EECE 5644 HW2

Owen McElhinney

Due: 2/25/2020

Contents

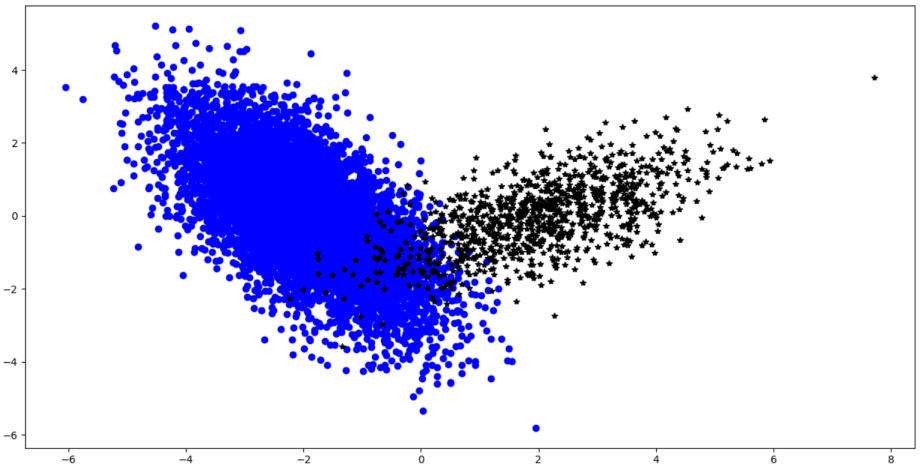

Figure 1: The data points used for the evaluation of all classifiers

1 Question 1

Question one investigates creating a maximum likelihood classifier for a binary problem using the sigmoid function. Each of the classes is generated as a bi-variate Gaussian distributions. The first part uses the true parameters to create a theoretical 'best-case' classifier. The second part then uses linear logistic regression at various numbers of training samples to build a classifier. Finally, part three uses a quadratic logistic regression algorithm to evaluate at various traing sample lengths. The training sample lengths tested are [10, 100, and 1000]

1.1 Data Distributions

The same data set was evaluated for every part of the following analysis. This data set was generated as a mixture of two bi-variate Gaussian distributions with the following parameters

$$\alpha_0 = 0.9; \mu_0 = [-2, 0]; cov_0 = \begin{pmatrix} 1 & -0.9 \\ -0.9 & 1 \end{pmatrix} \tag{1}$$

$$\alpha_0 = 0.1; \mu_0 = [2, 0]; cov_0 = \begin{pmatrix} 2 & 0.9 \\ 0.9 & 1 \end{pmatrix} \tag{2}$$

1.2 Optimal Classifier

The first part investigated the optimal classifier that used the distribution statistics to minimize error. 2 shows the results of this classifier on the data set. This

classifier was able to achieve a P(error)=0.013 at a P(False Alarm) = 0.002 and P(Hit) = 0.88

