Machine Vision

Coursework 1

Ting QIAO

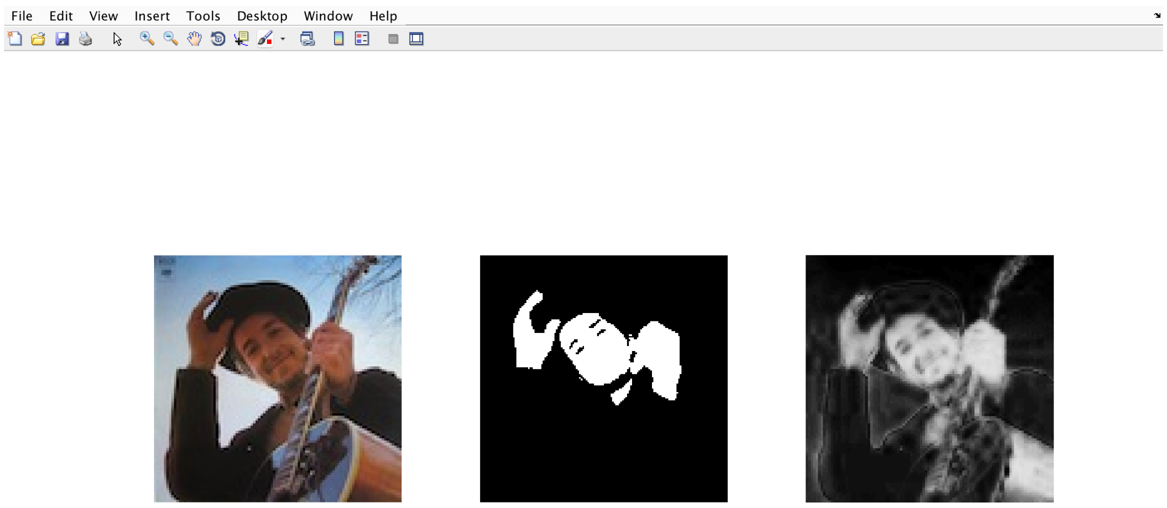
Nov 2016

Note: Some of the result images are in the folder of each part. The only the test images and test masks are in the folder called 'apples' and 'applesMask'.

Part I

Mixture of Gaussian Practical

Practical A:



Generally, the practical A is fitting the data with single normal distribution and then using the fitted normal distribution to classify the skin and non-skin pixel. The third image is the posterior probability calculated by using likelihood from the two normal distributions (skin and non-skin). The brighter pixel means the higher probability of being a skin pixel. Some part of the guitar was misclassified since the colour is similar to the people's face skin.

To Dos:

a. Calculating the data's mean value and the covariance of the data in different dimensions. Assume that the data is under a normal distribution, the mean value and covariance matrix is the parameters for the normal distribution. The disadvantage is that sometimes, the data may not be suited for the normal distribution. It can be more suitable for other distributions.

