ASSIGNMENT 2: REPORT

SUBMITTED BY: ISHA CHAUDHARY (2018EE30614)

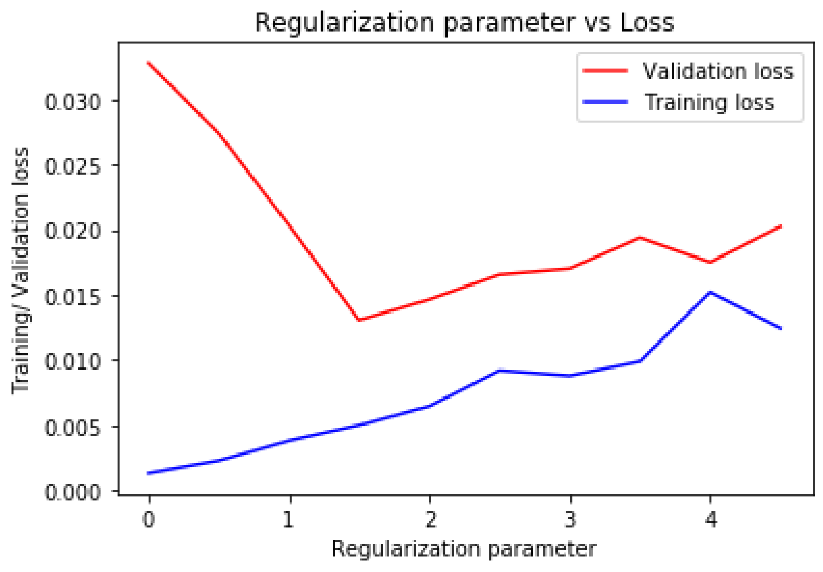
Part 1: Ridge Regression

Data set used: MNIST dataset with only digits ‘0’ and ‘1’ inputs.

Data set description: Training dataset contains 12665 examples and test set contains 2115 examples of (28 x 28) features.

Procedure:

1. Using the Numpy library in the helper function, get_01_samples(), the 0 or 1 labelled samples were picked out, for both the training and test sets and were stacked such that the samples labelled 0 were together in the upper half of each dataset and samples labelled 1 were together in the lower half of each dataset (training and test). The features were also normalized in this function.
2. The data was then randomized to help in better training.
3. A general class for the implementation of Logistic regression with L2 or Ridge regularization was created. The methods of this class include:
    a. In the _forward_pass() function, the final sigmoid activated output of the Logistic regression, along with the loss (if the correct labels are passed to the function) given out. The loss is the cross entropy loss (with no model complexity term included for simplicity).
    b. In the _backward_pass() function, the changes to make in the weights and the bias value are computed using gradient descent. Here the effect of the regularization term is utilized to control the model complexity.
    c. _update_params() function is used to apply the changes calculated in the _backward_pass() function to the weights and bias.
    d. predict() and evaluate() functions are utility functions for the determination of the model accuracy and loss.
    e. train() function is used to run the forward and backward passes for some number of epochs over the training dataset, to set the best weights.
4. Due to the large size of the data set, Batch Gradient Descent method is adopted per epoch.
5. At the end of the training, the validation accuracy is calculated.
6. This procedure is repeated for different values of the Regularization parameter, and the results are plotted as:



Interpretation of the results: The Regularization parameter controls the norm square of the weights of the Logistic Model and thus the model complexity. As the regularization parameter value increases the model's tendency to overfit the dataset reduces and thus the training loss increases and validation loss decreases. But after a certain value of Regularization parameter (= 1.5), the model begins to underfit the data, which results in the training loss and validation loss both to increase.

Speciality of this variant: This variant of regression considers the model complexity also and is expected to generalize well on unknown datasets.

Advantage of L2 regularization over the un-regularized Logistic Regression:

L2/ Ridge regularization is more helpful as compared to the un- regularized Logistic Regression as the tendency of the model to overfit the data is reduced. This kind of regularization paradigm tries to minimize the values of the weights used in the logistic regression. The objective function of its optimization problem is:

$$\min_{W, b} \sum^{n}_{i=1} y_i.\log(y_i') + (1-y_i).\log(1-y_i') + a.||W||^2$$

where a: regularization parameter

y_i: actual labels

$y_i' = X_i.W$; where X_i is the augmented feature vector

Disadvantages of L2 regularization:

1. It does not have the tendency to put certain less important weights to 0 for simplifying the model further as does its counterpart, L1 regression. This can be considered both as an advantage as well as disadvantage, depending on the situation.
2. For choosing the regularization parameter, extra cross validation costs need to be encountered.

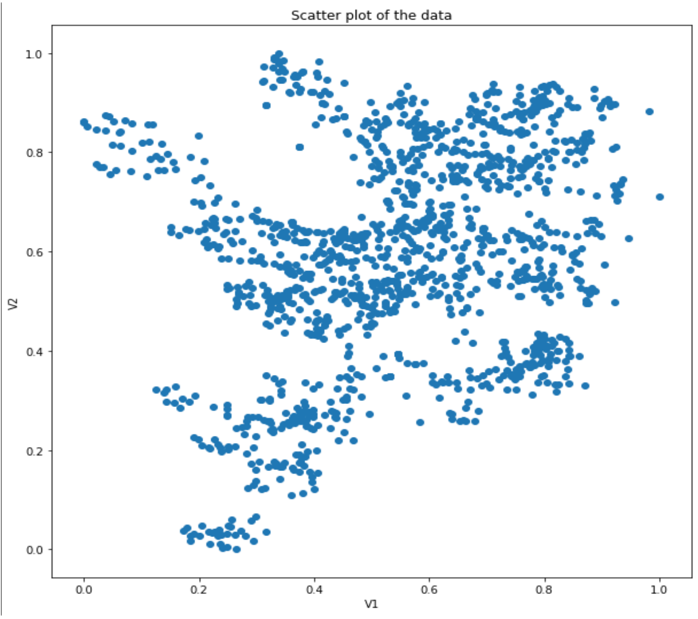
Part 2: Mini Batch K-Means Clustering

Data set used: Bank note authentication dataset from openml.org.

