CS 551: Pattern Recognition

Assignment #2:

Parametric non-Parametric Models

Omer Gozuacik

21301328

Question 1:

In this question, I implemented the EM algorithm for Gaussian mixtures with (1) same spherical covariance matrix, (2) different diagonal covariance matrix and (3) arbitrary covariance matrix for each component. I implemented EM algorithm as it is specified in the slides.

$$\boldsymbol{X} \text{ of } N \text{ feature vectors } \boldsymbol{x}_n, n = 1, \ldots, N,$$

$$\text{set of } K \text{ Gaussian component pdfs } \mathcal{N}_k \triangleq \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

$$K \text{ mixture weights } P(k), \quad k = 1, \ldots, K$$

$$p_{kn} \triangleq P(k|\boldsymbol{x}_n) = \frac{p(\boldsymbol{x}_n|k)P(k)}{p(\boldsymbol{x}_n)},$$

$$\hat{P}(k) = \frac{1}{N}\sum_{n=1}^{N} p_{kn}, \qquad \hat{\boldsymbol{\mu}}_k = \frac{\sum_n p_{kn}\boldsymbol{x}_n}{\sum_n p_{kn}},$$

$$\hat{\boldsymbol{\Sigma}}_k = \frac{\sum_n p_{kn}(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_k)^T}{\sum_n p_{kn}} \quad \textbf{(full cov)}$$

$$\hat{\sigma}_{ik}^2 = \frac{\sum_n p_{kn}(x_{in} - \hat{\mu}_{ik})^2}{\sum_n p_{kn}}, \quad i = 1, \ldots, D \quad \textbf{(diag cov)}$$

$$\hat{\sigma}_k^2 = \frac{\sum_n p_{kn} \|\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_k\|^2}{D\sum_n p_{kn}} \quad \textbf{(spherical cov)}$$

Figure 1: EM Algorithm

In data there are 3 classes and each one of them has 1000 observations. Each of these classes are randomly permutated and then 500 observations are selected from each. Then these samples of different classes are individually trained with EM algorithm with different cluster sizes.

By inspection, most of the data is around (20,20), therefore I assigned starting point (means) for every cluster in GM to (20,20). For sigma, I assigned them to [20 0 ; 0 20] randomly. Even though Sigmas are set random unlike previous case, it is in reasonable

range. If both Sigmas and center points(mu) were not close to the ideal range, likelihoods would be very low and there would be more error due to underflow. In my opinion, center points can be assigned randomly from the data points in the training set.

As likelihood is formed by very small numbers, log likelihood is used instead in order to prevent underflow.

EM algorithm is an iterative process. Whenever means for Gaussians are same as the previous step, the repetition is stopped as process converges. (|prevMu-Mu|<10e-15)

Classification is made by comparing the scores (class conditional probability) of a point with respect to pdfs acquired from the EM for each class. E.g: A data point is classified as class 1 if its score for class 1 is larger than the other two.

As data for classes are not shaped perfectly spherical or ellipsoid in one direction (only stretched in x or y), accuracy was significantly lower for spherical and diagonal covariances when number of Gaussians per class is 1.

