Machine Vision Assignment1

[redacted]

Part 1.

practicalMixGauss A.

Results:
As shown in the graphs below, the first one is the original image, the second one is the mask image and the third one is result of distinguishing skin pixels from non-skin pixels. For each pixel in the third graph, if the color of a pixel is whiter, brighter, it means that the probability of this pixel becoming a skin pixel is larger. By contrast, if the color of a pixel is darker, it is less likely for this pixel becoming a skin pixel.



When set the threshold equals 0.5, the result is shown below. Some pixels of guitar are mislabeled as skin pixels because its color is similar skin color.



Code:
a. Function fitGaussianModel returns the mean and covariance of data. Here add two equations to calculate the mean and covariance of data.

$$Cov(i,j) = Cov(j,i) = \frac{(X_i - \mu) * (X_j - \mu)'}{n}$$

$$(X_i, X_j \; are \; two \; varibles \; of \; data)$$

b. Given data and parameters: mean and covariance, function calcGaussianProb returns the probability (likelihood) of data. Here add a multivariate normal distribution function to return the likelihood of multivariate data.

c. Add three equations: first two is to calculate the likelihood for every pixel of becoming skin and non-skin pixel. Using the results of the first two equations and the prior of skin and non-skin, the third equation returns the posterior for every pixel of becoming skin pixel.

practicalMixGauss B.

Results:
The histograms in figure b-1 and figure b-2 shows the distributions of sample data. The green line in figure b-1 shows the values of mixture of Gaussians (probability) distribution of the original data. It is the true distribution of data. And the magenta line shows the distribution of every single Gaussian distribution.

