Contrasting k-Means, k-Nearest Neighbors and maximum likelihood for classification

Mohamed Nazaal Ibrahim and Roget Kou*
Department of Computer science, University of Bristol
(Dated: March 19, 2018)

I. INTRODUCTION

This report aims to describe, explain and analyze the results of applying 3 distinct classification algorithms on 2 data sets; k-Means clustering, k-Nearest Neighbors classification and Maximum-likelihood classification. Each data set consists of 150 labelled samples for training the model, and 15 unlabelled samples for testing the model.

II. FEATURE SELECTION AND CLASS IDENTIFICATION

The data provided had samples with 5 features; that is, each sample was defined by a vector $v \in \mathbb{R}^5$. Let D_{Train} be the $f \times m$ matrix storing the data samples, where f is the number of features, and m is the number of samples to be used in the model. For the training data in files mi16481.train and rk16699.train, $m = 150$, while for the testing data in files mi6481.test and rk16699.test, $m = 15$. $f = 5$ for all samples.

In classification problems, it is common to apply dimensionality reduction during pre-processing. This could be interpreted as selecting only relevant features for the given classification problem. Dimensionality reduction is helpful in the sense that it increases the speed of learning, and removes noise from the data, which include independent random noises, and unwanted degrees of freedom which may possibly be non-linear. [1] Since the training data is not labelled, it is required to manually select certain features which segment the data into separate clusters before training the model.

Choosing the relevant features is a separate and important task in itself. Plotting a scatter matrix for the given data would be helpful in this case. Given a data set of m samples where each sample has f features, a scatter matrix is a $f \times f$ matrix, where the $(i, j)^{th}$ element is a scatter plot of the i^{th} feature against the j^{th} feature. The scatter matrices for both training data sets is shown in Figures 1a and 1b.

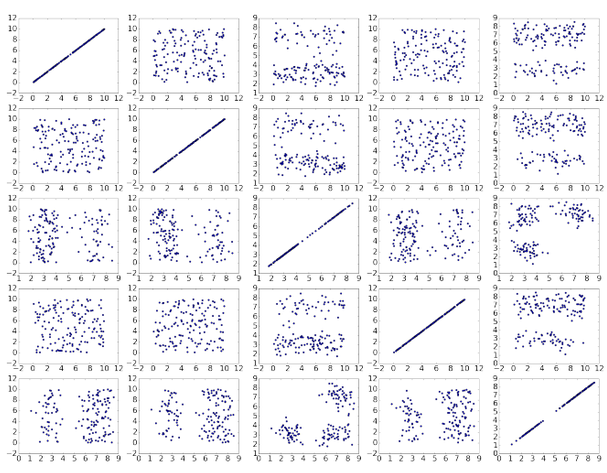

Figure 1a : Scatter matrix of of mi16481.train

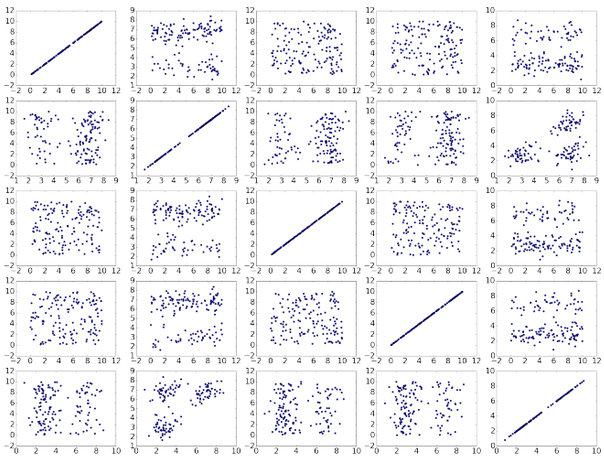

Figure 1b : Scatter matrix of rk16699.train

Note that the diagonal entries form a straight line with a gradient of 1 since they consist of the scatter plot of every feature with itself. Also, for each of the $(i, j)^{th}$ element not in the diagonal, the $(j, i)^{th}$ is a reflection of the former with respect to uni-variate identity function. From Figure 1a, it can be seen that the $(3, 5)^{th}$ and $(5, 3)^{th}$ elements have scatter plots which segment the data into 3 clusters. From Figure 1b, it can be seen that $(2, 5)^{th}$ and $(5, 2)^{th}$ show a scatter plot with a good clustering. By denoting D_{mi} and D_{rk} as the matrices storing the data from mi16481.train and rk16699.train respectively, the next step is to extract the 3^{rd} and 5^{th} rows from D_{mi} and the 2^{nd} and 5^{th} rows from D_{rk}. Let F_{mi} and F_{rk} be the mappings which apply dimensionality reduction to D_{mi} and D_{rk} respectively. F_{mi} and F_{rk} can be represented by the matrices

$$M = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

respectively.
Let X_{mi} and X_{rk} be the 2×150 matrices storing the data after dimensionality reduction.
Then $X_{mi} = F(D_{mi}) = MD_{mi}$ and $X_{rk} = F(D_{rk}) = RD_{rk}$. The same is done for the test data sets which would be required after training the model.
It should be noted however that due to the small number of features extracted, when it comes to implementation, it is more practical to extract relevant features via row indices instead of via a dimensionality reduction mapping.

