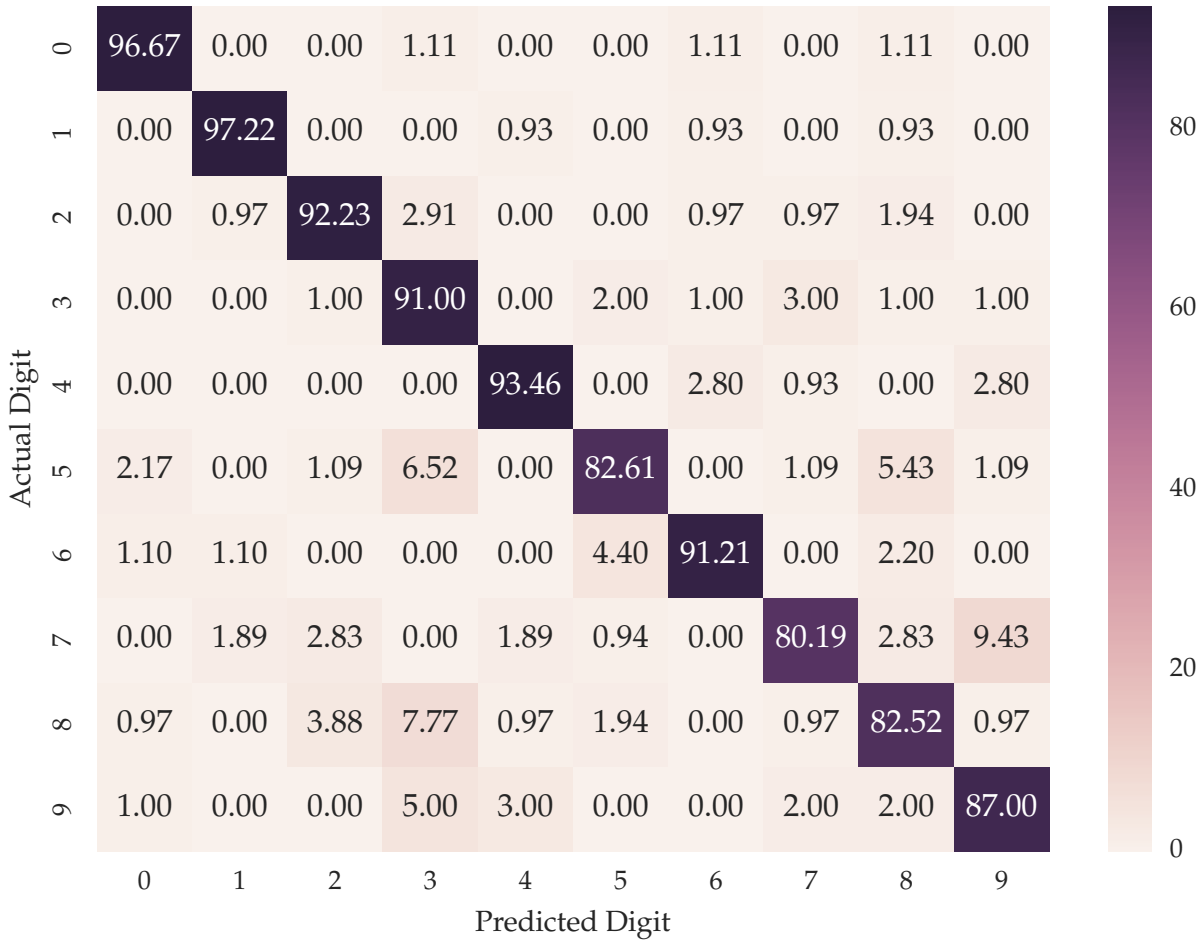


Figure 53: 4-Unit 4x2 Confusion Matrix (With Overlap)

```
4x4 Test:                    4x4_OLap=True
    Best Smoothing Factor:   0.020202
    Overall Accuracy:        89.500000
    Evaluation Time:         6384.94159484 seconds

                             Recall Recall Rate
    Digit          0      1      2     3      4      5      6      7     8     9
    Accuracy   97.78  98.15  90.29  93.0  95.33  79.35  87.91  82.08  83.5  87.0
```

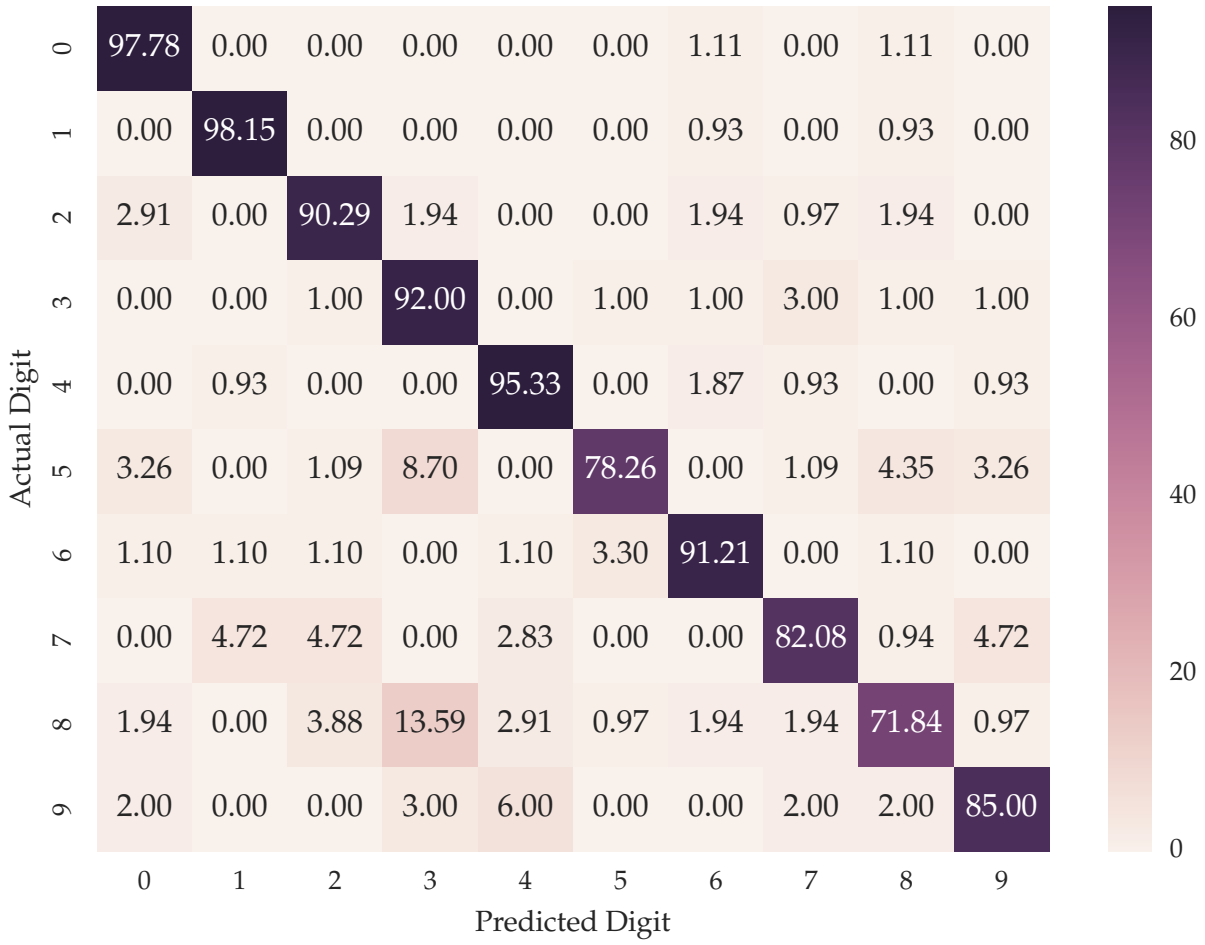


Figure 54: 4-Unit 4x4 Confusion Matrix (With Overlap)

A.3 Extra Credit Ternary Mode Operation

A.3.1 Single Pixel Results

```
1x1 Test:                    1x1_OLap=False
    Best Smoothing Factor:   0.020202
    Overall Accuracy:        77.900000
    Evaluation Time:         35.9816241264 seconds

                             Accuracy Matrix
    Digit          0      1      2     3     4      5      6     7      8     9
    Accuracy   84.44  94.44  77.67  80.0  75.7  68.48  82.42  71.7  62.14  82.0

                             Confusion Matrix
    Predicted Digit     0     1     2     3     4     5     6     7     8     9
    Actual Digit
    0                0.00  0.00  1.11  0.00  0.00  7.78  3.33  0.00  3.33  0.00
    1                0.00  0.00  0.93  0.00  0.00  1.85  0.93  0.00  1.85  0.00
```

2	0.97	1.94	0.00	5.83	0.97	0.00	5.83	1.94	4.85	0.00
3	0.00	0.00	0.00	0.00	0.00	5.00	2.00	6.00	2.00	5.00
4	0.00	0.00	0.00	0.00	0.00	0.93	3.74	0.93	1.87	16.82
5	1.09	0.00	1.09	15.22	3.26	0.00	1.09	1.09	2.17	6.52
6	0.00	2.20	3.30	0.00	4.40	4.40	0.00	0.00	3.30	0.00
7	0.00	4.72	3.77	0.00	2.83	0.00	0.00	0.00	2.83	14.15
8	0.97	2.91	2.91	11.65	3.88	6.80	0.00	0.97	0.00	7.77
9	1.00	0.00	0.00	2.00	9.00	1.00	0.00	2.00	3.00	0.00

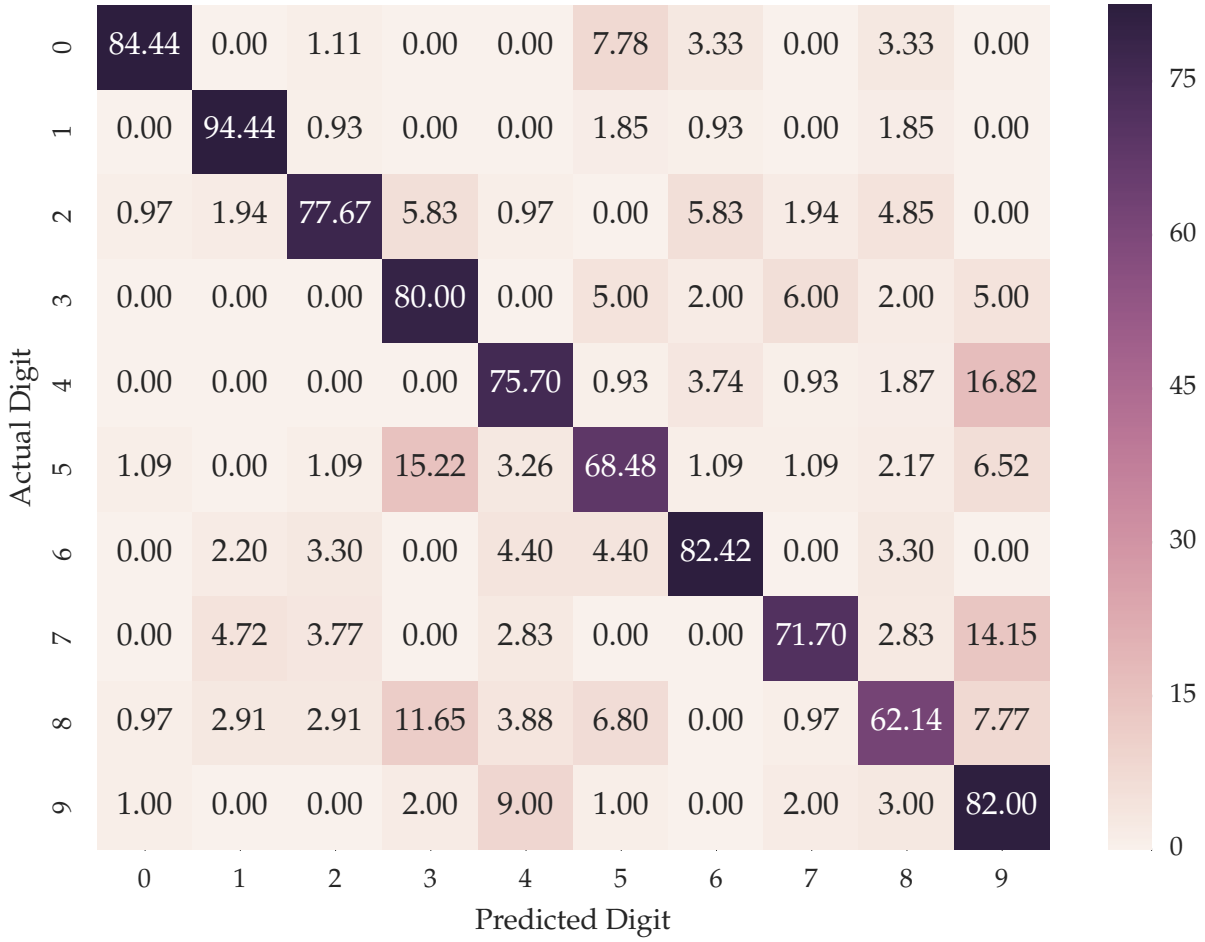


Figure 55: EC 1x1 Confusion Matrix (Digits Ternary Dataset)

A.3.2 Non-Overlapping Groupings

```
2x2 Test:                   2x2_OLap=False
    Best Smoothing Factor:  0.020202
    Overall Accuracy:       87.300000
    Evaluation Time:        233.774746895 seconds

                            Accuracy Matrix
```

Digit	0	1	2	3	4	5	6	7	8	9
Accuracy	96.67	94.44	88.35	90.0	89.72	79.35	87.91	79.25	83.5	84.0