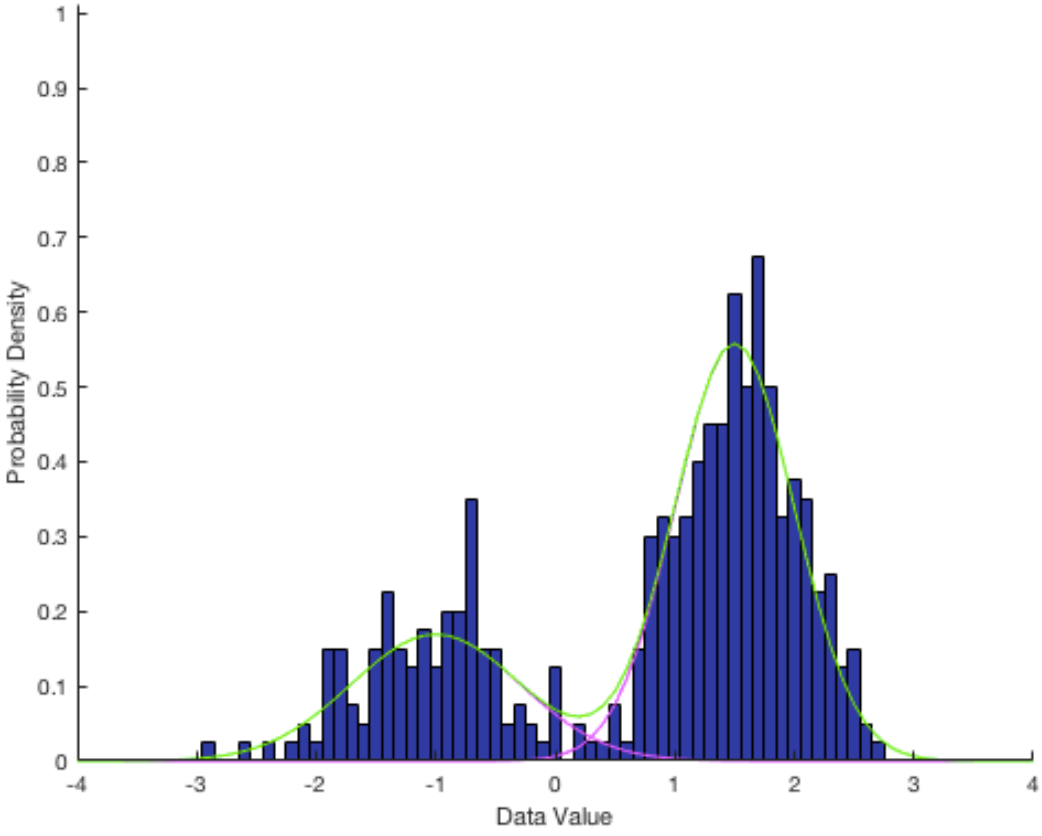

figure b-1

The green line and the magenta line in figure b-2 are the estimate distribution of original data from sample data. If the green line and the magenta line are more similar to the lines in figure b-1, that means the algorithm to estimate parameters of distribution is better. The lines in figure b-2 would change as the parameters update.

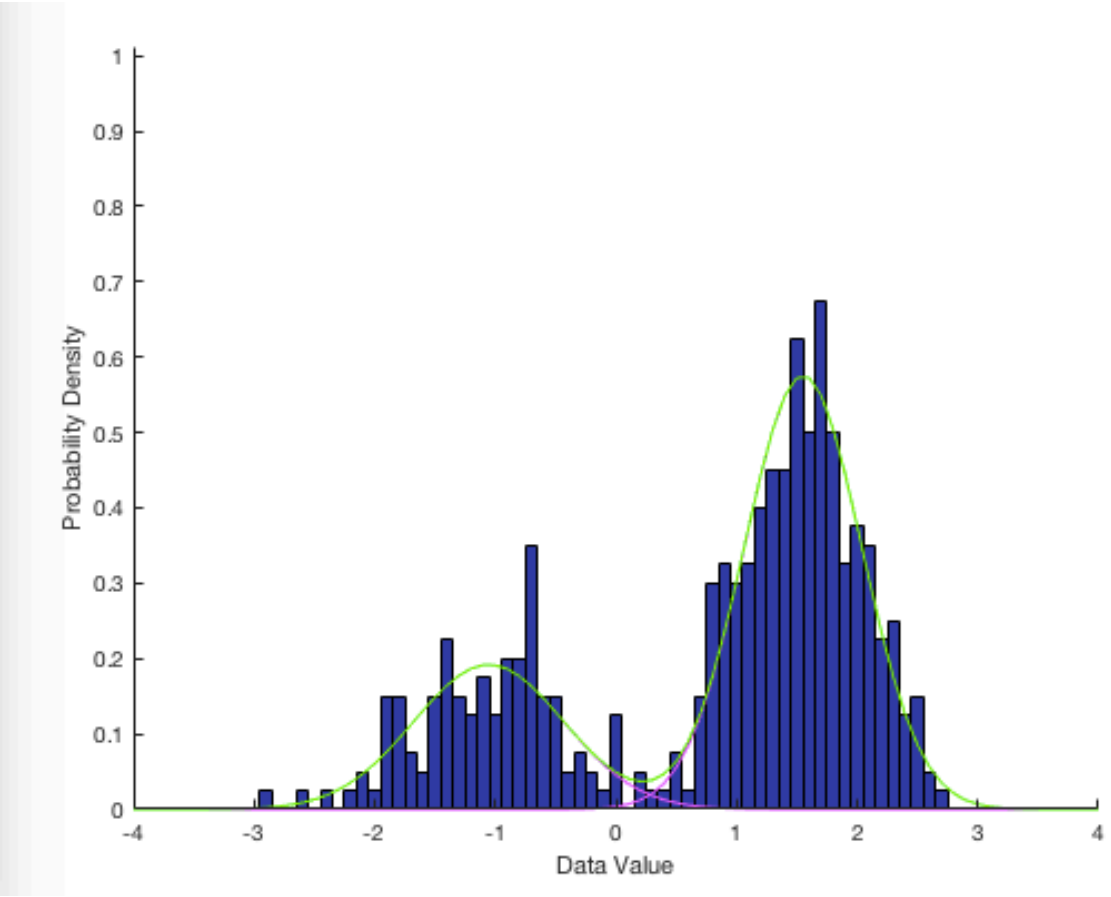

figure b-2

Code:
d. Function mixGaussGen1d generates data from a mixture of Gaussian distribution. Given mean and covariance of different Gaussian distribution, generating data from standard normal distribution using randn and then transforming them into different non-standard normal distribution.