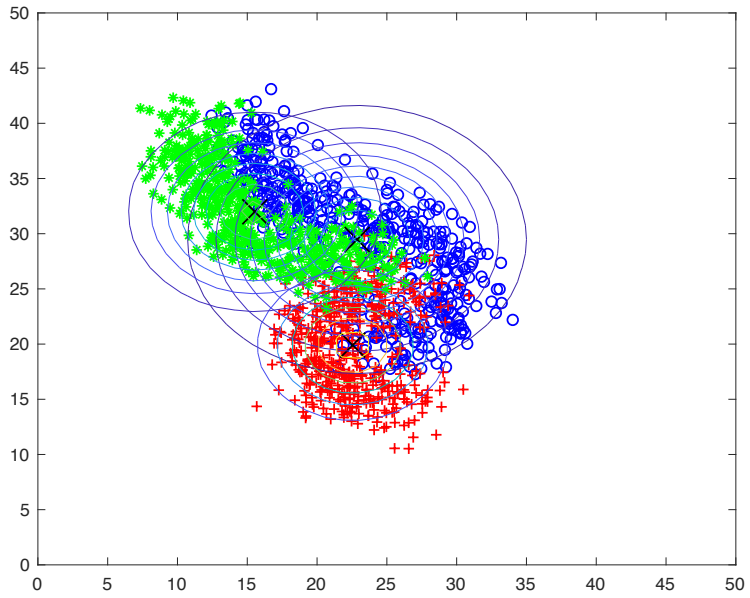

Figure 2: Plot for data and Gaussians when spherical covariance is used and number of Gaussians per class is 1

- Test and train accuracies are equal to 68.67% and 68.60% respectively.

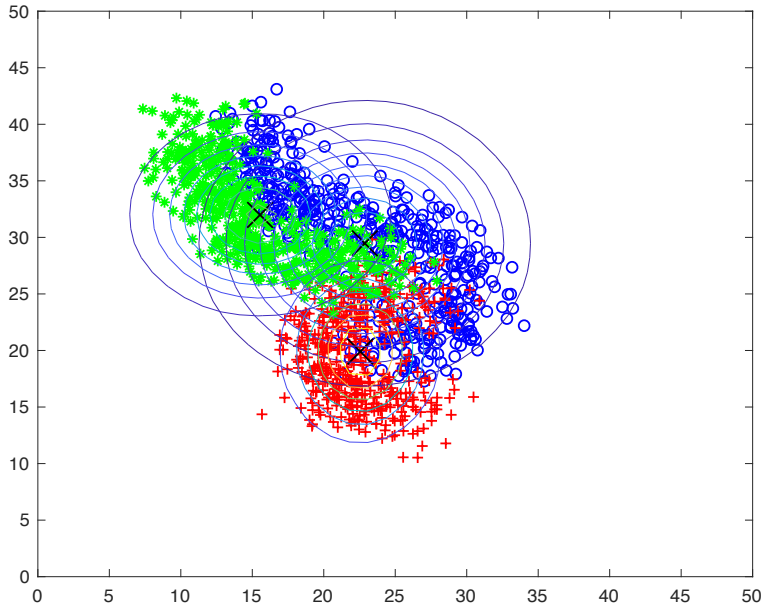

Figure 3: Plot for data and Gaussians when diagonal covariance is used and number of Gaussians per class is 1

- Test and train accuracies are equal to 70.27% and 70.93% respectively.

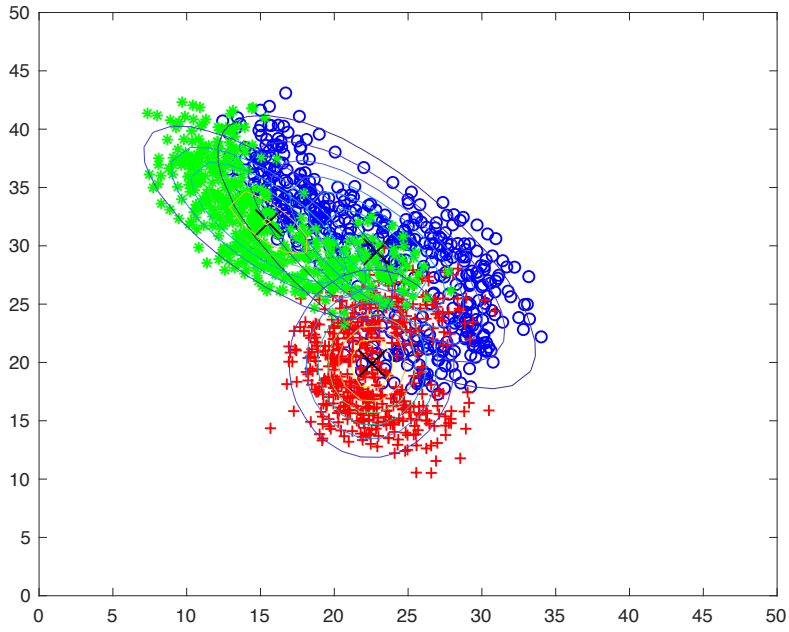

Figure 4: Plot for data and Gaussians when arbitrary covariance is used and number of Gaussians per class is 1

- Test and train accuracies are equal to 76.13% and 77.40% respectively.

