# Report of Building Classifiers for Pattern Analysis

## 1.Abstract

This report will present a study of classifiers in machine leraning area. The process of builing supervised and unsupervised classifiers will be introduced followed by a discussion on optimising a classifier. The result of the study will be presented at the end of the report.

## 2.Introduction

The classification is a process describing a machine which is able to identify datums from a given set.

This report will present the study of classification on classifiying 6 different classes of binary images. The techniques behind this idea is that for a class of data they will share some common features, the machine will analysis theses features and trying to match these features with the unidentified datum, once the datum is matched then is classified. This is the process of building and testing a classifier. The general way to build the classifer is to find the expression:

$$p(D|\Theta) = \prod_{i=1}^{N} p(x_i|\Theta). \quad (1)$$

This expression takes parameter Θ and measures the overall probability density of data $x_i$. Based on Maximum Likelihood(ML) method, work out the maximum value of p(D|Θ) will show the most likely data X that is generated from the given parameter Θ. In this report, I adpot Gaussian Mixture Model to fit the above expression:

$$p(x|\Theta) = \sum_{i=1}^{N} p(x|\theta_i)p(\theta_i). \quad (2)$$

The GMM will combine all the Gaussians and shows the overall density distribution of random variable x and is formed by weighting individual Multivariate Gaussian distribution p(x|θ):

$$p(x|\theta_i) \triangleq \frac{1}{(2\pi)^{K/2}|C_i|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_i)^T C_i^{-1}(x-\mu_i)\right) \quad (3)$$

And prior: $p(\theta_i) = N_i / N$.

The Multivariate Gaussian distribution p(x|θ) describes the likelihood of data x is generated by a given class θ. In statistic view, if a set of data points follows Gaussian Distribution, then its mean **μ** determines its location, and its covariance **C** shows how these points spreaded in space. So assume each class has only one Gaussian distribution, then there will be 6 different likelihoods corresponding to different classes. So, perform ML method will determine which class θ is more likely to represent data x. This is how classifiers is developed in this report.

The testing stage involves classifying a shape. The posterior p(θ|x) tells how likely the given data x belongs to class θ. By Maximum a posteriori (MAP) estimation, the maximum value of p(θ|x) gives the most estimation of x can be classified to class θ. By using Bay's law, the MAP estimation for p(θ|x) can be expressed as:

$$\arg\max_{\theta} p(\theta|\mathbf{x}) = \arg\max_{\theta} \frac{p(\mathbf{x}|\theta)p(\theta)}{p(\mathbf{x})} \quad (4)$$

This expression enables the machine finding the largest p(θ|x) of a datum x, and this class θ is what the machine will be referred as the output of classification.

At the end of testing stage, a confusion matrix will be presented to describe the accuracy of the classifier. The confusion matrix is a N × N size matrix, it is filled by the classification result. The row represents classes of data and columns represent classes of classifier under scrunity.

The above process introduces one way to train the machine which is so-called supervised learning. There is another way named unsupervised learning which trains the machine without providing any class type Θ. Both two methods involve GMM to build the classifier, however, the unsupervised method use EM algorithm to model the estimated parameters for GMM instead of calculating mean and covariance directly.

The following section will introduce both types of classifiers by applying the above theoris into practice and the implementation of EM algorithm.

## 3.1. Supervised Learning

The supervised learning is a human-intervention process, the machine will trained with labelled data. The data should represent the distinguishable feature of images which is named feature vector in the context. To transfer this idea into practice, I use the chain code to accomplish this. The chain code will trace over the boundary of image shape, and every step, it will record the dirction of its movement as a serises coordinates, then convert these coordinates into angles. These continuous series of angles can form a wave curve but jagged, these noise is usually in high-frequency so we can model the curve by using Fourier Transform. We keep N frequency and remove the rest by multiplying “top-hat” filter.

```
c = chainCode(im);
angles = c(3,:)*(2*pi/8);
anglesFFT = fft(angles); %fast fourier transform
filter = zeros(size(angles));
filter(1:N) = 1;
filter(end-N+2:end) = 1;
filteredFFT = anglesFFT .* filter; % Apply the filter by
featureVector = abs(filteredFFT);  % scalar multipliacation.
featureVector = featureVector(1,2:N); % discard zero frequency.
features      = vertcat(features,featureVector);
```

These steps will provide a feature vector with N dimension.The chain code translates the images into numbers and these valuable numbers socalled feature vectors X describe the shape of image, this is how machine read images.

