Pattern Recognition Lab #2 - Model Estimation and Discriminant Functions

Drew Gross - 20328775
Jake Nielsen - 20338042

Introduction

In this lab we investigate various ways of finding class boundaries from data. We apply parametric approaches with Gaussian, exponential, and uniform distributions, and we apply the Parzen Window and Sequential Discriminant type of non-parametric approaches.

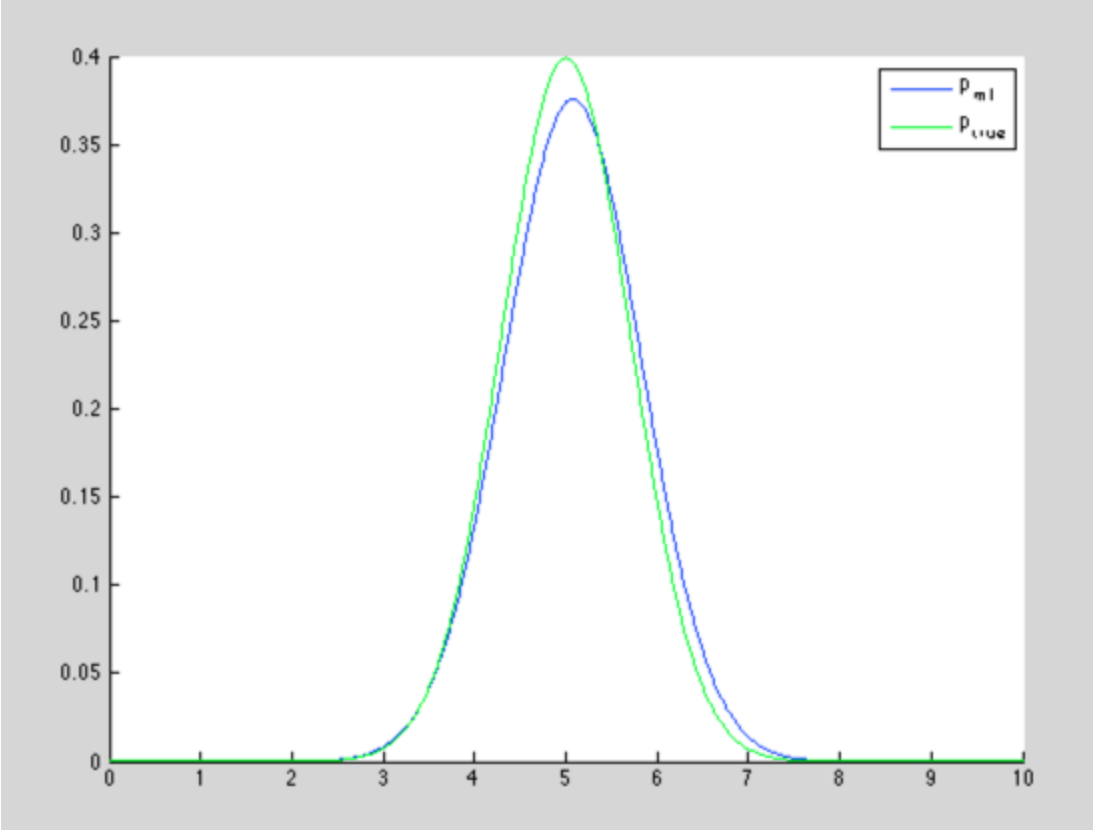
Model Estimation - 1D case

Parametric Estimation - Gaussian

Maximum Likelihood of *a* using Gaussian distribution. The formulae for determining θ and σ^2 is the one from the slides presented in class.

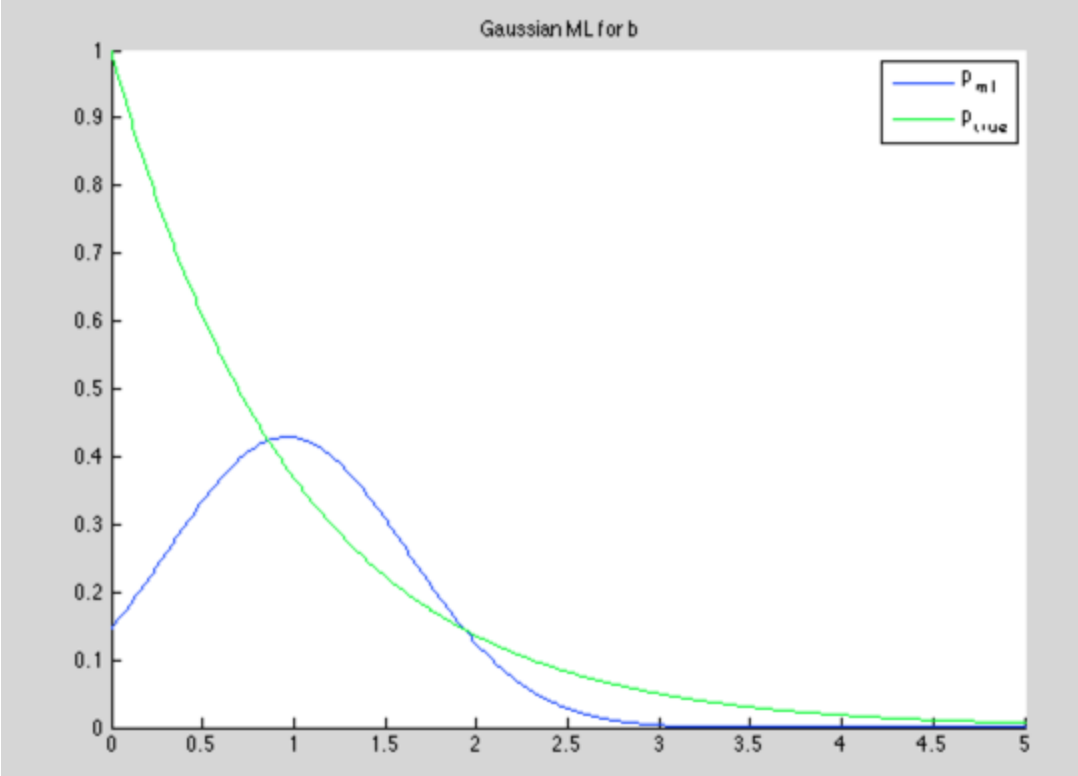
$\theta_a = 5.0763$
$\sigma_a^2 = 1.1274$

The ML mean is very close to the real mean, and differs by only 1.5%. The ML variance is not close, and differs from the real variance by 12.7%.