As number of Gaussians per class increases, both test and train accuracies increase for all types of sigmas. When number of components is 3 per class, accuracies for arbitrary covariance is around 85% and 82% for others. After this point, when number of components is increased, accuracy is almost same for arbitrary covariance but increases slightly for the others.

Covariance type and Accuracies (%)		Number of Components per Class		
		1	2	3
Spherical Covariance	Train Accuracy	68.60	82.00	83.73
	Test Accuracy	68.67	78.73	82.20
Diagonal Covariance	Train Accuracy	70.93	80.87	83.40
	Test Accuracy	70.27	78.67	81.53
Full Covariance	Train Accuracy	77.40	85.80	86.13
	Test Accuracy	76.13	84.60	85.67

Table 1: Accuracies for different covariance types and number of components per class

I set number of components per class to 3 after this point. The results are shown below.

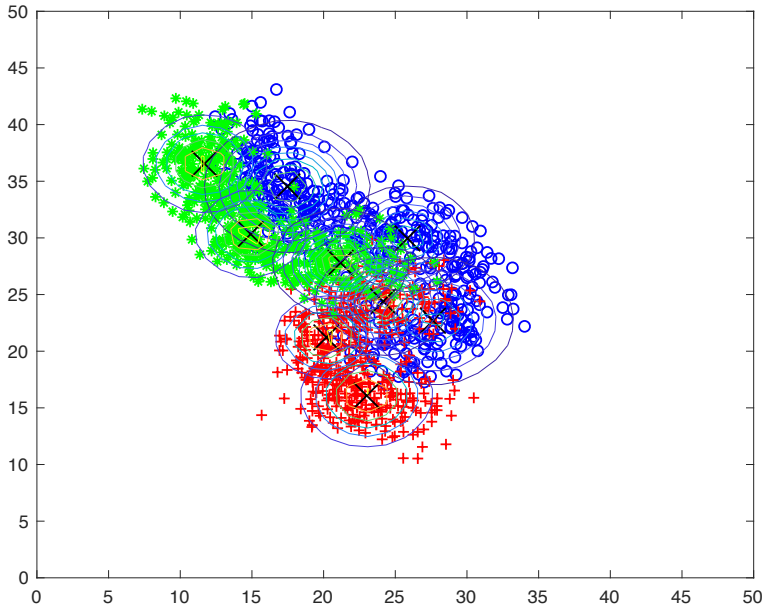

Figure 5: Plot for data and Gaussians when spherical covariance is used and number of Gaussians per class is 3

	Mu (X,Y)	Sigma	Weight
Class 1, Gaussian 1	25.8497 29.9572	5.5288 0 0 5.5288	0.2378
Class 1, Gaussian 2	17.4729 34.5409	4.6947 0 0 4.6947	0.4268
Class 1, Gaussian 3	27.6249 22.6914	4.8723 0 0 4.8723	0.3354
Class 2, Gaussian 1	24.1564 24.3933	7.4115 0 0 7.4115	0.2884
Class 2, Gaussian 2	20.1954 21.2154	3.3690 0 0 3.3690	0.2776
Class 2, Gaussian 3	23.0040 16.0823	3.6232 0 0 3.6232	0.4340
Class 3, Gaussian 1	11.6967 36.5476	8.4319 0	0.3882

		0 8.4319	
Class 3, Gaussian 2	14.9102 30.3354	5.5538 0 0 5.5538	0.3123
Class 3, Gaussian 3	21.1839 27.7967	4.7689 0 0 4.7689	0.2996

Table 2: Estimated parameters when spherical covariance is used

The final log-likelihood for train data = -8.0416 x 10^3

		Predicted (train,test)		
		1	2	3
Actual (train,test)	1	369, 340	50, 47	81, 113
	2	40, 44	440, 430	20, 26
	3	34, 25	19, 12	447, 463

Table 3: Combined version of confusion matrix for train&test data when spherical covariance is used