By iterating through the image folders, we can obtain all feature vectors and each feature vertor  can treated as a point in speace. Ideally,  the distribution for the same class points should tend to converge at a point. Such distribution forms the area,that is to say, any point under this area is more likely belong to this class. So, this enables the machine to determine the class of an unidentified point by working out which sector that the unlabelled point belongs to. The 2D visualisation for such cluster(Red: Star):

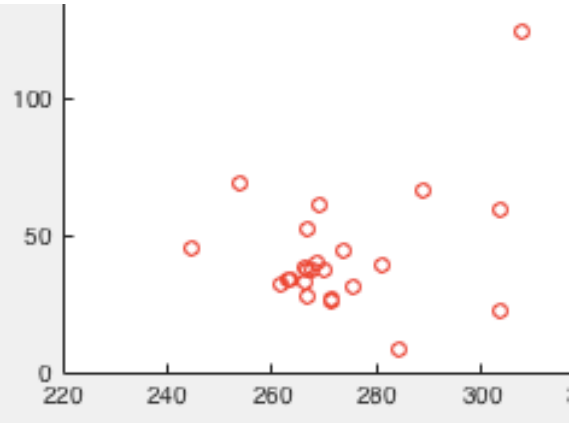


*Figure 1*

However, this method is not capable to fit all conditons. In practice, the distribution for a class points could be spreaded out and there will be intersection areas between different clusters. Thee 2D visualisation for these clusters(Red:Star, Magenta: Alien):

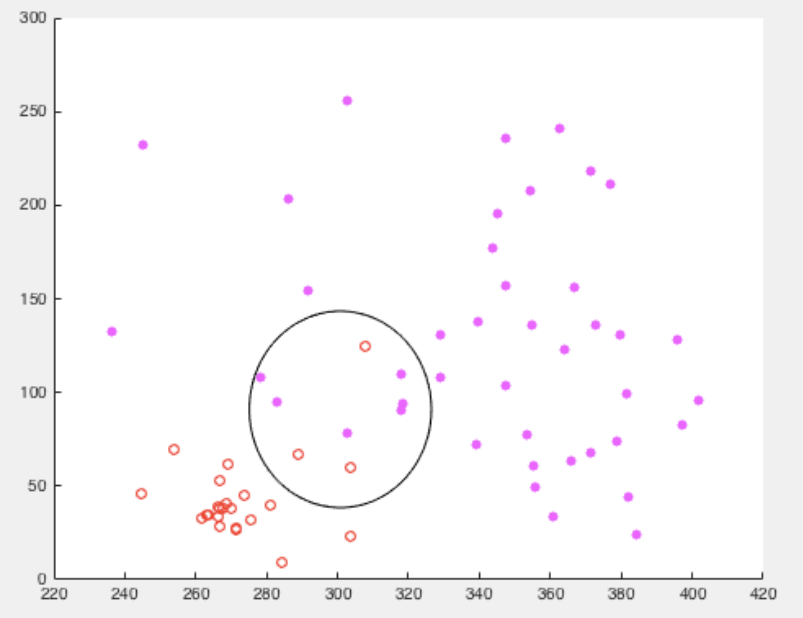


*Figure 2*

Such condition will provide an vague boundary between different classes(circled area), in this case, there is no certainty to show the class type for that point, instead, it shows the likelihood.

Recall from the previous section, we can use GMM to express the likelihood. So, we need to work out the parameters **μ** and **C** of each classes and fit the GMM(2):

```
for idx = 1:length(classes)
    class = classes{idx};
    models(idx).name = class;
    dataMatrix = getDataMatrix(imagedir, class, N); %Get feature vectors
    models(idx).mean = transpose(calcMean(dataMatrix));
    models(idx).cov = ensurePSD(calcCov(dataMatrix));%Non-singular Cov
    models(idx).prior = getNumImagesForClass (imagedir, class)/totalImages;
end
  save('models');
```

The overal density of x p(x|Θ) in GMM(2) is not used in this section. Because the MLE method wil only foucus on likelihood p(x|θ).