e. Function getMixGaussLogLike returns the log(likelihood) of data given parameters.

$$Likelihood = \prod_{i}^{I}(\sum_{k}^{K} \Pr\,(x_i, h = k|\theta)$$

$$\log(Likelihood) = \sum_{i}^{I} log(\sum_{k}^{K} (\Pr\,(x_i, h = k|\theta)))$$

Firstly, calculate the likelihood of each data point under mixture of Gaussians model and then add up the log(likelihood) of all data point.

practicalMixGauss C.

Results:
The scatters in figure c-1 and figure c-2 shows the distributions of sample data. The circles in figure c-1 shows three Gaussians distributions of the original data. It is the true distributions of data.

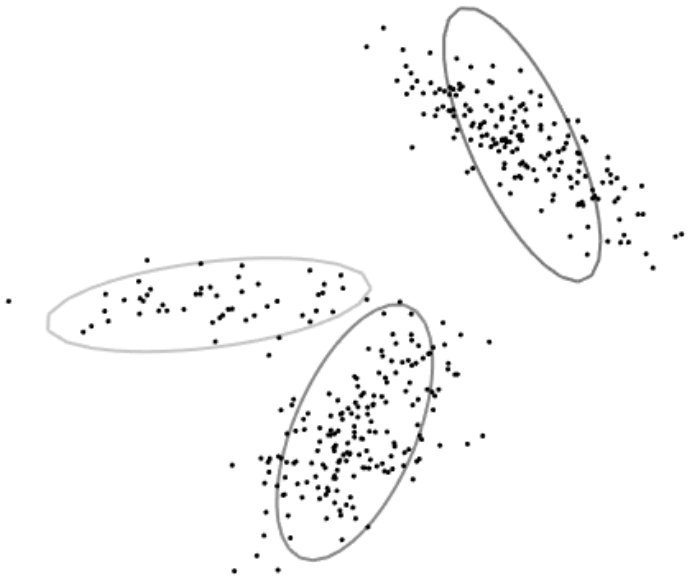

figure c-1

The circles in figure c-2 are the estimate distribution of original data from sample data. If the circles are more similar to the circles in figure c-1, that means the algorithm to estimate parameters of distribution is better. The circles in figure c-2 would change as the parameters update.

figure c-2

f. Function mixGaussGen generates data of a n-dimension mixture of Gaussian distribution. Given mean and covariance of different multivariate Gaussian distribution, generate data from random one multivariate Gaussian distribution and finally get a data set of a mixture of Gaussian distribution.

g. Firstly, calculate $P(h|x_i,\theta) = \sum_k^K P(h=k|x_i,\theta)$ for i-th data point, and then calculate $P(h=k|x_i,\theta)$ for the i-th data point for the k-th Gaussian. The responsibility for the k-th Gaussian i-th data point is $r_{ik} = P(h=k|x_i,\theta)/P(h|x_i,\theta)$, and finally get a k*ndata matrix.

h. By maximizing the bound, get an equation to update the weight of each Gaussian. Using this equation to update the weight of t+1and finally get a weight value for k-th Gaussian of t+1.

$$\mathcal{B}[\{q_i(h_i)\},\theta] = \sum_{i=1}^{I}\sum_{k=1}^{K} q_i(h_i)\log\left[\frac{\Pr\,(x,h_i|\theta)}{q_i(h_i)}\right]$$

$$\lambda_k^{[t+1]} = \frac{\sum_{i=1}^{I} r_{ik}}{\sum_{j=1}^{K}\sum_{i=1}^{I} r_{ik}}$$

i. By maximizing the bound, get an equation to update the mean of each Gaussian. Get a (nDim*k) vector for k-th Gaussian of time t+1.

$$\mu_k^{[t+1]} = \frac{\sum_{i=1}^{I} r_{ik}\, x_i}{\sum_{i=1}^{I} r_{ik}}$$

j. By maximum the bound, getting an equation to update the covariance of each Gaussian. Get a (nDim*nDim) matrix for k-th Gaussian of time t+1.

$$\Sigma_k^{[t+1]} = \frac{\sum_{i=1}^{I} r_{ik}\,(x_i-\mu_k^{[t+1]})(x_i-\mu_k^{[t+1]})^T}{\sum_{i=1}^{I} r_{ik}}$$

Part 2.