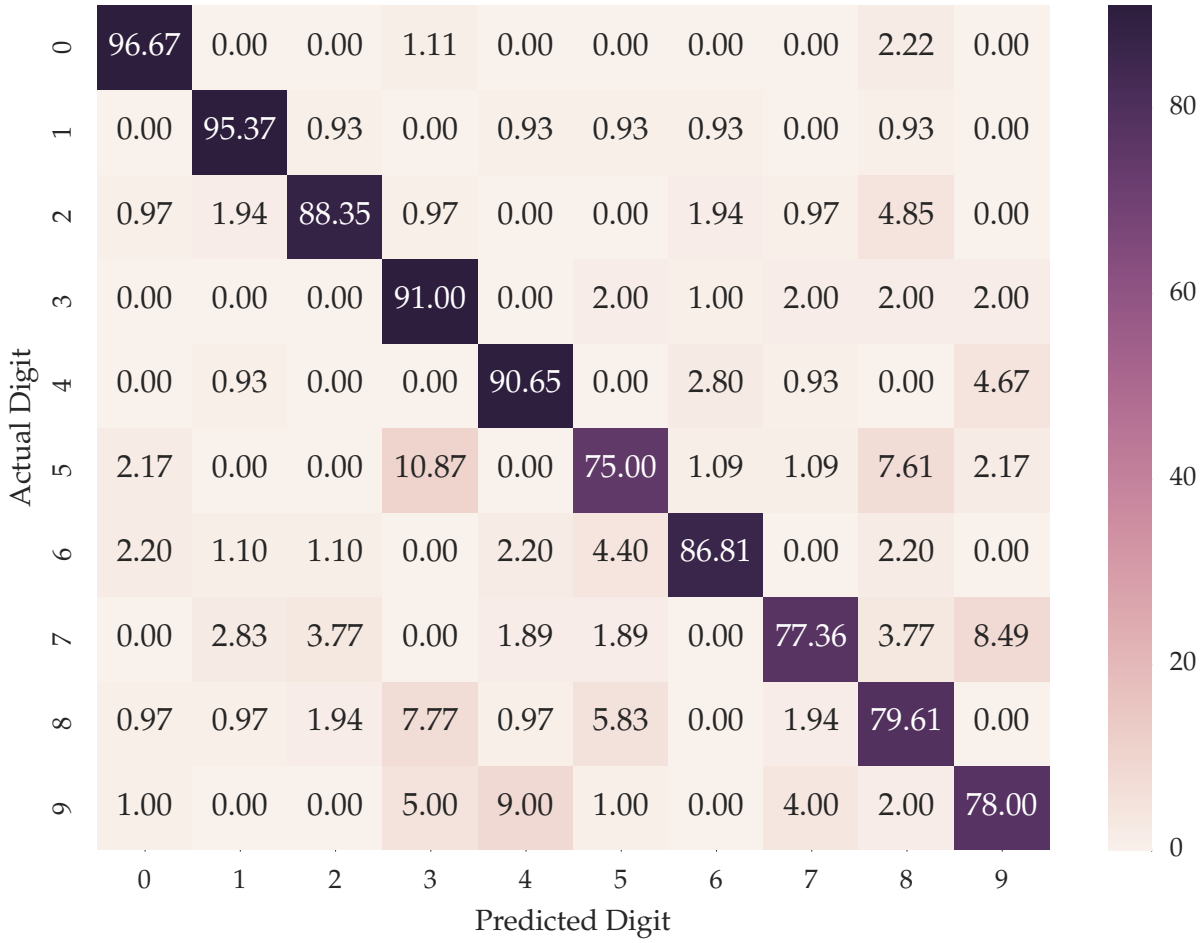


Figure 56: EC 2x2 Confusion Matrix (Digits Ternary Dataset)

2x4 Test: 2x4_OLap=False
Best Smoothing Factor: 0.020202
Overall Accuracy: 85.100000
Evaluation Time: 217.323404074 seconds

Accuracy Matrix

Digit	0	1	2	3	4	5	6	7	8	9
Accuracy	96.67	95.37	85.44	89.0	89.72	72.83	85.71	70.75	83.5	82.0

Confusion Matrix

Predicted Digit	0	1	2	3	4	5	6	7	8	9
Actual Digit										
0	0.00	0.00	0.00	1.11	0.00	0.00	0.00	1.11	1.11	0.00
1	0.00	0.00	1.85	0.00	0.00	0.93	0.93	0.00	0.93	0.00

2	1.94	0.97	0.00	0.97	1.94	0.00	1.94	1.94	4.85	0.00
3	0.00	0.00	1.00	0.00	0.00	5.00	0.00	3.00	1.00	1.00
4	0.00	0.93	0.00	0.00	0.00	0.00	1.87	0.00	1.87	5.61
5	2.17	0.00	0.00	11.96	0.00	0.00	0.00	0.00	9.78	3.26
6	2.20	1.10	0.00	0.00	2.20	4.40	0.00	0.00	4.40	0.00
7	0.00	3.77	6.60	0.00	3.77	1.89	0.00	0.00	1.89	11.32
8	0.97	0.97	3.88	5.83	0.00	0.97	1.94	0.97	0.00	0.97
9	1.00	0.00	0.00	4.00	8.00	1.00	0.00	3.00	1.00	0.00

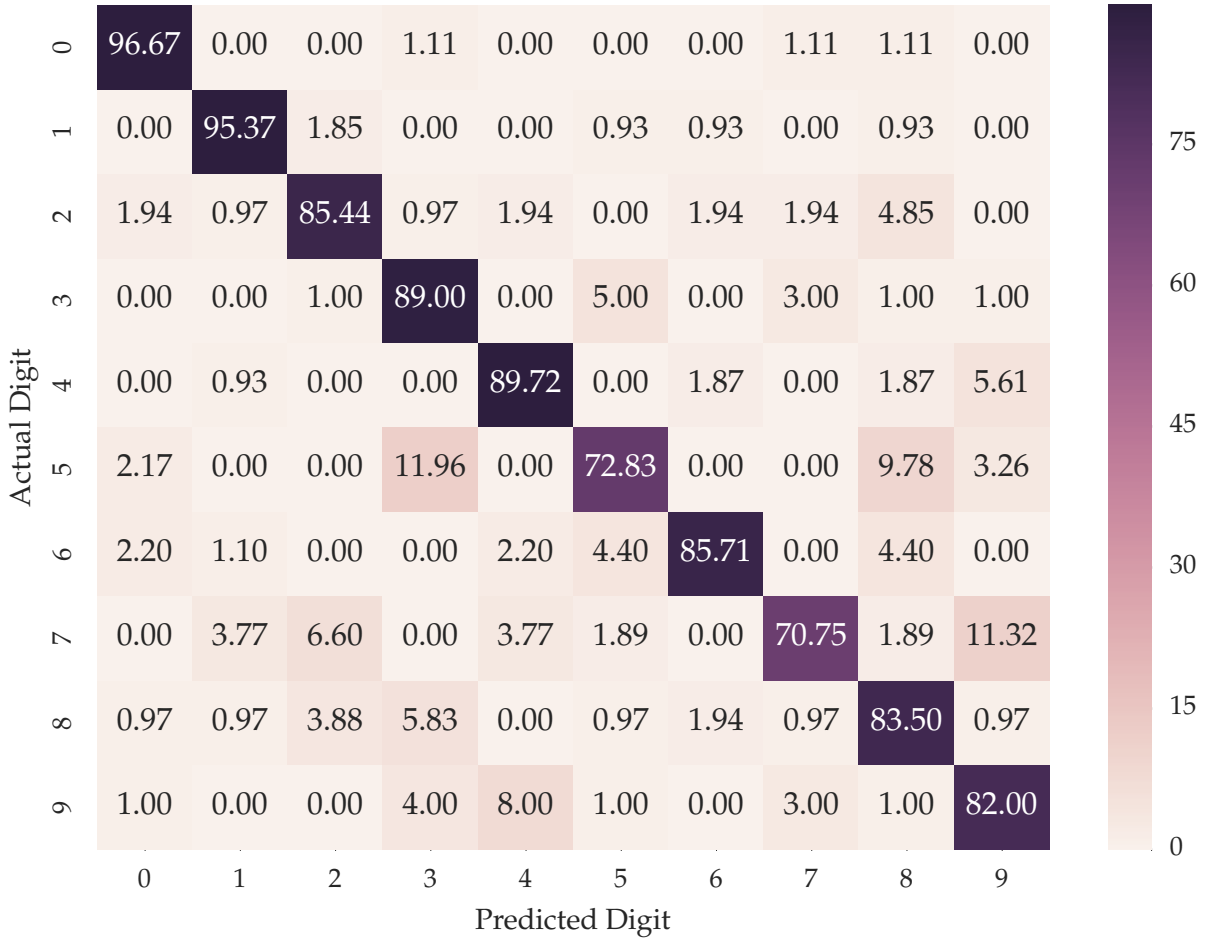


Figure 57: EC 2x4 Confusion Matrix (Digits Ternary Dataset)

4x2 Test: 4x2_OLap=False
Best Smoothing Factor: 0.020202
Overall Accuracy: 86.100000
Evaluation Time: 223.979768991 seconds

Accuracy Matrix

Digit	0	1	2	3	4	5	6	7	8	9
Accuracy	95.56	97.22	88.35	90.0	90.65	69.57	87.91	77.36	80.58	83.0

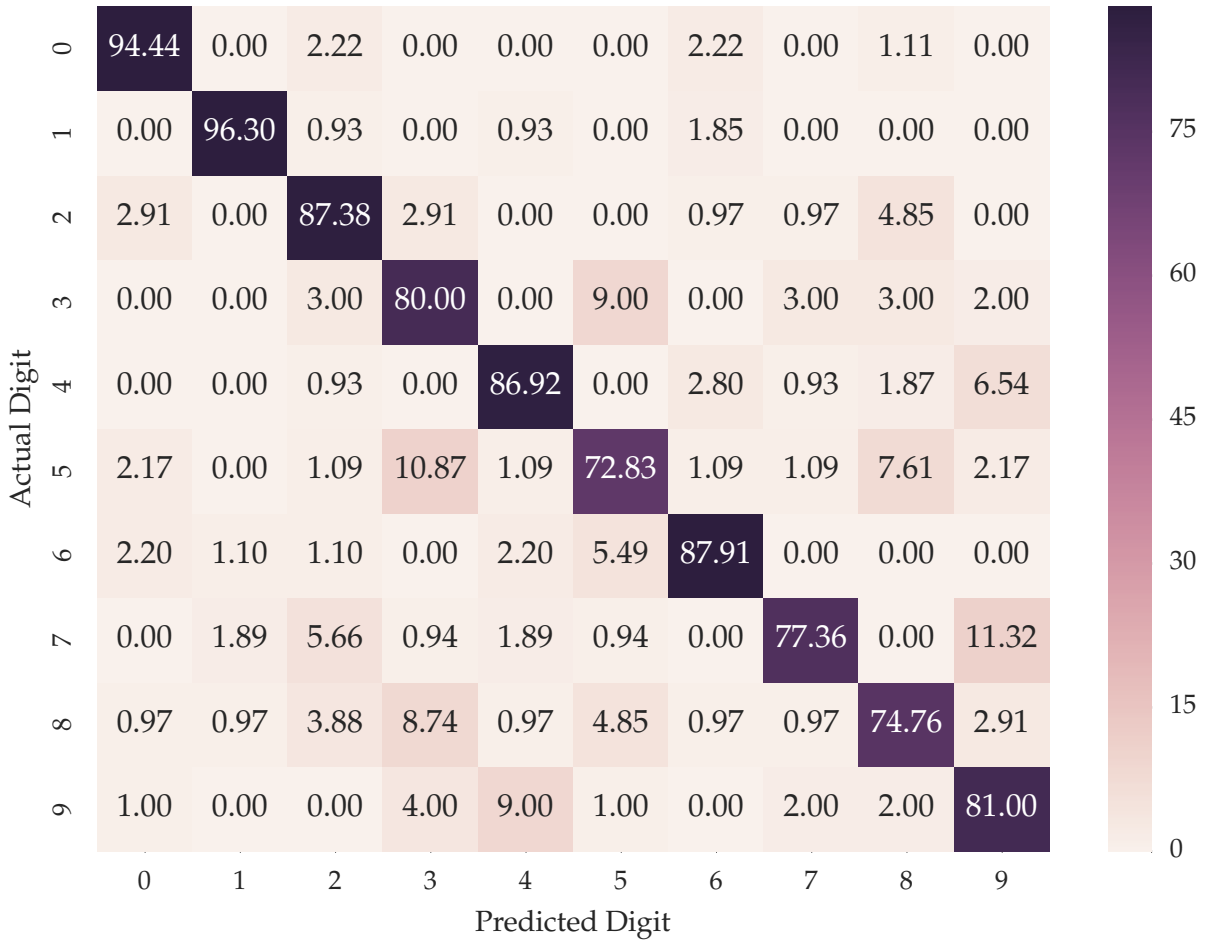


Figure 58: EC 4x2 Confusion Matrix (Digits Ternary Dataset)

4x4 NOT ENOUGH MEMORY, NOT ENOUGH TIME

Figure 59: EC 4x4 Confusion Matrix (Digits Ternary Dataset)

A.3.3 Overlapping Groupings

```
2x2 Test:                    2x2_OLap=True
    Best Smoothing Factor:   1.737374
    Overall Accuracy:        89.209536
    Evaluation Time:         383.162047863 seconds

                             Accuracy Matrix
    Digit          0      1      2     3      4      5      6      7
    Accuracy   93.33  98.15  82.52  92.0  95.33  78.26  86.81  85.85

                             Confusion Matrix
```

Predicted Digit	0	1	2	3	4	5	6	7
Actual Digit								
0	0.00	0.00	0.00	2.22	0.00	1.11	3.33	0.00
1	0.00	0.00	0.00	0.00	0.00	0.93	0.93	0.00
2	1.94	2.91	0.00	5.83	0.97	0.00	3.88	1.94
3	0.00	0.00	0.00	0.00	0.00	1.00	1.00	6.00
4	0.00	0.93	0.00	0.00	0.00	0.00	2.80	0.93
5	3.26	0.00	1.09	14.13	1.09	0.00	1.09	1.09
6	1.10	2.20	0.00	0.00	4.40	5.49	0.00	0.00
7	0.00	3.77	7.55	0.00	0.94	1.89	0.00	0.00
8	1.94	8.74	11.65	50.49	10.68	7.77	2.91	5.83
9	1.00	1.00	0.00	5.00	67.00	2.00	1.00	23.00