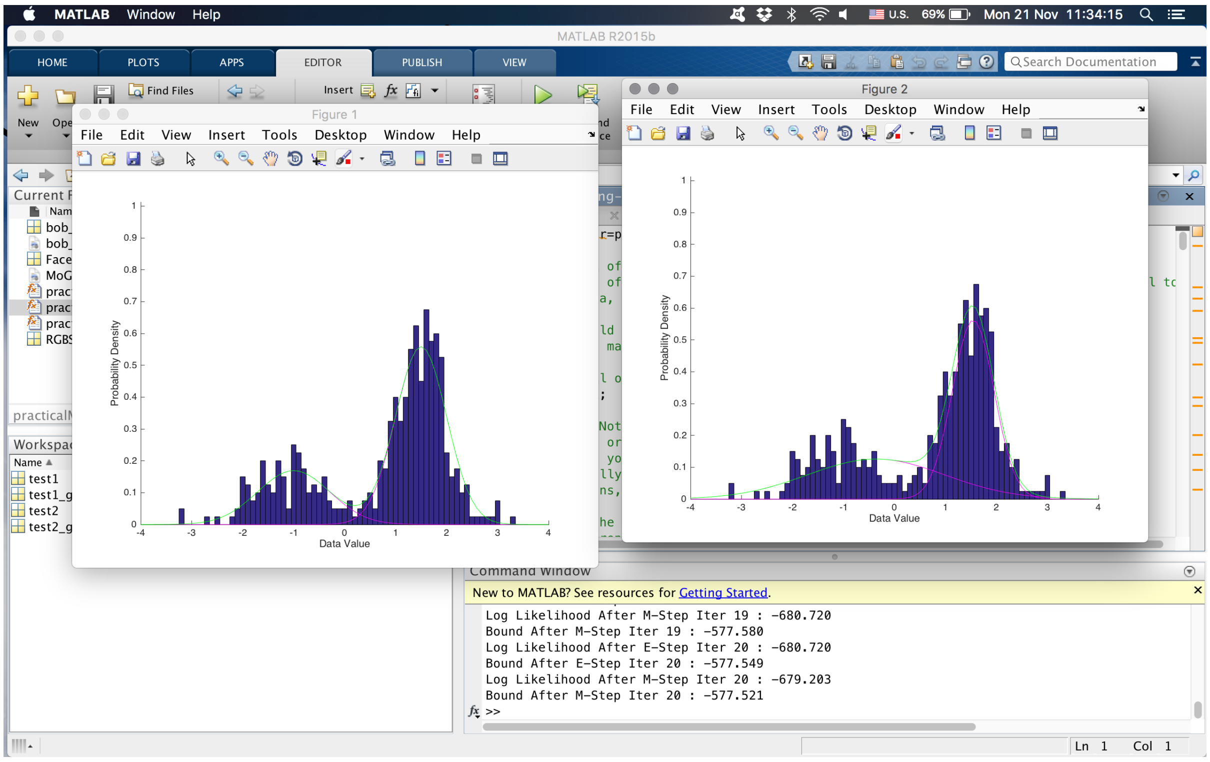
b. Given the parameters of a normal distribution model and data, we can work out the probability of the data is belonging to this normal distribution. The formula for multivariate normal distribution is shown below. The 'sigma' is the covariance matrix, the 'mu' is the mean vector. It is can also be called the likelihood (Pr(x|theta)).

$$y = f(x, \mu, \Sigma) = \frac{1}{\sqrt{|\Sigma|(2\pi)^d}} e^{-\frac{1}{2}(x-\mu)' \Sigma^{-1}(x-\mu)}$$

c. Based on the likelihood from both distribution(skin and non-skin), we can work out the posterior probability of the data is positive. The formula is shown below. In the formula, the Pr(x|w = 1) is the likelihood. The Pr(w = 1) is the prior pre-defined manually. The disadvantage here is that since we have 'infinite' data points, and we can not enumerate them, the prior can not be defined exactly.

$$Pr(w = 1|\mathbf{x}) = \frac{Pr(\mathbf{x}|w = 1)Pr(w = 1)}{\sum_{k=0}^{1} Pr(\mathbf{x}|w = k)Pr(w = k)}$$

Practical B:



```
Log Likelihood After M-Step Iter 19 : -680.720
Bound After M-Step Iter 19 : -577.580
Log Likelihood After E-Step Iter 20 : -680.720
Bound After E-Step Iter 20 : -577.549
Log Likelihood After M-Step Iter 20 : -679.203
Bound After M-Step Iter 20 : -577.521
>>
```

Instead of using single normal distribution to model the training data, the practical B is using the Mixture of Gaussian model to fit the data. Figure 1 is the normal distribution which was used to generate the training data. Figure 2 is the result of fitting the data by MoG. The result(Figure 2) is showing that the general shape of the fitted normal distribution is similar to the shape of the generative MoG, but not exactly the same. In Figure 2, the two normal distribution is affecting each other.