Using ML with a Gaussian distribution for *b* performed poorly, as expected

$\theta_b = 0.9633$
$\sigma_b^2 = 0.8643$



Parametric Estimation - Exponential

The formula for determining λ using ML with an exponential distribution is as follows:

$$P(\{x_i\}|\theta) = \prod_{i=1}^{N} \theta e^{-\theta x_i}$$

$$l(\theta) = \log \prod_{i=1}^{N} \theta e^{-\theta x_i} = \sum_{i=1}^{N} \log(\theta e^{-\theta x_i}) = \sum_{i=1}^{N} (\log(\theta) - \theta x_i)$$

$$= N \log\theta - \theta \sum_{i=1}^{N} x_i$$

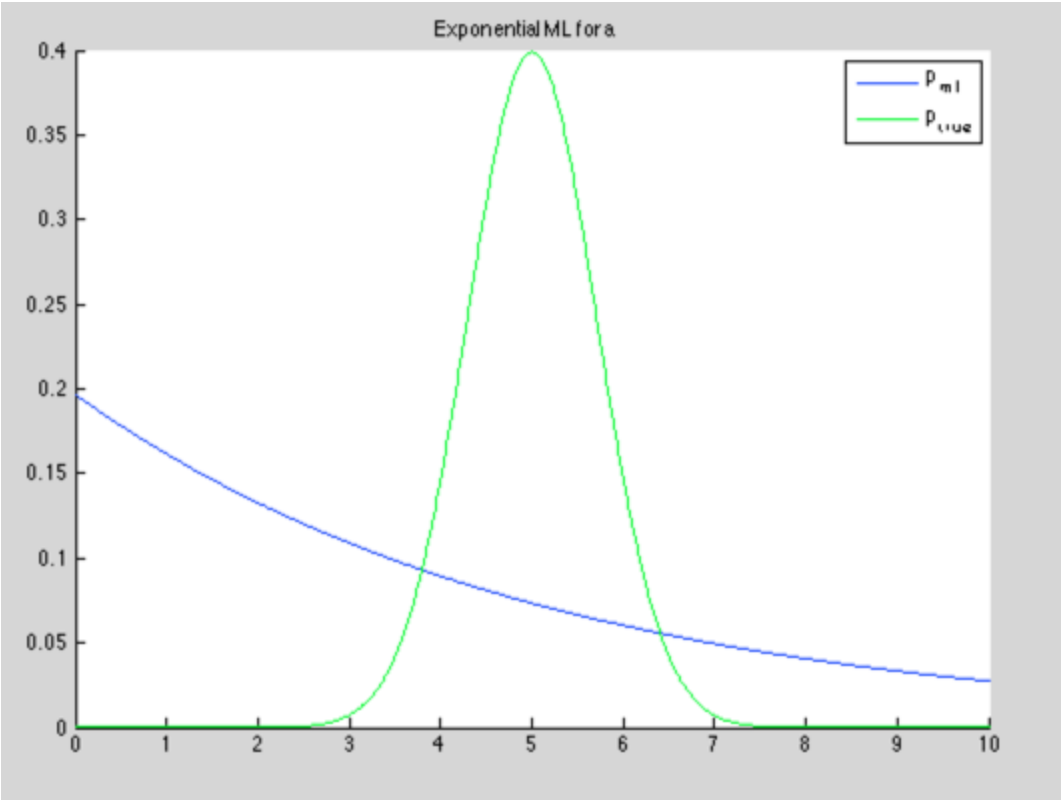
$$\frac{\partial}{\partial\theta} l(\theta) = \frac{N}{\theta} - \sum_{i=1}^{N} x_i = 0$$

$$\frac{N}{\theta} = \sum_{i=1}^{N} x_i$$

$$\frac{N}{\sum_{i=1}^{N} x_i} = \theta$$

Using this method for *a* performed poorly, as expected.

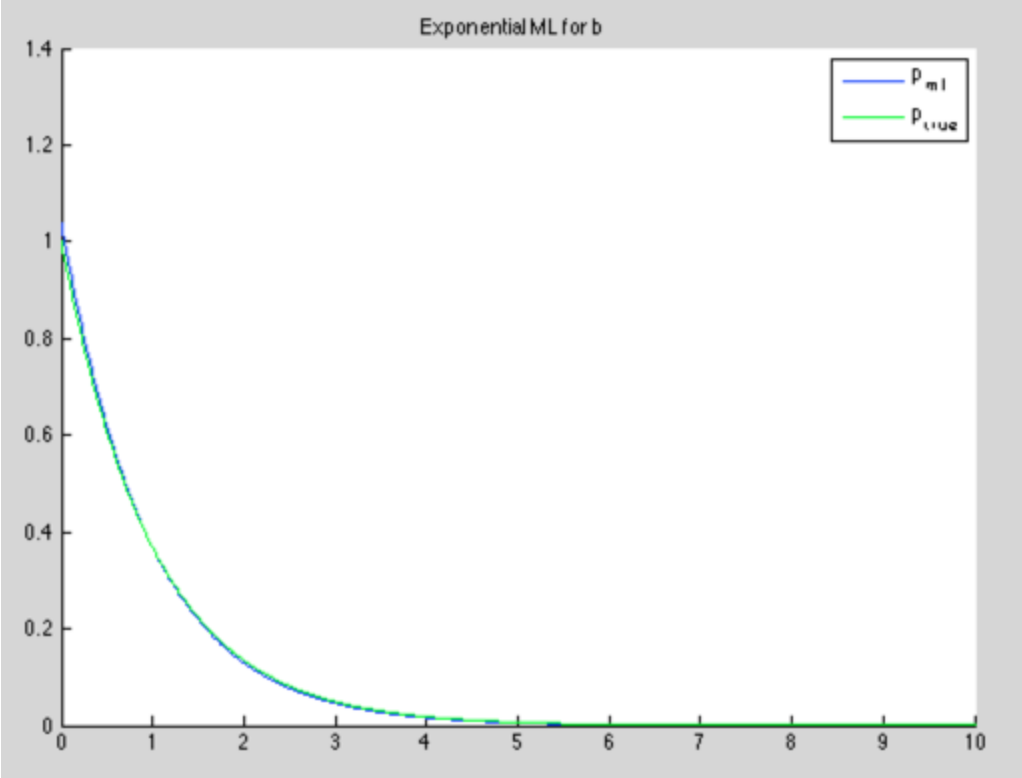
$\lambda_a = 0.1970$



Using ML for *b* was much more successful

$\lambda_b = 1.0381$

The learned lambda differs from the true lambda by 3.81%



Parametric Estimation - Uniform

The formula for determining *a* and *b* using ML is as follows:

$$P(\{x_i\}|\theta) = \prod_{i=1}^{N}\left(\frac{1}{\theta_2-\theta_1} \text{ for } \theta_1 \le x_i \le \theta_2,\ 0 \text{ else}\right)$$

$$l(\theta) = \sum_{i=1}^{N}\left(\log\left(\frac{1}{\theta_2-\theta_1}\right) \text{for } \theta_1 \le x_i \le x_2,\ \log(0) \text{else}\right)$$