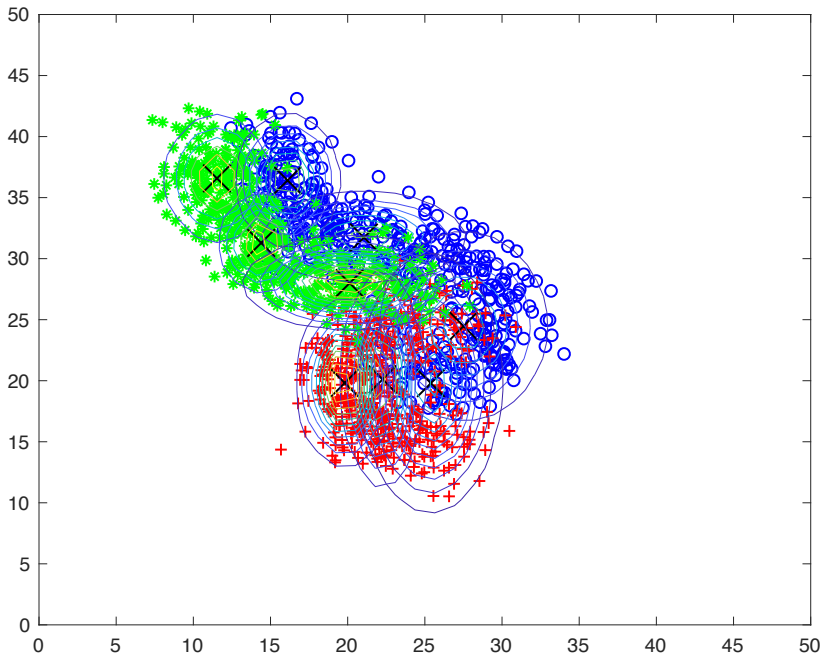

Figure 6: Plot for data and Gaussians when diagonal covariance is used and number of Gaussians per class is 3

	Mu (X,Y)	Sigma	Weight
Class 1, Gaussian 1	21.0499 31.7298	9.9792 0 0 3.0904	0.2979
Class 1, Gaussian 2	16.1050 36.4110	4.1649 0 0 23.0338	0.2381
Class 1, Gaussian 3	27.5087 24.4724	2.9003 0 0 7.0524	0.4641
Class 2, Gaussian 1	25.3925 19.8012	2.4752 0 0 6.6045	0.3528
Class 2, Gaussian 2	22.2639 20.1102	1.3446 0 0 16.7677	0.3166
Class 2, Gaussian 3	19.8104 19.8164	1.7794 0 0 4.1069	0.3306
Class 3, Gaussian 1	11.5297 36.5762	7.4475 0 0 15.5658	0.3726
Class 3, Gaussian 2	14.3940 31.2865	1.8525 0 0 9.3149	0.2410
Class 3, Gaussian 3	20.1277 27.9967	9.5227 0 0 2.9037	0.3864

Table 4: Estimated parameters when diagonal covariance is used

The final log-likelihood for train data = -8.0032 x 10^3

		Predicted (train,test)		
Actual (train,test)		1	2	3
	1	385, 364	57, 54	58, 82
	2	49, 65	433, 410	18, 25

	3	55, 34	12, 17	433, 449

Table 5: Combined version of confusion matrix for train&test data when diagonal covariance is used