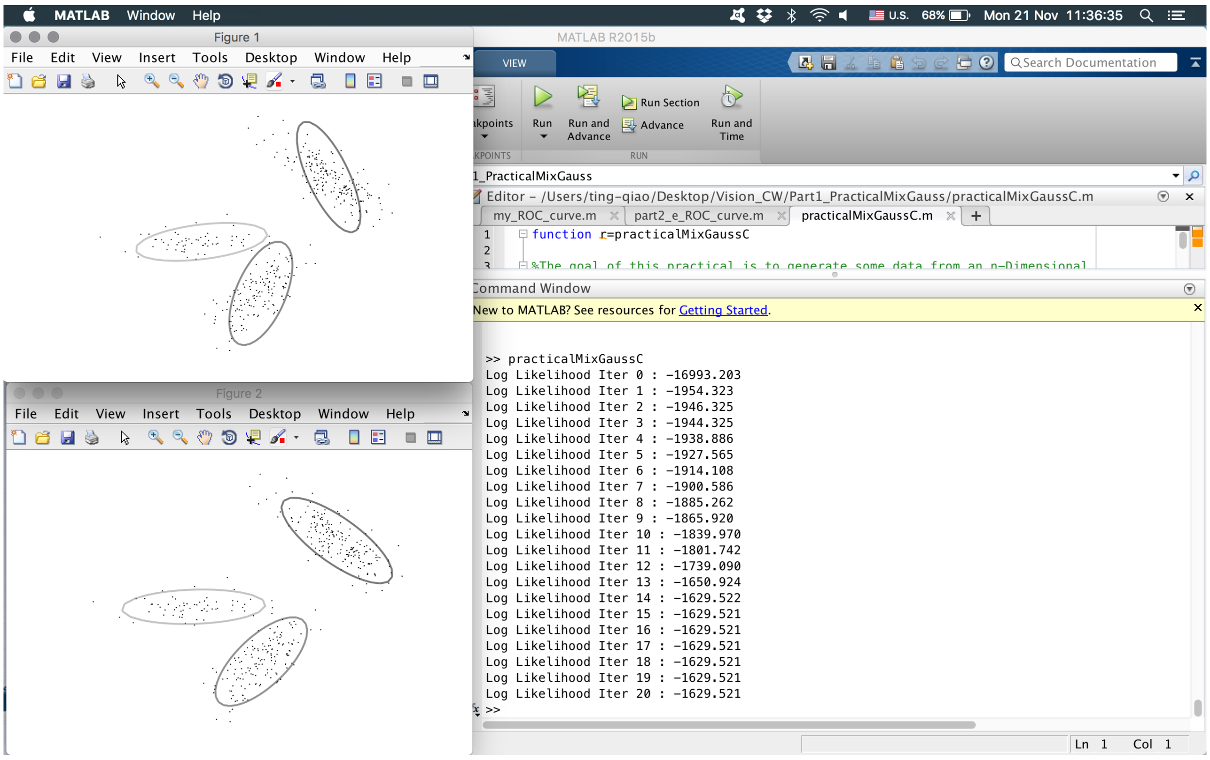
To do:

d. Given the parameters of a normal distribution, we can generate the data based on the distribution. Since the data is 'centred' at the mean value and have some 'difference' proportion to the standard deviation (covariance), the data is generated by the sum of mean value and standard deviation proportion to the random number (the random number is also fitting a normal distribution).

e. Since the data is modelled by the Mixture of Gaussian model, the likelihood is affected by different normal distributions. The likelihood is the weighted sum of the probability from different normal distribution. The formula is shown below. The 'K' is the hidden variable indicating that which Gaussian model it is belonging to. The 'lambda' is the weight for each Gaussian model.

$$Pr(\mathbf{x}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \lambda_k \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k]$$

Practical C:



```
function r=practicalMixGaussC

%The goal of this practical is to generate some data from an n-Dimensional
```

New to MATLAB? See resources for Getting Started.

```
>> practicalMixGaussC
Log Likelihood Iter 0 : -16993.203
Log Likelihood Iter 1 : -1954.323
Log Likelihood Iter 2 : -1946.325
Log Likelihood Iter 3 : -1944.325
Log Likelihood Iter 4 : -1938.886
Log Likelihood Iter 5 : -1927.565
Log Likelihood Iter 6 : -1914.108
Log Likelihood Iter 7 : -1900.586
Log Likelihood Iter 8 : -1885.262
Log Likelihood Iter 9 : -1865.920
Log Likelihood Iter 10 : -1839.970
Log Likelihood Iter 11 : -1801.742
Log Likelihood Iter 12 : -1739.090
Log Likelihood Iter 13 : -1650.924
Log Likelihood Iter 14 : -1629.522
Log Likelihood Iter 15 : -1629.521
Log Likelihood Iter 16 : -1629.521
Log Likelihood Iter 17 : -1629.521
Log Likelihood Iter 18 : -1629.521
Log Likelihood Iter 19 : -1629.521
Log Likelihood Iter 20 : -1629.521
>>
```