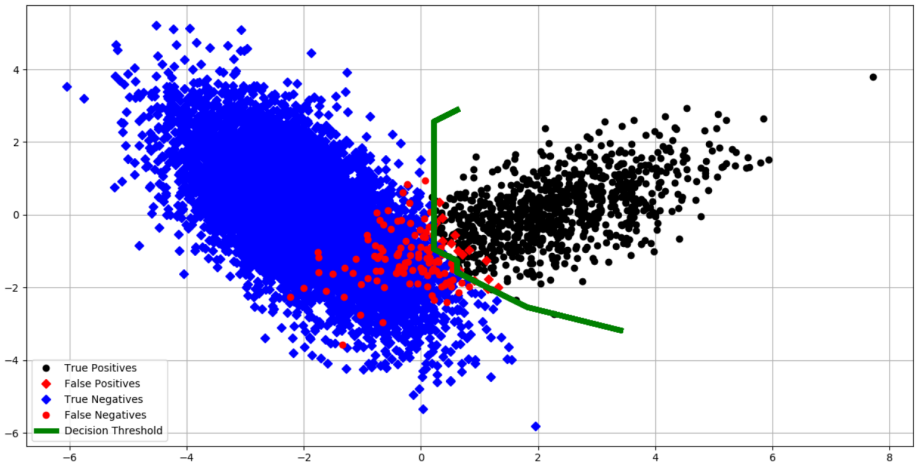

Figure 2: Results for the optimal classifier, with all priors known

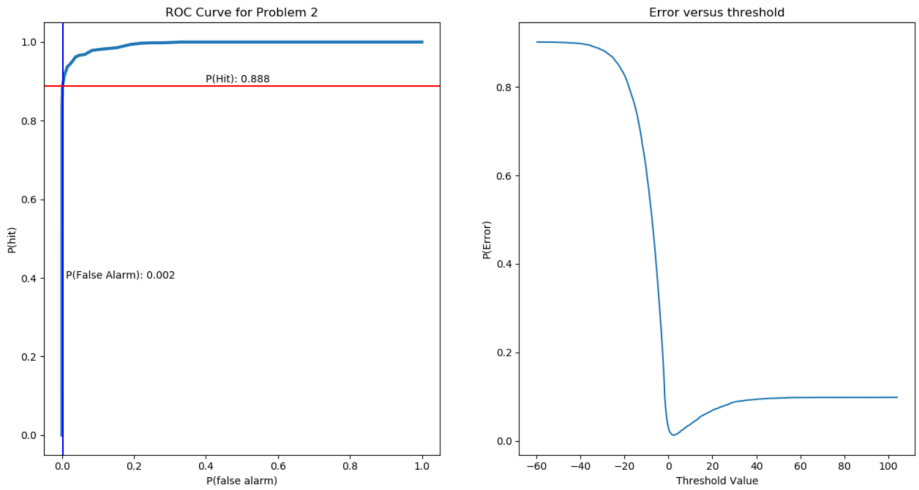

Figure 3: ROC curve and P(error) vs threshold for optimal classifier

1.3 Linear Logistic Classifier

The next section of this exam explored the linear logistic classifier. This classifier was defined using the Sigmoid function given as:

$$h(x, w) = \frac{1}{1 + e^{-w^T z(x)}} \tag{3}$$

where z(x) $= [1, x^T]^T$ and w represents our parameter vector. This function was used to classify each data point as either a 1 or a 0. This was compared to the true labels of the training set to get a cost function. The total difference was then plugged in to the gradient of the cost function function, given as:

$$C' = x(h(z) - y) \tag{4}$$

where x is the training data and y is the true label. This gives an updated value for w, and the routine is repeated until an optimal w is found.

This assumes a linear relationship of $x_2 = ax_1 + b$. The results of this classifier are tabulated below, and an example of the output is shown in 4

Training Points	P(error)
10	0.102
100	0.025
1000	0.027

1.4 Quadratic Logistic Classifier

The final section evaluated the quadratic logistic classifier under a similar Maximum Likelihood optimization technique. The quadratic case still uses the sigmoid function from the linear classifier, but this time out input vector z =

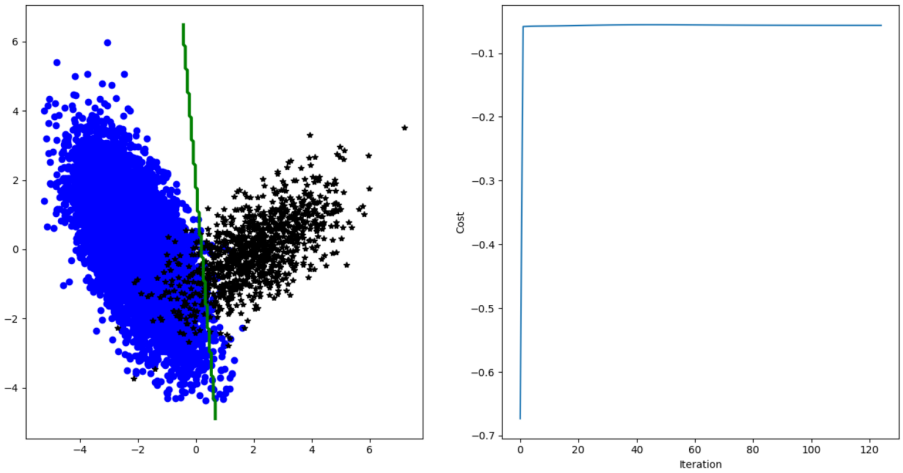

Figure 4: Decision boundary for the logistic linear classifier as well as the cost vs iteration plot for Question 1