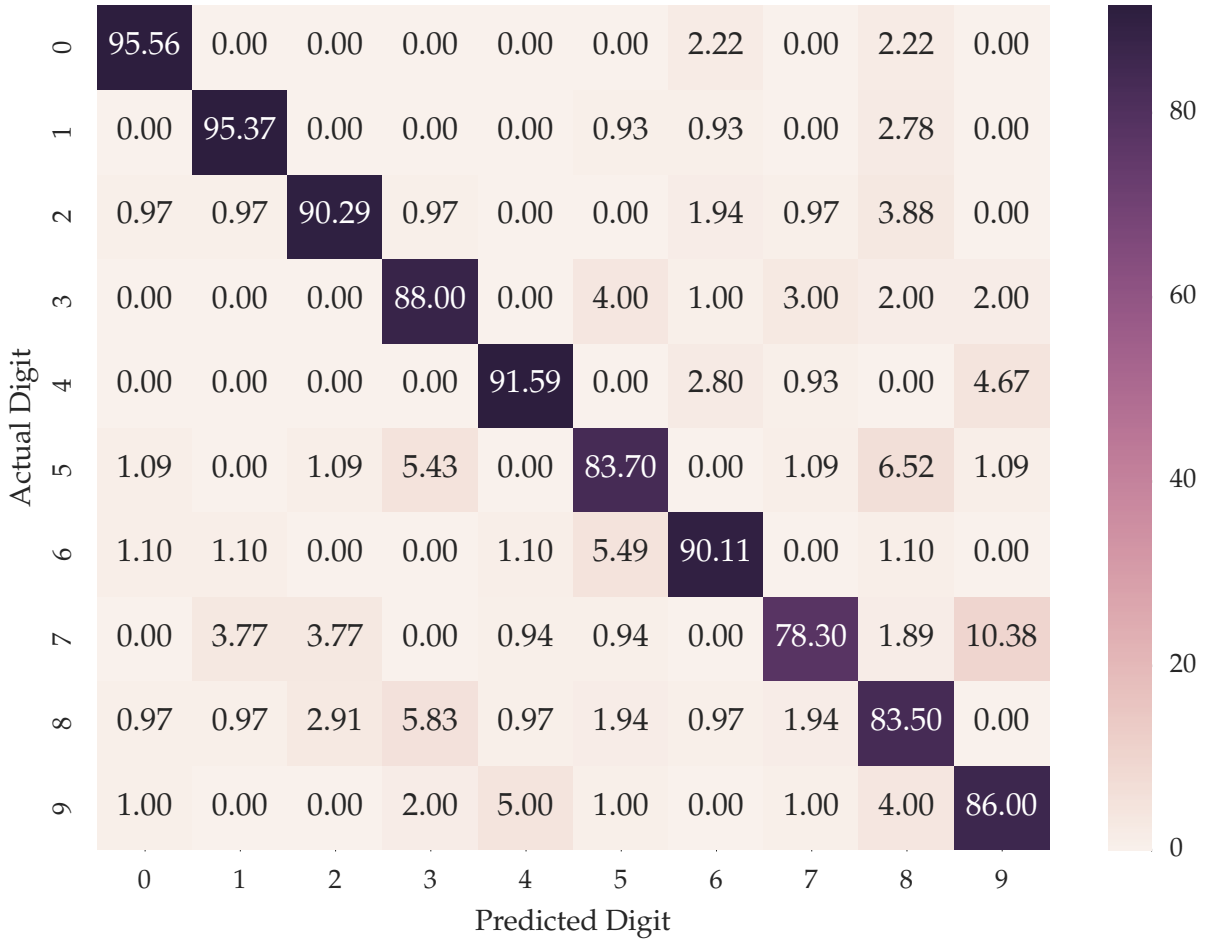


Figure 60: EC 2x2 Confusion Matrix (Digits Ternary Dataset, With Overlap)

```
2x3 Test:                    2x3_OLap=True
    Best Smoothing Factor:   0.020202
    Overall Accuracy:        88.700000
    Evaluation Time:         115.206239939 seconds
```

Accuracy Matrix

Digit	0	1	2	3	4	5	6	7	8	9
Accuracy	96.67	97.22	90.29	88.0	93.46	86.96	90.11	77.36	80.58	87.0

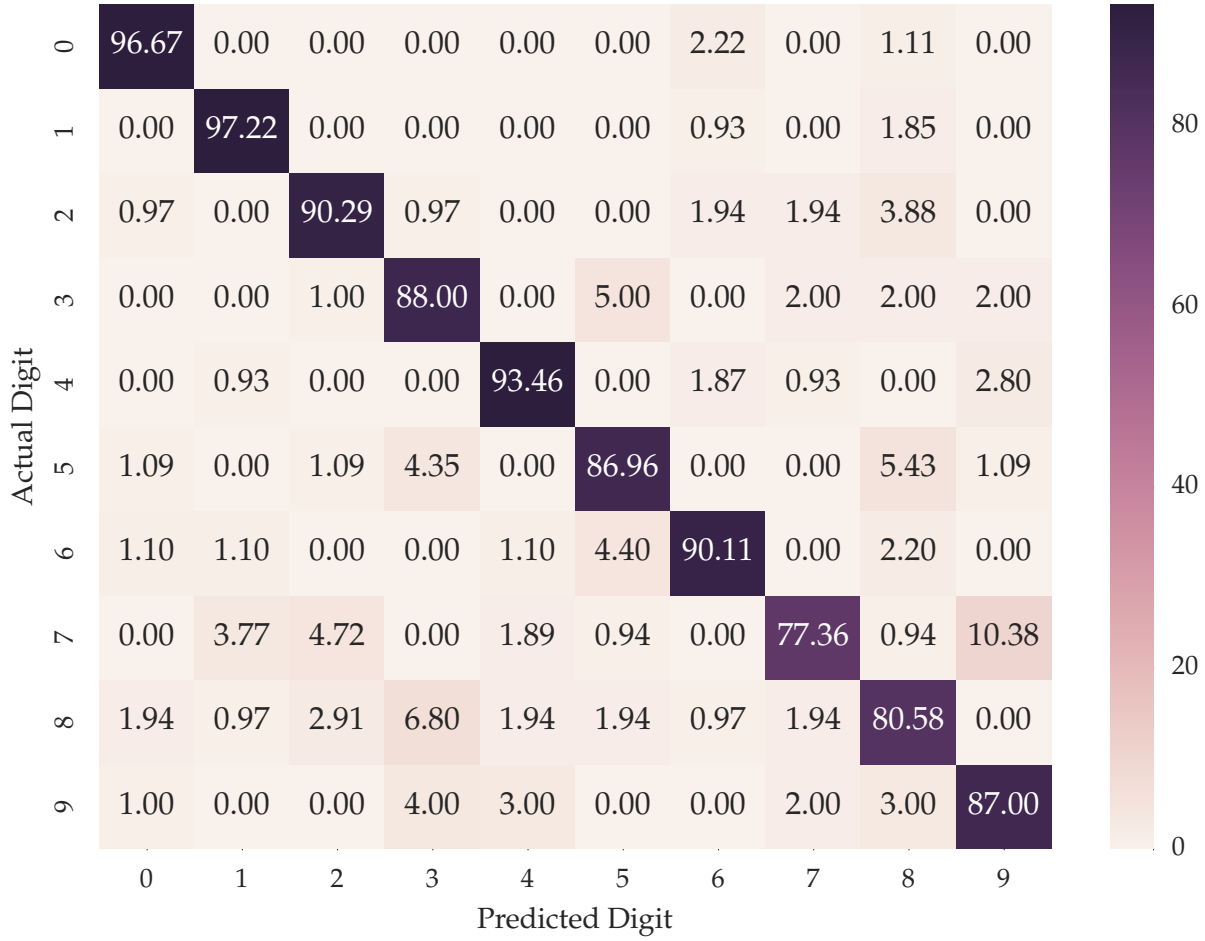


Figure 61: EC 2x3 Confusion Matrix (Digits Ternary Dataset, With Overlap)

2x4 Test: 2x4_OLap=True
Best Smoothing Factor: 0.020202
Overall Accuracy: 88.600000
Evaluation Time: 1185.30268502 seconds

Accuracy Matrix

Digit	0	1	2	3	4	5	6	7	8	9
Accuracy	98.89	97.22	89.32	88.0	93.46	84.78	90.11	76.42	82.52	86.0

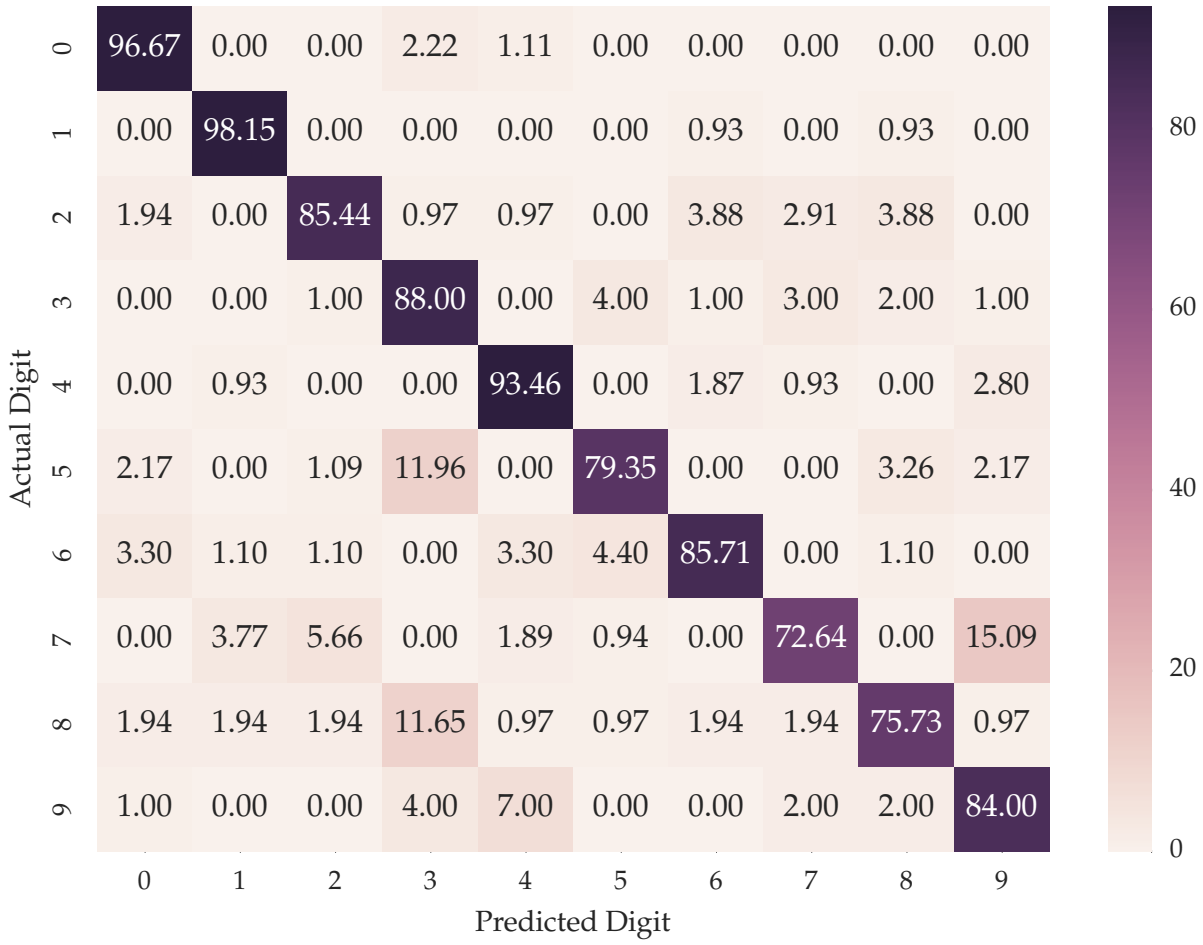


Figure 62: EC 2x4 Confusion Matrix (Digits Ternary Dataset, With Overlap)

```
3x2 Test:                    3x2_OLap=True
    Best Smoothing Factor:   0.020202
    Overall Accuracy:        88.700000
    Evaluation Time:         162.374095917 seconds

                             Accuracy Matrix
    Digit          0      1      2     3      4      5      6      7      8     9
    Accuracy   96.67  98.15  91.26  89.0  95.33  81.52  90.11  77.36  78.64  89.0
```