III. K-MEANS CLUSTERING

k-Means clustering is an unsupervised classification algorithm, that is, an algorithm used to derive structure out of unstructured data. The aim of the k-means algorithm is to divide m points in f dimensions into K clusters so that the within-cluster sum of squares, also called inertia, is minimized.[2]
The algorithm is given below;
Let k be the number of clusters to group the data
Generate k centroids, where each centroid $c_i \in \mathbb{R}^f$.
Then, repeat the following steps until convergence.
For each sample point x_i:
- find the nearest centroid c_j via the given metric

* m.nazaal.2016@bristol.ac.uk; rk16699@my.bristol.ac.uk

- assign the sample point x_i to cluster w_j
Then for each cluster:
- assign the new centroid to be the arithmetic mean of the feature vectors currently in the cluster

Convergence occurs when cluster assignments stop happening. This means that within each cluster the inertia is minimized. However, it is not guaranteed that a global minimum would be achieved. As a result, in practical applications, it is common to run the algorithm a few times and use the minimum inertia is achieved.

The result of applying K-Means clustering with $k = 3$ on both training data sets is shown in Figures 2a and 2b.

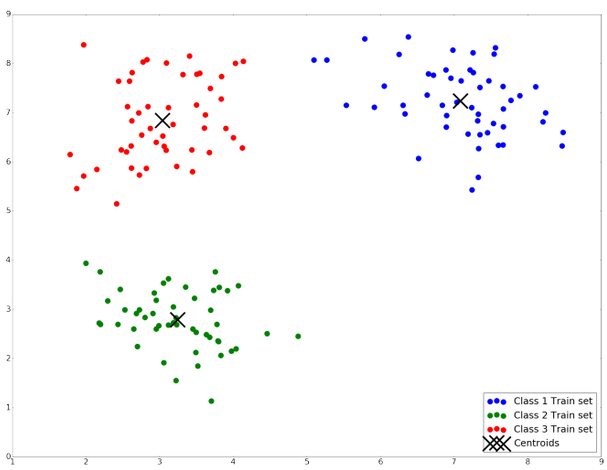

Figure 2a : k-Means clustering applied to mi16481.train

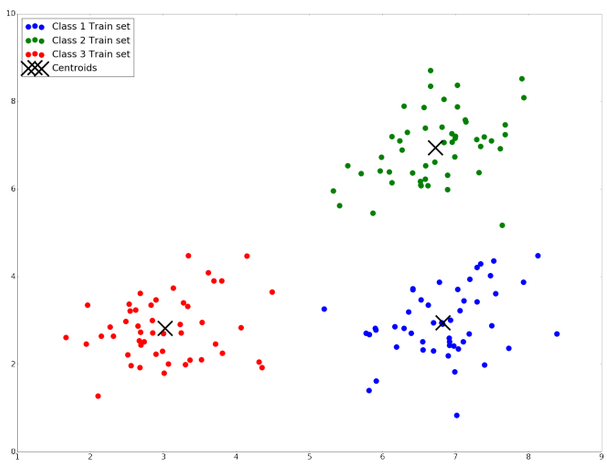

Figure 2b : k-Means clustering applied to rk16699.train

To visualize the effects of a non-optimal clustering, it is needed to maximize the sum of the distances squares, or the inertia, within each cluster. Limiting the number of iterations of the algorithm will help to maximize the inertia, since it would stop before any signs of convergence. To make sure the clustering it non-optimal, it is safe to run the algorithm a few hundred times (500 in this case) and then taking the clustering where the inertia is maximum. The results of the non-optimal clustering is shown in Figures 3a and 3b.

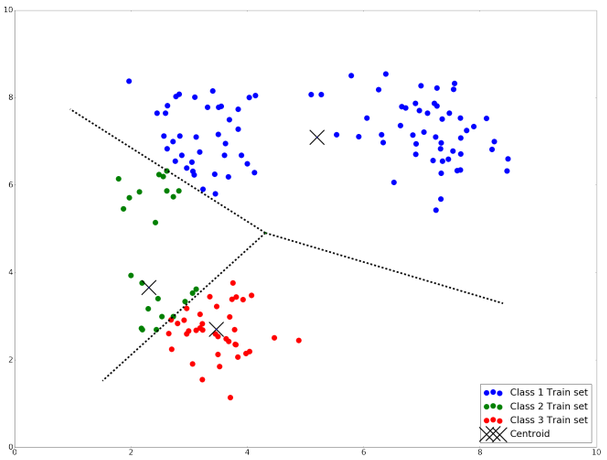

Figure 3a : Non-optimal k-Means clustering applied to mi16481.train with Voronoi diagram

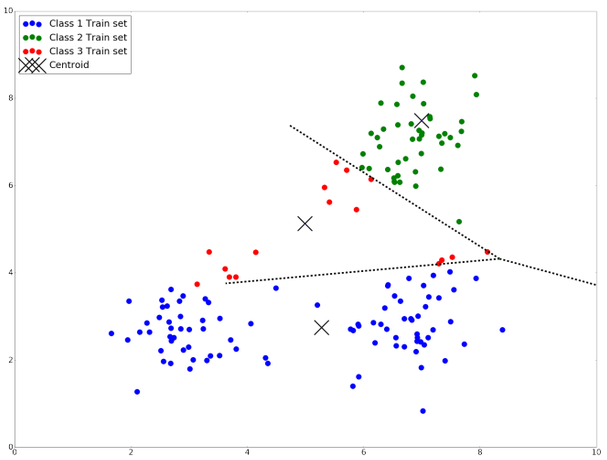

Figure 3b : Non-optimal k-Means clustering applied to rk16699.train with Voronoi diagram

The classifier is clearly inaccurate for both data sets. For mi16481.train. the centroid for class 1 has very few points near it, whilst the decision boundary for the classes 2 and 3 itself contains a lot of samples on it. For rk16699.train, the centroid for the classes 1 and 3 have very few points near it.