$[1, x_1, x_2, x_1^2, x_2^2, x_1x_2]$ and w is just a larger weight vector. The results of this classifier are tabulated below, and the output decision boundary is shown in 5.

Training Points	P(error)
10	0.175
100	0.020
1000	0.016

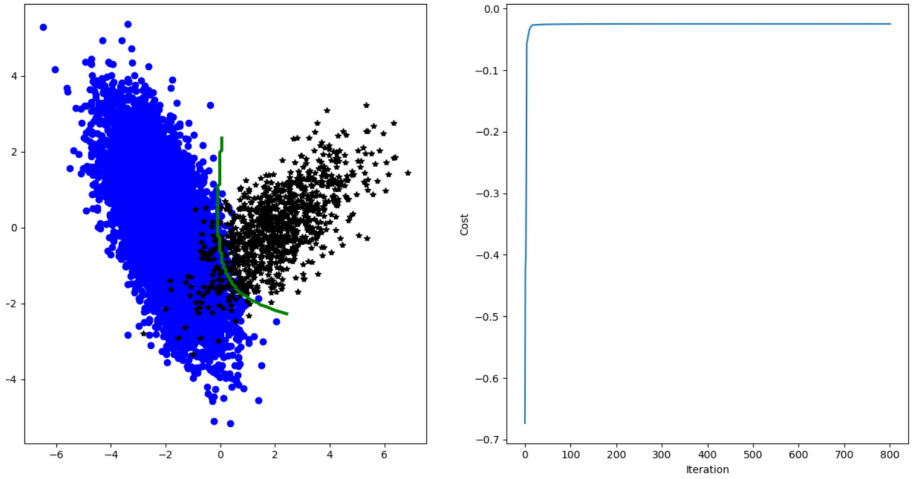

Figure 5: Decision boundary for the logistic quadratic classifier as well as the cost vs iteration plot for Question 1

1.5 Discussion

Compared to the "optimal" classifier in part 1 that assumed all statistics were known, the quadratic classifier was able to get a P(error) that was 0.011 lower. The major benefit of the linear classifier, is that while it may be less accurate, it converged in only 126 iterations vs 796 iterations for the quadratic classifier. This trade-off between accuracy and computability is one of the major driving considerations in problems of machine learning, and this problem did a great job in illustrating this difference.

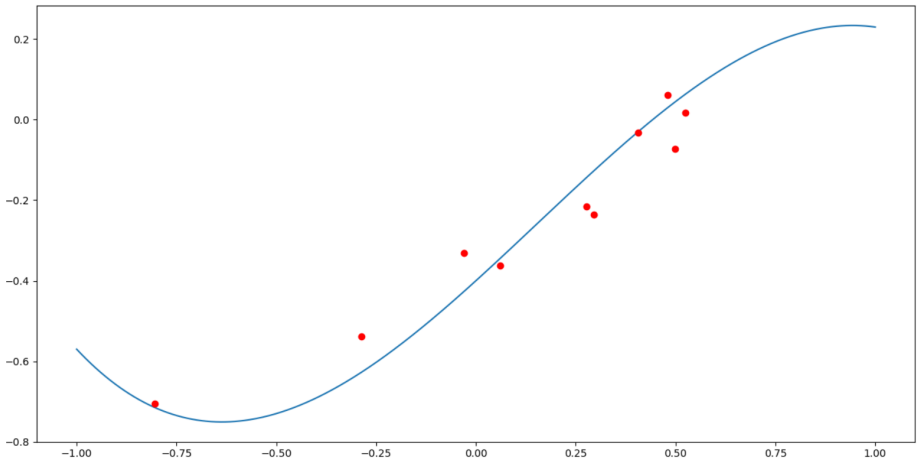

Figure 6: Example of one 10-point training set. The red dots represent outputs y, and the line represents the true model