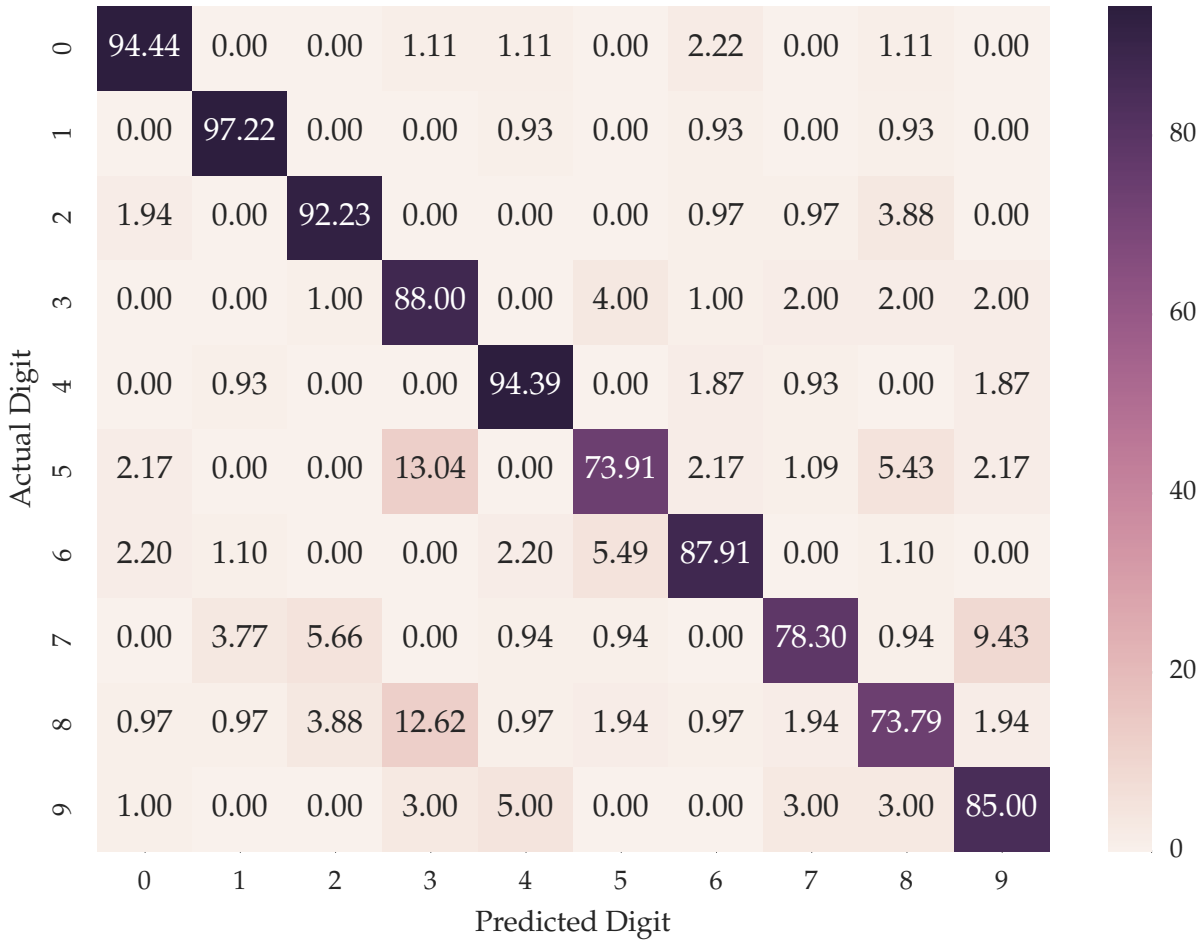


Figure 63: EC 2x2 Confusion Matrix (Digits Ternary Dataset, With Overlap)

3x3 Test: 3x3_OLap=True
Best Smoothing Factor: 0.020202
Overall Accuracy: 88.000000
Evaluation Time: 2866.77030396 seconds

Accuracy Matrix

Digit	0	1	2	3	4	5	6	7	8	9
Accuracy	96.67	99.07	90.29	89.0	94.39	77.17	90.11	77.36	77.67	88.0

4x2 Test: 4x2_OLap=True
Best Smoothing Factor: 0.020202
Overall Accuracy: 88.200000
Evaluation Time: 1159.87088299 seconds

Accuracy Matrix

Digit	0	1	2	3	4	5	6	7	8	9
Accuracy	96.67	97.22	91.26	90.0	95.33	73.91	90.11	81.13	78.64	87.0

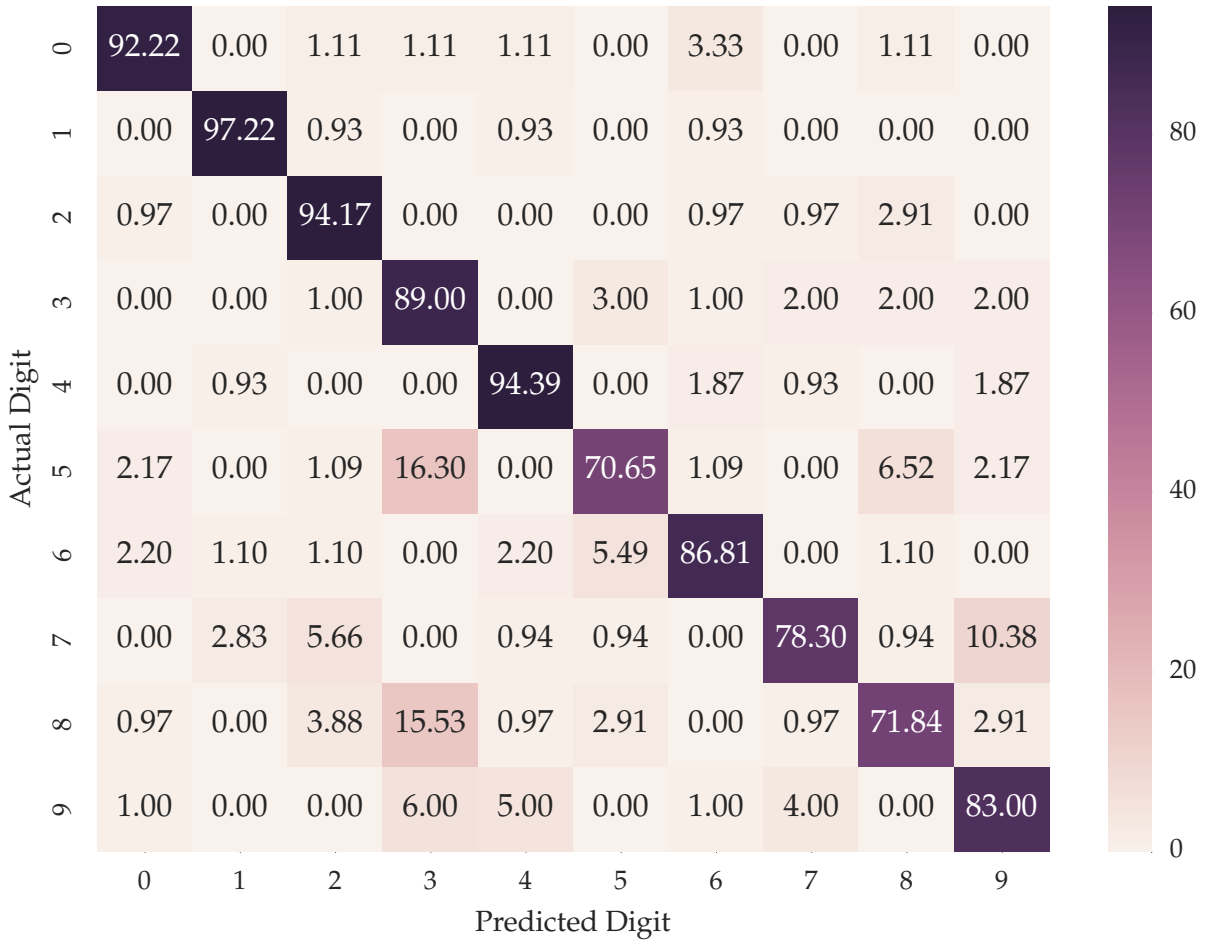


Figure 64: EC 4x2 Confusion Matrix (Digits Ternary Dataset, With Overlap)

4x4 NOT ENOUGH MEMORY, NOT ENOUGH TIME

A.4 Extra Credit Face Data Set

A.4.1 Single Pixel Results

```
1x1 Test:                    binary_face_1x1_OLap=False
    Best Smoothing Factor:   0.606061
    Overall Accuracy:        88.000000
    Evaluation Time:         36.2563018799 seconds

                             Accuracy Matrix
    Digit          0      1
    Accuracy   92.21  83.56

                             Confusion Matrix
    Predicted Digit      0     1
    Actual Digit
```

```
0                0.00  7.79
1               16.44  0.00
```

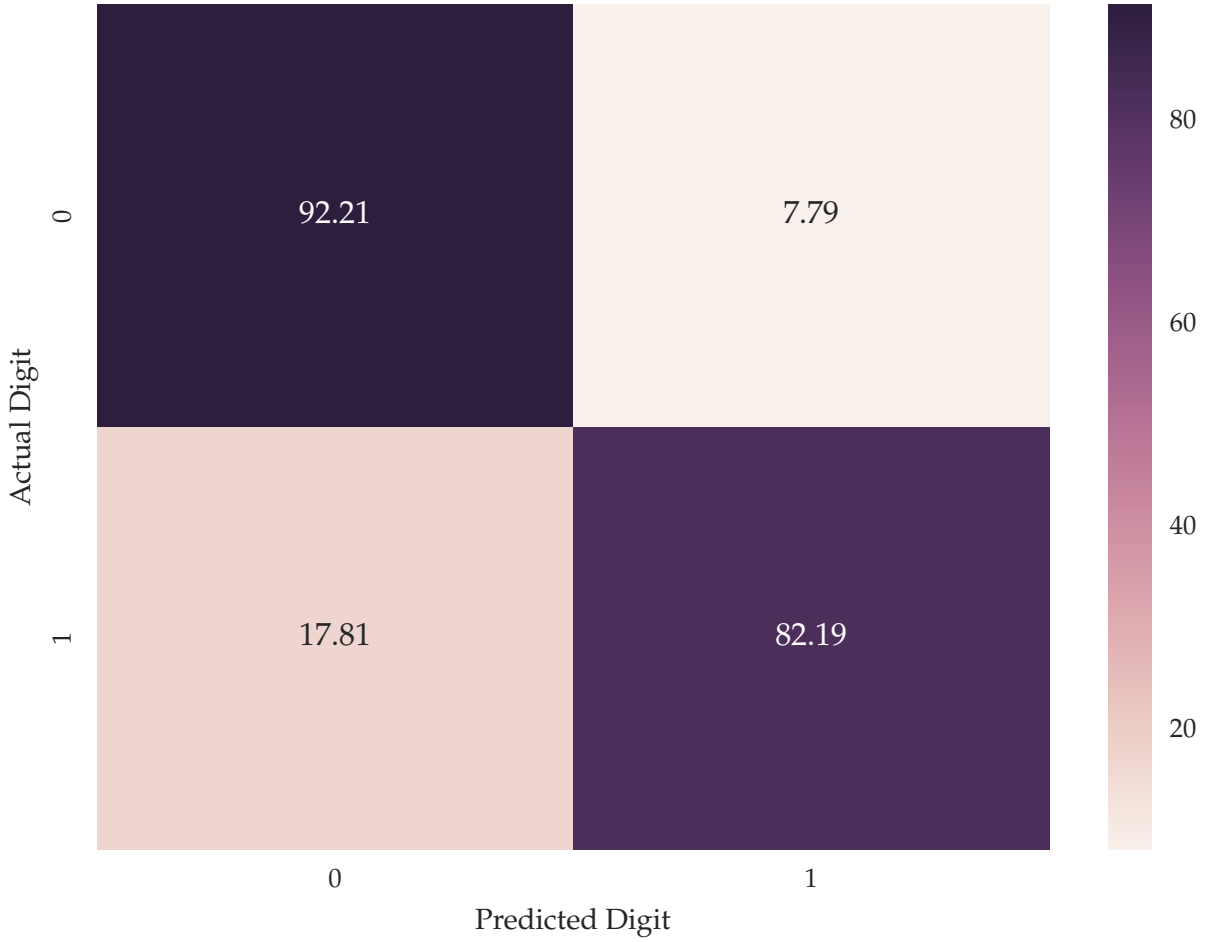


Figure 65: EC 1x1 Confusion Matrix (Face Dataset)

A.4.2 Non-Overlapping Groupings

```
2x2 Test:                    binary_face_2x2_OLap=False
    Best Smoothing Factor:   0.383838
    Overall Accuracy:        99.333333
    Evaluation Time:         68.9599370956 seconds

                             Accuracy Matrix
    Digit         0      1
    Accuracy  100.0  98.63

                             Confusion Matrix
    Predicted Digit     0
    Actual Digit
    1                1.37
```

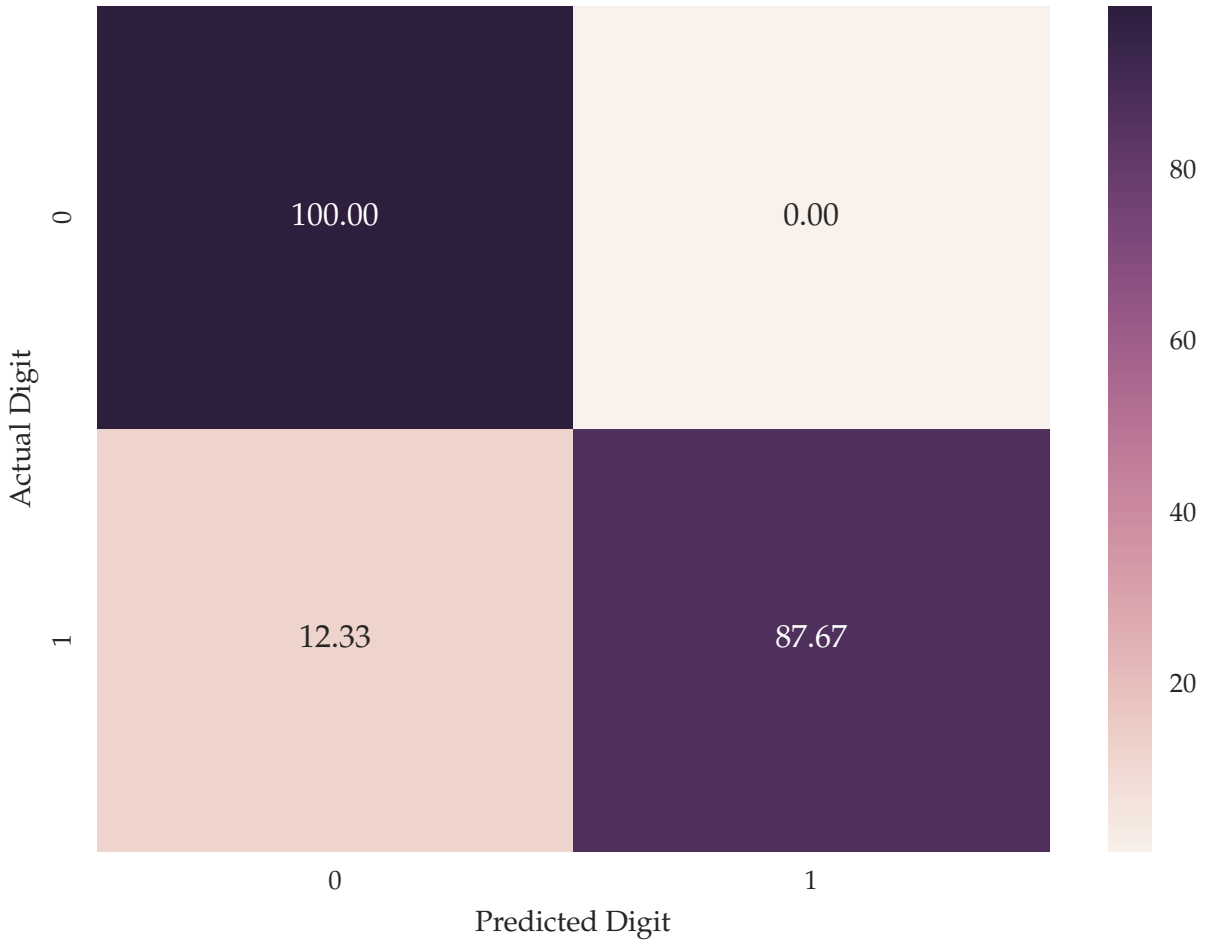


Figure 66: EC 2x2 Confusion Matrix (Face Dataset)

```
2x4 Test:                    binary_face_2x4_OLap=False
    Best Smoothing Factor:   0.121212
    Overall Accuracy:        99.333333
    Evaluation Time:         472.658554077 seconds

                             Accuracy Matrix
    Digit         0      1
    Accuracy  100.0  98.63

                             Confusion Matrix
    Predicted Digit     0
    Actual Digit
    1                1.37
```