Train mixture of Gaussians models

1. Get the apple and non-apple data.
For every color image and the corresponding mask image in file apples, get the pixels in color image where labeled 1 in mask image as apple data and the pixels in color image where labeled 0 in mask image as non-apple data. Connect three apple data from three training images and get a 3*ndata matrix apple pixel training data set. Connect three non-apple data from three training images and get a 3*ndata matrix non-apple pixel training data set.

2. Our aim is to distinguish apple pixels from non-apple pixels, which equals to calculate the posterior probability of a pixel being apple pixel (w=1) given apple data.

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

To calculate the posterior probability of w=1, firstly model Pr(x|w). Since the data is continuous and multivariate, choose mixture of Gaussians models to model Pr(x|w=1) and Pr(x|w=0).

$$\Pr(x|\theta) = \sum_{k=1}^{K}\Pr\,(x,h=k|\theta)$$

Secondly, set the prior probability of apple and non-apple:

$$\Pr(w=1)=\frac{num(apple)}{num(apple)+num(non-apple)}$$

$$\Pr(w=0)=1-\Pr(w=1)$$

num(apple): the number of apple pixel
num(non-apple): the number of non-apple pixel

3. Initialize the values of parameters.

The number of Gaussian: K=3. Assume that the distribution of data from each image is a Gaussian distribution.

Dimension: nDim=3. Use red, green and blue as dimensions.

The weights of each Gaussian: 1/K

The mean of the mixture of Gaussian: mean=randn (nDim, K).

The covariance of the mixture of Gaussian:
cov(:, :, k)=(1.5+1*rand(1))*eye(nDim,nDim)
Use random numbers to initialize the covariance of mixture of Gaussian.

4. E-M algorithm

To estimate the parameters, use the E-M algorithm. Maximizing the likelihood $\Pr(x|\theta)=\sum_{k=1}^{K}\Pr(x,h=k|\theta)$ equals to maximize the bound.

$$\mathcal{B}[\{q_i(h_i)\},\theta]=\sum_{i=1}^{I}\sum_{k=1}^{K}q_i(h_i)\log\left[\frac{\Pr(x,h_i|\theta)}{q_i(h_i)}\right]\leq\sum_{i=1}^{I}\log\left[\sum_{k=1}^{K}\Pr(x,h=k|\theta)\right]$$

E-step: calculate $q_i(h_i)$

$$q_i(h_i)=\Pr(h_i=k|x_i,\theta^t)=\frac{\lambda_k Norm_{xi}[\mu_k,\Sigma_k]}{\sum_{j=1}^{K}\lambda_j Norm_{xi}[\mu_k,\Sigma_k]}$$

M-step: Update the parameters

$$\lambda_k^{[t+1]}=\frac{\sum_{i=1}^{I}r_{ik}}{\sum_{j=1}^{K}\sum_{i=1}^{I}r_{ik}}$$

$$\mu_k^{[t+1]}=\frac{\sum_{i=1}^{I}r_{ik}\,x_i}{\sum_{i=1}^{I}r_{ik}}$$

$$\Sigma_k^{[t+1]} = \frac{\sum_{i=1}^{I} r_{ik} \, (x_i - \mu_k^{[t+1]})(x_i - \mu_k^{[t+1]})^T}{\sum_{i=1}^{I} r_{ik}}$$

Repeat E-step and M-step until the likelihood become maximum which means likelihood no longer increases.

Here we get two mixture of Gaussians models: mixGaussApple and mixGaussnonApple.