By the idea of Maximum Likelihood Eestimation (MLE), the maximum among these values is more likely to generate the feature vector x. The maximum likelihood can be obtained by the following expression:

$$\arg\max_{\mu_i, \mathbf{C}_i} \sum_{n=1}^{N} p(\mathbf{x}|\theta_i) \quad (5)$$

Which is to calculate:

$$\mu_i = E[\mathbf{x}] = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x_n}$$

$$\mathbf{C_i} = E\left[(\mathbf{x}-\mu_i)(\mathbf{x}-\mu_i)^\top\right] = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}-\mu_i)(\mathbf{x}-\mu_i)^\top \quad (6)$$

Implementation in matlab code:

```
function meanVec = calcMean(dataMatrix)
    vectorNum    = length(dataMatrix);
    meanVec = sum(dataMatrix)/vectorNum;
end
```

**$\mu_i$** :

**$C_i$** :

```
function covMat = calcCov(dataMatrix)
    dataNum                 = size(dataMatrix,1);
    oneOneM                 = ones(dataNum);
    myDeviation             = dataMatrix - (oneOneM * dataMatrix / dataNum);
    covMat                  = transpose(myDeviation) * myDeviation / dataNum;
end
```

The covariance matrix **C** is obatined by refering:

$$C = x'x\ (\ 1 / n\ ) \quad (7)$$

Where

$$x = X - 11'X\ (\ 1 / n\ )$$

11’ is an n*n identity matrix, X is the data matrix and n is the number of the data element. Thus, we complete our supervised training step.

## 3.2.Unsupervised Learning

The practical difference between the unsupervised and supervised learning is that the way to fit GMM model is different. In terms of unsupervised learning, we obtained the clusters without any label:

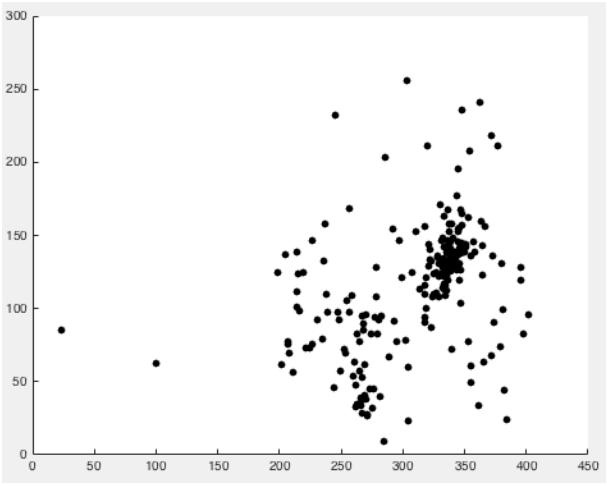

*Figure 3*

Thus, we can not fit GMM with individual Gaussians. Recall Equation (2), N represent there are N components have been mixed in GMM, so in this case N is 6. This number 6 tells the machine that is are in total 6 different clusters, and this is what KM algorithms based on. So, for a component $K_i$ its corresponding p(X) from GMM shows the likelihood of $x_i$ is generated from componet $K_i$ over all clusters. This means, all we need to do is to find the largest p(X). By ML method on GMM, we can obtain the largest p(X) as follow expression:

$$\sum_{i=1}^{N} \log \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k) \right\} \quad (8)$$

Where K is equivalent to N from Equation(2).

This process will label each point from clusters, so the rest of work is similar to the supervised classifier.