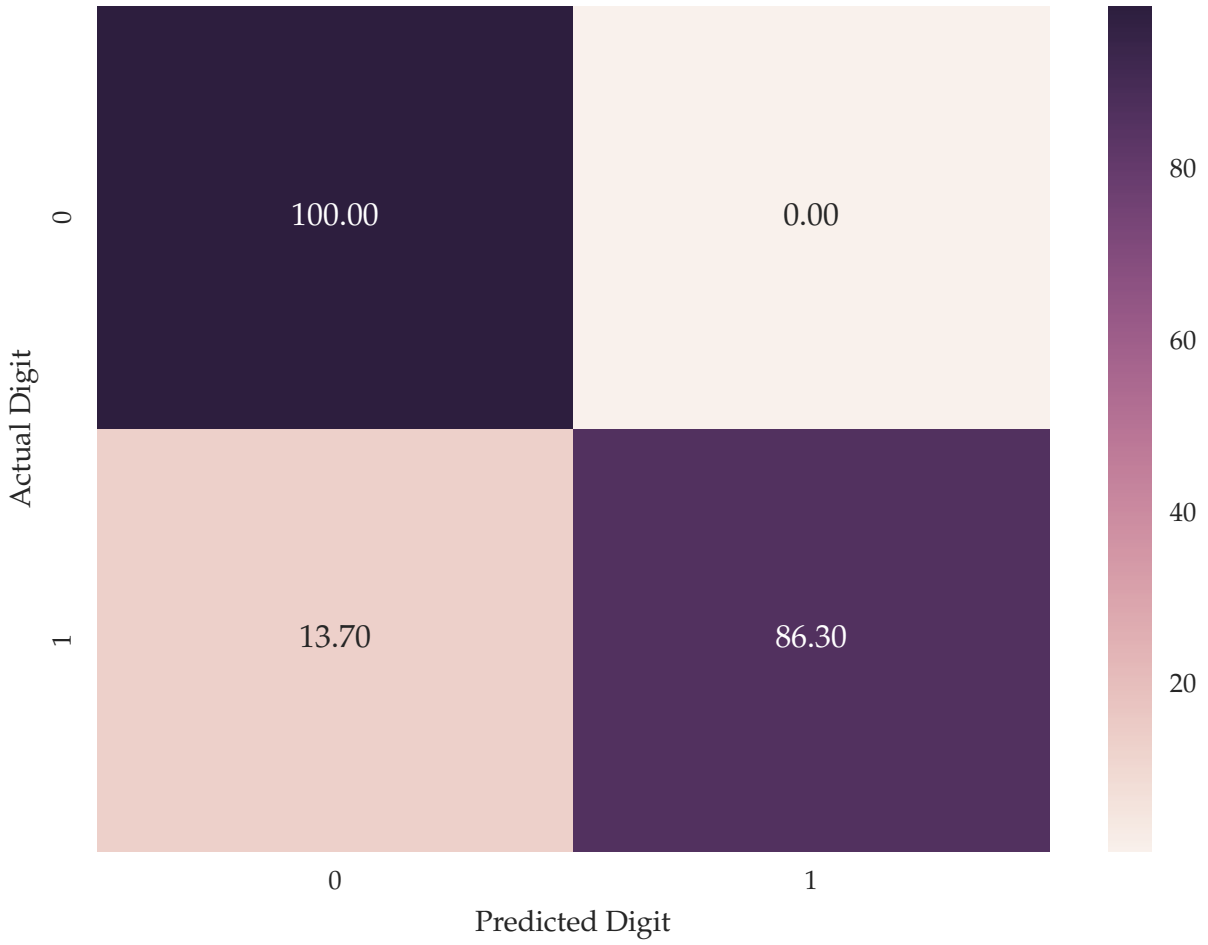


Figure 67: EC 2x4 Confusion Matrix (Face Dataset)

```
4x2 Test:                    binary_face_4x2_OLap=False
    Best Smoothing Factor:   0.565657
    Overall Accuracy:        98.000000
    Evaluation Time:         448.871038198 seconds

                             Accuracy Matrix
    Digit        0      1
    Accuracy  96.1  100.0

                             Confusion Matrix
    Predicted Digit    1
    Actual Digit
    0                3.9
```

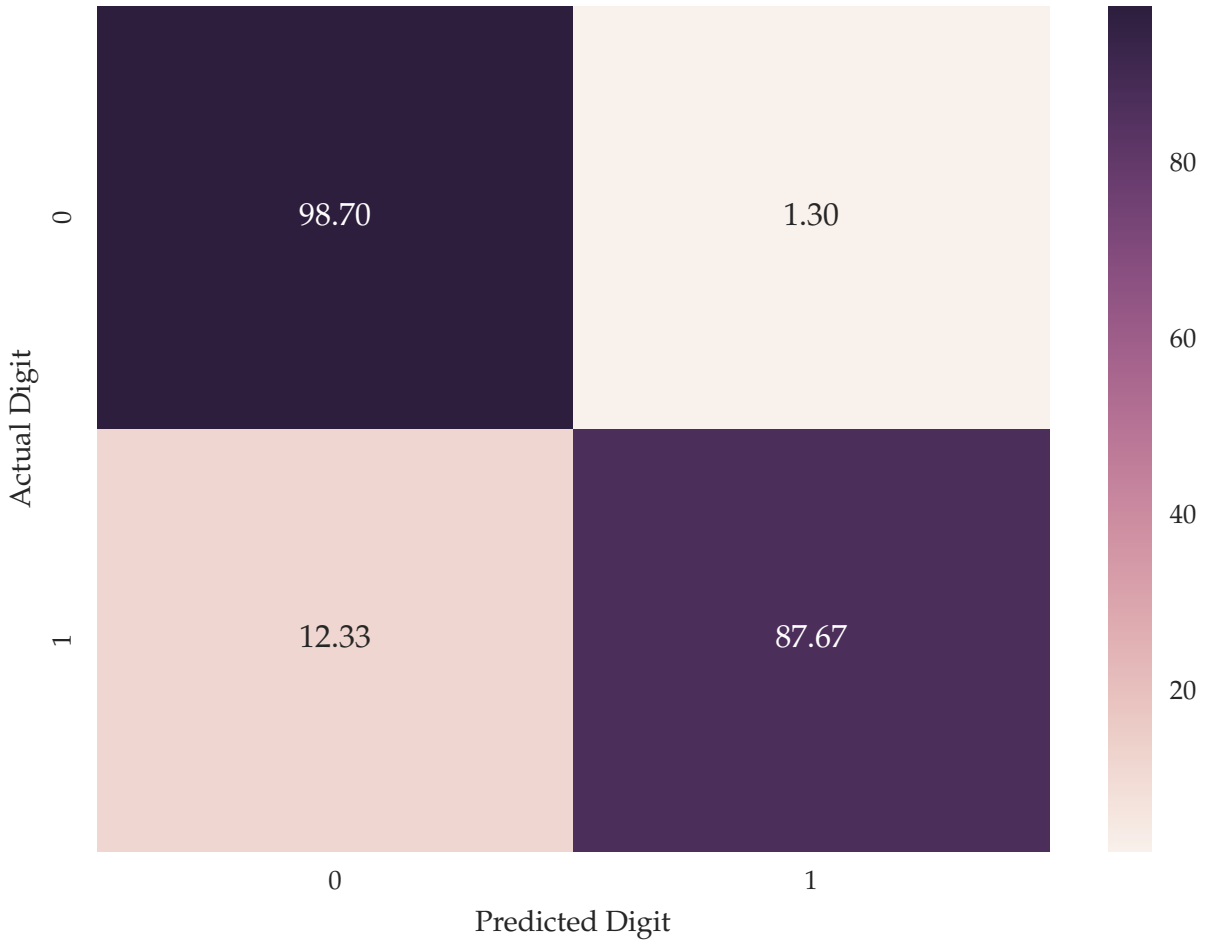


Figure 68: EC 4x2 Confusion Matrix (Face Dataset)

```
4x4 Test:                    binary_face_4x4_OLap=False
    Best Smoothing Factor:   0.020202
    Overall Accuracy:        97.333333
    Evaluation Time:         1208.2900362 seconds

                             Accuracy Matrix
    Digit         0      1
    Accuracy  94.81  100.0

                             Confusion Matrix
    Predicted Digit     1
    Actual Digit
    0                5.19
```

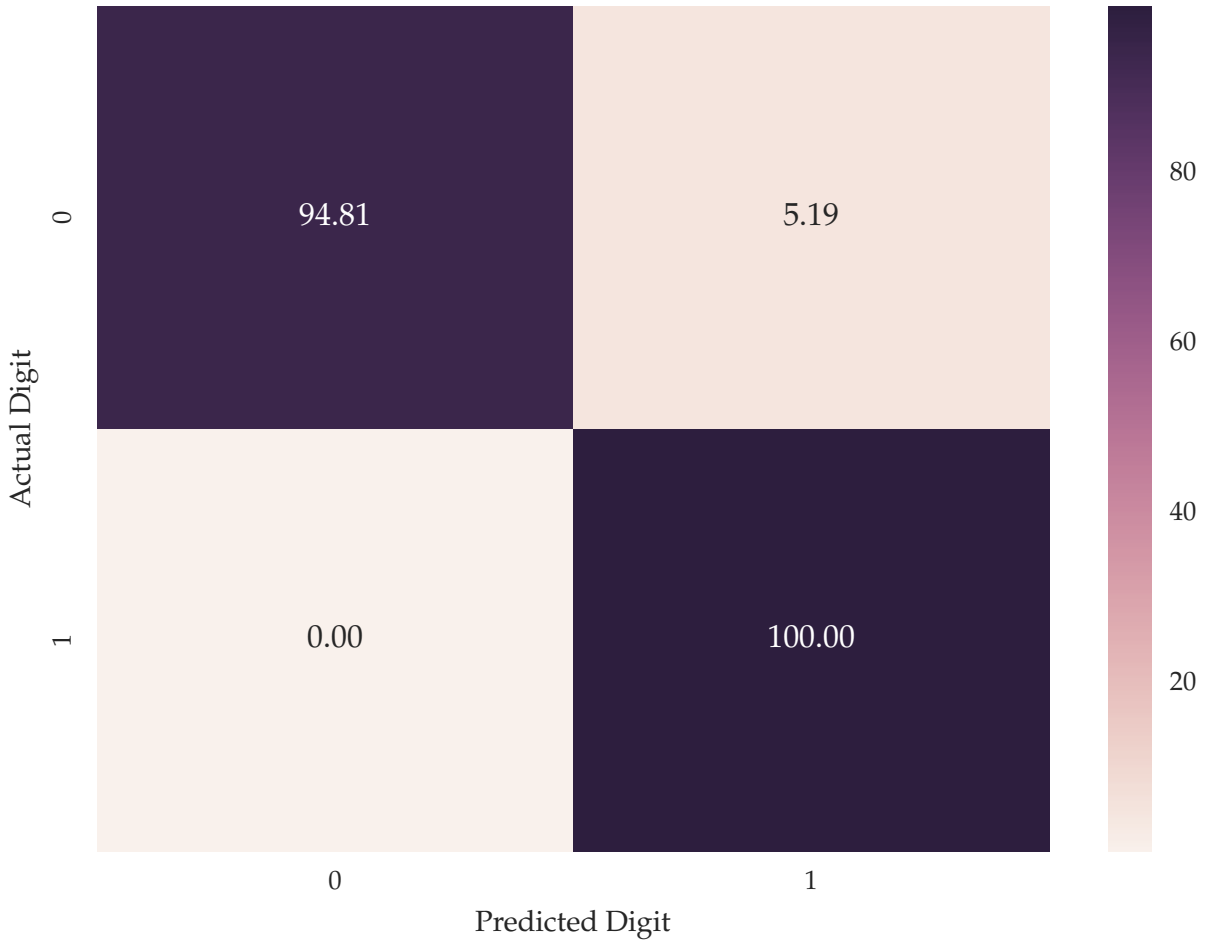


Figure 69: EC 4x4 Confusion Matrix (Face Dataset)

A.4.3 Overlapping Groupings

```
2x2 Test:                    binary_face_2x2_OLap=True
    Best Smoothing Factor:   1.070707
    Overall Accuracy:        98.666667
    Evaluation Time:         141.701714039 seconds
```

Figure 70: EC 2x2 Confusion Matrix (Face Dataset, With Overlap)

```
2x3 Test:                    binay_face_2x3_OLap=True
    Best Smoothing Factor:   0.020202
    Overall Accuracy:        94.666667

                            Accuracy Matrix
    Digit         0      1
    Accuracy  100.0  89.04
```

```
                        Confusion Matrix
Predicted Digit       0      1
Actual Digit
0                100.00   0.00
1                 10.96  89.04
```