Test the models

Results:
Use the images in file testApples to test models mixGaussApple and mixGaussnonApple. The results are shown below: the larger the posterior probability of being ‘apple’ is, the whiter and brighter a pixel is. In figure 2-1, most red apple pixels can be correctly distinguished while the posterior probability of yellow and green apple pixels are small. However, in figure 2-2, the green apple pixels are well distinguished while most red apple pixels with low posterior probability. Moreover, some leaf pixels are with high posterior probability. In figure 2-3, all apple pixels are correctly distinguished, but some orange pixels are still mislabeled. Generally, these two models can distinguish apple from non-apple pixels, but not precisely.

figure 2-1

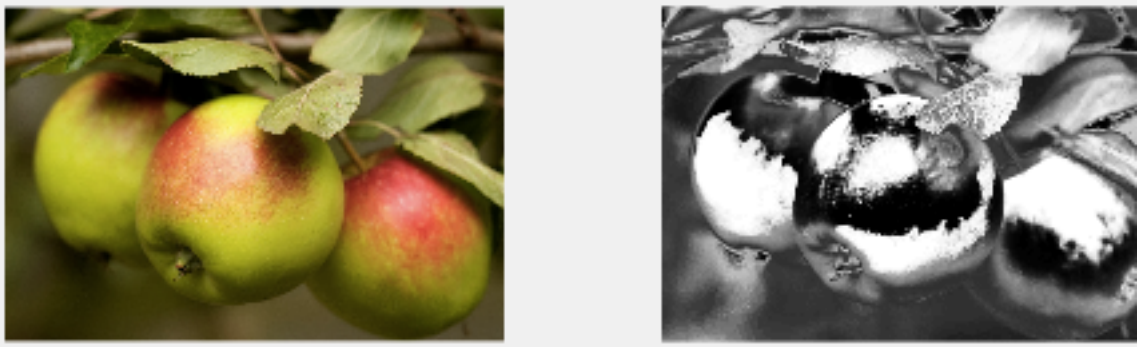

figure 2-2

figure 2-3

Set the threshold=0.5. When threshold=0.5, some apple pixels cannot be distinguished while some non-apple pixels are mislabeled as apple pixel.

Validate the model

Quantify the result by plot the ROC curve of figure 2-3.

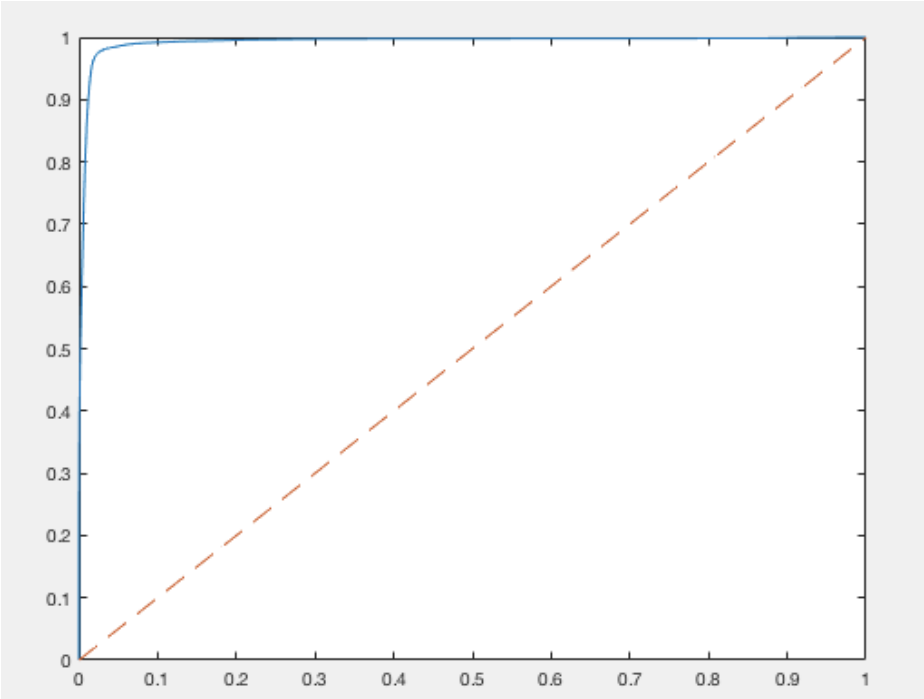

Figure 2-4

In figure 2-4, x axis shows the false positive rate FPR and y axis shows the false positive rate TPR.