Data set description: This dataset is actually a labelled dataset, which contains bank note samples and the labels classify each sample as a genuine or forged bank note. The features used were obtained by performing a Wavelet transform on the bank note images. In this implementation, two of the features (V1 and V2) were used to implement clustering and for observing the various trends in the dataset, without the labels. This dataset contains 1372 samples.

Procedure:

1. The features were normalized and a scatter plot of the dataset was created. (Only 2 features are being used for the ease of visualization.)



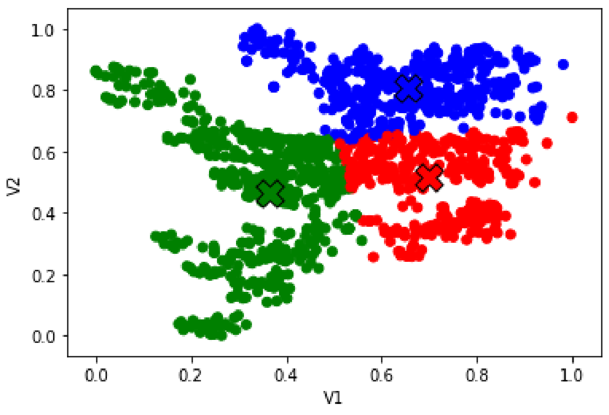
2. With some helper functions: get_random_rows(), find_nearest_center(), distortion_function(), plot_clusters(), the variations in the clusters with each iteration of the mini- batch K-Means clustering are observed.
    a. The cluster centres were initialized using random rows of the input data.
    b. In each iteration of the mini batch K- means clustering, the cluster centres were updated using a random batch of rows from the input data.
    c. The nearest of the current cluster centres is assigned to each data point in the mini batch (size = 300), thus assigning the clusters to each of the data points
    d. The mean of all the points in the mini batch, belonging to the same cluster is taken and the values of the cluster centres are put as these values.
    e. The loss function used to evaluate the clusters is:

       Distortion function = $\sum^{n}_{i=1} || X_i - C[\text{corresponding to } X_i] ||^2$

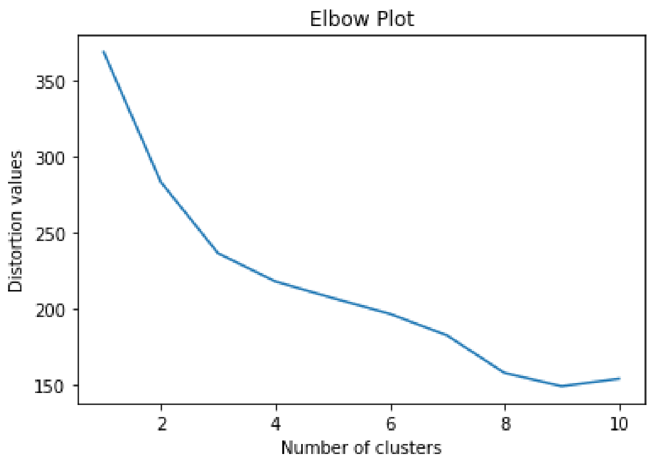
       where ||.|| signifies the Euclidean norm

    f. With each iteration, the improvement in the distortion function is tracked and the loop ends after sufficient number of iterations.
    g. Results of this step are presented after the procedure
3. As the randomized initialization of cluster centres may at times not lead to good final clusters, as the algorithm may settle at some local minimum of the distortion function, so in the next part of this implementation, the entire procedure of mini batch K- means clustering is run several times, each time with a different set of randomly selected initial cluster centres. The initialization which results in the least distortion function at the end of the algorithm is selected and displayed as the best clustering.

```
Least Distortion observed:  235.1662242420826
Cluster Centers: [[0.6999751  0.51542936]
 [0.65411633 0.80581477]
 [0.36452694 0.46544663]]
```



4. To observe the effect of change in the number of clusters on the distortion function, and obtain the right number of clusters to give the best analysis of this data, an elbow plot was created for the mini- batch K-means clustering, given as follows:



The value of ‘k’ at which the Elbow plot changes its slope most drastically to become less steep, is 3.

Therefore, the most appropriate number of clusters to have is for k = 3.