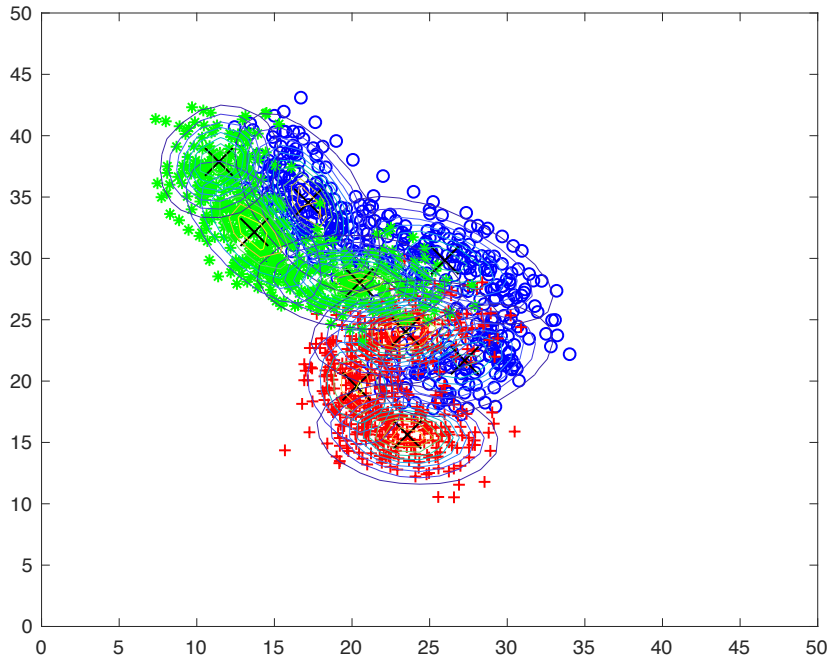

Figure 7: Plot for data and Gaussians when arbitrary covariance is used and number of Gaussians per class is 3

	Mu (X,Y)	Sigma	Weight
Class 1, Gaussian 1	25.9709 29.8371	11.5943 -5.2630 -5.2630 6.3357	0.3368
Class 1, Gaussian 2	17.1773 34.5442	2.3541 -0.6846 -0.6846 5.8832	0.3968
Class 1, Gaussian 3	27.2641 21.6810	9.2337 -0.8873 -0.8873 3.0218	0.2707
Class 2, Gaussian 1	20.2846 19.5663	4.4190 -3.9951 -3.9951 10.8742	0.2967

Class 2, Gaussian 2	23.4663 24.0198	7.2276 0.9579 0.9579 3.5173	0.3702
Class 2, Gaussian 3	23.5686 15.6299	4.7938 -4.0023 -4.0023 8.5020	0.3331
Class 3, Gaussian 1	20.4873 28.0245	7.6167 2.3064 2.3064 4.8805	0.3452
Class 3, Gaussian 2	13.7072 32.1408	6.9417 -0.7697 -0.7697 3.2904	0.4337
Class 3, Gaussian 3	11.4234 37.8657	3.2179 0.5006 0.5006 4.7665	0.2212

Table 6: Estimated parameters when arbitrary covariance is used

The final log-likelihood for train data = -7.8847 x 10^3

		Predicted (train,test)		
Actual (train,test)		1	2	3
	1	408, 383	37, 33	55, 84
	2	35, 27	448, 451	17, 22
	3	42, 30	22, 19	436, 451

Table 7: Combined version of confusion matrix for train&test data when arbitrary covariance is used

With arbitrary covarianced models, more complex shaped data distributions can be fitted and it can be seen from the figures above. Therefore, accuracies are higher as well and the likelihood is larger than the other two. If number of components is increased, other two types of covariances also provide good results for this data but still arbitrary is better. When number of components per class is increased over 3, additional components have very small weight compared to other 3. On one case when I ran for 5 components. The weight distribution was almost like 0.31 0.31 0.31 0.045 0.045. Increasing the number of

components makes us have slightly better results but I believe it does not worth its computational cost.

Question 2:

In this question, the class conditional probabilities are estimated using histograms. As in previous question, data is randomly permutated and 500 observations from each class is chosen to train the classifier. I implemented a code to run for different bin counts per axis and choose the best bin count for test accuracy in the end. Data is divided into bins in between 0-50 for both x&y axes. As data seems to be in this range from the scatter plot, I set these values as my limits. One other way can be setting minimums and maximums for the data in these axes. Classification is made according to the count of components in a specific bin. If any class has more components in a bin than the others, that bin is classified as that class. E.g: A new point is classified as class 1 if class 1 has more samples in the bin that point belongs to compared to class 2&3. If a bin does not contain any observations from these 3 classes, it is not classified (classified as 0).