2 Question 2

Question 2 explores the Maximum A-Posterior approach to parameter estimation. This technique uses some assumes we have some known information about the prior distributions of the parameters being estimated in order to improve the accuracy of our estimates. The problem seeks to estimate the parameters [a, b, c, d] from the cubic equation:

$$y = ax^3 + bx^2 + cx + d + v \tag{5}$$

where $v\tilde{\mathcal{N}}(0, \sigma^2)$. The first subsection will outline the equations used in order to solve this problem, and the second sub-section will provide the final results of the classifier and discussion on said results. An example of the y data being tested on, can be seen in figure 6. The true parameters were chosen to be: [-0.5, 0.23, 0.9, -0.4]

The analysis was run over a wide range of values for γ. At each γ value, the estimator was run 100 times on various sample data y, to see the full range of possible outputs. The results section will show that this method is highly dependent on the randomness in the noise as well as which data points are sampled.

2.1 Analysis

As stated above, the solution to this problem uses a Maximum A Posterior approach defined as:

$$\hat{\theta} = argmax_\theta \Big[p(x|\theta)p(\theta) \Big] \tag{6}$$

For the case of regression, this equation becomes:

$$\hat{\theta_{MAP}} = argmin_\theta \sum_{i=1}^{N} (Y - \theta^T Z_i) + (\frac{\sigma_v^2}{\gamma})\theta^T\theta \tag{7}$$

To solve, take the derivative of (7) with respect to θ and set it equal to zero. With some calculus and algerba, the final equation becomes:

$$\hat{\theta} = (\sum (Z_i Z_i^T) + (\frac{\sigma_v^2}{\gamma}))^{-1} (\sum Y_i^T Z_i^T) \tag{8}$$

2.2 Results

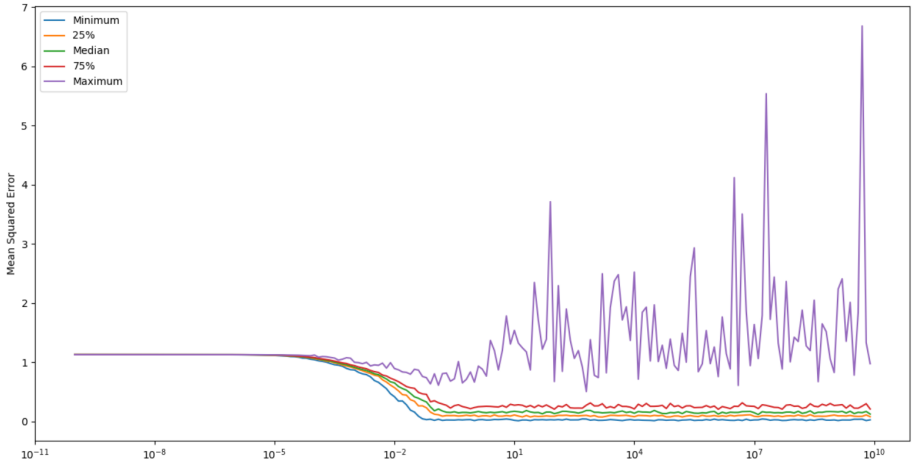

Figure 7: Range of accuracy estimates for each gamma value. Each estimate is evaluated based on it's MSE from the true parameter values. Results for minimum, maximum, and various percentiles are shown

Figure 7 shows the range of Mean Square Error (MSE) values over a wide range of gamma values. It can be observed that the output is very dependent on what is sampled as a y value. If only one measurement of y is taken and used as the final estimation, the result could lie on the maximum line and be very far from the truth. By taking 100 samples at each value, estimations were found that lay extremely close to the truth.