Results: Some selected and important steps of the final results of each step of the mini batch K-means algorithm are shown. The hyper parameters used are:

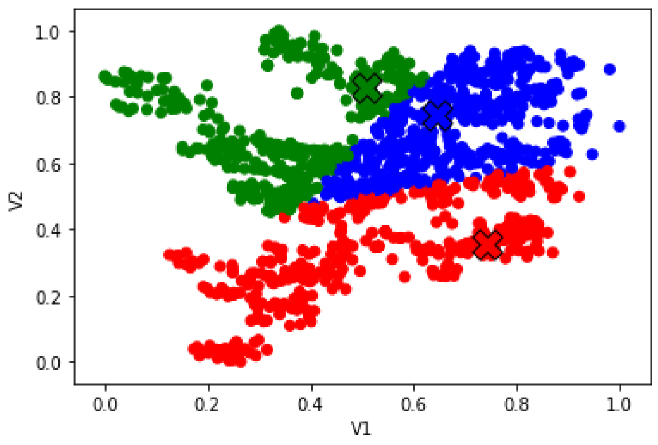
K = 3

Batch size = 200

Number of iterations = 100

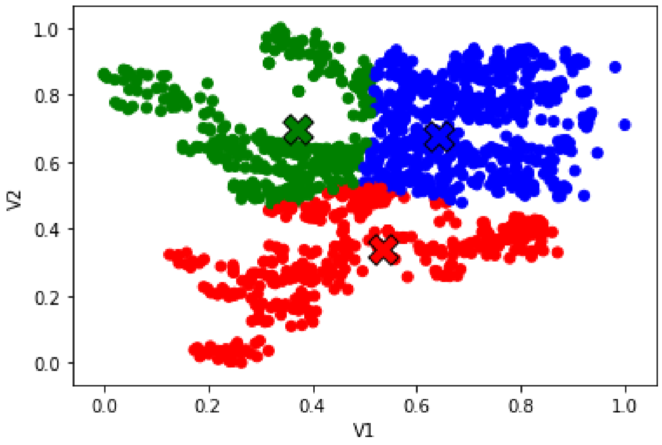
(Crosses denote the mean values of the clusters.)

1. Initial distortion loss function: 310.65

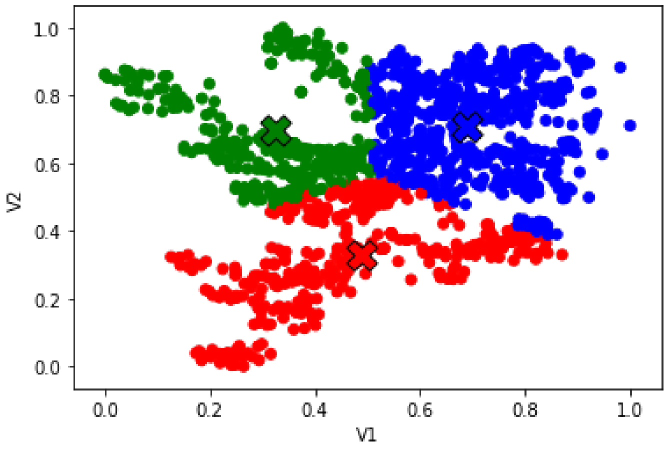


2. After 0th iteration, the distortion loss function is 252.16, with an improvement of 58.49 Cluster: [[0.53554548 0.33846039]

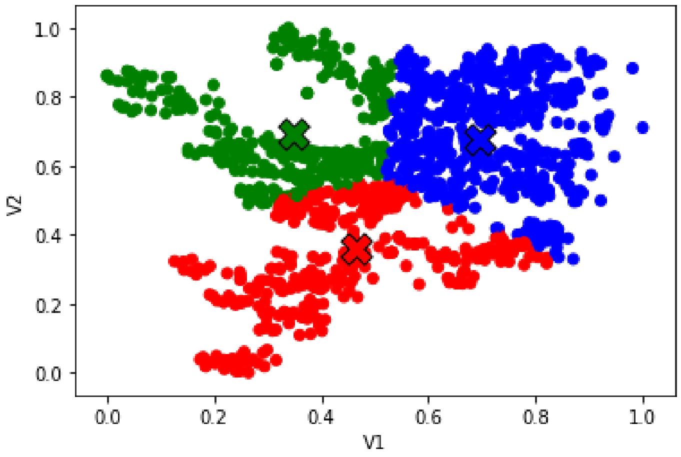
[0.64238763 0.67753073]
[0.372104   0.69973611]]



3. After 1th iteration, the distortion loss function is 245.42, with an improvement of 6.74
   Cluster: [[0.48824906 0.32954465]
   [0.69044965 0.70780085]
   [0.32597556 0.69908837]]

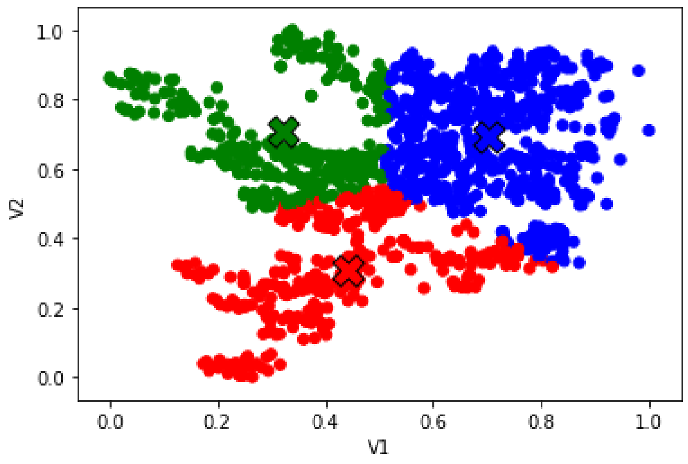


4. After 2th iteration, the distortion loss function is 244.29, with an improvement of 1.13
   Cluster: [[0.46588143 0.36223745]
   [0.69578486 0.67913099]
   [0.34880647 0.69466438]]