IV. K-NEAREST NEIGHBORS

The k-Nearest Neighbor algorithm is an algorithm which can be either used for classification or regression. It falls under the category of non-parametric regression techniques. A non-parametric regression technique makes its prediction by making use of information previously derived from the data, without taking any predetermined form. [3] The algorithm is defined below;

If used for classification:
Let m denote the size of the training data set, k denote the number of samples used for comparison, and $(\boldsymbol{x}_i, y_i)$ denote a labelled training sample, where $\boldsymbol{x}_i \in \mathbb{R}^f$ represents the i^{th} feature vector, and y_i is it's class. Note that $k < m$, and ideally $k << m$. Generate m regions $R_i \subset \mathbb{R}^f$ called the Voronoi cell, such that they form a partition in $\mathbb{R}^f$. If $k = 1$, the boundaries formed where training data samples from every 2 classes meet form the decision boundary between the classes.
For each sample point z_i in the training data set:
- find the k nearest points to $\boldsymbol{z}_i$ based on the given metric, and get the classes they belong to.
- classify $\boldsymbol{z}_i$ to be in the class which is the majority class from the k nearest points found above.

If used for regression, on a function $a : \mathbb{R}^f \to \mathbb{R}$:
Let m and k be the same as before with the same restrictions, and $(\boldsymbol{x}_i, y_i)$ denote a training sample

where $\boldsymbol{x}_i \in \mathbb{R}^f$ is the independent variable and $y_i \in \mathbb{R}$ is the dependent variable.
To estimate the value of $a(\boldsymbol{z})$, where $\boldsymbol{z} \in \mathbb{R}^f$
- Find the k nearest independent variables in the training data set to z with respect to the given metric.
- Compute the arithmetic means of the value of a at those points. This is the estimate value of $a(\boldsymbol{z})$.

In the given case, the training data set is the sample of 3 centroids generated from k-Means clustering in the previous section, alongside their labels. Since there is only 1 sample from each class, the only plausible action to do is to apply k-Nearest Neighbors with $k = 1$, since $k = 2$ or $k = 3$ will lead to the sample being classified in more than 1 class. Hence, k-Nearest Neighbours is applied to this training data set to partition $\mathbb{R}^2$ into 3 segments, where the metric is the Euclidean distance. This is visualized in Figures 4a and 4b, where the training sample points and test sample points are plotted to check the accuracy of the classifier.

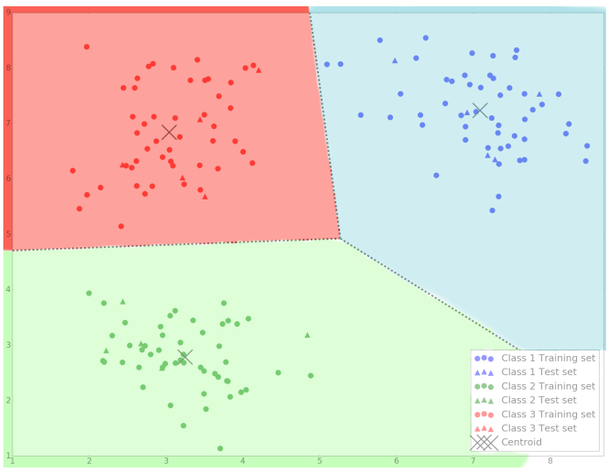

Figure 4a : k-Nearest Neighbor classification with Voronoi diagram

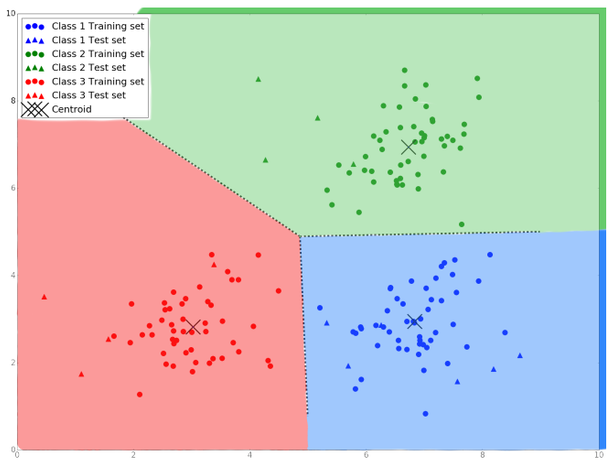

Figure 4b : k-Nearest Neighbor classification with Voronoi diagram

As seen from the diagrams above, the algorithm does a good job classifying the test data samples correctly, without any issues. However, it must be noted that k-Nearest Neighbors with $k = 1$ has the limitation that if the training data has a lot of samples, the decision boundary might be over-fitted. Also, suppose that a point $\boldsymbol{x}_k$ from class i had some outlier feature values and as a result is nearer to points in class j. This forms a Voronoi cell for $\boldsymbol{x}_k$ in the space which should belong to class j. In this case, the classifier may label a test point near $\boldsymbol{x}_k$ to be in class i while in fact it belongs to class j. This outlier problem is a side-effect of the fact that k-Nearest Neighbors algorithm assumes a uniform prior probability distribution.

V. MAXIMUM LIKELIHOOD CLASSIFICATION

In this algorithm, the data is assumed to have a multivariate Gaussian distribution with f dimensions, with mean vector $\boldsymbol{\mu} \in \mathbb{R}^f$ and $f \times f$ co-variance matrix $\boldsymbol{\Sigma}$. It has a probability density function of

$$f_{\mathbf{X}}(x_1, \ldots, x_f) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^f|\boldsymbol{\Sigma}|}} \tag{1}$$

This happens to be a special case of an elliptical distribution.[4] In this model, each class is assumed to have a 2-D Gaussian distribution. The unbiased sample mean for class c, $\widehat{\boldsymbol{\mu}_c} = \frac{1}{m}\sum_{n=1}^{m}\boldsymbol{x}_n$. The unbiased sample co-variance matrix for class c, $\widehat{\boldsymbol{\Sigma}_c}$ is a $f \times f$ matrix where the $(i,j)^{th}$ element is the sample co-variance between the i^{th} and j^{th} element, measured as $\frac{1}{m-1}\sum_{n=0}^{m}(x_{ni} - \boldsymbol{\mu}_i)(x_{nj} - \boldsymbol{\mu}_j)$, where x_{ni} is the n^{th} value of the i^{th} feature, f and m are the number of features and samples respectively. After calculating the sample mean vector and sample co-variance matrices for the training data, and calculating the probability density for each class, it is possible to get the following plot for the density at each value of the feature space, where for each class, the volume under the surface is corresponds to the probability.

