Assignment

Archit Kumar, Avinav Sanyal, Sakhile Naga Koti Reddy

Saturday 11th November, 2017

Contents

1 Theory

The aim of this assignment is to make bayes classifiers using knn(k-nearest-neighbours) method and dhmm(discrete hidden markov models).

1.1 K-nearest Neighbors

It is a non-parametric method used for classification and regression (supervised algorithm). First the distance of a point from all the points is taken, and then sorted according to distance. Then k least distant are taken into consideration. An object is classified by a majority vote of its neighbors. If k = 1, then the object is assigned to the class of that single nearest neighbor.

1.1.1 Algorithm

The training examples are vectors in a multidimensional feature space (here 39d), each with a class label. First all the feature vectors and class labels of the training samples are stored in a data-structure and then k (number of neighbors to consider) is chosen. A commonly used distance metric for continuous variables is Euclidean distance. We have used DTW (Dynamic Time Warping) to calculate the distance between two observation sequences.

1.1.2 Dynamic time warping

In time series analysis, dynamic time warping (DTW) is one of the algorithms for measuring similarity between two temporal sequences, which may vary in speed. For instance, similarities in walking could be detected using DTW, even if one person was walking faster than the other, or if there were accelerations and decelerations during the course of an observation. DTW has been applied to temporal sequences of video, audio, and graphics data indeed, any data that can be turned into a linear sequence can be analyzed with DTW. A well known application has been automatic speech recognition, to cope with different speaking speeds. Other applications include speaker recognition and online signature recognition. Also it is seen that it can be used in partial shape matching application.

Implementation:

This example illustrates the implementation of the dynamic time warping algorithm when the two sequences *s* and *t* are strings of discrete symbols. For two symbols *x* and *y*, `d(x, y)` is a distance between the symbols, e.g. `d(x, y)` = $|x - y|$

```
int DTWDistance(s: array [1..n], t: array [1..m]) {
   DTW := array [0..n, 0..m]

   for i := 1 to n
       DTW[i, 0] := infinity
   for i := 1 to m
       DTW[0, i] := infinity
   DTW[0, 0] := 0

   for i := 1 to n
       for j := 1 to m
           cost := d(s[i], t[j])
           DTW[i, j] := cost + minimum(DTW[i-1, j  ],    // insertion
                                       DTW[i  , j-1],    // deletion
                                       DTW[i-1, j-1])    // match

   return DTW[n, m]
}
```

1.2 K-means clustering and vector quantization

The observations in observation sequences which we were given to do HMM upon was 39 dimensional, to make computation easier it was converted into 1dimensional. The k-means algorithm takes as input the number of clusters to generate, k, and a set of observation vectors to cluster. It returns a set of centroids, one for each of the k clusters. An observation vector is classified with the cluster number or centroid index of the centroid closest to it. A vector v belongs to cluster i if it is closer to centroid i than any other centroids. If v belongs to i, we say centroid i is the dominating centroid of v. The k-means algorithm tries to minimize distortion, which is defined as the sum of the squared distances between each observation vector and its dominating centroid. Each step of the k-means algorithm refines the choices of centroids to reduce distortion. The change in distortion is used as a stopping criterion: when the change is lower than a threshold, the k-means algorithm is not making sufficient progress and terminates. One can also define a maximum number of iterations. Since vector quantization is a natural application for k-means, information theory terminology is often used. The centroid index or cluster index is also referred to as a code and the table mapping codes to centroids and vice versa is often referred as a code book. The result of k-means, a set of centroids, can be used to quantize vectors. Quantization aims to find an encoding of vectors that reduces the expected distortion.

1.3 Hidden Markov Model

For an HMM with N hidden states and an observation sequences of T observations, there are N^T possible hidden sequences. For real tasks, where N and T are both large, N^T is a very large number, so we cannot compute the total observation likelihood for each hidden state sequence and then summing them. Instead of using such and extrememly exponential algorithm, we use and efficient $O(N^2T)$ algorithm called the **forward algorithm**. The forward algorithm is a kind of **dynamic programming** algorithm, that is, an algorith that uses a table to store intermediate values as it builds up the probability of the observation sequence. The forward algorithm computes the observation probability by summing over the probabilities of all possible hidden state paths that could generate the observvation sequence, but it does so efficiently by implicitly folding each of these paths into a single forward trellis.