$$FPR = \frac{FP}{FP + TN}$$

$$TPR = \frac{TP}{TP + FN}$$

TP: true positive; FP: false positive; TN: true negative; FN: false negative

Our aim is to get a classifier with high TPR and low FPR which means the curve is close to the upper left corner. Area Under Curve (AUC) is larger and more close to 1, the classifier is more effective. For figure 2-4, the AUC value is near 0.99, which shows that compare to classify pixels randomly, this model can classify apple and non-apple pixels effectively.

To choose a reasonable threshold, we need to calculate the accuracy. The larger the accuracy is, the more pixels are correctly classified. Here the threshold which maximize the accuracy is near 0.7374. Hence, we set the threshold=0.7.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

In addition, to get the best K value (number of Gaussian), we can use different K value to train models and calculate the AUC of these models, and finally choose k which maximize the AUC. Here, our optimal K is K=3 for mixGaussApple and K=5 for mixGaussnonApple.

Test the models 2

Use other images to test the models, results are as shown below.

figure 2-5

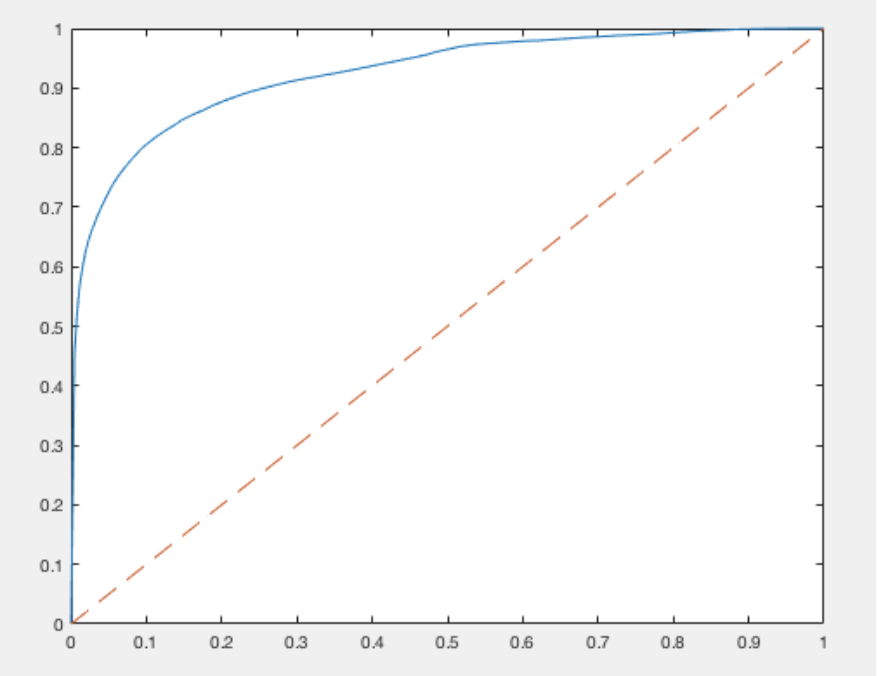

figure 2-6

As shown in figure 2-5, most red apple pixels are correctly classified. Although the curve is not as close to the upper left corner as figure 2-4, the AUC is 0.9274 > 0.5 shows that this classifier is effective.