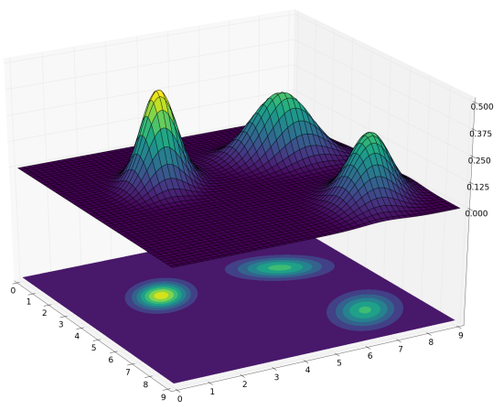

Figure 4a : Probability density and contour levels for mi16481.train (not normalized)

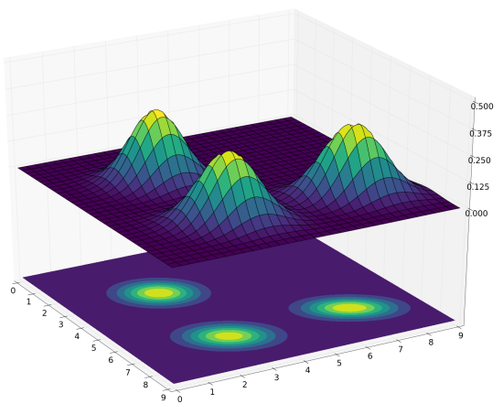

Figure 4b : Probability density and contour levels for rk16699.train (not normalized)

The Mahalabonis distance, d_M, is a generalization of the Euclidean metric to include the variation between sample points.
${d_M}^2 = (\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})$. It is the distance metric used in the multivariate Gaussian distribution, and has a chi-squared distribution with f degrees of freedom,[5] where f is the number of dimensions of the distribution, which in this case is 2.

Analytically, an ellipsoid in $\mathbb{R}^f$ with center $\boldsymbol{x_0}$ is denoted as $\mathcal{E} = \{\boldsymbol{x} : (\boldsymbol{x} - \boldsymbol{x_0})^T \mathbf{C} (\boldsymbol{x} - \boldsymbol{x_0}) = z\}$, for some $z \in \mathbb{R}$ where $\boldsymbol{x} \in \mathbb{R}^f$ and $\mathbf{C}$ is a $f \times f$ symmetric positive definite matrix.[6] In the case $\boldsymbol{x}_0 = \boldsymbol{\mu}$ and $\mathbf{C} = \boldsymbol{\Sigma^{-1}}$, where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma^{-1}}$ are the mean and inverse of the co-variance matrix of a Gaussian distribution respectively, then the region enclosed by the ellipsoid $\mathcal{E}$, which shall be denoted as $\mathcal{E}_R$, represents the α % confidence ellipsoid of the distribution, where z is the quantile function output, or the inverse cumulative density function value for χ^2_f, i.e. the chi-squared distribution with f degrees of freedom. This is because the Mahalabonis distance and the chi-square distribution with f degrees of freedom have the same probability density function. The given data has samples with 2 features, thus in this case, $f = 2$. This means the confidence ellipsoid is in fact an ellipse in 2D space. The probability distribution of χ^2_2 is also the same as the exponential distribution with a rate of $\frac{1}{2}$ which has a quantile function of $Q(p) = -2log(1 - p)$. At 0.95%, it equates to $-2log(0.05) \approx 5.9915$. The confidence ellipse is the contour of the probability density value when $(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) = 5.9915$ for each class in each training data set. This calculation yields the following results.

Table 1 - results from mi16481.train

Class	$\hat{\boldsymbol{\mu}}_i$	$\hat{\boldsymbol{\Sigma}}_i$	t_i
1	$\begin{bmatrix} 3.0423 \\ 6.8405 \end{bmatrix}$	$\begin{bmatrix} 0.3763 & 0.1537 \\ 0.1537 & 0.7055 \end{bmatrix}$	0.0157
2	$\begin{bmatrix} 3.2477 \\ 2.7853 \end{bmatrix}$	$\begin{bmatrix} 0.390 & 0.1071 \\ 0.1071 & 0.3445 \end{bmatrix}$	0.0227
3	$\begin{bmatrix} 7.0871 \\ 7.2337 \end{bmatrix}$	$\begin{bmatrix} 0.5764 & -0.2038 \\ -0.2038 & 0.5204 \end{bmatrix}$	0.0162

Table 2 - results from rk16699.train

Class	$\hat{\boldsymbol{\mu}}_i$	$\hat{\boldsymbol{\Sigma}}_i$	t_i
1	$\begin{bmatrix} 6.8308 \\ 2.9453 \end{bmatrix}$	$\begin{bmatrix} 0.4131 & 0.1569 \\ 0.1569 & 0.6061 \end{bmatrix}$	0.0167
2	$\begin{bmatrix} 6.7270 \\ 6.9452 \end{bmatrix}$	$\begin{bmatrix} 0.3908 & 0.2082 \\ 0.2082 & 0.6476 \end{bmatrix}$	0.0174
3	$\begin{bmatrix} 3.024687 \\ 2.8228 \end{bmatrix}$	$\begin{bmatrix} 0.4290 & 0.0977 \\ 0.0977 & 0.5144 \end{bmatrix}$	0.0173

The results from plotting the contours for the probability density surface of each class in each dataset at the required threshold level t_i is shown below.