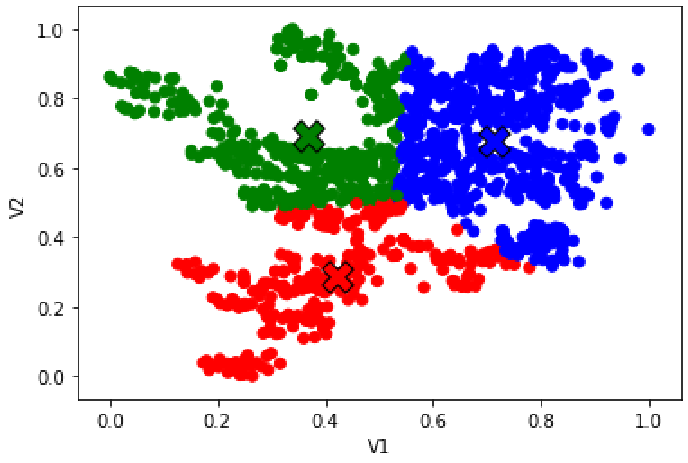


5. After 4th iteration, the distortion loss function is 245.30, with an improvement of -2.27
   Cluster: [[0.4405837 0.30981505]
   [0.7034269 0.69164751]

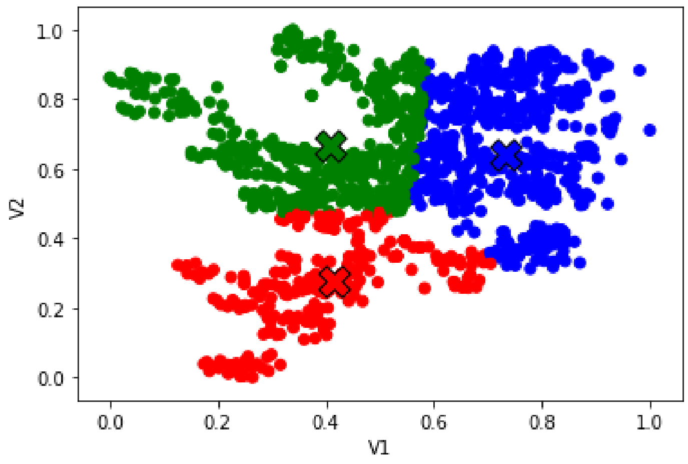
[0.32164969 0.70790705]]



6. After 6th iteration, the distortion loss function is 242.10, with an improvement of 0.12
   Cluster: [[0.42114899 0.28885966]
    [0.71157287 0.67835154]
    [0.36964323 0.69291821]]

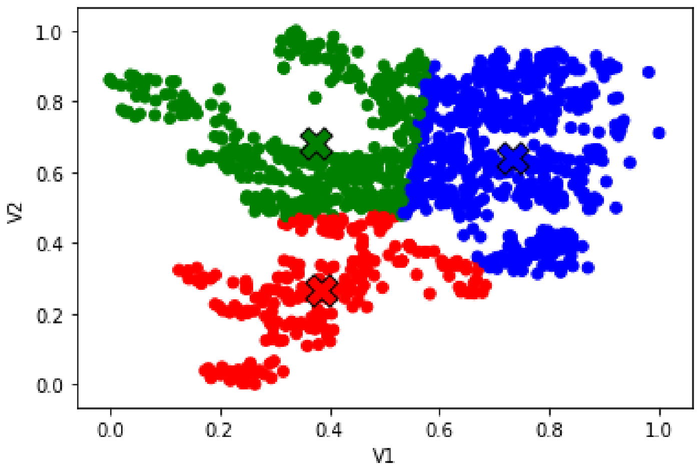


7. After 50th iteration, the distortion loss function is 240.15, with an improvement of -0.51
   Cluster: [[0.41610943 0.27670649]
    [0.73332255 0.64299311]
    [0.40683175 0.66845098]]



8. After 99th iteration, the distortion loss function is 241.32, with an improvement of 1.52
   Cluster: [[0.3843899 0.26868067]

```
[0.73157202 0.64054529]
[0.37507336 0.67997408]]
```



```
Final clustering!
```

Interpretation of results:

1. *Interpretation of the steps of the mini-batch K- means:* From the results shown above, it appears that the initial clustering is majorly rectified during the first 3-4 iterations.
2. *Interpretation of the selection of the best clusters method:* The randomized selection of initial cluster centres results in lesser minimization of the distortion function at times. So the algorithm is run again and again and the best clusters are selected.
3. *Interpretation of the selection of the best value for k (number of clusters):* The elbow plot indicates that the best value for k is 3.

Speciality of the Mini-batch version over the normal K-Means Clustering:

When we have really large datasets, then the normal K-Means Clustering becomes too costly, computationally as well as with respect to the storage requirements. It also takes a lot of time for one iteration. So a Mini-batch version of this clustering algorithm helps to reduce the storage and computation costs and also allows for the unavailability of the entire dataset at once.

Advantages of the Mini-batch version:

1. Computational costs and time are lesser.
2. Storage requirements are lesser.

Disadvantages of the Mini-batch version:

1. It is a sub-optimal way of clustering.
2. It can lead the iterations to settle at the local minima of the distortion function,.

3. The improvements in the distortion function fluctuate a lot between small negative and positive values, instead of settling down at 0.00.

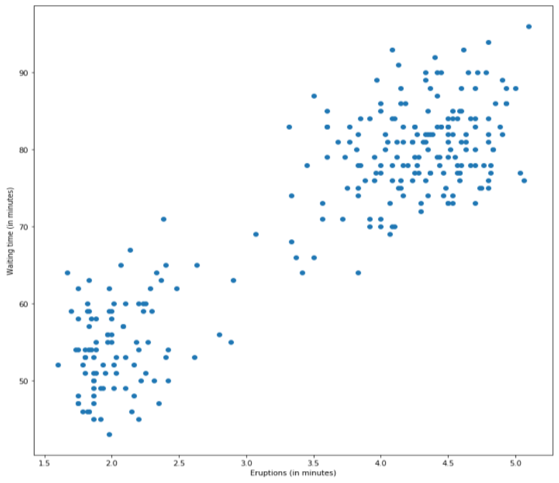
Part 3: ML Estimation of Gaussian Mixture Models using the EM Algorithm:

Data set used: Old Faithful Geyser dataset.