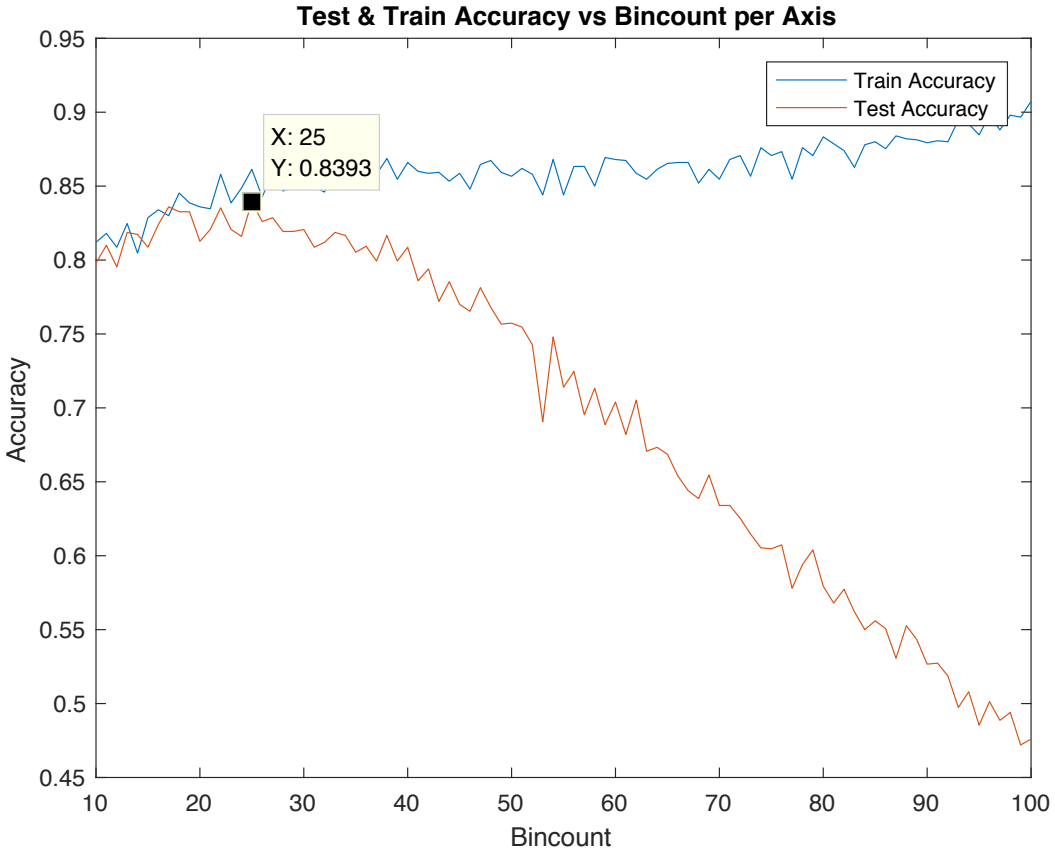

Figure 8: Bin count per axis vs train&test accuracy

As it can be seen from the graph above, test accuracy increases up to a point when bin count is increased. When bin count is equal to 25, test accuracy is maximized. After that point as bin count increases, train accuracy continues to increase but test accuracy decreases as it causes overfitting. The number of bins which do not contain any data from train data increases and these bins are not classified as any class. As they remain unclassified, test accuracy decreases. If we increase the bin count so that every point in train set has one bin, train accuracy would be equal to 100%. As it can be seen from the figure above, the classifier performs best for test data when bin count per axis is equal to 25. The total bin size for that number is 25x25 = 625. The classification matrix and histogram for that bin count are shown below.