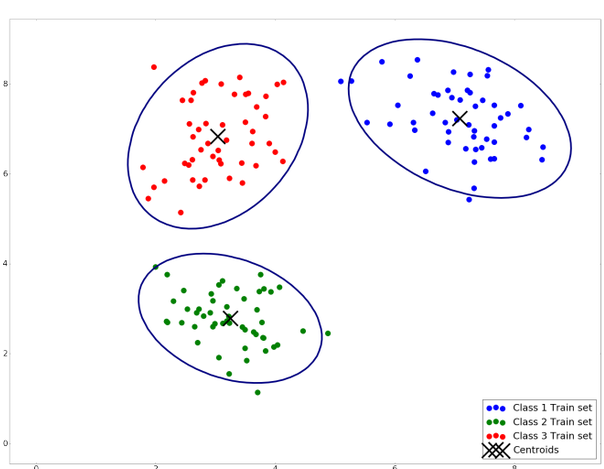

Figure 5a : 95% contour ellipses for mi16481.train

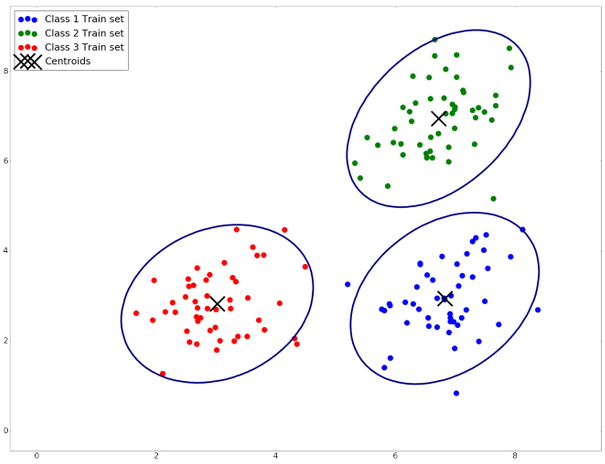

Figure 5b : 95% contour ellipses for rk16699.train

The noticeable difference in the size of one ellipse compared to the other 2 in Figure 5a matches with the intuition that since the class with the mean vector nearest to $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$ forms a probability density surface with a higher peak than the other 2 classes (due to lesser spread), the 3-D shape formed between the surface and the feature space plane with volume 0.95 has a smaller base-area, where the base is the ellipse. The other two classes in this data set have samples with a relatively higher spread, thus to keep the volume of the 3-D shape formed between the surface and the feature space plane constant at 0.95, the base-area of the ellipse must be larger. Meanwhile, Figure 5b has ellipses with roughly the same area, and this can be predicted from the probability density surface in Figure 4b, where the height and spread of the function is roughly similar for all classes. The ellipses in this figure being tilted to such an angle is a result of the structure of the co-variance matrices, which can be found in Table 2. The co-variance between the the features in all 3 classes in this data set is positive, indicating a positive correlation.

To find the decision boundaries for each pair of classes in each of the training data sets, it is required to calculate the pairwise likelihood ratios and plot their contours. The decision boundary between class i and j is a contour plot of the discriminant function $\boldsymbol{g}_{(i,j)} = \frac{\mathbb{P}(\boldsymbol{x}|\omega i)\mathbb{P}(\omega_i)}{\mathbb{P}(\boldsymbol{x}|\omega j)\mathbb{P}(\omega_j)}$, where ω_k refers to class k. The choice of this discriminant function is not unique. When $\boldsymbol{g}_{(i,j)} > 1$, the sample is classified into class class i, and if $\boldsymbol{g}_{(i,j)} < 1$, the sample is classified into class j. The level curve formed when $\boldsymbol{g}_{(i,j)} = 1$ determines the space where the sample is equally likely to be in either of the 2 classes, thus this curve forms the decision boundary. Hence, the placement of the likelihood ratio on the numerator and denominator does not make a difference when plotting the decision boundary. The result of calculating the 3 pairwise discriminant functions and plotting their contours when $\boldsymbol{g}_{(i,j)} = 1$ for all the combinations of i,j is given in Figures 6a and 6b. In this case, the prior probability distribution is assumed to be uniform.