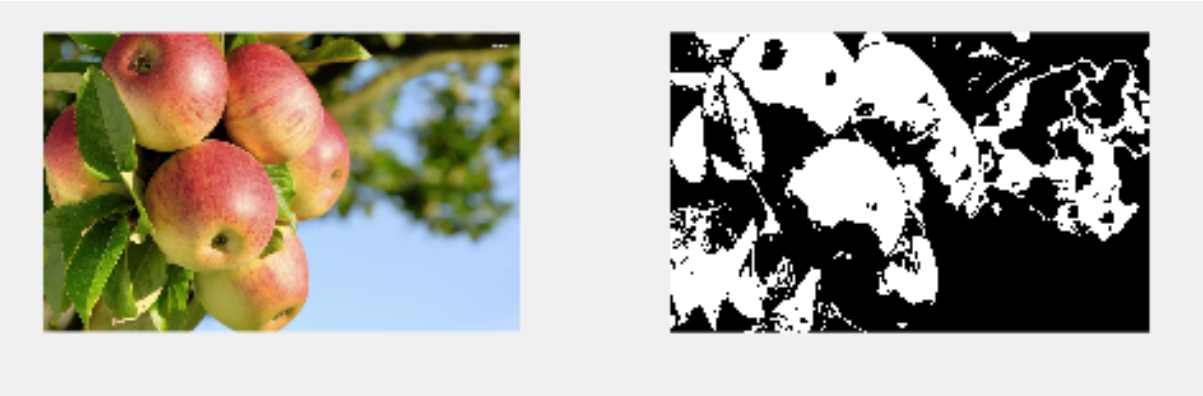

figure 2-7

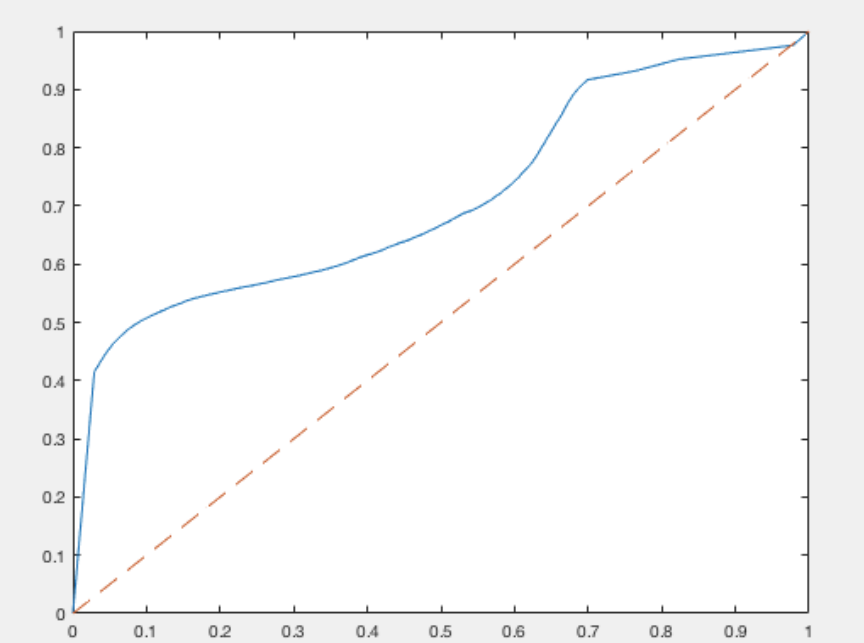

figure 2-7

As shown in figure 2-6, some red apple pixels are correctly classified. However, some leaf pixels are mislabeled. The ROC curve is also far away from the upper left corner the AUC is 0.7226. Although the AUC is still larger than 0.5 which means this classifier is better that an uninformative classifier. However, this classifier is not effective for this picture and needed to improve.

Training set, validation set and test set

A training set is for learning: estimating parameters and getting a model. A validation set is for tuning the model's hyperparameters. And the test set is used to evaluate the model and show the effectiveness.

In part 2, we use three images as a training set to get a classifier model. By training these data to fit our model, we get a classifier and some parameters of this classifier, like mean, covariance. However, some hyperparameters, like the K, threshold, are not estimated during the learning process. Therefore, we use one image (figure 2-3) as a validation set to validate the model, and to choose the optimal values of these hyperparameters. Finally we use four images (figure 2-1,2-2,2-5,2-7) as a test set to test the performance of this classifier.