$\log(0)$ can't be allowed, so $[a, b$ must contain all data points a least

$$l(\theta) = \sum_{i=1}^{N}\log\left(\frac{1}{\theta_2-\theta_1}\right)$$

$$= N \cdot \log\left(\frac{1}{\theta_2-\theta_1}\right)$$

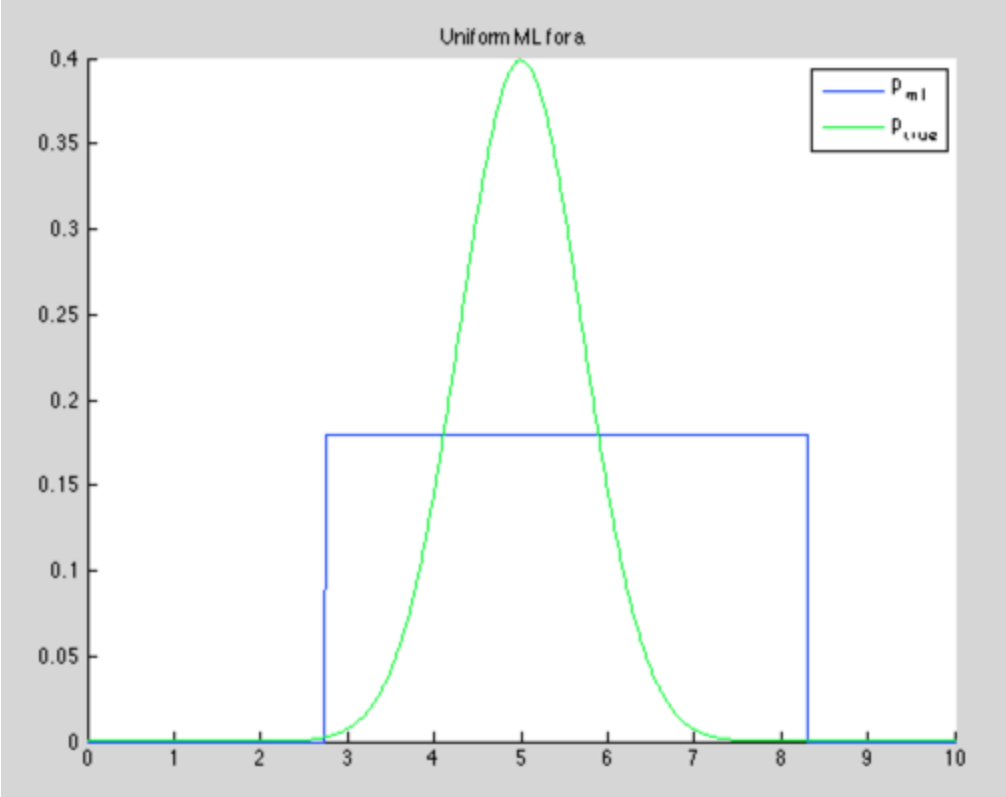
$$\frac{dl(\theta)}{d\theta} = 0 = \left[\frac{N}{\theta_2-\theta_1}, \frac{N}{\theta_1-\theta_2}\right]$$

$\theta_2 > \theta_1$, so minimize θ_2 and maximize θ_1 while containing all points