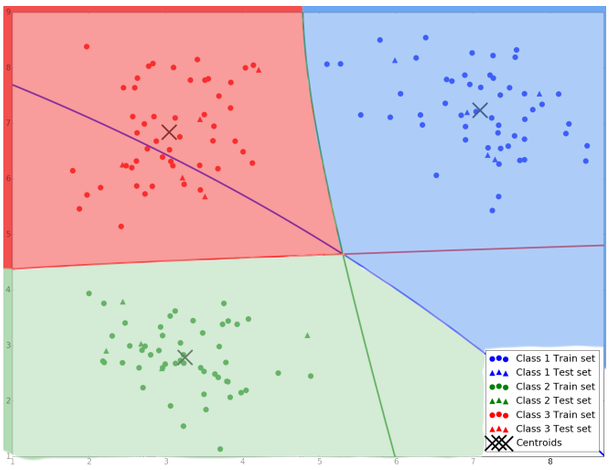

Figure 6a : Non-Linear decision boundaries on mi16481.train and mi16481.test

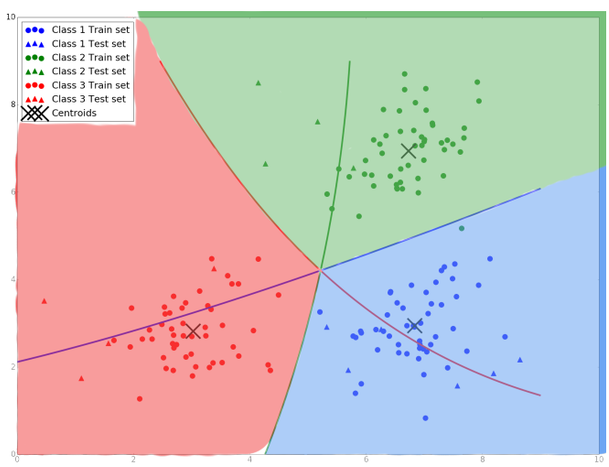

Figure 6b : Non-Linear decision boundaries on rk16699.train and rk16699.test

In both figures, the blue line is the decision boundary between classes 1 2, the red line divides for classes 2 and 3, while the green line is the decision boundary for classes 1 and 3. Also for both figures,the light red area is the region $\{\frac{\mathbb{P}(\boldsymbol{x}|\omega 3)\mathbb{P}(\omega_3)}{\mathbb{P}(\boldsymbol{x}|\omega 2)\mathbb{P}(\omega_2)} > 1 \cap \frac{\mathbb{P}(\boldsymbol{x}|\omega 3)\mathbb{P}(\omega_3)}{\mathbb{P}(\boldsymbol{x}|\omega 1)\mathbb{P}(\omega_1)} > 1\}$, the light green area represents the region $\{\frac{\mathbb{P}(\boldsymbol{x}|\omega 2)\mathbb{P}(\omega_2)}{\mathbb{P}(\boldsymbol{x}|\omega 1)\mathbb{P}(\omega_1)} > 1 \cap \frac{\mathbb{P}(\boldsymbol{x}|\omega 2)\mathbb{P}(\omega_2)}{\mathbb{P}(\boldsymbol{x}|\omega 3)\mathbb{P}(\omega_3)} > 1\}$, and the light blue area represents the region $\{\frac{\mathbb{P}(\boldsymbol{x}|\omega 1)\mathbb{P}(\omega_1)}{\mathbb{P}(\boldsymbol{x}|\omega 2)\mathbb{P}(\omega_2)} > 1 \cap \frac{\mathbb{P}(\boldsymbol{x}|\omega 1)\mathbb{P}(\omega_1)}{\mathbb{P}(\boldsymbol{x}|\omega 3)\mathbb{P}(\omega_3)} > 1\}$.

Note that $\boldsymbol{g}_{(i,j)} > 1 \iff \frac{\mathbb{P}(\boldsymbol{x}|\omega_i)\mathbb{P}(\omega_i)}{\mathbb{P}(\boldsymbol{x}|\omega_j)\mathbb{P}(\omega_j)} > 1 \implies \frac{log(\mathbb{P}(\boldsymbol{x}|\omega_i)\mathbb{P}(\omega_i))}{log(\mathbb{P}(\boldsymbol{x}|\omega_j)\mathbb{P}(\omega_j))} > 1$, since $log(x)$ is monotone increasing $\forall x \in \mathbb{R}$ where $log(x)$ is shorthand $log_e(x)$. Also, $\frac{log(\mathbb{P}(\boldsymbol{x}|\omega_i)\mathbb{P}(\omega_i))}{log(\mathbb{P}(\boldsymbol{x}|\omega_j)\mathbb{P}(\omega_j))} > 1 \implies \boldsymbol{g}_{(i,j)} > 1$ since $log(x)$ is invertible $\forall x \in \mathbb{R}$.

Hence, $\boldsymbol{g}_{(i,j)} > 1 \iff \frac{log(\mathbb{P}(\boldsymbol{x}|\omega_i)\mathbb{P}(\omega_i))}{log(\mathbb{P}(\boldsymbol{x}|\omega_j)\mathbb{P}(\omega_j))} > 1$.

Rearranging the terms further shows that $\boldsymbol{g}_{(i,j)} > 1 \iff log(\mathbb{P}(\boldsymbol{x}|\omega_i)\mathbb{P}(\omega_i)) - log(\mathbb{P}(\boldsymbol{x}|\omega_j)\mathbb{P}(\omega_j)) > 0$. Let $\boldsymbol{h}_{(i,j)} = log(\mathbb{P}(\boldsymbol{x}|\omega_i)\mathbb{P}(\omega_i)) - log(\mathbb{P}(\boldsymbol{x}|\omega_j)\mathbb{P}(\omega_j))$.

For the given data, the co-variance matrices contain non-zero correlation values as seen in Table 1. let Thus, the discriminant function $\boldsymbol{h}_{(i,j)}$ for each pair of classes can be written in the form

$$\boldsymbol{h}_{(i,j)} = [-\tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}_i^{-1}\boldsymbol{x} + \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i\boldsymbol{x} - \tfrac{1}{2}\boldsymbol{\mu}_i^T\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i - \tfrac{1}{2}log|\boldsymbol{\Sigma}_i^{-1}| + log(\mathbb{P}(\omega_i)] - [-\tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}_j^{-1}\boldsymbol{x} + \boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j\boldsymbol{x} - \tfrac{1}{2}\boldsymbol{\mu}_j^T\boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j - \tfrac{1}{2}log|\boldsymbol{\Sigma}_j^{-1}| + log(\mathbb{P}(\omega_j)].[7]$$

Where $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ denote the the sample mean and sample co-variance matrix for class i. This further simplifies to

$$\boldsymbol{h}_{(i,j)} = [-\tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}_i^{-1}\boldsymbol{x} + \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i\boldsymbol{x} - \tfrac{1}{2}\boldsymbol{\mu}_i^T\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i] - [-\tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}_j^{-1}\boldsymbol{x} + \boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j\boldsymbol{x} - \tfrac{1}{2}\boldsymbol{\mu}_j^T\boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j] + log\Big(\frac{|\boldsymbol{\Sigma}_i|\mathbb{P}(\omega_i)}{|\boldsymbol{\Sigma}_j|\mathbb{P}(\omega_j)}\Big)$$

This is a non-linear function,[7] which explains why the decision boundaries are non-linear.