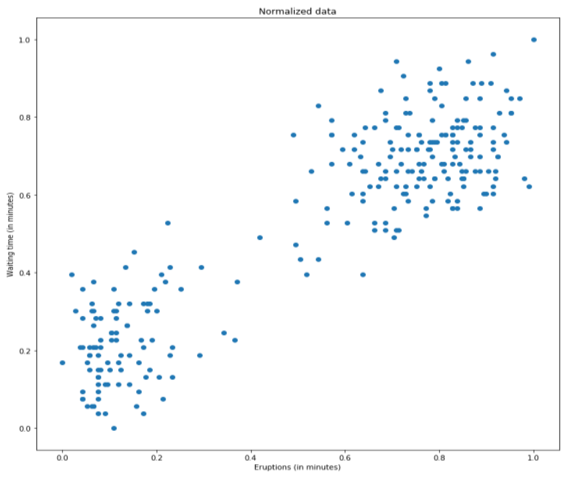
Data set description: This dataset contains the times between eruptions and the durations of those eruptions for the Old Faithful Geyser in Yellowstone National Park, United States. It contains 272 observations, each with two features, i.e. the time between eruptions and the duration of eruption.

Procedure:

1. The data set was visualized as follows:



2. The data set was then normalized and visualized as follows:

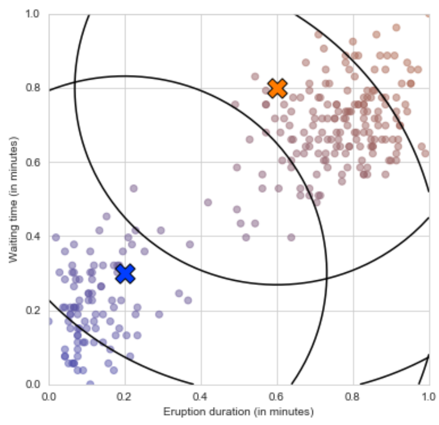


Visually, it appears to be a mixture of two bivariate Gaussian distributions.

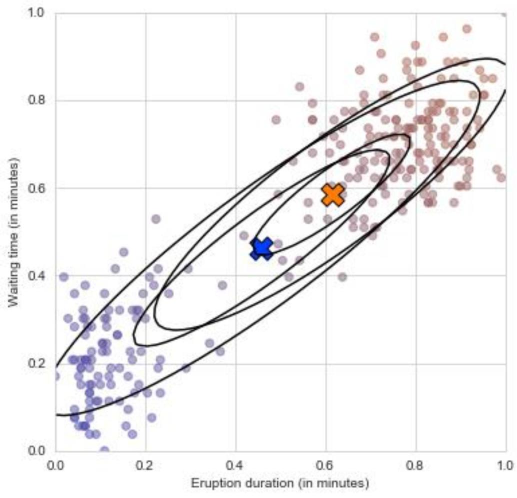
3. Log likelihood for the (incomplete, i.e. without the information of probability of coming from each Gaussian distribution) data was then calculated in the gmm_log_likelihood() function.

4. In the function E_step(), the responsibilities of each of the training data points, with respect to each of the components of the Gaussian mixture were calculated. Basically, this is obtained by finding the expectation of the log likelihood of the complete data.
5. In the function M_step(), the expectation obtained in the E-step is maximized by varying the parameters of the distributions, i.e. the means, Covariance matrices and mixing coefficients for each of the distributions. Thus the means, Covariance matrices and mixing coefficients are updated for each iteration of the algorithm, going to convergence.
6. Final loop for running the EM Algorithm keeps track of the log likelihood at each step and the improvement in its value as compared to the previous step. The means, Covariance matrices and mixing coefficients are updated till convergence is reached. The result is as follows:

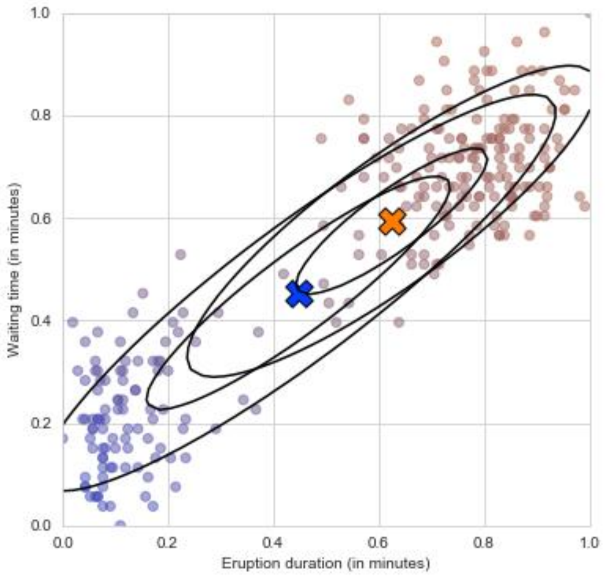
   a. Initially, log likelihood is -382.57812548433213