The Expectation-Maximisation (EM) algorithm is used to find p(X) according to Equation(8). The EM algorithm consists of two steps: Expectation and Maximisation. The Expectation step fit Equation(3) with different **μ** and **C** iteratively in order to find the likelihood p(x|θ) of the best fit:

$$\gamma(i,k) = \frac{\pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_i|\mu_j, \Sigma_j)} \quad (9)$$

Equation(9) is a rewite form of Equation(3), where γ(i,k) is equivalent to p(x|θ) and Σ is **C**. Implementation in matlab:

```
Px = calcPosterior(dataM,components,pMiu,pSigma);
prodM = size(Px);
%pGa = coe * N(x|pMu(k),pSig(k))
for i = 1:N
    for j = 1:K
        prodM(i,j) = pPi(j) * Px(i,j);%numerator
    end
end
pGa = size(dataM);%Likelihood function
pDenominator = sum(prodM, 2);%Denominator of likelihood F

for i = 1:N
    for j = 1:K
        pGa(i,j) = prodM(i,j) / pDenominator(i);
    end
end
```

The above step will generate N points vary from γ(1,K) to γ (N,K), and Maximisation will ganrantee we obtain the largest likelihood γ (x,K) among them followed from equation (6). Iterating through the above 2 steps until the likelihood function is convergence. Then the clusters has been labelled by 6 different classes, and GMM is fitted.

## 4.Classify Images

The idea of MAP estimation is to get the largest posterior values p(θ|x). The equation (4) shows how to achieve MAP method, since p(x) in our case is a constant, it is not depend on θ, thus this term can be removed:

$$= \arg\max_{\theta} p(\mathbf{x}|\theta)p(\theta) \tag{10}$$

The result for MAP is usually small, so the computer will round this value to 0. This precision problem can be solved by using logarithm:

$$\begin{aligned} \arg\max_{\theta} p(\theta|\mathbf{x}) &= \arg\max_{\theta} \log\left(p(\mathbf{x}|\theta)p(\theta)\right) \\ &= \arg\max_{\theta} \left[\log\left(p(\mathbf{x}|\theta)\right) + \log\left(p(\theta)\right)\right] \end{aligned} \tag{11}$$

By substituting equation (3):

$$\arg\max_{\theta} p(\theta|\mathbf{x}) = \arg\max_{\theta} \left[\log(p(\theta)) - 0.5\log(|\mathbf{C}_\theta|) - 0.5(\mathbf{x}-\mu_\theta)^T {\mathbf{C}_\theta}^{-1}(\mathbf{x}-\mu_\theta)\right] \tag{12}$$

The implementation of MAP in matlab:

```
%Find out which class has the highest score
for idx = 1:length(models)
  model = models(idx);
  modelMean = model.mean;
  transMean = transpose(modelMean);
  modelCovariance = model.cov;
  modelPrior = model.prior;
  score = log(modelPrior) - 0.5*log(det(modelCovariance)) ...
      - 0.5*(features -transMean)*inv(modelCovariance)...
      *(features - transMean)';
  if score > maxscore
    maxscore = score;
    bestidx = idx;
  end
end
classname = classes(bestidx);
```

From this code, the score represents the previous MAP equation(12), and the unidentified data X will be labelled the class with the highest score.

## 5.Optimising Supervised Classifier

The dimensions of feature vectors deponds on how we choose to fliter the chain code by assigning different N to keep the highst N frequencies. So, the N can not be too high or too low, because such N would be considered as lossy. The performance of Classifier can be seen on Result section, it shows that the performance for classifer is peaked at N = {5,6,7} with accuracy 81%, and the decresed tendency can be observed as changing the value of N towards both ends.

From equation (12) about MAP, it shows that the prior p(θ) do have influen on the finding the maximum posterior. The term prior $p(\theta_i) = N_i / N$ describes the proportion of a class θ over all the classes. So the prior is a probability that shows the chance of seeing a class θ. The result for decreasing the value of prior is unpredictable. From Figure 6, the accuracy is decreasing by lower the prior for Face. From Figure 7 the curve can not give significant information to describe the influence of prior for accuracy. The Figure 3 4 5 (Magenta:Alien Red: Face) depict such phenomenon, the way I lower the prior is simply remove the relevant class images from training set, lower the number of faces will dilute the density of centriod, thus a Face image in test set is more likely to be classified to Alien. On the other hand, lower the number of Alien will possibliy remove the centroid points but it could also remove the others. This is the reason why the accuracy is fluctuated.