Another explanation for the non-linearity is due to the co-variance between the two features within each class being non-zero. This can be seen from the decision boundary since it curves while moving along the feature space, implying one feature influences the other. If instead the pairwise co-variance between the 2 features were 0, this implies that within each class, the 2 features are independent. In this case, the decision boundaries do not curve when they move along the feature space, since they do not influence one another, and thus are linear.

In order to make the decision boundaries the same as the one's calculated from the k-Nearest Neighbors classifier previously, it is required to make the discriminant function $\boldsymbol{h}_{(i,j)}$ linear, whilst preserving the accuracy of the classifier. By introducing the assumption of independence between the 2 features within each class, and letting $\boldsymbol{\Sigma_i} = \sigma^2\boldsymbol{I}$[7] for all classes, the discriminant function $\boldsymbol{h}_{(i,j)}$ can be re-written in the form

$$\boldsymbol{h}_{(i,j)} = \tfrac{1}{2\sigma^2}(-2\boldsymbol{\mu}_i^T\boldsymbol{x} + 2\boldsymbol{\mu}_j^T\boldsymbol{x} + \boldsymbol{\mu}_i^T\boldsymbol{\mu}_i - \boldsymbol{\mu}_j^T\boldsymbol{\mu}_j) + log\Big(\frac{\mathbb{P}(\omega_i)}{\mathbb{P}(\omega_j)}\Big)$$

This is a linear function. Moreover, since the prior probability distribution is assumed to be a discrete uniform distribution, the logarithm vanishes. Hence, the decision boundary, which is the contour level where $\boldsymbol{h}_{(i,j)} = 0$ does not depend on σ^2 at all since $\frac{1}{2\sigma^2}(-2\boldsymbol{\mu}_i^T\boldsymbol{x} + 2\boldsymbol{\mu}_j^T\boldsymbol{x} + \boldsymbol{\mu}_i^T\boldsymbol{\mu}_i - \boldsymbol{\mu}_j^T\boldsymbol{\mu}_j) = 0 \implies -2\boldsymbol{\mu}_i^T\boldsymbol{x} + 2\boldsymbol{\mu}_j^T\boldsymbol{x} + \boldsymbol{\mu}_i^T\boldsymbol{\mu}_i - \boldsymbol{\mu}_j^T\boldsymbol{\mu}_j = 0$. The results of plotting the contour levels where $\boldsymbol{g}_{(i,j)} = 1$ or equivalently, where $\boldsymbol{h}_{(i,j)} = 0$ is shown in Figures 7a and 7b.

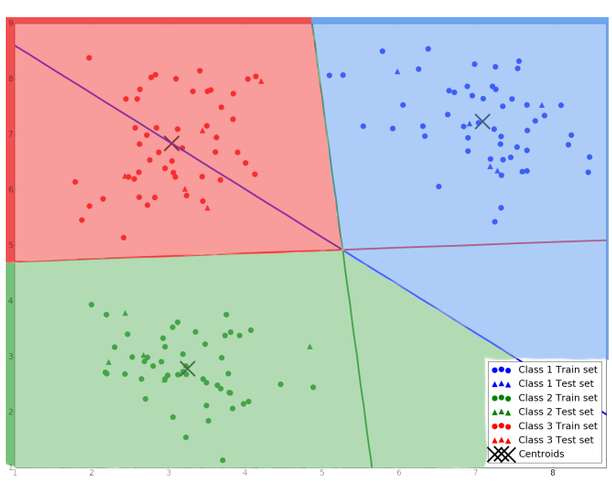

Figure 7a : Linear decision boundaries on mi16481.train and mi16481.test

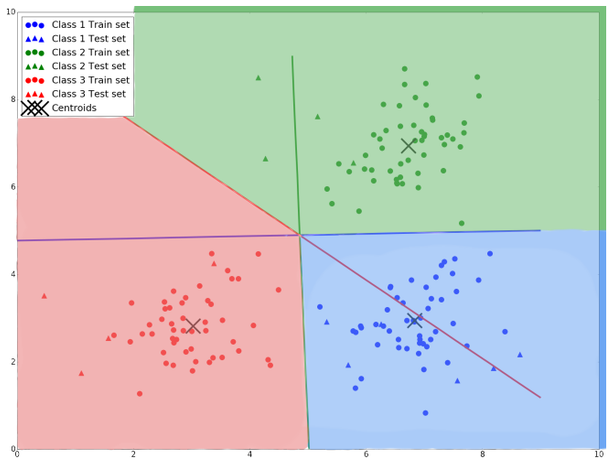

Figure 7b : Linear decision boundaries on rk16699.train and rk16699.test

These decision boundaries align to the ones generated from the k-Nearest Neighbors algorithm from Figures 4a and 4b. This is because both classifiers only use the Euclidean distance from the training data sample means to construct the decision boundaries.

The introduction of a non-uniform prior probability distribution would make the model more realistic. Suppose one class is twice as more likely to be observed compared to the other two. Without loss of generality, assume $\mathbb{P}(\omega_1) = 0.5$, and $\mathbb{P}(\omega_2) = \mathbb{P}(\omega_3) = 0.25$ for both data sets. The effect of this prior distribution can be seen in Figures 8a and 8b. The non-linear decision boundaries where the prior probability distribution was assumed to be uniform is also plotted on the same axes to visualize the difference made by the new non-uniform prior probability distribution.