$$\theta_2 = \max(x_i)$$
$$\theta_1 = \min(x_i)$$

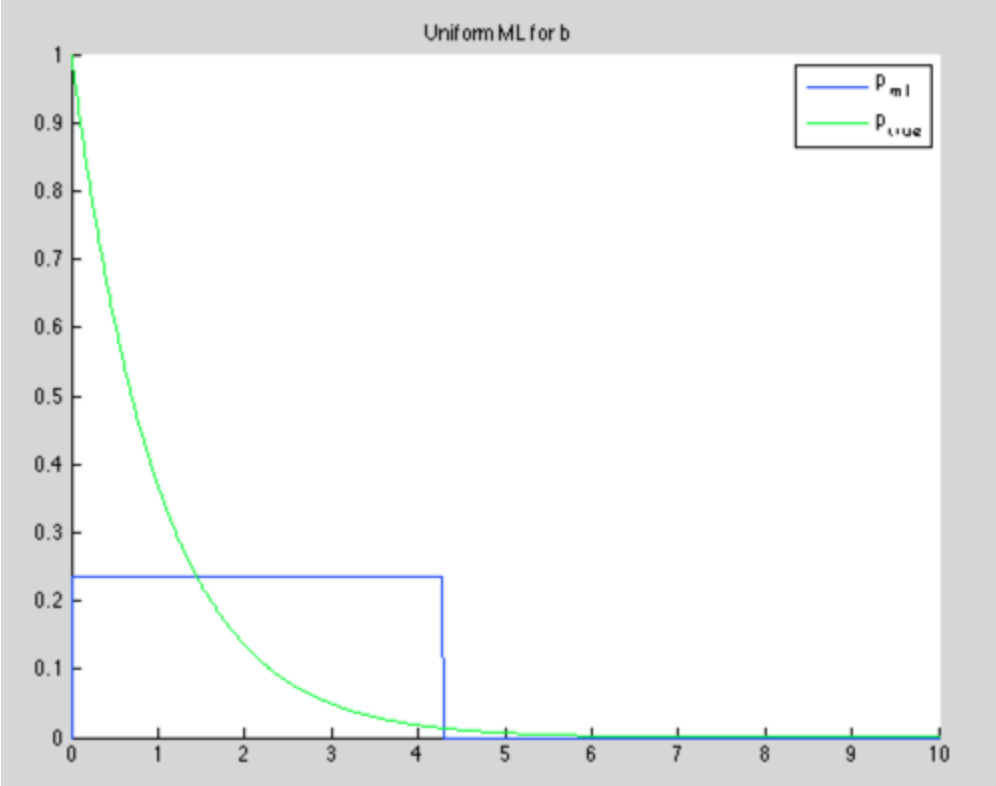
For data set *a* this results in

$a = 2.7406$
$b = 8.3079$



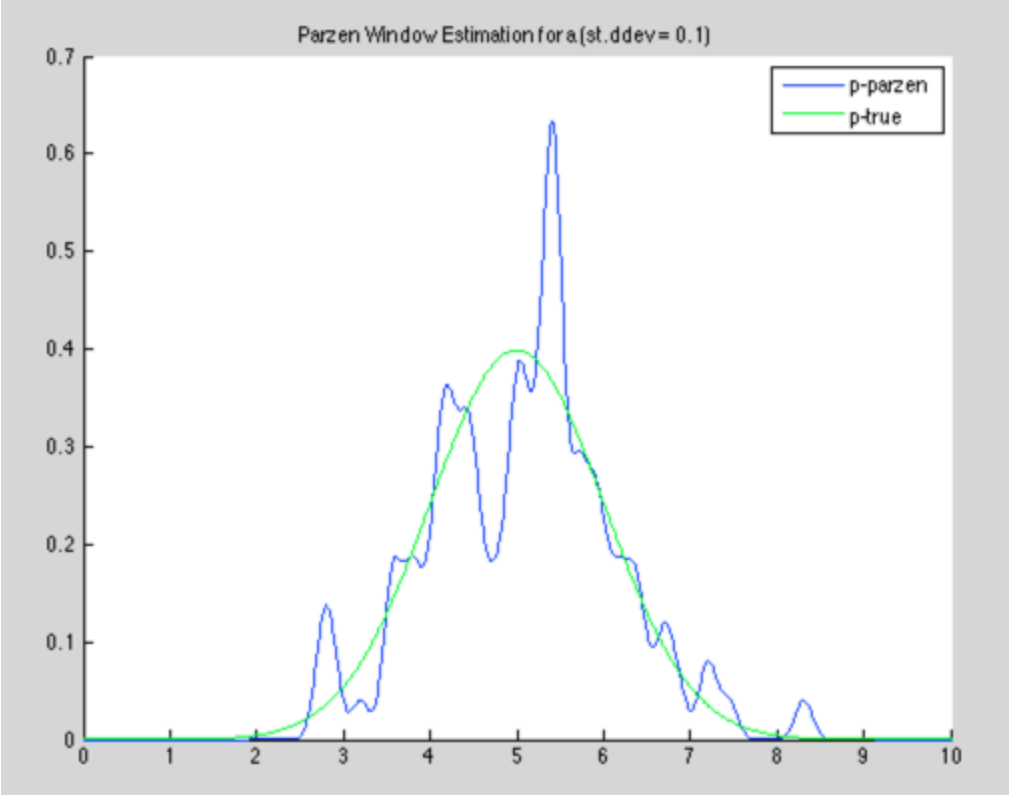
For data set *b* this results in

$a = 0.0143$
$b = 4.2802$

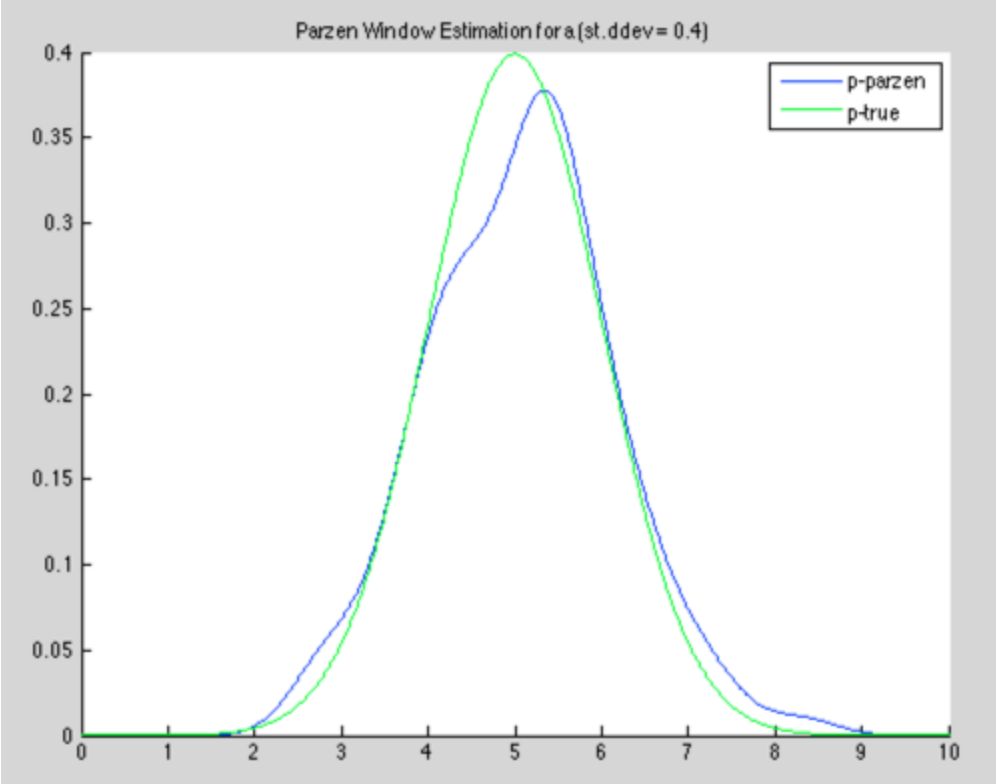


Parzen Window Estimation

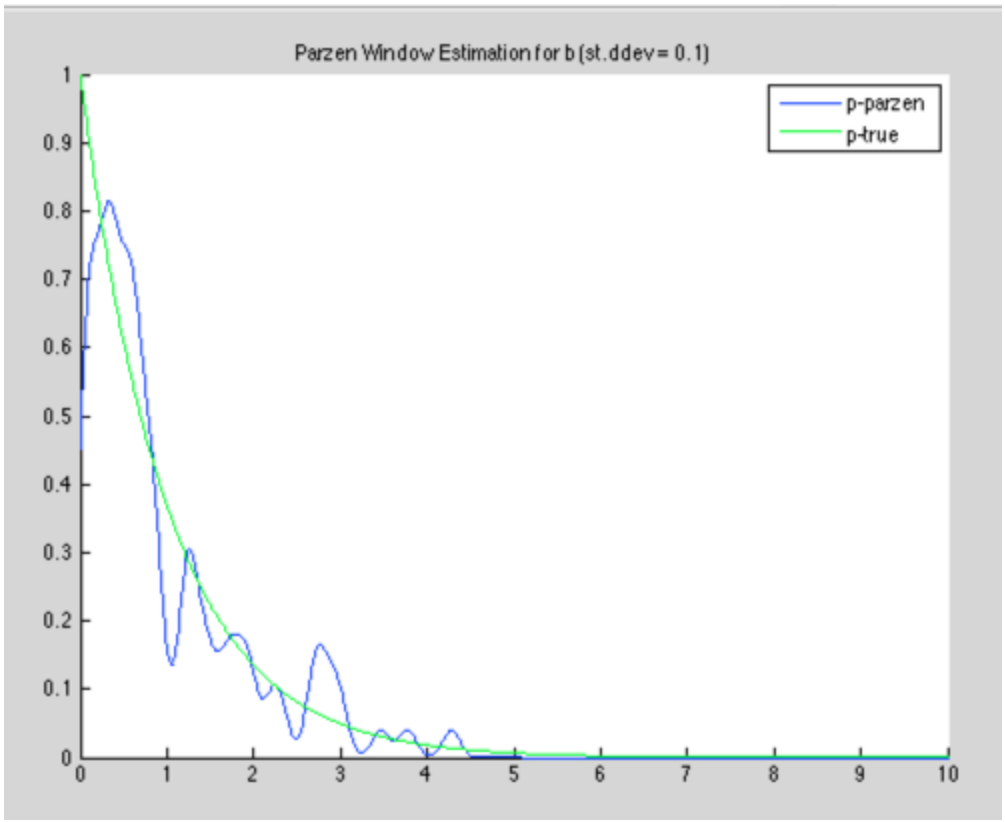
Using Parzen Window Estimation with a Gaussian Distribution with standard deviation 0.1 on dataset a resulted in this distribution:



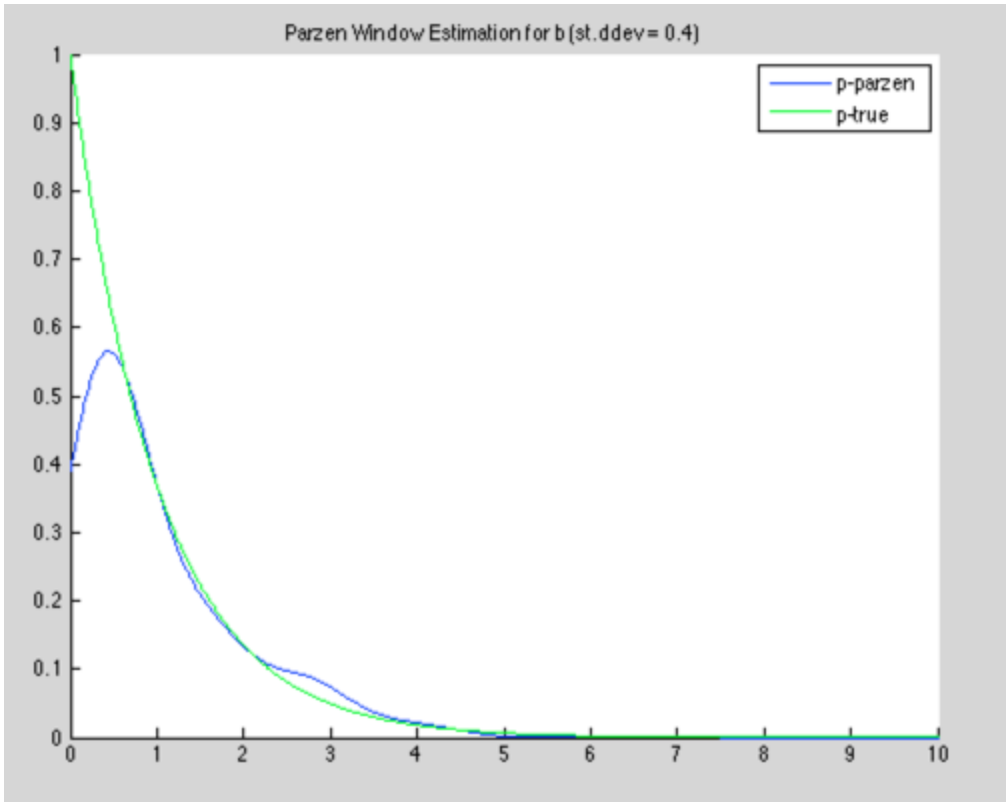
Repeating with a standard deviation of 0.4 resulted in this distribution:



Using the same technique for dataset b resulted in this distribution for a standard deviation of 0.1:



And this distribution for a standard deviation of 0.4:



For both datasets, using parametric estimation with the function used to generate the original data was the most successful method, as determined by how close the blue line is to the green line. Non-parametric did not work as well, but worked much better than using parametric with the wrong parameterized function. In general, non-parametric worked quite well, with the 0.4 standard deviation window working better than the 0.1 standard deviation window. The 0.1 standard deviation result exhibited a lot of noise. Parzen estimation also failed to find the high but narrow peak in the exponential distribution.

It is not always possible to use parametric estimation because the form of the true distribution is not always known. When the form of the distribution is know, it is generally better to use parametric unless the true PDF is very complex.

Model Estimation - 2D case

ML classification

We used the mean() and cov() MATLAB methods to find the mean and covariance for the 3 data sets.

mean for *a1*: [347.1600 131.2000]
covariance for *a1*:

```
  [  1766.6   -1610.6
   -1610.6    3343.5 ]
```

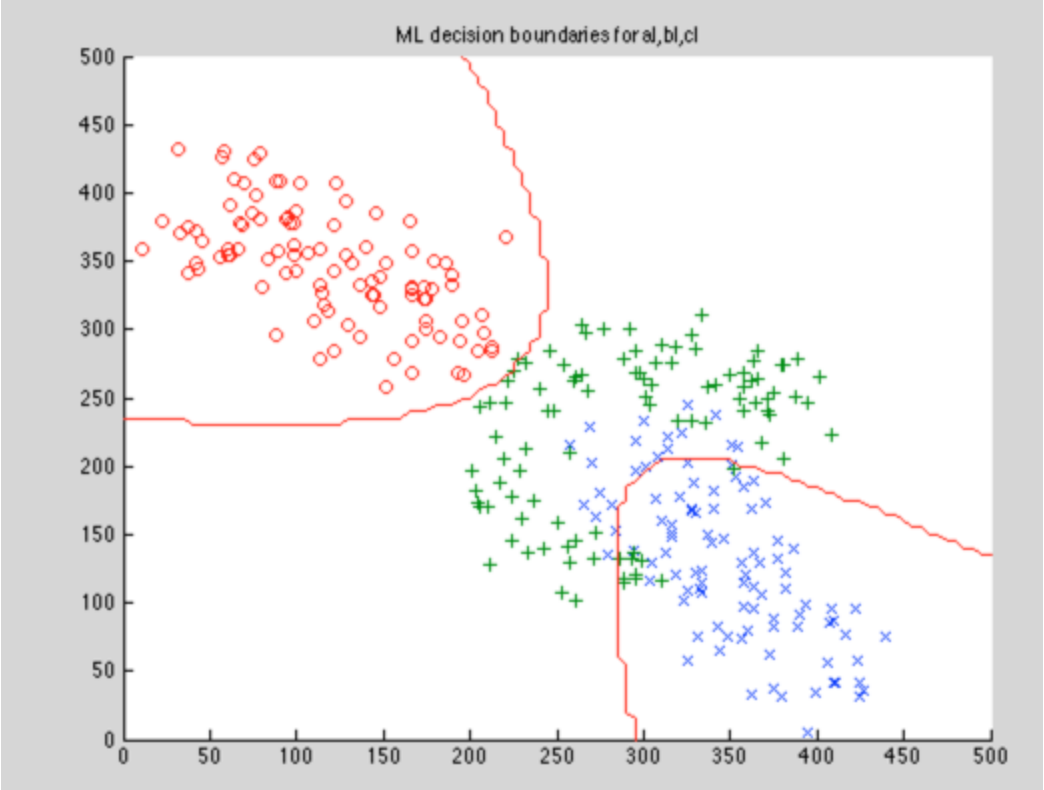
mean for *b1*: [291.8400 224.0200]
covariance for *b1*:

```
[ 3315.7    1176.0
  1176.0    3414.0 ]
```

mean for *c1*: [119.5500 346.6700]
covariance for *c1*:

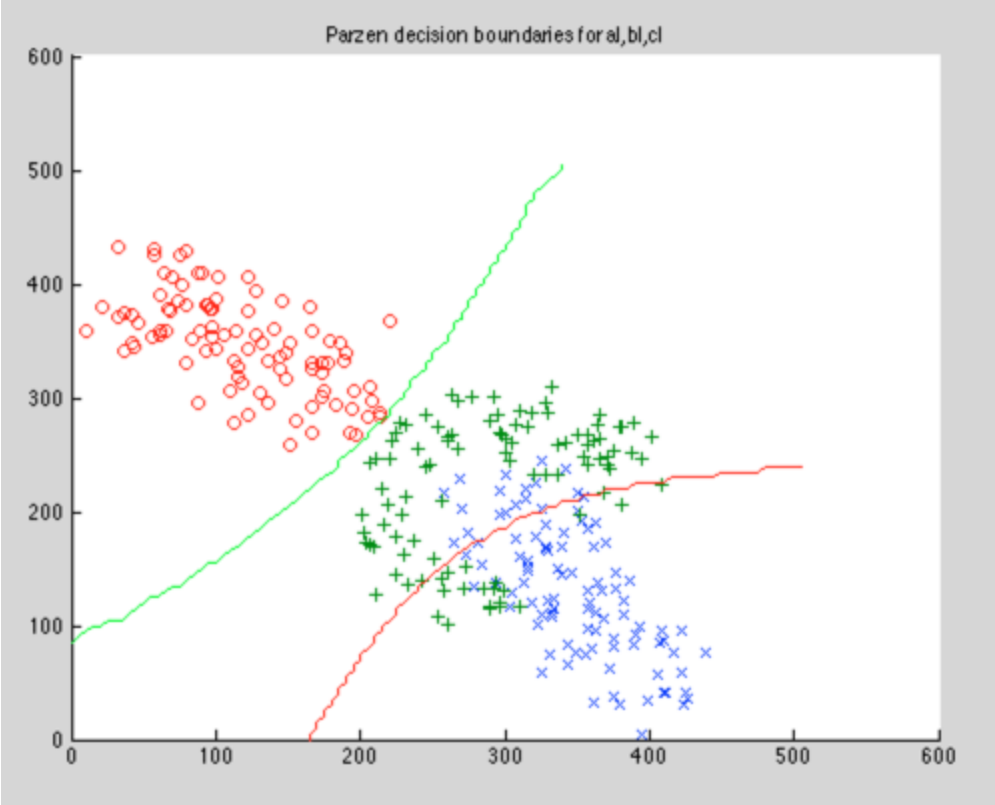
```
  [  2738.5   -1327.2
   -1327.2    1699.3 ]
```

The plots of the data along with the decision boundaries are as follows:

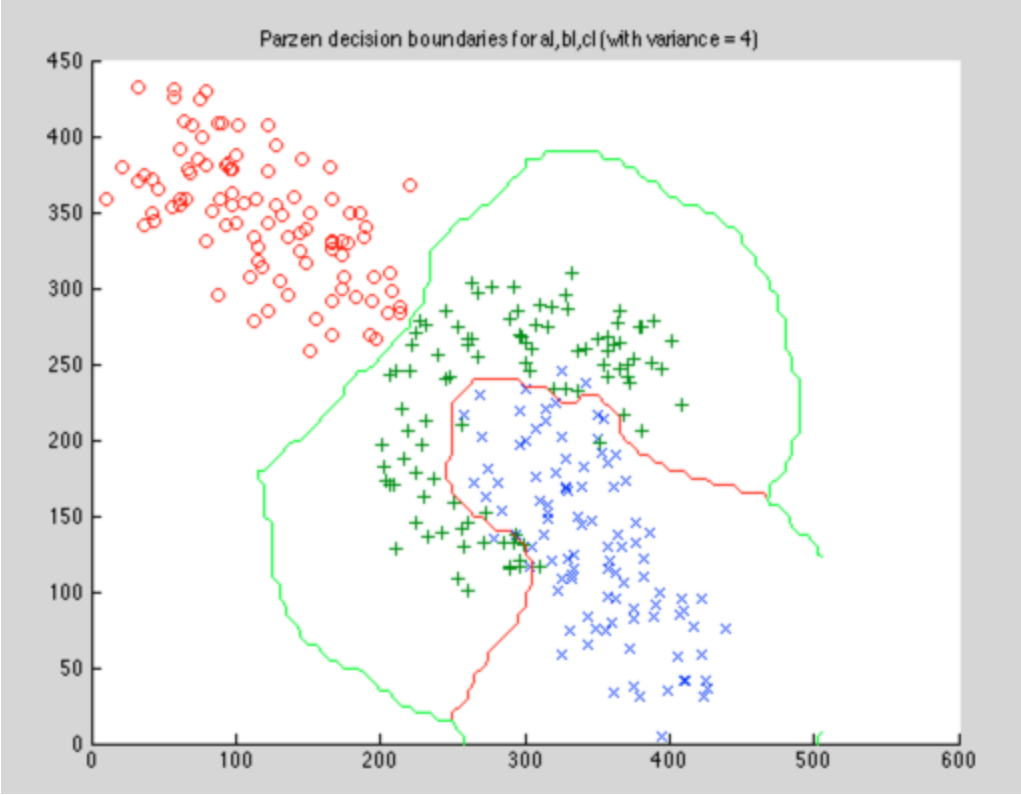


This method distinguishes between cl (red) and the others very well, but not particularly well between al (blue) and bl (green).

Next, we used Parzen Window discrimination, implemented using the provided matlab function. We used the ‘fspecial’ function to create the window matrix. The result is a numerical solution to the problem. As instructed by the lab, we began with a window with variance=400. This was the result:



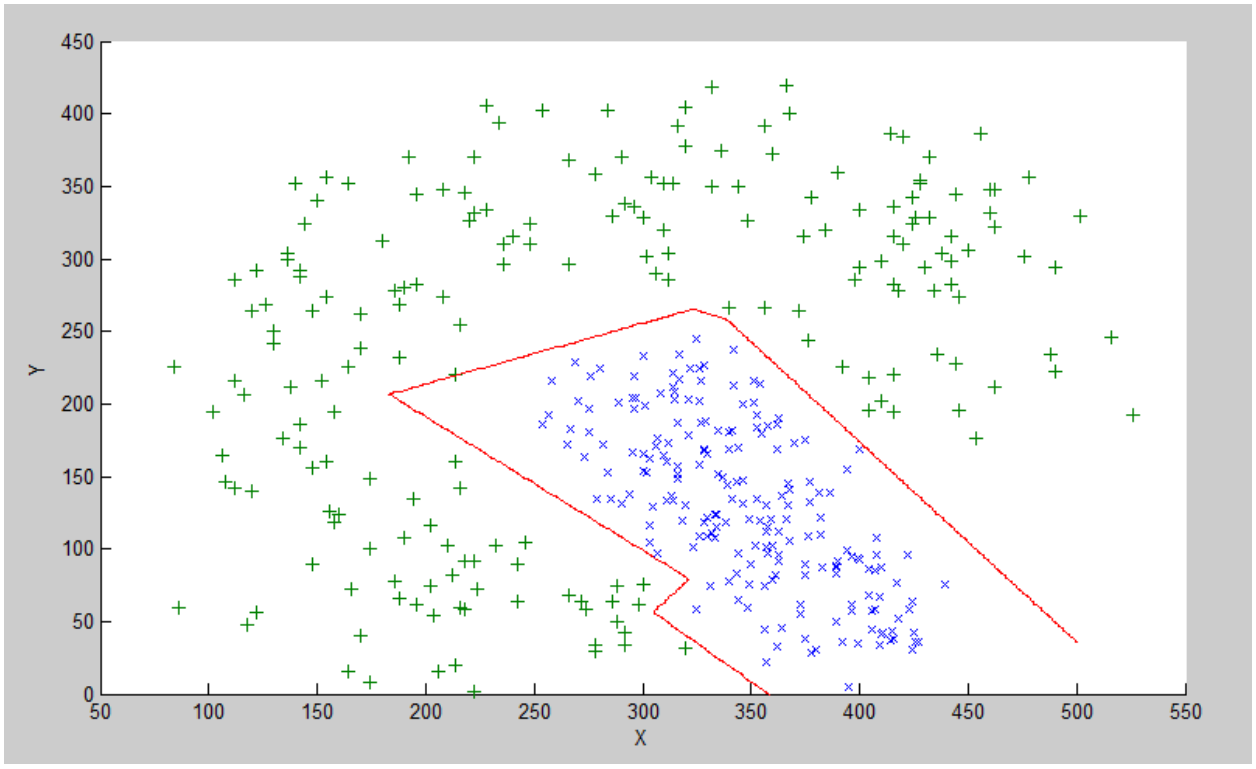
This result was unexpectedly poor. We tried other variances and found that variance=4 worked much better:

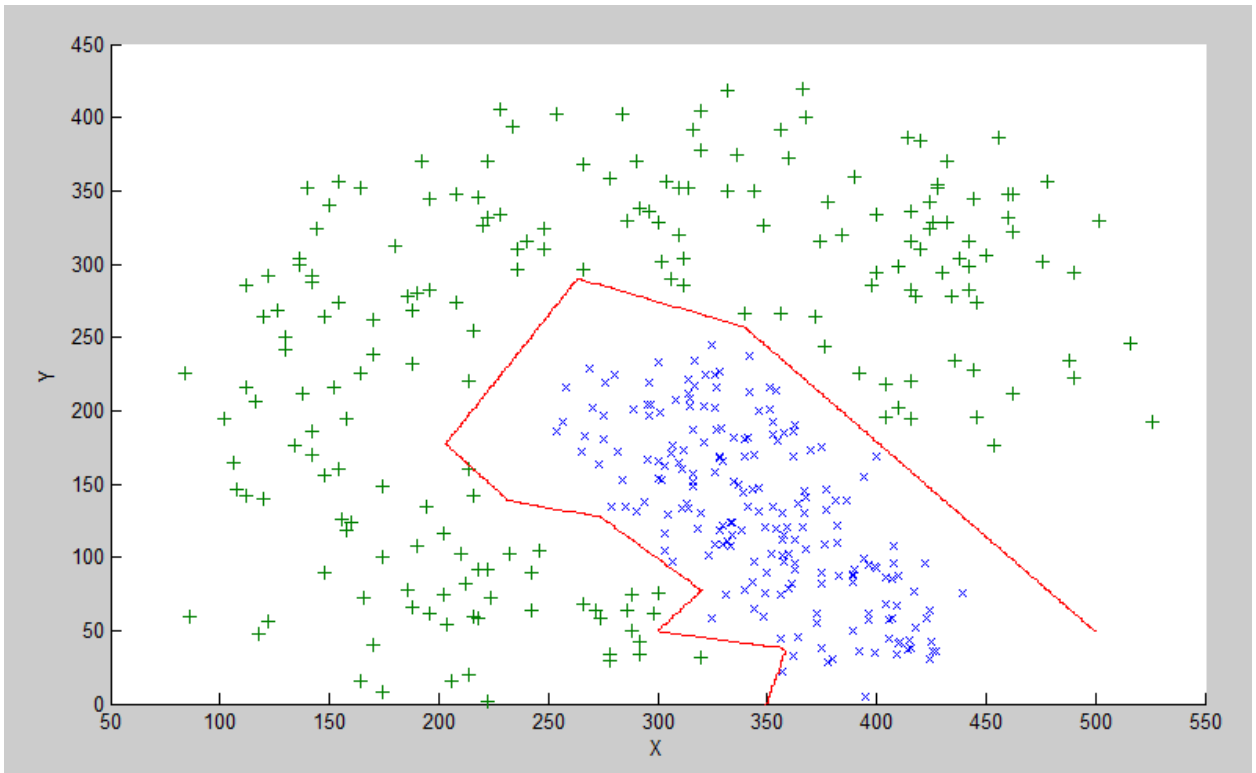


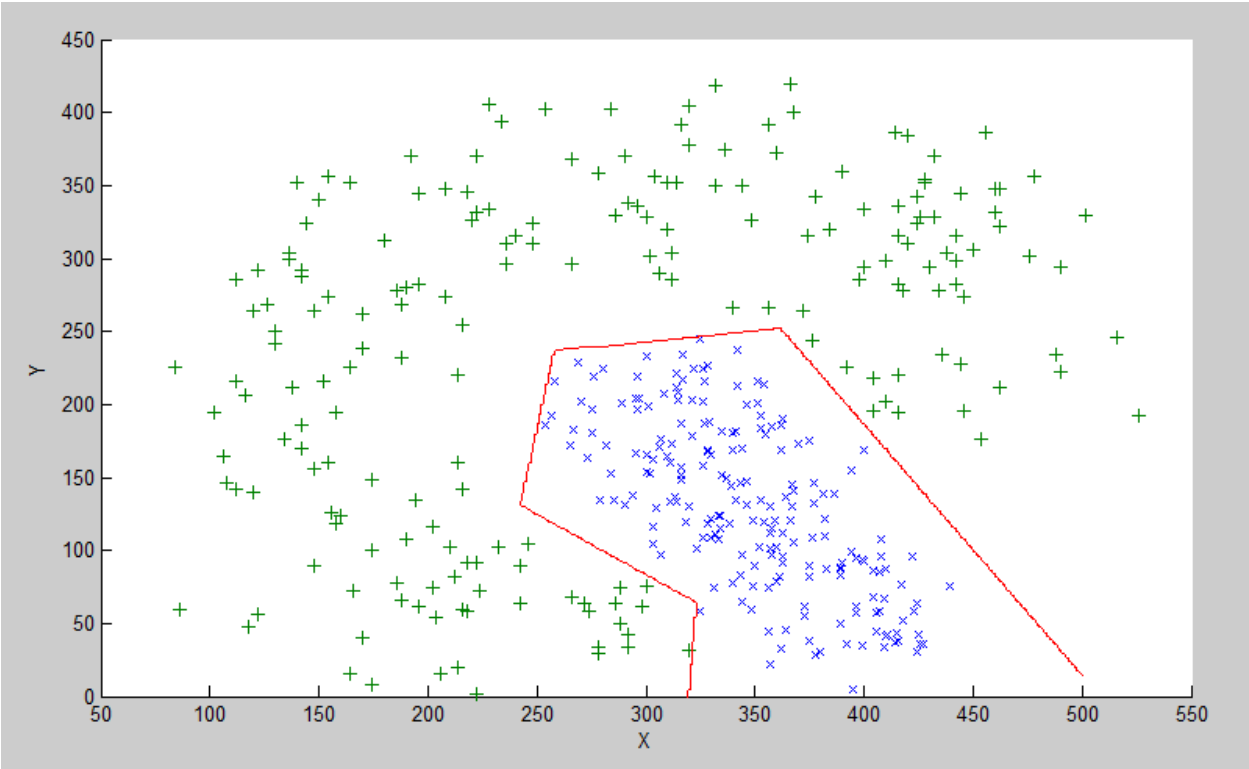
For 2d data the parametric approach will be even more difficult than for the 1d case as determining the correct function to use will be more difficult, and solving the sum will also be more difficult. In these cases, the non parametric approach will be the best solution. If the true PDF is known and not overly complex, the parametric approach is probably best.

Sequential Discriminants

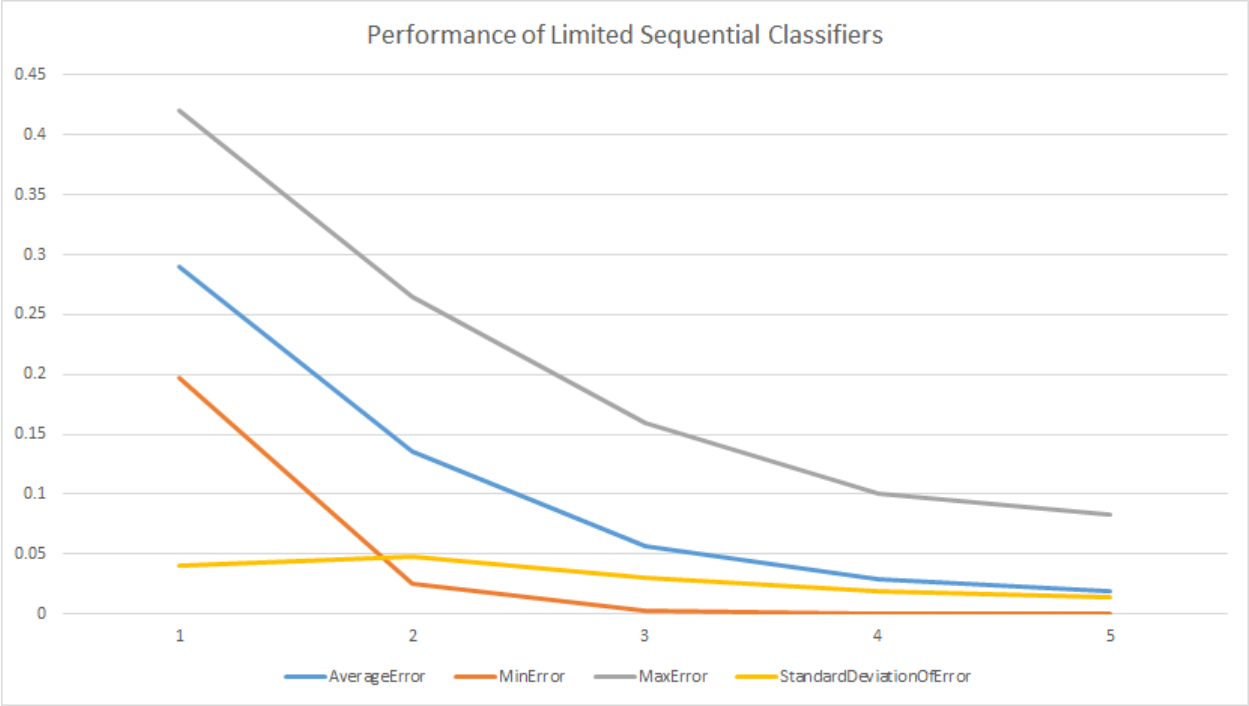
Here are three sequential classifiers:


Y
X
0
50
100
150
200
250
300
350
400
450
50
100
150
200
250
300
350
400
450
500
550


Y
X
0
50
100
150
200
250
300
350
400
450
50
100
150
200
250
300
350
400
450
500
550



As seen in the plots, if we were to test these classifiers with the test data, we would get 0% error. This is because the sequential classifiers are constructed in such a way that makes sure that all of the test data is correctly classified.



As shown in the plot, the average error rate, minimum error rate, maximum error rate, and the standard deviation of the error rate for limited sequential classifiers gets better as the number of composite classifiers increases.

The challenge in coming up with a limited sequential classifier was figuring out what to do with the points in the region that can't be classified by the sequence of classifiers. For this implementation, the first classifier was taken to be canon.

Probably the most correct thing to do would be to tally the remaining unclassified points in the training set, and whichever class was more represented within the unclassified points, simply assume that any point that lands in that region belongs to that set.

If the number of pairs that could be tested was limited, then the classifier would probably be forced to make all of the tests that it was allowed, and then decide amongst the classifiers it came up with, which are the most definitive, and which are the least, and then make classifications in order from most definitive to least definitive. Clearly this would result in some misclassification, even for the training data, but it would probably be the best solution under the given circumstances.

Conclusions

In conclusion, we found that for 1 dimensional data, where it is not hard to determine the true form of the PDF, parametric estimation works best. However, Parzen window approach also worked well if the window function is selected appropriately. Too wide of a windows causes a smearing of the PDF, while too narrow of a window causes noise in the PDF, which is a result of overfitting / too much dependence on the data.

For 2 dimensional data, the Parzen window approach worked substantially better than the ML approach, because our data was not Gaussian, yet our ML classifier assumed a Gaussian distribution. The Parzen Window method worked much better for the 2 dimensional data.

We also tried the Sequential Discriminant method for 2 dimensional data. The Sequential Discriminant method, of course, worked perfectly on the training data when allowed to grow to an arbitrarily large size. Even with just 5 discriminants, the Sequential Discriminant method pushed the error rate below 2% for the training data. By using a limited number of discriminants, the method can be tuned to take more or less time depending on the application, which makes it a very flexible technique to use.