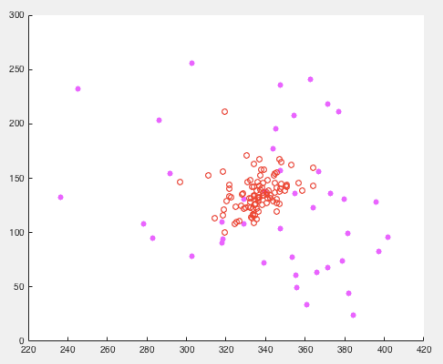

*Figure 4*

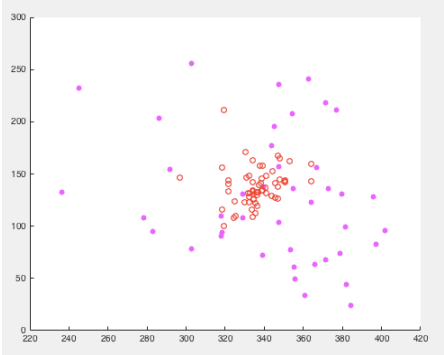

*Figure 5*

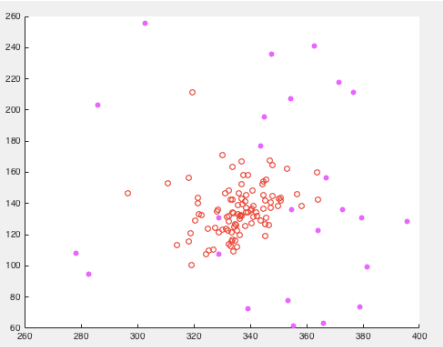

*Figure 6*

## 6.Result

The performance for both classifiers are represented by confusion matrix. The diagonal of confusion matrix shows the number of corrections. So the accuracy can be calculated by suming diagonal elements and divided by total images number.

```
Confusion Matrix =
                  Alien        Arrow    Butterfly         Face         Star     Toonhead
        Alien        37            0            5            0            0            0
        Arrow         0            0            2            0            0            0
    Butterfly         2            0           48            0            0            0
         Face        25            0            1           73            0            0
         Star         2            0            3            0           21            0
     Toonhead         2            0            0            0            0            0

accuracy =

    0.8100
```

*Figure 7 Confusion Matrix for supervised classifier at N =7*

```
Confusion Matrix =
                  Alien        Arrow    Butterfly         Face         Star     Toonhead
        Alien         6            9           18            0            9            0
        Arrow         1            0            0            0            2            0
    Butterfly        21            0           27            0            0            1
         Face         0            4            7           72            3           14
         Star         0            0            6           11            7            0
     Toonhead         0            1            0            0            1            1


accuracy =

    0.5113
```

*Figure 8 Confusion Matrix for unsupervised classifier*

Figures 9 shows the performance of supervised classifier with different N.

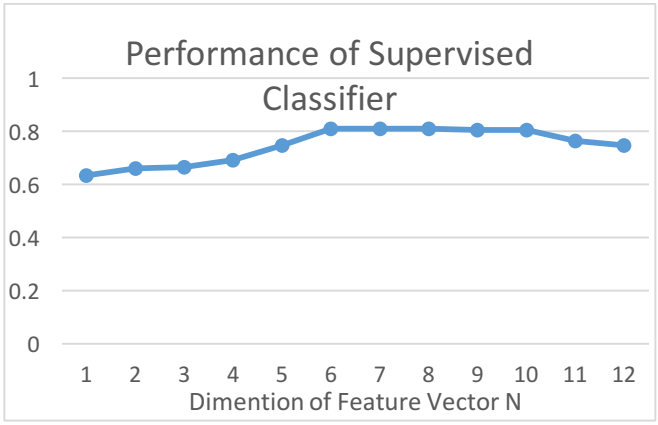


*Figure 9*

Figuer 10 shows the performance of unsupervised classifier over 10 times. As the EM algorithm based on predicting  the paramenters for GMM. Thus, EM can not give a certain value of paramenters, so the performance of unclassifier is not guaranteed.