3 Question 3

Question 3 explores the application of the expectation maximization algorithm to Model Order Selection. The underlying problem is that the output data is created by a mixture of N independent Gaussian distributions, each with it;s own mixture parameters and prior probabilities. This problem seeks to estimate the how many Gaussians are present and their parameters through an unsupervised algorithm.

For this problem, the mixture parameters were chosen as such:

Label	Prior	Mean	Covariance
0	0.2	[5, 3]	$\begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$
1	0.3	[3, -2]	$\begin{pmatrix} 1 & -0.25 \\ -0.25 & 1 \end{pmatrix}$
2	0.1	[-2, 3]	$\begin{pmatrix} 1 & 0.1 \\ 0.1 & 1 \end{pmatrix}$
3	0.4	[-4, -3]	$\begin{pmatrix} 1 & 0.75 \\ 0.75 & 1 \end{pmatrix}$

The algorithm was run a large number of times in order to improve accuracy. The algorithm was run 100 times, and the order with the highest expected value on the most iterations was chosen. This was done for training sets of both 100 and 1000 points. Within the algorithm itself, the training data was chosen using a bootstrapping technique. This was done 10 times, to generate 10 different training sets and the results were averaged over all ten sets.

The following section will overview some of the equations and concepts used in the actual implementation of the algorithm, and the final section will discuss the results and some of the issues that were encountered in this implementation.

3.1 Analysis

The Expectation Maximization algorithm can be broken down in to two general steps:

1. Assume a distribution, and find the expected value of the data points under this distribution.
2. Find the parameters that maximize the expectation under this assumption, and use those as the new assumed values

For this case of Gaussian Mixtures, the expected value is calculated as

$$E[p(x)] = \sum_i \alpha_i \mathcal{N}(\mu_i, \sigma_i^2) = \gamma \tag{9}$$

This provides a weight matrix, which can be used to update the model parameters as such:

$$\hat{\alpha_k} = \frac{N_k}{n} \tag{10}$$

$$\hat{\mu} = x\gamma^T \tag{11}$$

$$\hat{\sigma^2} = \gamma(x - \hat{\mu})^2 \tag{12}$$

Thus, under proper convergence conditions, this will provide a best fit and can be run over many order numbers and data sets.

3.2 Results

As stated in the introduction, this EM procedure was run 100 times and each iteration consisted of an average on 10 bootstrapped validation sets. It was first run with validation sets of 100 points and later by validation sets of 1000 points. The problem asked that it also be run at validation sets of 10 points but there were convergence issues at that criteria, to be discussed in the next section. At the conclusion, the algorithm selected m =3 and m = 5

Label	Prior	Mean	Covariance
0	0.152	[-3.451, -2.828]	$\begin{pmatrix} 0.085 & 0.0285 \\ 0.0285 & 0.116 \end{pmatrix}$
1	0.448	[-3.654, -0.933]	$\begin{pmatrix} 2.531 & 4.543 \\ 4.543 & 8.752 \end{pmatrix}$
2	0.400	[3.562, -0.513]	$\begin{pmatrix} 2.295 & 1.934 \\ 1.934 & 5.152 \end{pmatrix}$

Table 1: Results for 100 point data sets

Label	Prior	Mean	Covariance
0	0.007	[3.082, 5.938]	$\begin{pmatrix} 15.074 & 3.328 \\ 3.328 & 0.735 \end{pmatrix}$
1	0.029	[6.201, 4.063]	$\begin{pmatrix} 0.381 & 0.405 \\ 0.405 & 0.783 \end{pmatrix}$
2	0.399	[-3.299, -1.016]	$\begin{pmatrix} 2.136 & 3.379 \\ 3.379 & 8.636 \end{pmatrix}$
3	0.547	[3.477, 0.044]	$\begin{pmatrix} 1.983 & 2.52 \\ 2.52 & 6.159 \end{pmatrix}$
4	0.016	[5.333, 5.857]	$\begin{pmatrix} 0.150 & -0.183 \\ -0.183 & 0.263 \end{pmatrix}$