The practical C is very similar to the practical B. The difference is practical C is modelling the multi-dimension data(2 dimensions). So the implementation of the formula will be changed to matrix manipulation. Figure 2 is the fitting result. Compared with the generative Gaussian, the position and size of Gaussian distribution is similar, but the orientation is not. The reason could be the size of training data set is small. It is not enough to represent the distribution.

f. Similar to question 'd.', the 'f' is using the normal distribution model to generate data points. The difference is this question is generating the multi-dimension data. So the implementation is matrix manipulation and we are using the 'chol' function to get the 'square root' of the covariance matrix(equivalent to standard deviation in 1D sample).

g. The question 'g' is updating the responsibility for each data point. The formula was shown below. The responsibility respect to each data point to each hidden variable is the weighted probability of the data point belongs to each Gaussian distributions. It was also normalised by the weighted sum of probability belongs to each Gaussian model.

$$\begin{aligned} q_i(h_i) = Pr(h_i = k|\mathbf{x}_i, \boldsymbol{\theta}^{[t]}) &= \frac{Pr(\mathbf{x}_i|h_i = k, \boldsymbol{\theta}^{[t]})Pr(h_i = k, \boldsymbol{\theta}^{[t]})}{\sum_{j=1}^{K} Pr(\mathbf{x}_i|h_i = j, \boldsymbol{\theta}^{[t]})Pr(h_i = j, \boldsymbol{\theta}^{[t]})} \\ &= \frac{\lambda_k \text{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k]}{\sum_{j=1}^{K} \lambda_j \text{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j]} \\ &= r_{ik}. \end{aligned} \tag{7.17}$$

h.;i.;j; Those question is updating the parameters for each Gaussian model. The formula is shown below.

$$\begin{aligned} \lambda_k^{[t+1]} &= \frac{\sum_{i=1}^{I} r_{ik}}{\sum_{j=1}^{K}\sum_{i=1}^{I} r_{ij}} \\ \boldsymbol{\mu}_k^{[t+1]} &= \frac{\sum_{i=1}^{I} r_{ik}\mathbf{x}_i}{\sum_{i=1}^{I} r_{ik}} \\ \boldsymbol{\Sigma}_k^{[t+1]} &= \frac{\sum_{i=1}^{I} r_{ik}(\mathbf{x}_i - \boldsymbol{\mu}_k^{[t+1]})(\mathbf{x}_i - \boldsymbol{\mu}_k^{[t+1]})^T}{\sum_{i=1}^{I} r_{ik}}. \end{aligned} \tag{7.19}$$

Discussion:
Since the parameters was initialised randomly. The initial parameters have huge influence on the final result. However, the random initialisation can not be controlled.

Part I I

Apple

A. The image was 'tided up' to two arrays based on the Ground Truth image. The apple and non-apple data are 3D data.

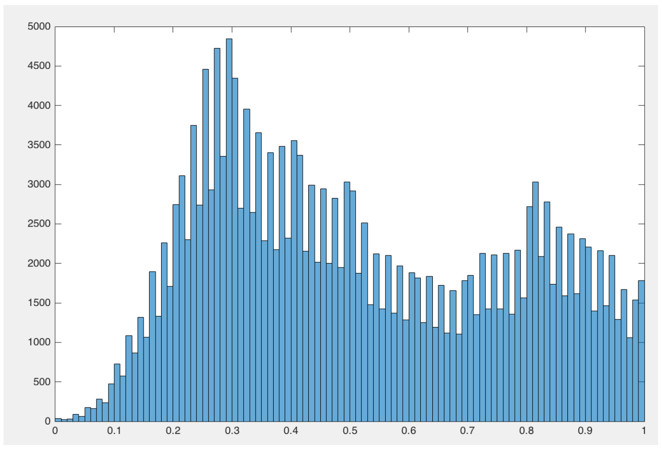
B. Decisions making:

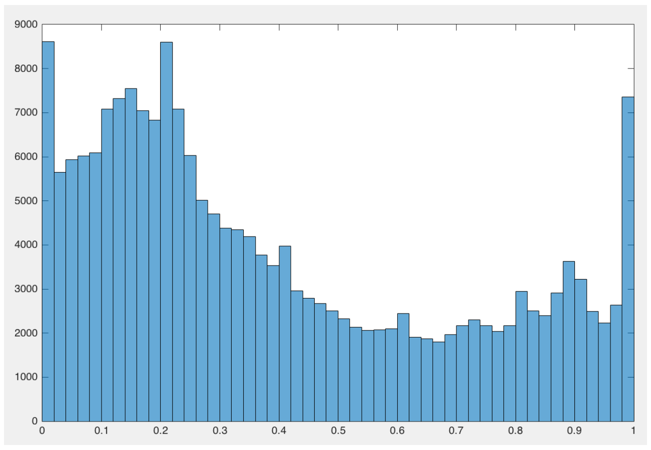
1. Saving the parameters as file.

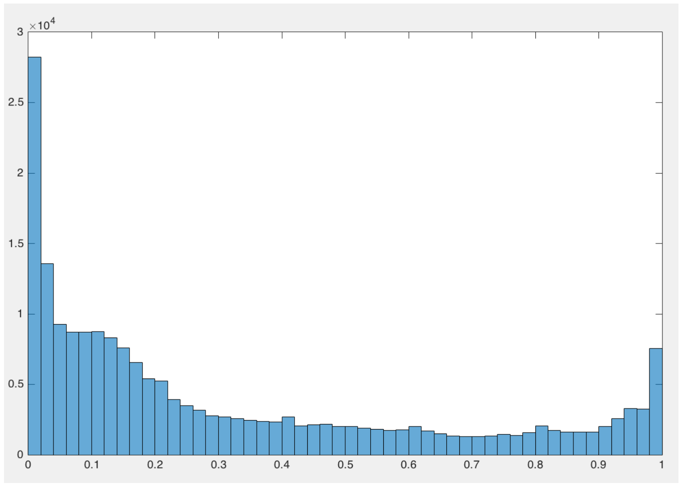
In order to reuse the trained parameters, I saved the parameters as file. The trained parameters was used in the following part.

2. The selection of k

The sample distributions of RGB 3 dimensions are shown below.