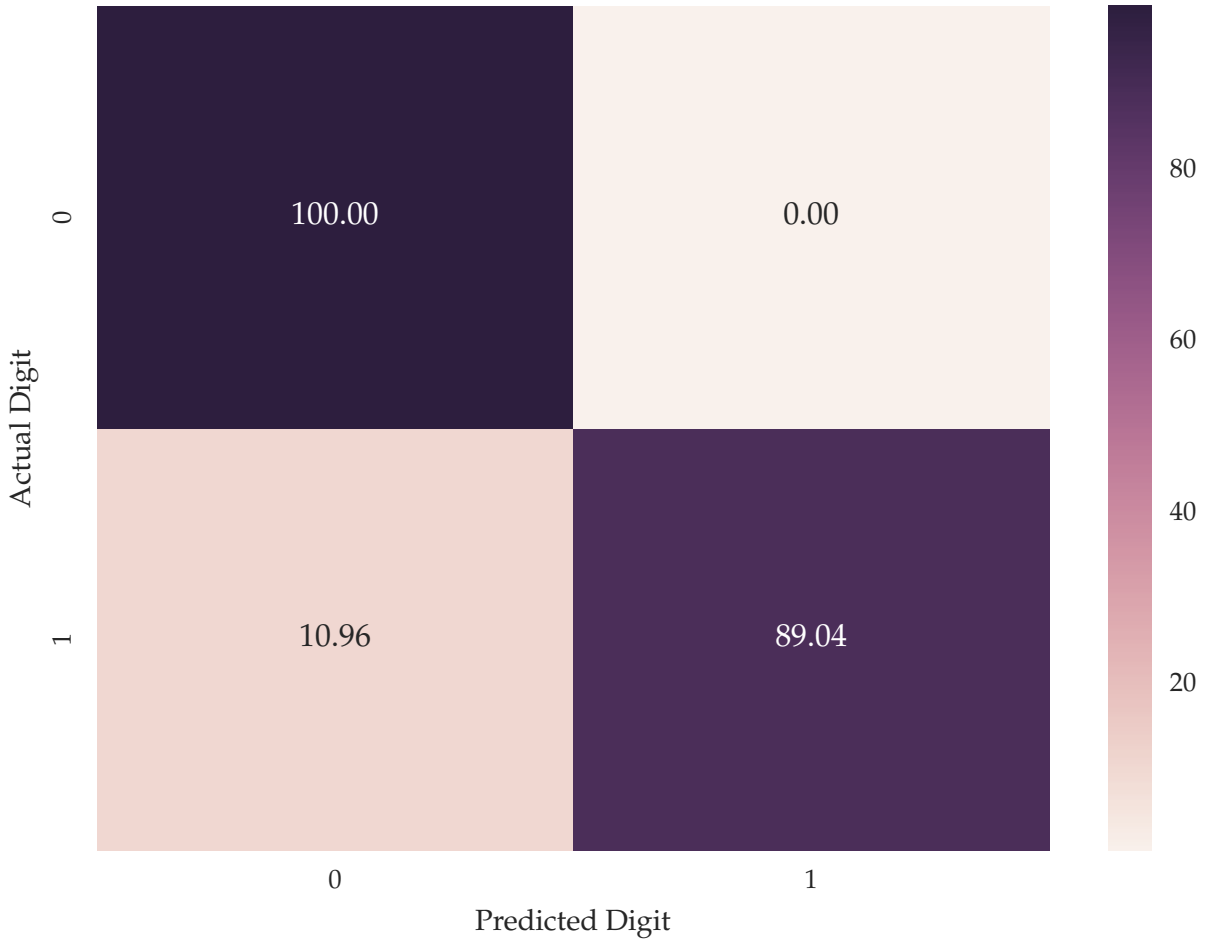


Figure 71: EC 2x3 Confusion Matrix (Face Dataset, With Overlap)

```
2x4 Test:                   binary_face_2x4_OLap=True
    Best Smoothing Factor:  0.040404
    Overall Accuracy:       98.666667
    Evaluation Time:        2729.28366303 seconds
```

Figure 72: EC 2x4 Confusion Matrix (Face Dataset, With Overlap)

```
3x2 Test:                   binay_face_3x2_OLap=True
    Best Smoothing Factor:  0.020202
    Overall Accuracy:       94.666667
```

```
                        Accuracy Matrix
Digit          0      1
Accuracy  100.0  89.04

                        Confusion Matrix
Predicted Digit       0      1
Actual Digit
0                100.00   0.00
1                 10.96  89.04
```

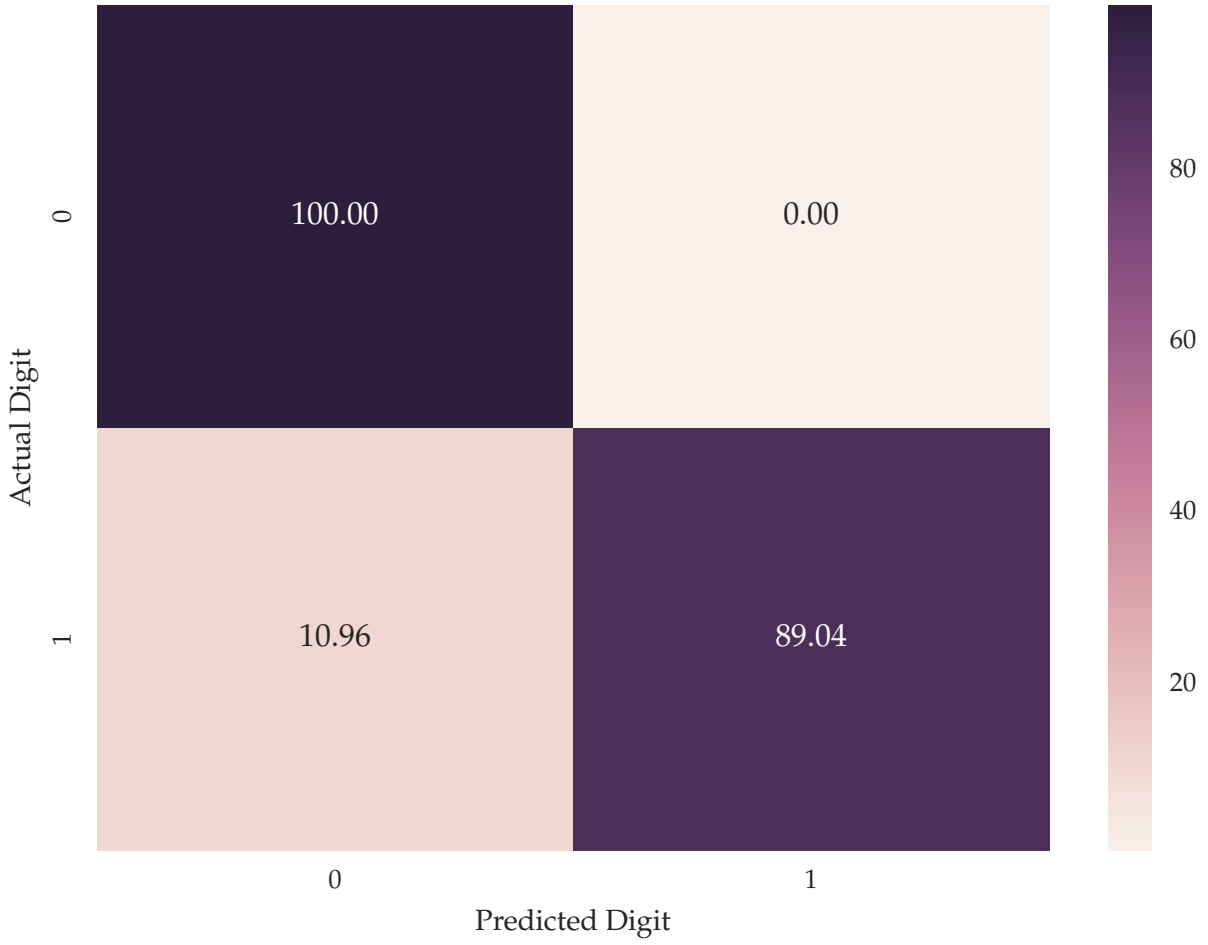


Figure 73: EC 2x2 Confusion Matrix (Face Dataset, With Overlap)

3x3 Out of memory, out of time

Figure 74: EC 2x2 Confusion Matrix (Face Dataset, With Overlap)

4x2 Out of memory, out of time

Figure 75: EC 4x2 Confusion Matrix (Face Dataset, With Overlap)

4x4 `Out of memory, out of time`

Figure 76: EC 4x4 Confusion Matrix (Face Dataset, With Overlap)

B
Part Two Results

B.1 Sentiment analysis of movie reviews results

B.1.1 Multinomial Naive Bayes Classifier

```
HINT: row index is actual value, column index is predicted value
-------------------------------------------------------------
Confusion Matrix
 [0,           1]
0[0.756, 0.244]
1[0.234, 0.766]

-------------------------------------------------------------
Classification rate for each digit:
The class 0: 0.756
The class 1: 0.766
-------------------------------------------------------------------------------------------------------
Top 10 words with highest likelihood for class 0:   Top 10 words with highest likelihood for class 1:
movie: -4.6759392568762985                          film: -4.704509111705017
film: -4.91991689509335                             movie: -5.12405895969492
like: -5.249396096223593                            --: -5.440519996696744
one: -5.379449224471791                             one: -5.642002051229775
--: -5.604330395684541                              like: -5.755330736536778
bad: -5.8719257095695845                            story: -5.806624030924328
story: -5.894915227794283                           good: -5.917849666034552
much: -5.918445725204478                            comedy: -5.929684123681556
time: -6.01852918376146                             way: -5.96605176785243
even: -6.086582647006476                            even: -6.016695500671185

-------------------------------------------------------------------------------------------------------

Top 10 words with highest odds ratio for class 0:   Top 10 words with highest odds ratio for class 1
flat: 2.719288599579288                             disturbing: 2.696811802625132
stale: 2.6502957280923365                           refreshingly: 2.386656874321293
dull: 2.6139280839214614                            haunting: 2.386656874321293
tired: 2.4961450482650784                           engrossing: 2.386656874321293
plain: 2.4091336712754483                           grief: 2.386656874321293
mediocre: 2.4091336712754483                        refreshing: 2.2913466945169674
unfunny: 2.3138234914711244                         gripping: 2.2913466945169674
forced: 2.3138234914711244                          inventive: 2.2913466945169674
poorly: 2.3138234914711244                          polished: 2.2913466945169674
bore: 2.3138234914711244                            gem: 2.2913466945169674

-------------------------------------------------------------------------------------------------------
The accuracy is:  0.761
Execution time is: 0.373941s
```

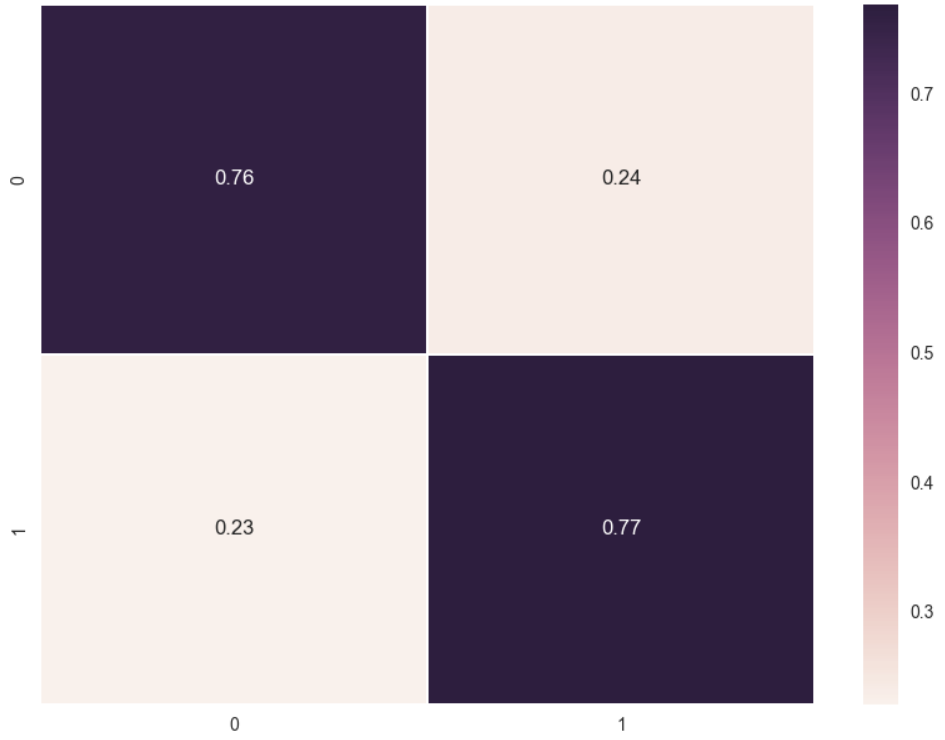


Figure 77: Confusion matrix Movie corpus Multinomial

B.1.2 Bernoulli Naive Bayes Classifier

```
HINT: row index is actual value, column index is predicted value
-------------------------------------------------------------
Confusion Matrix
 [0,      1]
0[0.768, 0.232]
1[0.258, 0.742]

-------------------------------------------------------------
Classification rate for each digit:
The class 0: 0.768
The class 1: 0.742
-------------------------------------------------------------------------------------------------------
Top 10 words with highest likelihood for class 0:   Top 10 words with highest likelihood for class 1:
movie: -1.9529277217139598                          film: -1.9599948889370522
film: -2.1902559080201263                           movie: -2.4089451089849554
like: -2.545656154526858                            one: -2.9014215940827497
one: -2.674648274717961                             like: -3.006782109740576
story: -3.1592507033848496                          --: -3.016934481204594
much: -3.183061352078568                            story: -3.0910424533583156
--: -3.207452805202727                              comedy: -3.195182712610913
bad: -3.258096538021482                             way: -3.2324541074081443
```

```
time: -3.311442518726775                                    even: -3.258096538021482
even: -3.353406717825807                                    good: -3.297836866670996

-------------------------------------------------------------------------------------------------------------

Top 10 words with highest odds ratio for class 0:   Top 10 words with highest odds ratio for class 1:
flat: 2.7080502011022096                            disturbing: 2.639057329615259
stale: 2.639057329615259                            refreshingly: 2.3978952727983707
dull: 2.5649493574615363                            grief: 2.3978952727983707
tired: 2.484906649788                               engrossing: 2.3978952727983707
mediocre: 2.3978952727983707                        haunting: 2.3978952727983707
plain: 2.3978952727983707                           refreshing: 2.302585092994046
poorly: 2.302585092994046                           gripping: 2.302585092994046
forced: 2.302585092994046                           polished: 2.302585092994046
unfunny: 2.302585092994046                          affecting: 2.302585092994046
bore: 2.302585092994046                             inventive: 2.302585092994046

-------------------------------------------------------------------------------------------------------------
The accuracy is:  0.755
Execution time is: 54.626312s
```