Table 2: Results for 1000 point data sets

3.3 Issues

Several issues were encountered in the implementation of this problem. The first and most problematic was in the calculation of covariances. Under some conditions, Numpy would be unable to return an estimate for a covariance

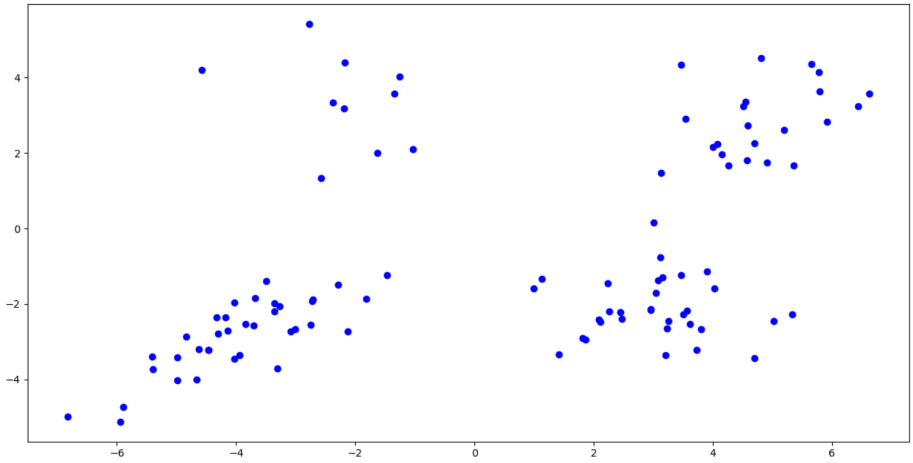

Figure 8: Example of 100 data points used for training with clear separation between the distributions

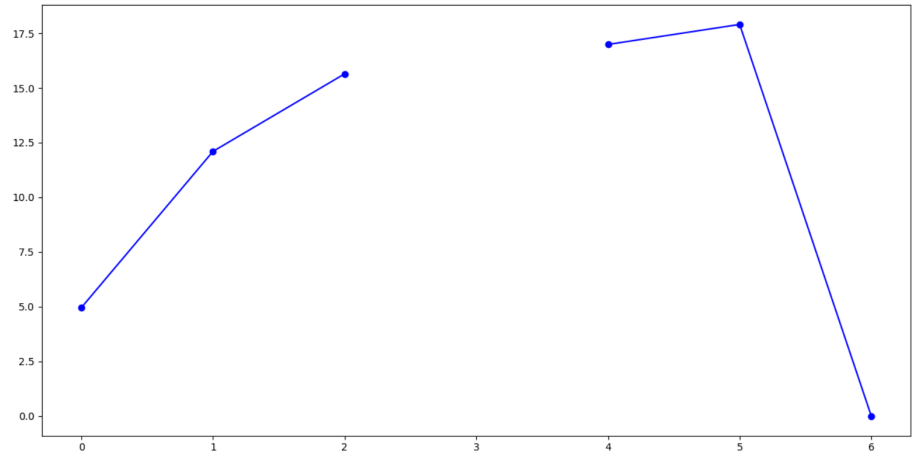

Figure 9: Example of output results. The x-axis represents model order and the y-axis is likelihood. Missing data points are explained in the next section

matrix. These failure conditions are presented below as well as some of the attempts used to correct them.

1. If only one point was assigned to a label for the initial estimates, there was not enough data to find a covariance. It was attempted to just use the identity matrix as an initial guess, but this led to the prior for that group dropping immediately to almost zero.

2. If a data point was selected twice in the bootstrapping routine, it would lead to a covariance matrix of 0 and propagate a NaN value, which is what causes the missing data point in 9. This affect is more common in the case for only generating ten data points, so this case was ignored for the presentation of results.

3.4 Discussion

With training sets of only ten points, the results were far behind in accuracy. The algorithm tended towards a 2 to 3 mixtures with high variance and large priors with any outliers being picked up by a small mixture with a tight variance. The results could be improved by increasing the number of data points used for training.