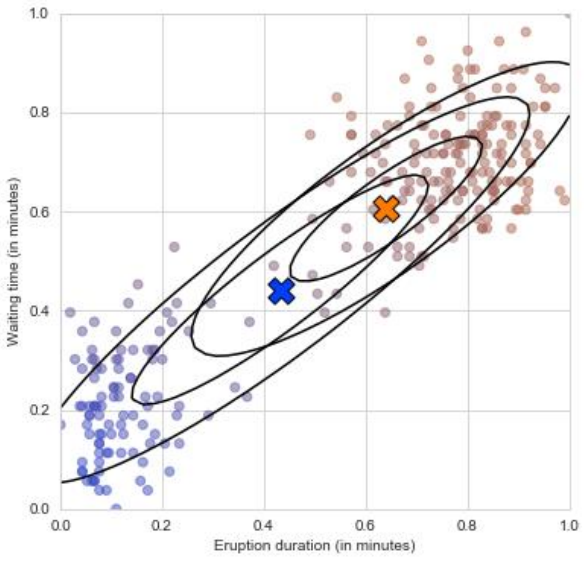
   b. After iteration 0, log likelihood is 132.35, with improvement 514.93



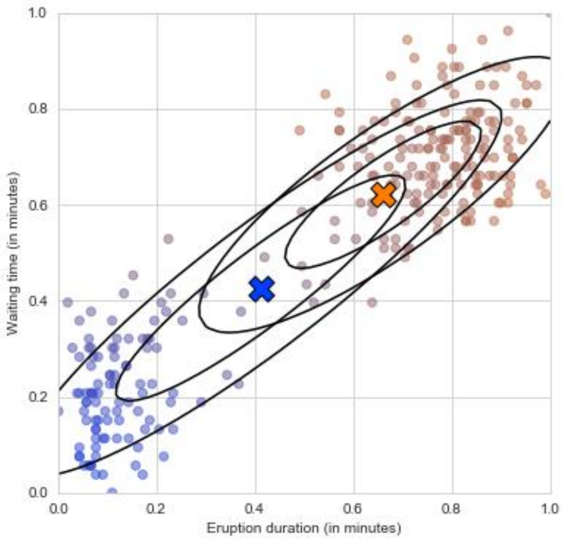
   c. After iteration 1, log likelihood is 133.20, with improvement 0.85



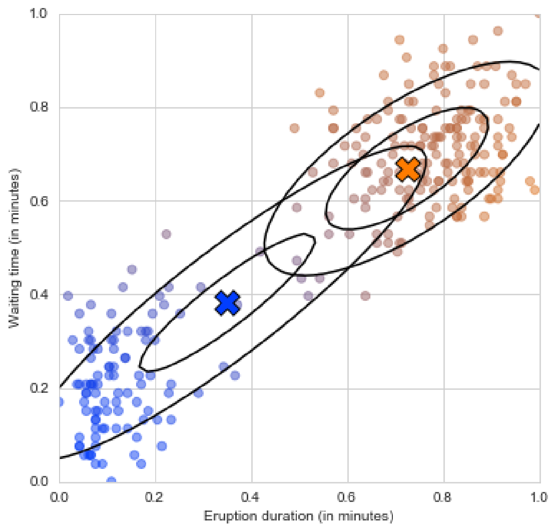
d. After iteration 2, log likelihood is 134.64, with improvement 1.43



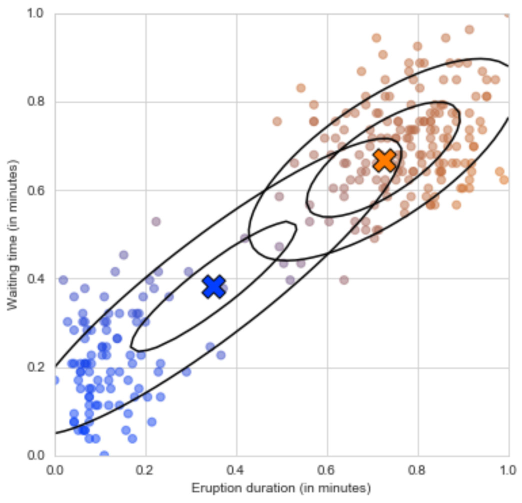
e. After iteration 3, log likelihood is 137.41, with improvement 2.78



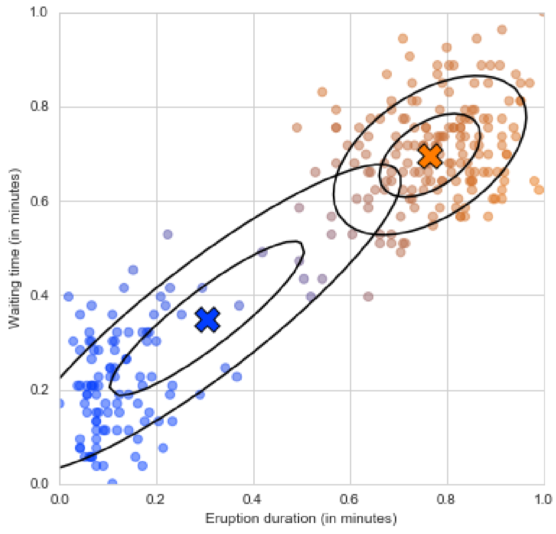
f. After iteration 4, log likelihood is 143.78, with improvement 6.37