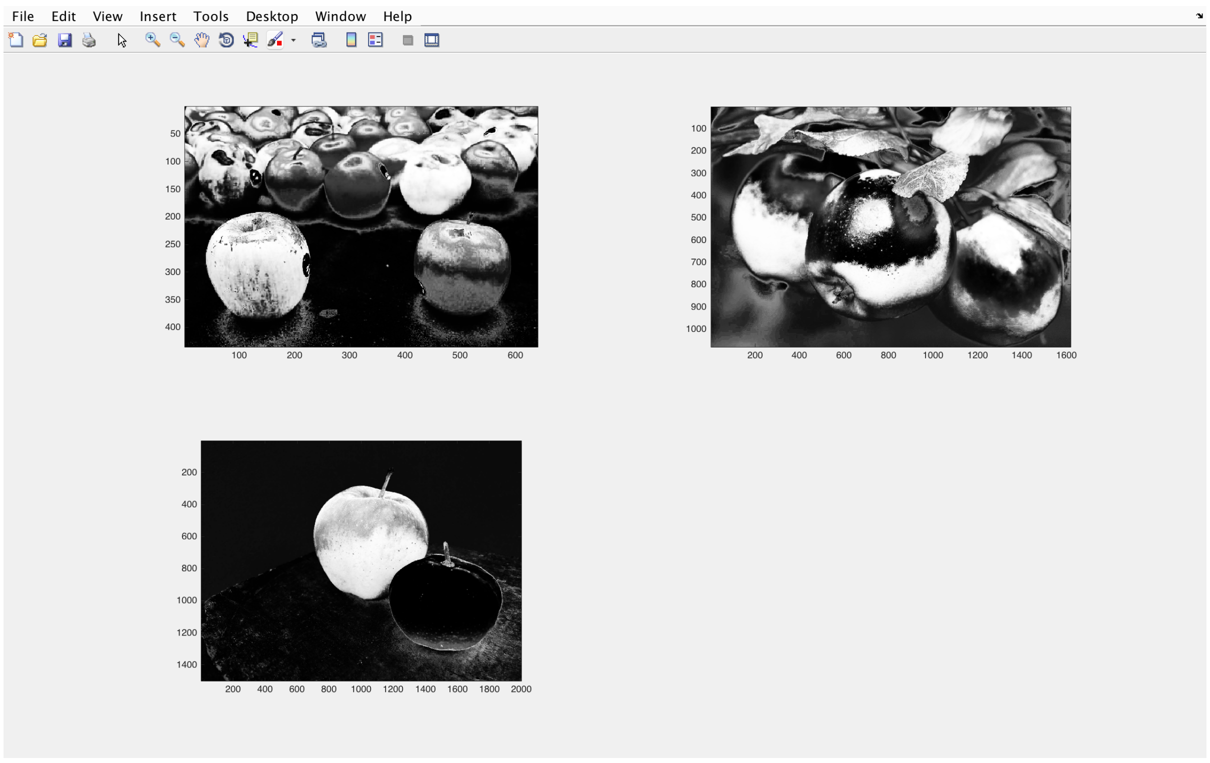




As the distribution suggested, normally the data has three peaks. So normally, the number of Gaussian model is three is a good choice.

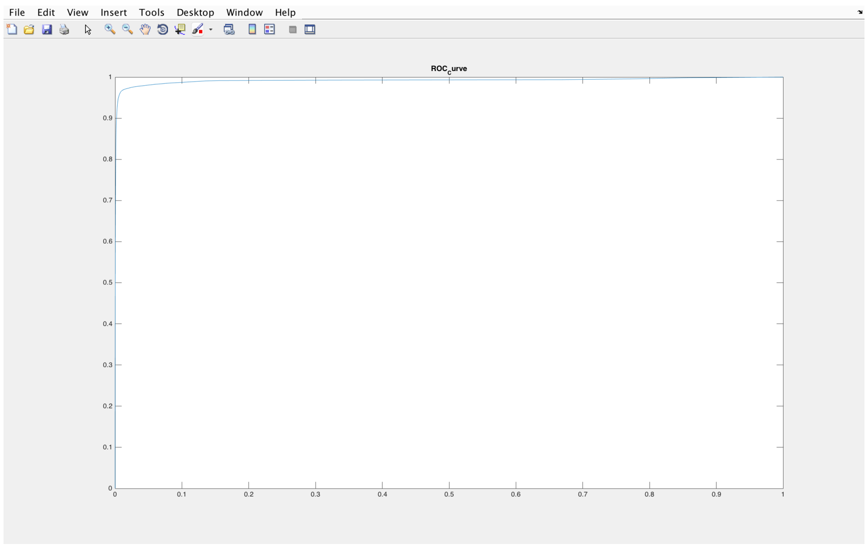
Using how many hidden variables to describe the data set is kind of the property of the data set itself. The more hidden variables can lead to more accurate fitting of the data. However, more hidden variables also means higher computational cost and the over-fitting problem.

C.



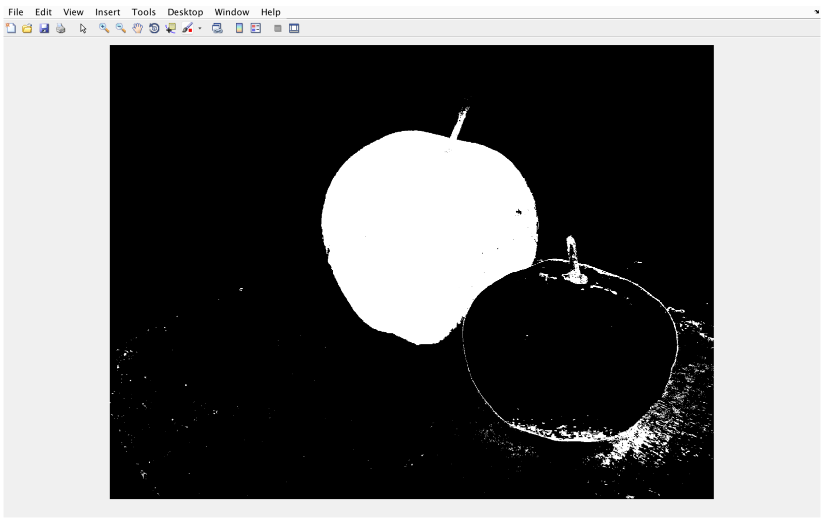
D. ROC curve. Precision, Recall, F1 measurement