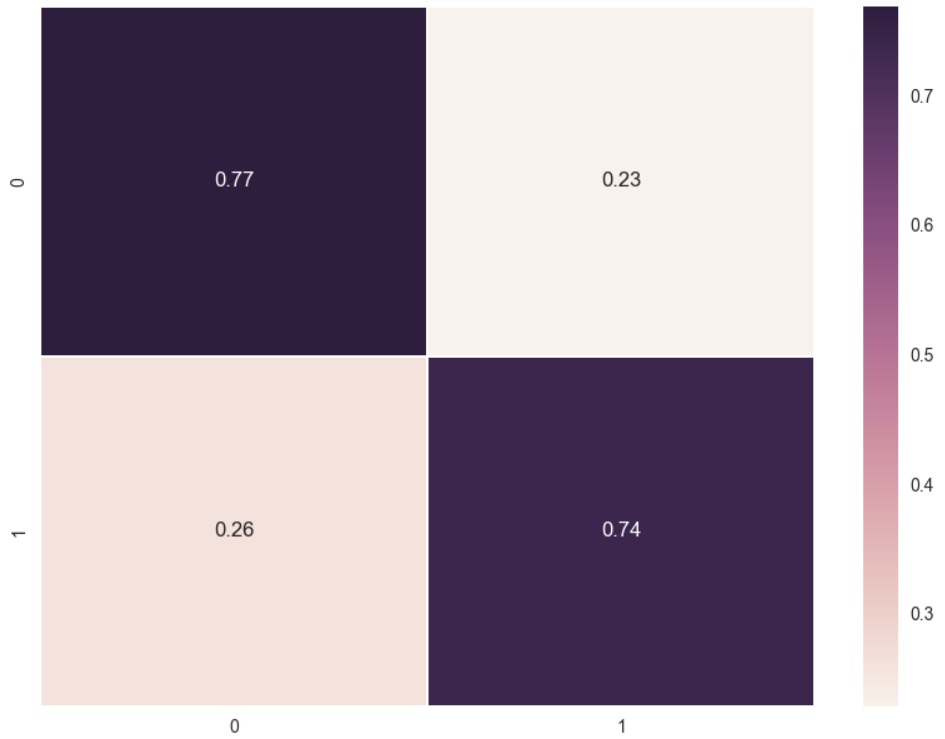


Figure 78: Confusion matrix Movie corpus Bernolli

B.2 Binary conversation topic classification results

B.2.1 Multinomial Naive Bayes Classifier

```
HINT: row index is actual value, column index is predicted value
-------------------------------------------------------------
  [0, 1]
0[0.9387755102040817, 0.061224489795918366]
1[0.10204081632653061, 0.8979591836734694]

-------------------------------------------------------------
Classification rate for each digit:
The class 0: 0.9387755102040817
The class 1: 0.8979591836734694
-------------------------------------------------------------------------------------------------------
Top 10 words with highest likelihood for class 0:   Top 10 words with highest likelihood for class 1:
know: -2.910374098291088                            know: -2.9656753972801257
yeah: -3.09338333161108                             yeah: -3.0920674144329356
uh: -3.495268685128181                              like: -3.5442710676555786
like: -3.5192182486052146                           uh: -3.8203013626144338
um: -3.772754801261674                              um: -3.932062959614287
right: -3.946199695203724                           right: -3.9908900394304148
just: -4.018611645280864                            don: -4.055063301882509
think: -4.03495890770624                            think: -4.070541403616782
oh: -4.120772339973308                              just: -4.104863993157005
don: -4.141399503703598                             oh: -4.21200027495845

-------------------------------------------------------------------------------------------------------

Top 10 words with highest odds ratio for class 0:   Top 10 words with highest odds ratio for class 1:
relationship: 5.288602653961092                     wage: 5.013675699632688
compatibility: 4.785951946878692                    minimum: 5.012695787746339
communication: 4.7590444939587675                   welfare: 4.857760486458663
marriage: 4.719641946092739                         wages: 4.724752243205215
partner: 4.687729819710658                          inflation: 4.505784063301485
relationships: 4.56575390579424                     waitresses: 4.348155119097903
friendship: 4.558373798496616                       tax: 4.335885026506088
attracted: 4.547200497898491                        waitress: 4.23189531298204
dating: 4.489380927009665                           salary: 4.175542376430909
compatible: 4.415272954855943                       increase: 4.06857025687874

-------------------------------------------------------------------------------------------------------
The accuracy is:  0.9183673469387755
Execution time is: 0.573207s
```

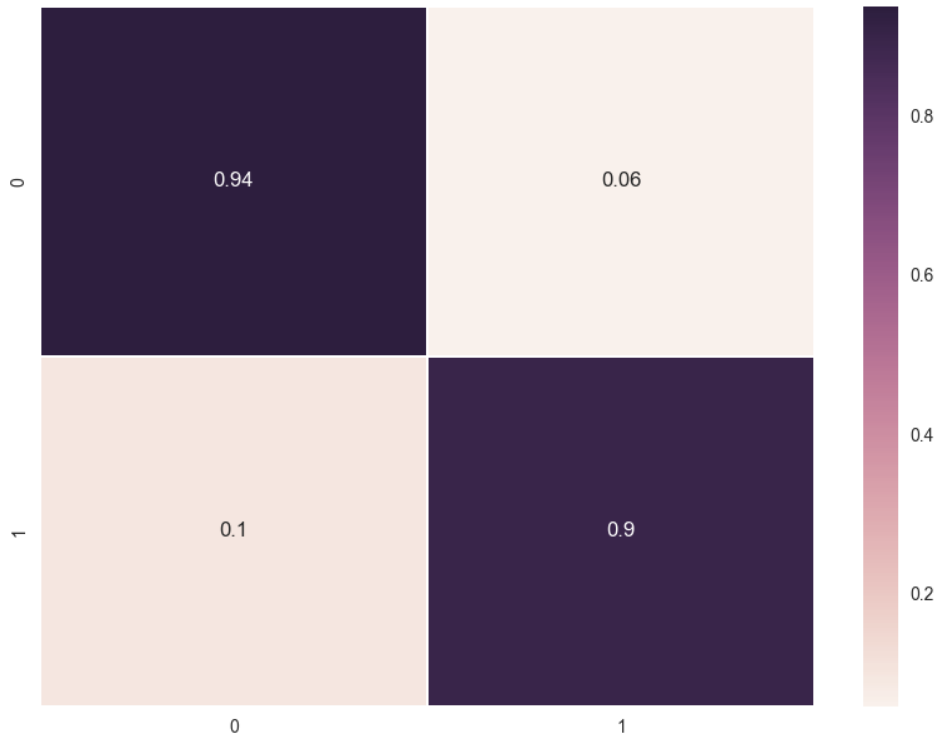


Figure 79: Confusion matrix two topics Multinomial

B.2.2 Bernoulli Naive Bayes Classifier

```
HINT: row index is actual value, column index is predicted value
-------------------------------------------------------------
Confusion Matrix
[0,                                          1]
0[0.9183673469387755, 0.08163265306122448]
1[0.10204081632653061, 0.8979591836734694]

-------------------------------------------------------------
Classification rate for each digit:
The class 0: 0.9183673469387755
The class 1: 0.8979591836734694
-------------------------------------------------------------------------------------------------------------
Top 10 words with highest likelihood for class 0:   Top 10 words with highest likelihood for class 1:
know: -3.6370465104852876                           know: -3.6326209953369784
like: -3.6370465104852876                           um: -3.6326209953369784
just: -3.6393270131840127                           think: -3.6326209953369784
think: -3.6416127284648687                          don: -3.6326209953369784
don: -3.6416127284648687                            just: -3.6326209953369784
yeah: -3.6416127284648687                           like: -3.6326209953369784
um: -3.6531203353163484                             people: -3.6348911438715175
right: -3.6554378334567112                          yeah: -3.6348911438715175
```

```
oh: -3.657760714872851                                  oh: -3.6394469604073785
really: -3.662422727978662                              really: -3.6417326756882344

-------------------------------------------------------------------------------------------------------------

Top 10 words with highest odds ratio for class 0:   Top 10 words with highest odds ratio for class 1:
attracted: 4.1590030305830386                       wage: 4.343685474630318
compatibility: 4.127254332268458                    wages: 4.233986557373894
relationship: 4.0431712150579155                    inflation: 4.127014437821726
relationships: 3.8782414009758295                   waitresses: 4.007213238009106
attraction: 3.8713209581312578                      minimum: 4.007213238009106
friendship: 3.8713209581312578                      waitress: 3.9701719663287562
communication: 3.8178322731802714                   welfare: 3.93170568550096
marriage: 3.8178322731802705                        salary: 3.817592378733539
dating: 3.791856786777009                           retail: 3.6887595068905705
qualities: 3.737789565506735                        increase: 3.663441698906281

-----------------------------------------------
The accuracy is:  0.9081632653061225
Execution time is: 9.274343s
```

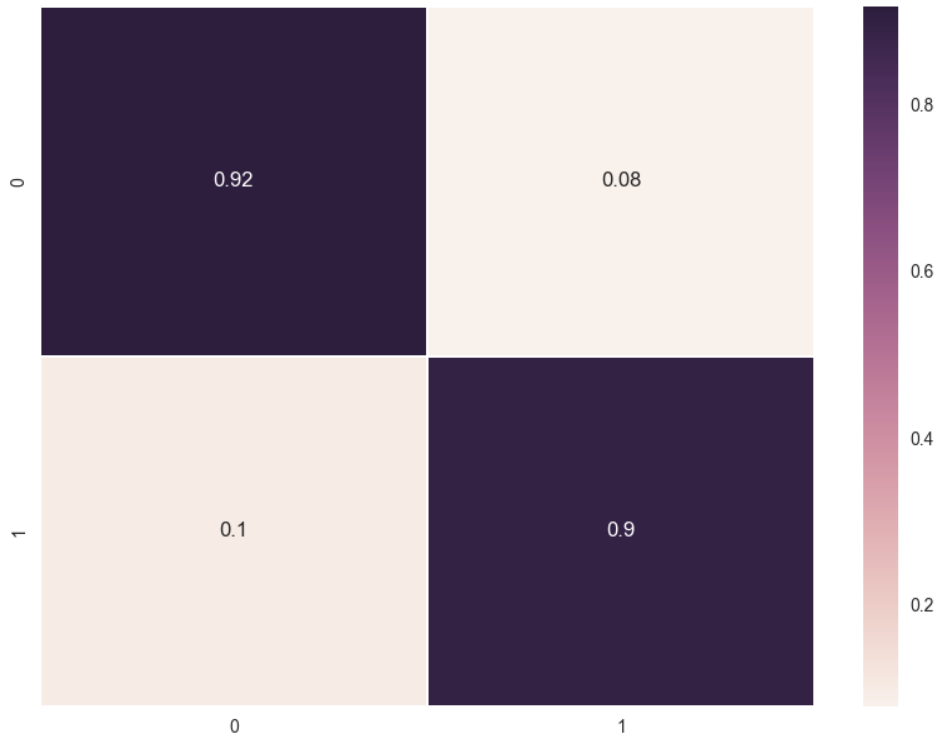


Figure 80: Confusion matrix two topics Bernoulli

B.3 full 40-topic corpus classification results

B.3.1 Multinomial Naive Bayes Classifier with k = 1

```
-------------------------------------------------------------------------------------
Classification rate for each digit:                    Most likely confused topic:
The class 0: 0.8444444444444444                        0: 23
The class 1: 0.8928571428571429                        1: 25
The class 2: 0.9183673469387755                        2: 39
The class 3: 0.8367346938775511                        3: 23
The class 4: 0.9545454545454546                        4: 29
The class 5: 0.5384615384615384                        5: 2
The class 6: 0.08333333333333333                       6: 23
The class 7: 0.8571428571428572                        7: 33
The class 8: 0.8235294117647058                        8: 2
The class 9: 0.8823529411764706                        9: 29
The class 10: 0.9714285714285714                       10: 28
The class 11: 0.8333333333333334                       11: 37
The class 12: 1.0                                      12: 39
The class 13: 1.0                                      13: 39
The class 14: 0.8                                      14: 30
The class 15: 0.8571428571428572                       15: 36
The class 16: 1.0                                      16: 39
The class 17: 0.125                                    17: 3
The class 18: 0.5714285714285715                       18: 36
The class 19: 0.6666666666666666                       19: 34
The class 20: 0.07142857142857142                      20: 29
The class 21: 1.0                                      21: 39
The class 22: 1.0                                      22: 39
The class 23: 0.9                                      23: 37
The class 24: 0.8518518518518519                       24: 0
The class 25: 0.6956521739130435                       25: 23
The class 26: 0.23809523809523808                      26: 26
The class 27: 0.8275862068965517                       27: 37
The class 28: 1.0                                      28: 39
The class 29: 0.9722222222222222                       29: 1
The class 30: 0.76                                     30: 3
The class 31: 0.8076923076923077                       31: 12
The class 32: 0.7083333333333334                       32: 2
The class 33: 0.96875                                  33: 2
The class 34: 0.9705882352941176                       34: 7
The class 35: 0.8461538461538461                       35: 36
The class 36: 1.0                                      36: 39
The class 37: 0.8928571428571429                       37: 23
The class 38: 0.9615384615384616                       38: 1
The class 39: 0.43478260869565216                      39: 23
-------------------------------------------------------------------------------------
Smoothing: k = 1
The accuracy is:  0.8388214904679376
Execution time is: 28.472286s
```