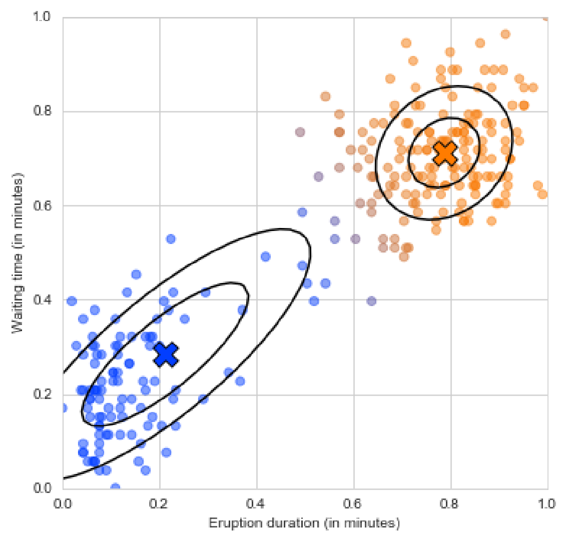
g. After iteration 5, log likelihood is 161.28, with improvement 17.50



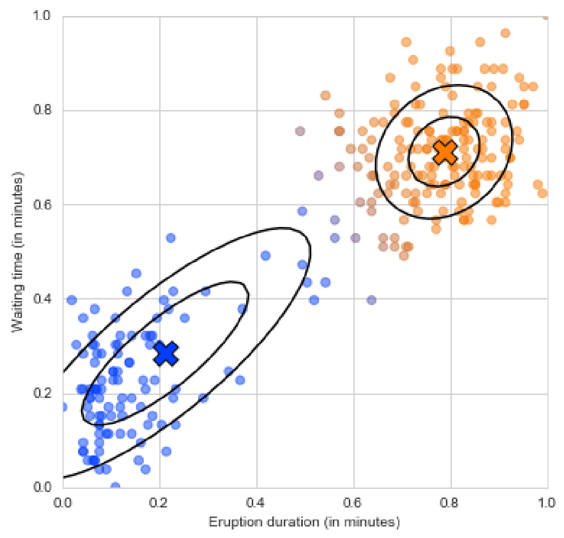
h. After iteration 6, log likelihood is 202.52, with improvement 41.24



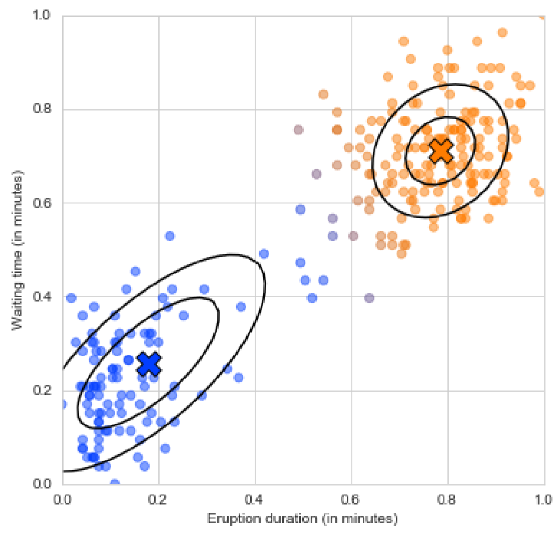
i. After iteration 7, log likelihood is 229.68, with improvement 27.16



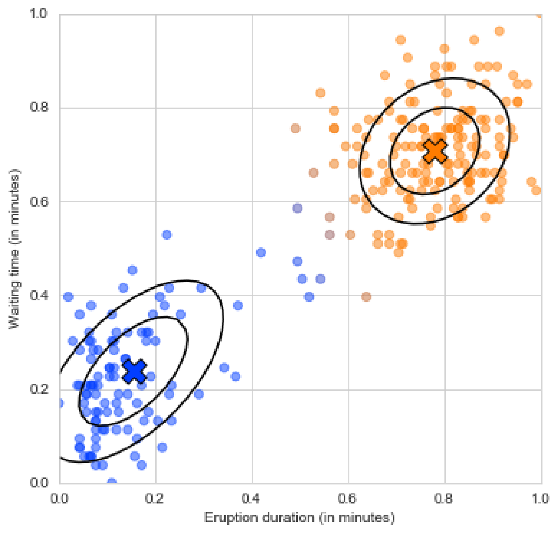
j. After iteration 8, log likelihood is 241.92, with improvement 12.24



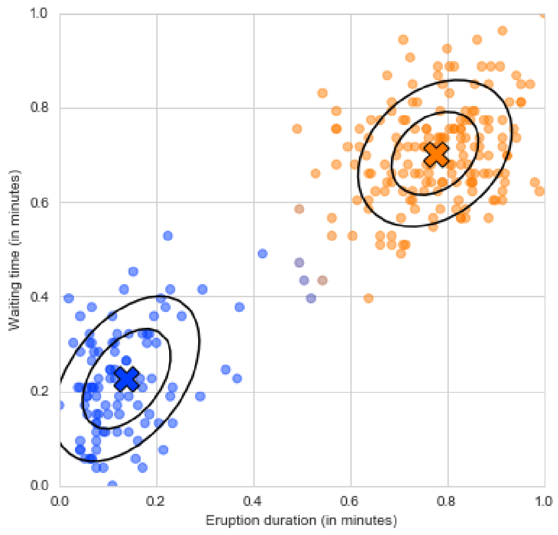
k. After iteration 9, log likelihood is 255.86, with improvement 13.93