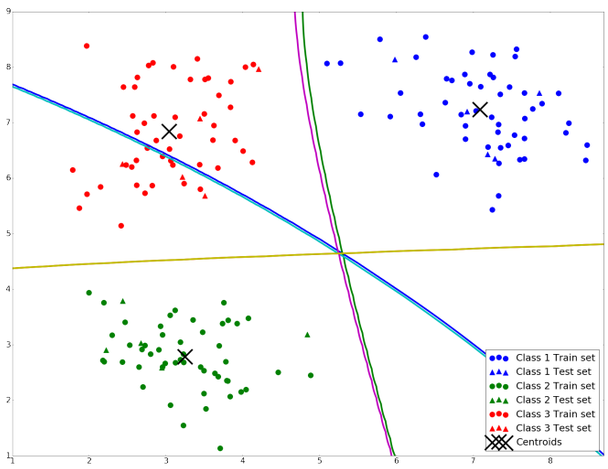

Figure 8a : Non-linear decision boundaries with non-uniform prior probability distribution on mi16481.train and mi16481.test

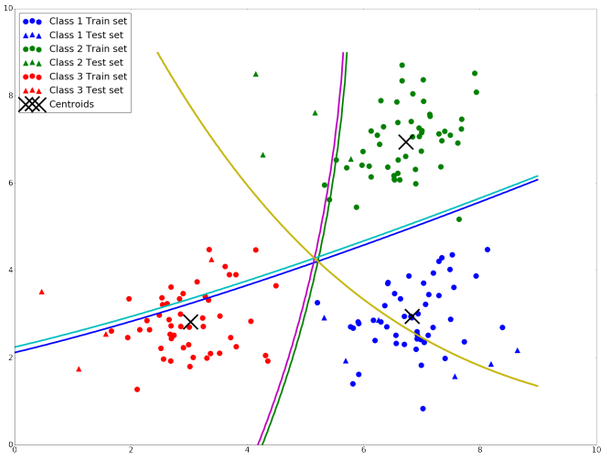

Figure 8b : Non-linear decision boundaries with non-uniform prior probability distribution on rk16699.train and rk16699.test

It can be seen that the decision boundaries generated by $\frac{\mathbb{P}(\boldsymbol{x}|\omega 1)\mathbb{P}(\omega_1)}{\mathbb{P}(\boldsymbol{x}|\omega 2)\mathbb{P}(\omega_2)}$ and $\frac{\mathbb{P}(\boldsymbol{x}|\omega 1)\mathbb{P}(\omega_1)}{\mathbb{P}(\boldsymbol{x}|\omega 3)\mathbb{P}(\omega_3)}$ shift from the green line to the purple line, and from the dark blue line to cyan line respectively. This shift is altogether away from the samples in class 1, whilst the decision boundary generated by $\frac{\mathbb{P}(\boldsymbol{x}|\omega 2)\mathbb{P}(\omega_2)}{\mathbb{P}(\boldsymbol{x}|\omega 3)\mathbb{P}(\omega_3)}$ stays the same, and is plotted on top of the previous decision boundary in yellow, since $\mathbb{P}(\omega_2) = \mathbb{P}(\omega_3) = 0.25$

VI. FURTHER DISCUSSION

The application of probabilistic and statistical models for prediction and classification is not limited to the algorithms mentioned above. k-Means clustering for example is a special case of the Gaussian mixture model, thus meaning that K-Means clustering can be further generalized. The 3 algorithms outlined in this report can also be modified further to adapt for certain conditions. Changes could be introduced in the distance metric which would enable the k-Nearest Neighbors algorithm to handle uncertain data.[8] The maximum-likelihood classifier can also be improved in the presence of strongly irregular spectral distributions through the use of a transition matrix which can be computed from contingency tables derived from previous classifications.[9] Table lookup methods are also used in the implementation of maximum likelihood classifiers on large images to improve their running time. [10]

1. References

[1] Weiran Wang and Miguel A. Carreira-Perpi´n˜an *The Role of Dimensionality Reduction in Classification.* Electrical Engineering and Computer Science, School of Engineering, University of California, Merced

[2] J. A. Hartigan and M. A. Wong *Algorithm AS 136: A K-Means Clustering Algorithm* Journal of the Royal Statistical Society. Series C (Applied Statistics), Vol. 28, No. 1 (1979), pp. 100-108

[3] Altman, N. S. (1992) *"An introduction to kernel and nearest-neighbor nonparametric regression". The American Statistician*

[4] Ma'rcia D. Branco, Dipak K. Dey *A General Class of Multivariate Skew-Elliptical Distributions* Journal of Multivariate Analysis 79, 99-113 (2001)

[5] Johanna Hardin, David M. Rocke *The Distribution of Robust Distances* University of California at Davis

[6] Michael Friendly and Georges Monette and John Fox *Elliptical Insights: Understanding Statistical Methods Through Elliptical Geometry* Psychology Department and Statistical Consulting Service York University

[7] Peter E. Hart, Richard O. Duda, David G. Stork *Pattern Classification, page 25, 20* Wiley-Blackwell; 2Rev Ed edition (21 Nov. 2000)

[8] Rashmi Agrawal, Babu Ra *A Modified K-Nearest Neighbor Algorithm to Handle Uncertain Data* IEEE

[9] Claudio Conese and Fabio MaseUi *Use of Error Matrices to Improve Area Estimates with Maximum Likelihood Classification Procedures* REMOTE SENS. ENVIRON. 40:113-124 (1992)

[10] Paul V. Bolstad* and T. M. Lillesand *Rapid Maximum Likelihood Classification* Environmental Remote Sensing Center,University of Wisconsin-Madison