B.3.2 Multinomial Naive Bayes Classifier with k = 0.1

```
-------------------------------------------------------------------------------------
Classification rate for each digit:                    Most likely confused topic:
The class 0: 0.8444444444444444                        topic 0:23
The class 1: 0.8928571428571429                        topic 1:25
The class 2: 0.8775510204081632                        topic 2:39
The class 3: 0.8163265306122449                        topic 3:17
The class 4: 0.9545454545454546                        topic 4:29
The class 5: 1.0                                       topic 5:39
The class 6: 0.9166666666666666                        topic 6:8
The class 7: 0.8571428571428572                        topic 7:33
The class 8: 0.8823529411764706                        topic 8:27
The class 9: 0.8823529411764706                        topic 9:29
The class 10: 0.9714285714285714                       topic 10:28
The class 11: 0.8333333333333334                       topic 11:26
The class 12: 1.0                                      topic 12:39
The class 13: 1.0                                      topic 13:39
The class 14: 0.84                                     topic 14:30
The class 15: 0.8928571428571429                       topic 15:31
The class 16: 1.0                                      topic 16:39
The class 17: 0.5625                                   topic 17:23
The class 18: 0.9285714285714286                       topic 18:15
The class 19: 1.0                                      topic 19:39
The class 20: 0.8571428571428571                       topic 20:15
The class 21: 1.0                                      topic 21:39
The class 22: 0.96875                                  topic 22:31
The class 23: 0.9                                      topic 23:37
The class 24: 0.8518518518518519                       topic 24:31
The class 25: 0.782608695652174                        topic 25:23
The class 26: 0.7142857142857143                       topic 26:37
The class 27: 0.8620689655172413                       topic 27:37
The class 28: 1.0                                      topic 28:39
The class 29: 0.9444444444444444                       topic 29:8
The class 30: 0.88                                     topic 30:27
The class 31: 0.8846153846153846                       topic 31:12
The class 32: 0.875                                    topic 32:31
The class 33: 0.96875                                  topic 33:2
The class 34: 0.9705882352941176                       topic 34:7
The class 35: 0.8461538461538461                       topic 35:36
The class 36: 1.0                                      topic 36:39
The class 37: 0.9285714285714286                       topic 37:27
The class 38: 0.9615384615384616                       topic 38:1
The class 39: 0.6521739130434783                       topic 39:2
-------------------------------------------------------------------------------------
Smoothing k = 0.1
The accuracy is:  0.8994800693240901
Execution time is: 27.553080s
```

B.3.3 Bernoulli Naive Bayes Classifier with k = 1

```
-------------------------------------------------------------------------------------
Classification rate for each digit:                    Most likely confused topic:
The class 0: 0.8444444444444444                        topic 0:23
The class 1: 0.8928571428571429                        topic 1:23
The class 2: 0.9183673469387755                        topic 2:3
The class 3: 0.8367346938775511                        topic 3:23
The class 4: 0.9545454545454546                        topic 4:3
The class 5: 0.0                                       topic 5:3
The class 6: 0.0                                       topic 6:3
The class 7: 0.17857142857142858                       topic 7:7
The class 8: 0.2647058823529412                        topic 8:8
The class 9: 0.5882352941176471                        topic 9:2
The class 10: 0.9428571428571428                       topic 10:28
The class 11: 0.1111111111111111                       topic 11:2
The class 12: 0.21739130434782608                      topic 12:12
The class 13: 0.0                                      topic 13:0
The class 14: 0.2                                      topic 14:14
The class 15: 0.32142857142857145                      topic 15:2
The class 16: 0.6521739130434783                       topic 16:3
The class 17: 0.0                                      topic 17:3
The class 18: 0.0                                      topic 18:23
The class 19: 0.0                                      topic 19:2
The class 20: 0.0                                      topic 20:39
The class 21: 0.8666666666666667                       topic 21:2
The class 22: 0.9375                                   topic 22:2
The class 23: 0.975                                    topic 23:3
The class 24: 0.22222222222222224                      topic 24:24
The class 25: 0.08695652173913043                      topic 25:1
The class 26: 0.0                                      topic 26:37
The class 27: 0.6896551724137931                       topic 27:23
The class 28: 1.0                                      topic 28:39
The class 29: 0.7222222222222222                       topic 29:2
The class 30: 0.32                                     topic 30:30
The class 31: 0.5                                      topic 31:0
The class 32: 0.0                                      topic 32:29
The class 33: 0.96875                                  topic 33:2
The class 34: 0.9705882352941176                       topic 34:1
The class 35: 0.576923076923077                        topic 35:2
The class 36: 1.0                                      topic 36:39
The class 37: 0.7142857142857143                       topic 37:23
The class 38: 0.5384615384615384                       topic 38:29
The class 39: 0.043478260869565216                     topic 39:2
-------------------------------------------------------------------------------------
Smoothing k = 1
The accuracy is:  0.5953206239168111
Execution time is: 4663.196538s
```

B.3.4 Bernoulli Naive Bayes Classifier with k = 0.1

```
-------------------------------------------------------------------------------------
Classification rate for each digit:                    Most likely confused topic
The class 0: 0.8222222222222222                        topic 0:23
The class 1: 0.8928571428571429                        topic 1:25
The class 2: 0.8775510204081632                        topic 2:39
The class 3: 0.8163265306122449                        topic 3:39
The class 4: 0.9318181818181819                        topic 4:29
The class 5: 0.8461538461538461                        topic 5:8
The class 6: 0.6666666666666667                        topic 6:9
The class 7: 0.8214285714285715                        topic 7:33
The class 8: 0.8529411764705882                        topic 8:29
The class 9: 0.8529411764705882                        topic 9:36
The class 10: 0.9714285714285714                       topic 10:28
The class 11: 0.8333333333333334                       topic 11:8
The class 12: 1.0                                      topic 12:39
The class 13: 0.9375                                   topic 13:28
The class 14: 0.84                                     topic 14:3
The class 15: 0.7500000000000001                       topic 15:31
The class 16: 0.9565217391304348                       topic 16:35
The class 17: 0.25                                     topic 17:17
The class 18: 0.8571428571428572                       topic 18:39
The class 19: 1.0                                      topic 19:39
The class 20: 0.35714285714285715                      topic 20:20
The class 21: 1.0                                      topic 21:39
The class 22: 1.0                                      topic 22:39
The class 23: 0.925                                    topic 23:37
The class 24: 0.8148148148148148                       topic 24:23
The class 25: 0.782608695652174                        topic 25:23
The class 26: 0.5238095238095238                       topic 26:37
The class 27: 0.8620689655172413                       topic 27:37
The class 28: 1.0                                      topic 28:39
The class 29: 0.9166666666666666                       topic 29:15
The class 30: 0.88                                     topic 30:27
The class 31: 0.8846153846153846                       topic 31:12
The class 32: 0.5                                      topic 32:2
The class 33: 0.96875                                  topic 33:29
The class 34: 1.0                                      topic 34:39
The class 35: 0.8076923076923077                       topic 35:16
The class 36: 1.0                                      topic 36:39
The class 37: 0.8571428571428571                       topic 37:27
The class 38: 0.9615384615384616                       topic 38:1
The class 39: 0.6086956521739131                       topic 39:2
-------------------------------------------------------------------------------------
Smoothing k = 0.1
The accuracy is:  0.8604852686308492
Execution time is: 4619.236173s
```

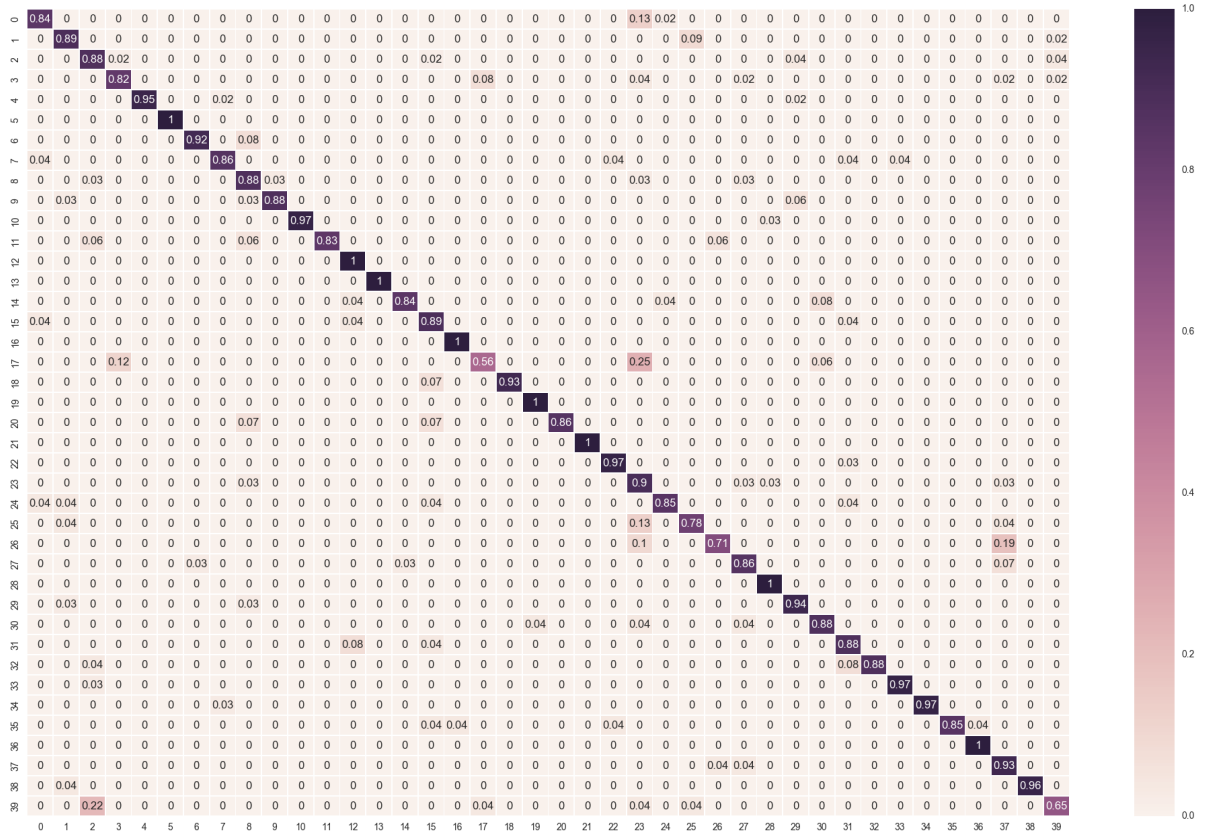


Figure 81: Confusion matrix 40 topics Multinomial

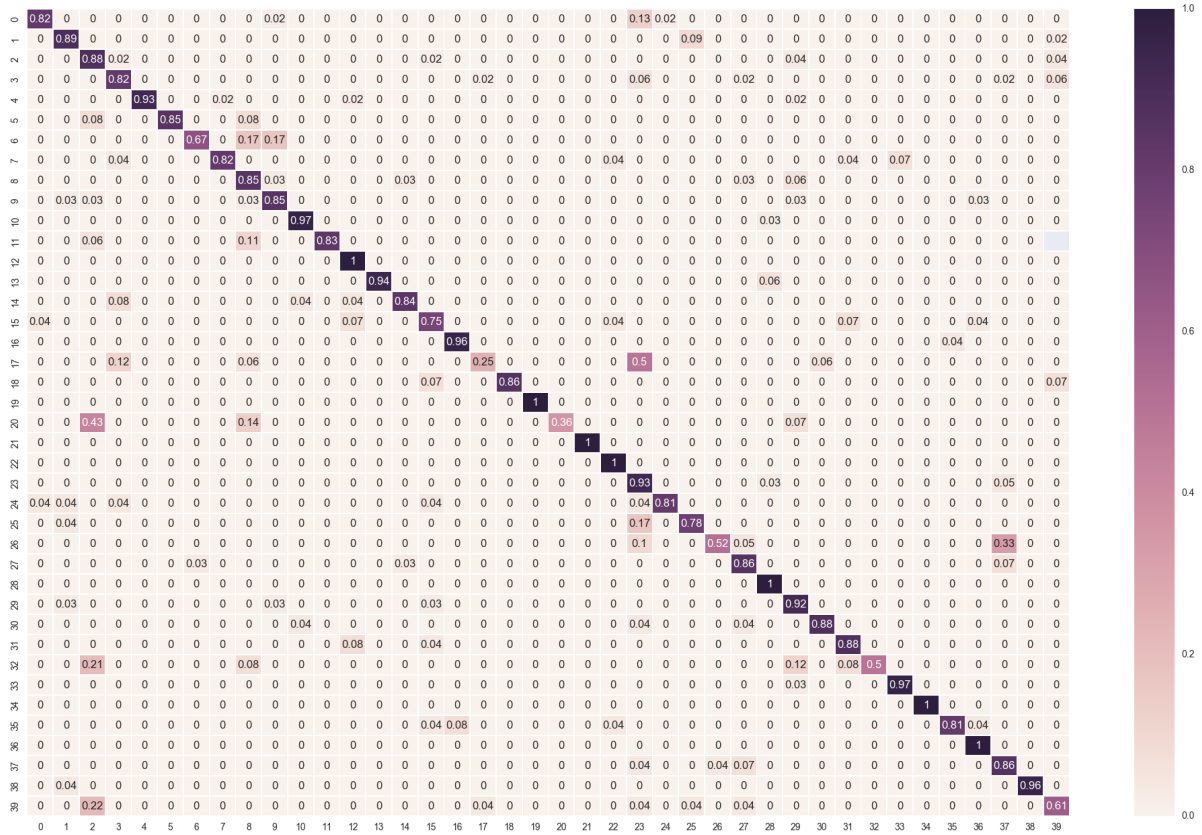


Figure 82: Confusion matrix 40 topics Bernoulli

References

[1] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[2] Wes McKinney. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython.* " O'Reilly Media, Inc.", 2012.

[3] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial intelligence: a modern approach*, volume 2. Prentice hall Upper Saddle River, 2003.

[4] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[5] Wikipedia. Naive bayes classifier — wikipedia, the free encyclopedia, 2016. [Online; accessed 1-November-2016].