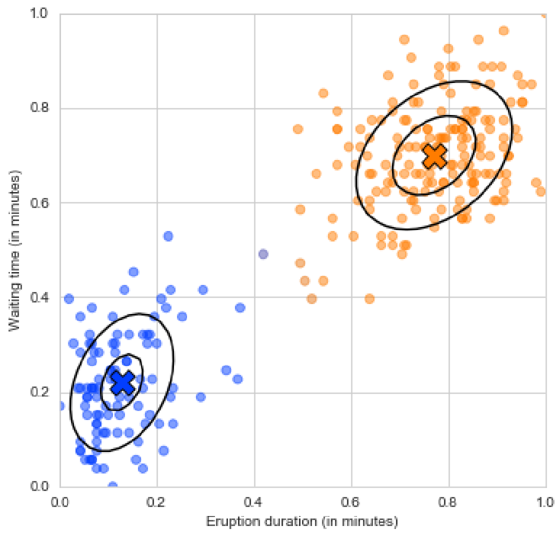
l. After iteration 10, log likelihood is 270.47, with improvement 14.61



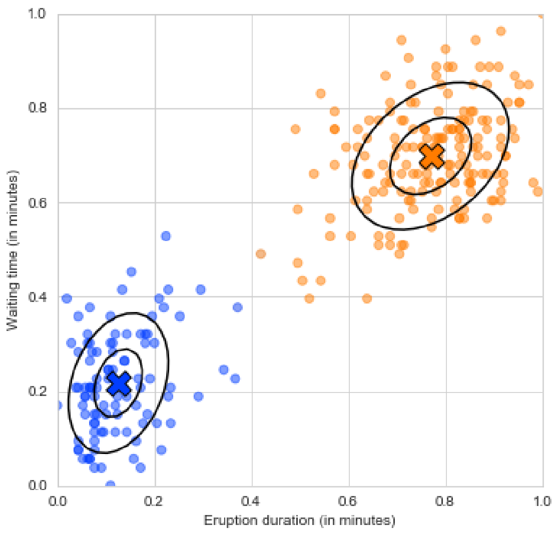
m. After iteration 11, log likelihood is 282.15, with improvement 11.68



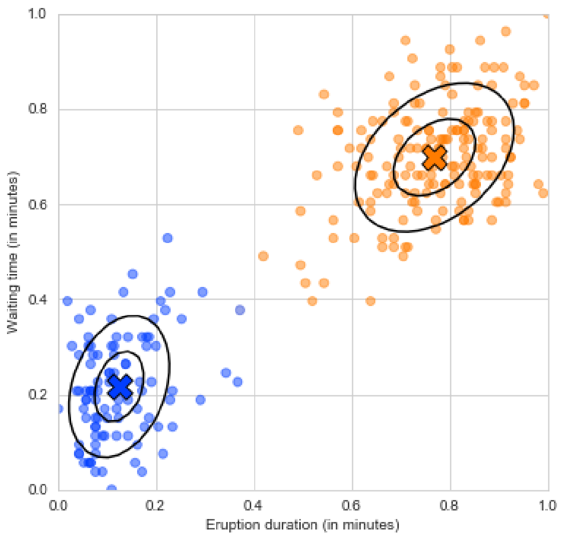
n. After iteration 12, log likelihood is 289.52, with improvement 7.38



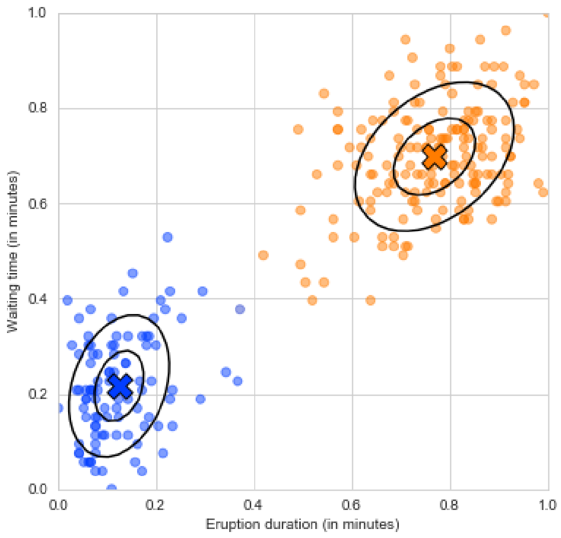
o. After iteration 13, log likelihood is 290.38, with improvement 0.86



p. After iteration 14, log likelihood is 290.41, with improvement 0.03



q. After iteration 15, log likelihood is 290.41, with improvement 0.00



The final values of the means, covariance matrices and mixing coefficients are:

Means: [[0.12469026, 0.216581], [0.76848179, 0.69751715]]
Covariance Matrices:
[[0.00564812, 0.00234715], [0.00234715, 0.01199674]] for first component
[[0.01387246, 0.00506856], [0.00506856, 0.01283083]] for second component
Mixing Coefficients: [[0.35588414], [0.64411586]]

Speciality of this procedure:

This is a nice iterative procedure to obtain the means, covariance matrices and mixing coefficients of Gaussian Mixture Models. This can be treated as a variant of clustering of datasets and of figuring out the ML estimates of the parameters for the assumed Gaussian Probability density function of the data. As this procedure gives us the additional information of the covariance matrices and mixing coefficients, this is better than the clustering algorithms (like K-means) which just group data points together based on the information of the ML estimates of the means.

Advantages:

1. This algorithm is very advantageous as it enables the computation of the probability density of the data based Gaussian Mixture Models, using the information of the hypothesized “complete” data (i.e. hypothesizing the probabilities of each data point drawn from each component of the distribution), in an iterative way.
2. At each step of the algorithm, we maximize the log likelihood further.
3. This algorithm does not require any optimizer for its implementation as it is derivative-free.
4. Using this, the complete data probability density can be constructed. 4
5. In most cases, closed form solutions exist.

Disadvantages:

1. This algorithm converges very slowly.
2. Being an iterative algorithm, it can even converge to local minima.