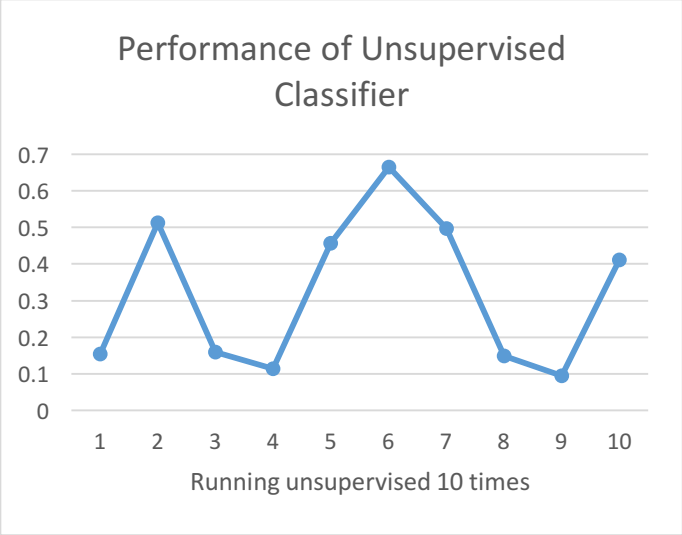


*Figure 10*

Figure 11 and 12 show the performance of supervised classifier by decreasing prior values.

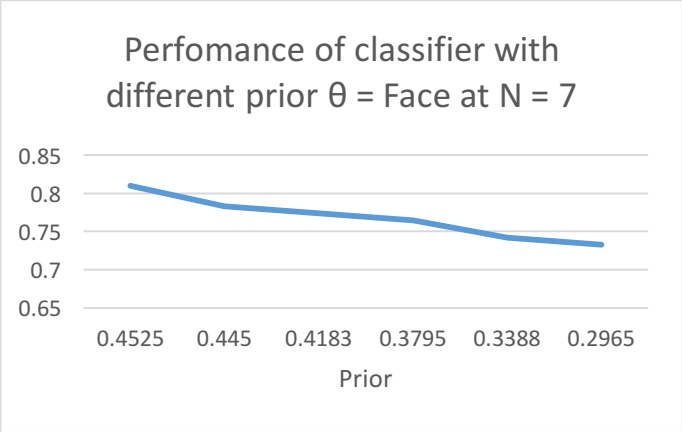


*Figure 11*

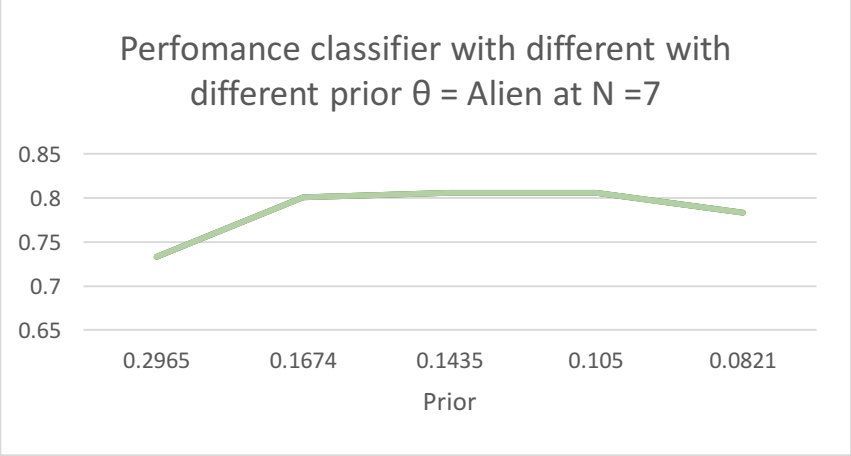


*Figure 12*

## 7.Summary and Conclusion

This report introduces the process of builing and testing classifiers. The classifer is built on two different technique: supervised and unsupervised, and both classifiers are under same scrutiny method. In this report, I use GMM to build classifiers and use MAP method to maintain the performances. The report also provides a discussion to refine supervised classifier.

By studying on the results, the performance of supervised classifier can be improved by assigning N value to [6,8]. Changing prior value also have impact on overall performance, but to improve accuracy through this way is still doubtlable.Other suggest ways to modify the value of prior is to add irrelevant images to the training set, or add more class images. However both of these approaches will influence the feature vector, which will lead to the similar situation as the one in Section 5.

The result figure of unsupervised classifer shows the uncertainty performance. It is becaused the random choosed initial value for EM algorithm. The naïve way of improving the performance can be running classifier N times and pick the highest accuracy as result.