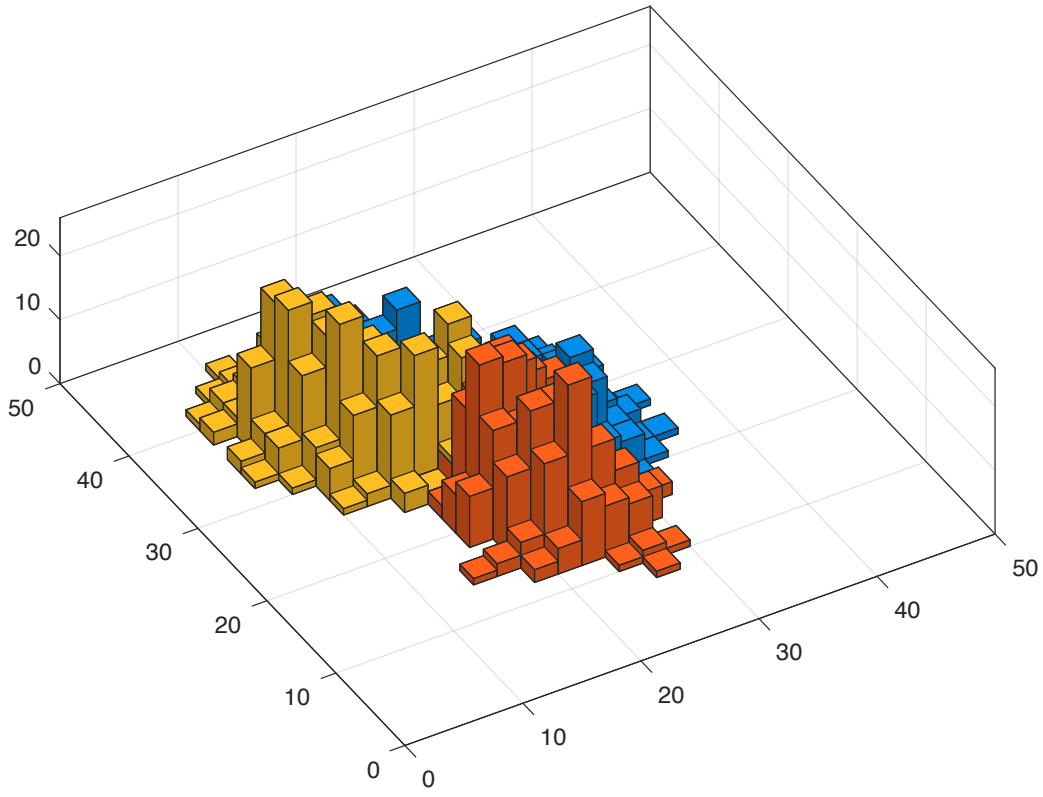

Figure 9: Histogram plot when bincount per axis is 25

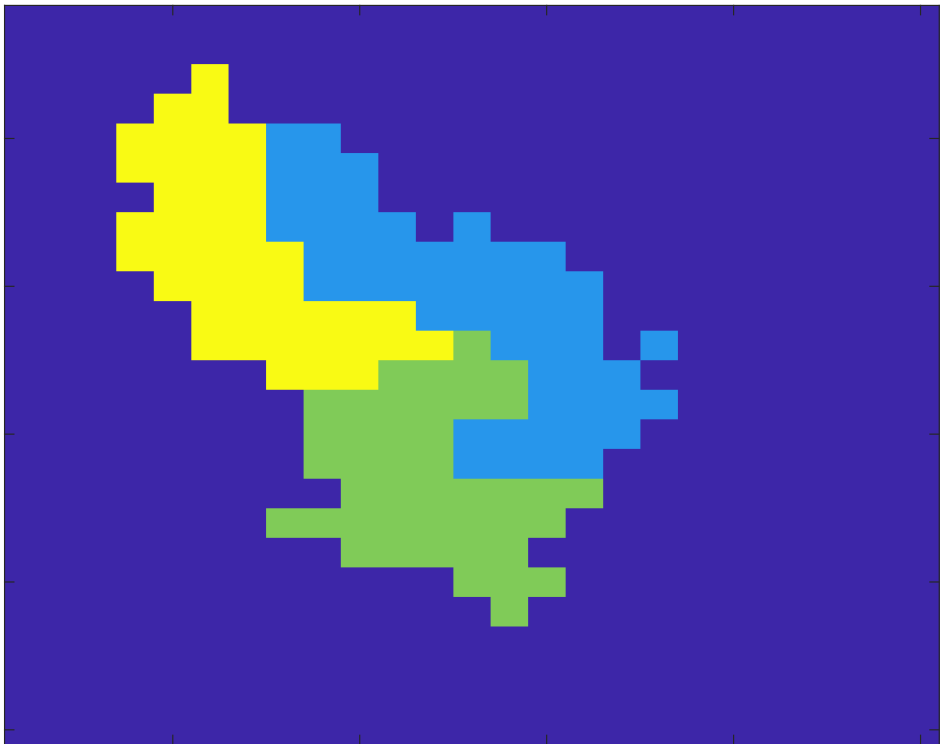

Figure 10: Classification plot when bin count per axis is 25 (purple areas are labelled as not classified)