In order to find the best threshold to segment the apple, we are using the ROC curve to find the best threshold. The best threshold have the smallest distance to the top left corner. That means the rate of TruePositive/FalsePositive are maximised. The ROC curve is:



The number of thresholds of the ROC curve in this case is 100. That means 100 thresholds between 0 to 1 was tried to do the thresholding. The reason why number of threshold is 100 is that the meaning of thresholds is probability and 100 is accurate enough to two decimals. Someone suggested that the thresholds should be 256, since the image grayscale is from 0 to 256. I don't think it is meaningful.

The best threshold is 37 * 1/100 = 0.37. The thresholding result is shown below.



After the best threshold was find, the precision rate, recall rate and f1 measurement should be calculated. In this case, the precision rate and recall rate are equally important. So the f1 measurement's parameter **Beta Square =** 1.

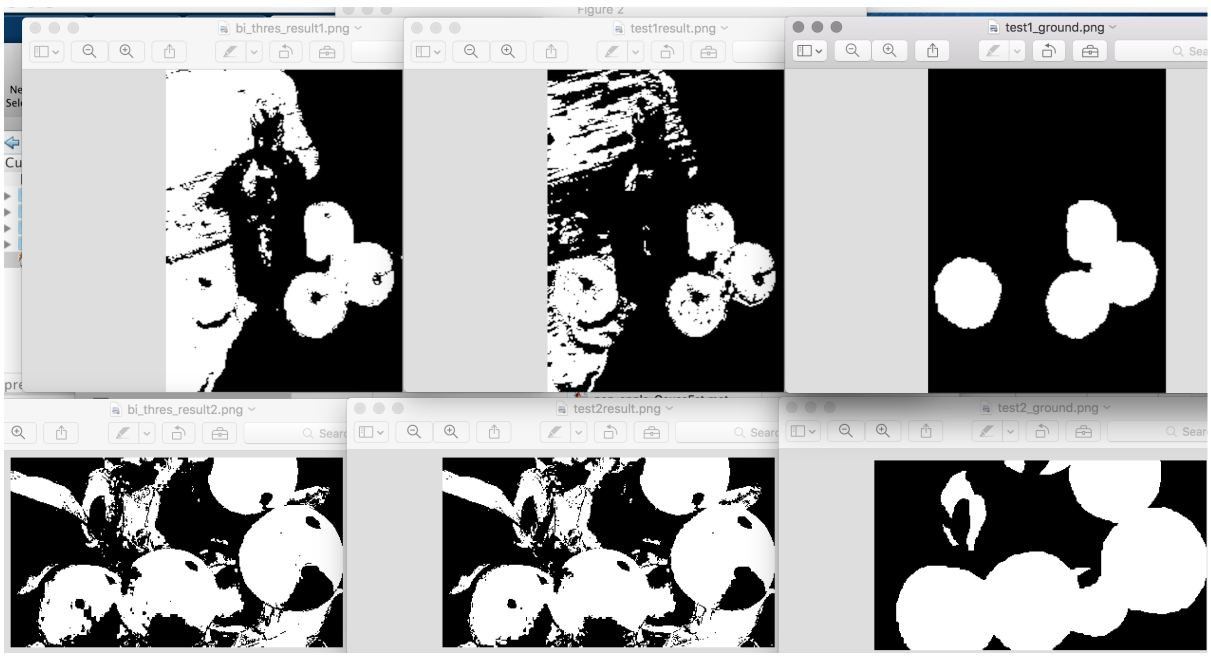
	Precision	Recall	F1 measurement
	0.9072	0.9706	0.9379

As the f1 measurement indicate, 93% is very high rate. It is a good model for classify the apple/non-apple pixel for this image.

E. I downloaded two images from the Google. I used the GIMP software to produce the ground truth manually. In this case, I used the ROC curve to find the best threshold for each of the test image. I also used the threshold from previous part(Part D) which is threshold = 0.37.