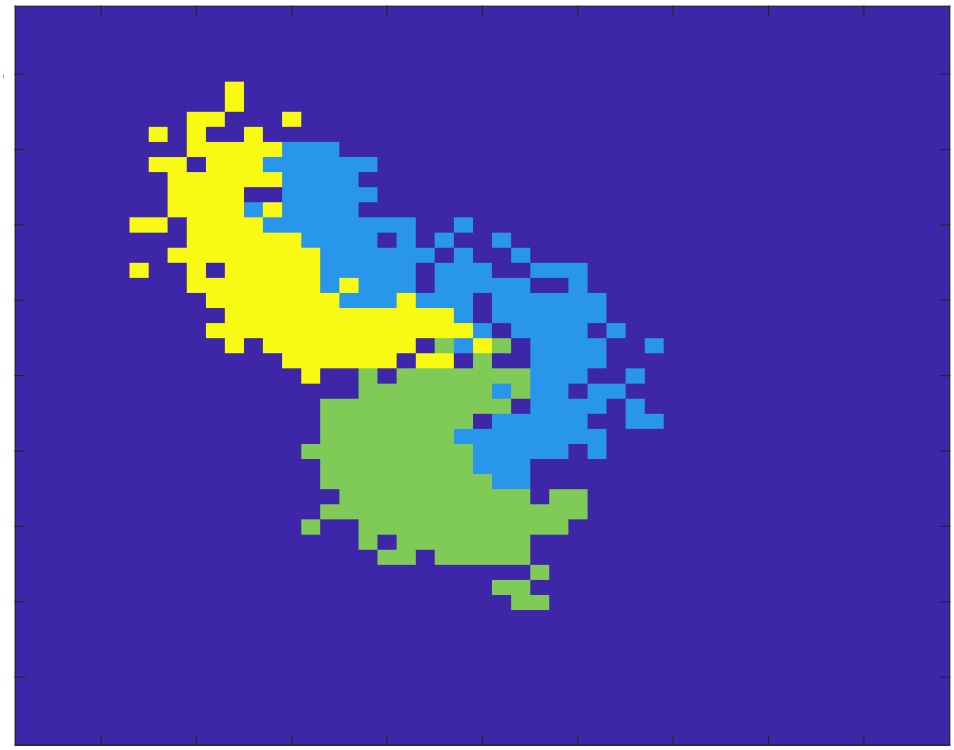

Figure 11: Classification plot when bin count per axis is 50 (purple areas are labelled as not classified)

As it can be seen from the figure above, when bin count per axis is high, there are bins labelled not classified even though they should be labelled because they are in the range for that class. This is the result of overfitting the train data. On the other hand when bin count is optimal (25), unlabelled bins are acceptable in general.

Test accuracy when bin count per axis is 25 = 85.93%

Train accuracy when bin count per axis is 25 = 86.13%

		Predicted		
Actual		1	2	3
	1	397	42	61
	2	22	462	16
	3	51	16	433

Table 8: Confusion matrix for train data when bin count per axis is 25

			Predicted		
Actual		not-labelled	1	2	3
	not-labelled	0	0	0	0
	1	9	412	40	39
	2	4	32	440	24
	3	1	66	26	407

Table 9: Confusion matrix for test data when bin count per axis is 25

In the end, the results were consistent with my expectations. When bin count per axis is high, classifier overfits the train data and this ends up low test accuracy as classifier does not label empty bins. When bin counts for 10 to 100 is ran, best is found to be 25 which contains 625 bins in total.