	Precision	Recall	F1 Measurement
Image test1: Threshold = 0.37	0.3427	0.9213	0.4995
Image test2: Threshold = 0.37	0.7528	0.7199	0.7359
Image test1: ROC Threshold = 0.69	0.4717	0.8033	0.5944
Image test2: ROC Threshold = 0.29	0.7397	0.7657	0.7525

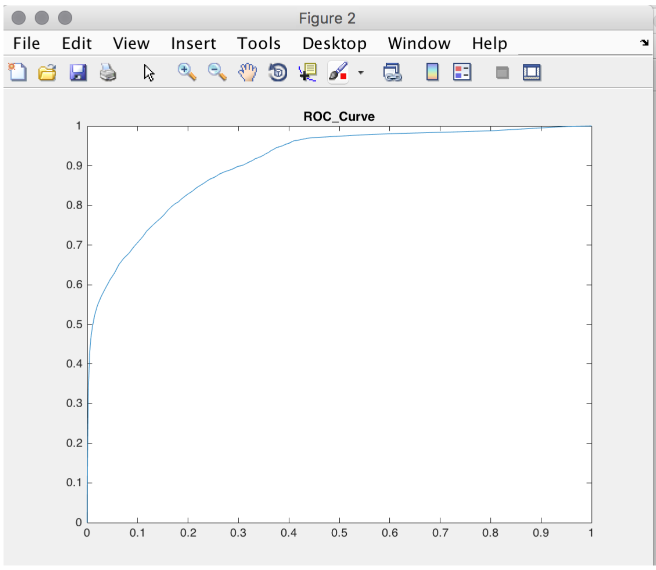
The result image was shown below:

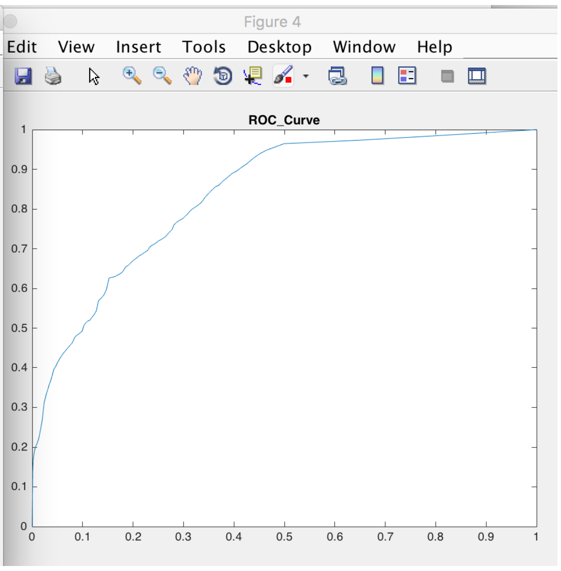


The upper row is the test1 image set. The lower row is the test2 image set. From left to right, the first column is the image thresholded by the threshold from previous section(threshold = 0.37). The second row is the image thresholded by the threshold from the ROC curve. The third column is the ground truth.

The result is poor, especially for the test image 1. The reason is that the test1 image is a painting picture. It has different style from the training set images. The second reason could be the ground truth is not good enough. The smooth edges is difficult to be classified.

The ROC curve for test1 and test2 is shown below.





F.

The training set here is used in the section A and B to find the parameters of the Gaussian model. The training set's folder's name is 'apples'.

The validation set here is used to find the threshold for general classification. In this case, the threshold is 0.37. The validation set here is the image called 'Bbr98ad4z0A-ctgXo3gdwu8-original.jpg'. This part was implemented at section D.

The test set here is the images found by me. The threshold = 0.37 was applied. The result was analysed in the previous section. Only the images thresholded by threshold = 0.37 can be called as the test set.

Generally, the training set is used for training the parameters of the model. The cost function measures the difference between predicted value and the target for each data point in the training set. The derivative of the cost function determines the magnitude to update the parameters.

The validation set is used to find the relations between different models. For instance, finding the weights of the output from different decision trees in the random forest algorithm should use the validation set. However, find the branches' parameters should use the training set.

The test set is only used once for the last test. The test result is used as the final result to report. The test set can never be mixed with another data set and can only be used once. Do not mix any data point from test set with other sets!!! It is a restricted