A hidden Markov model (HMM) allows us to talk about both observed events (like words that we see in the input) and hidden events (like part-of-speech tags) that we think of as causal factors in our probabilistic model.
An HMM is specified by the following components :

$Q = q_1 q_2 \dots q_N$	a set of N **states**
$A = a_{11} a_{12} \dots a_{n1} \dots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \dots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i
q_0, q_F	a special **start state** and **end (final) state** that are not associated with observations, together with transition probabilities $a_{01} a_{02} \dots a_{0n}$ out of the start state and $a_{1F} a_{2F} \dots a_{nF}$ into the end state

$\alpha_{t-1}(i)$	the **previous forward path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

$$\alpha_1(j) \;=\; a_{0j} b_j(o_1) \;\; 1 \le j \le N$$

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t); \quad 1 \leq j \leq N, 1 < t \leq T$$

1.4 EM for HMM

Algorithm [edit]

Set $\theta = (A, B, \pi)$ with random initial conditions. They can also be set using prior information about the parameters if it is available; this can speed up the algorithm and also steer it toward the desired local maximum.

Forward procedure [edit]

Let $\alpha_i(t) = P(Y_1 = y_1, \ldots, Y_t = y_t, X_t = i|\theta)$, the probability of seeing the $y_1, y_2, \ldots, y_t$ and being in state i at time t. This is found recursively:

1. $\alpha_i(1) = \pi_i b_i(y_1)$,
2. $\alpha_i(t+1) = b_i(y_{t+1}) \sum_{j=1}^{N} \alpha_j(t) a_{ji}$.

Backward procedure [edit]

Let $\beta_i(t) = P(Y_{t+1} = y_{t+1}, \ldots, Y_T = y_T|X_t = i, \theta)$ that is the probability of the ending partial sequence $y_{t+1}, \ldots, y_T$ given starting state i at time t . We calculate $\beta_i(t)$ as,

1. $\beta_i(T) = 1$,
2. $\beta_i(t) = \sum_{j=1}^{N} \beta_j(t+1) a_{ij} b_j(y_{t+1})$.

Update [edit]

We can now calculate the temporary variables, according to Bayes' theorem:

$$\gamma_i(t) = P(X_t = i|Y, \theta) = \frac{P(X_t = i, Y|\theta)}{P(Y|\theta)} = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)},$$

which is the probability of being in state i at time t given the observed sequence Y and the parameters θ

$$\xi_{ij}(t) = P(X_t = i, X_{t+1} = j|Y, \theta) = \frac{P(X_t = i, X_{t+1} = j, Y|\theta)}{P(Y|\theta)} = \frac{\alpha_i(t)a_{ij}\beta_j(t+1)b_j(y_{t+1})}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i(t)a_{ij}\beta_j(t+1)b_j(y_{t+1})},$$

which is the probability of being in state i and j at times t and $t+1$ respectively given the observed sequence Y and parameters θ.

The denominators of $\gamma_i(t)$ and $\xi_{ij}(t)$ are the same ; they represent the probability of making the observation Y given the parameters θ.

The parameters of the hidden Markov model θ can now be updated:

- $\pi_i^* = \gamma_i(1),$

which is the expected frequency spent in state i at time 1.

- $a_{ij}^* = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)},$

which is the expected number of transitions from state *i* to state *j* compared to the expected total number of transitions away from state *i*. To clarify, the number of transitions away from state *i* does not mean transitions to a different state *j*, but to any state including itself. This is equivalent to the number of times state *i* is observed in the sequence from $t = 1$ to $t = T - 1$.

- $b_i^*(v_k) = \frac{\sum_{t=1}^{T} 1_{y_t = v_k}\gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)},$

where

$$1_{y_t = v_k} = \begin{cases} 1 & \text{if } y_t = v_k, \\ 0 & \text{otherwise} \end{cases}$$

is an indicator function, and $b_i^*(v_k)$ is the expected number of times the output observations have been equal to v_k while in state i over the expected total number of times in state i.

These steps are now repeated iteratively until a desired level of convergence.

Note: It is possible to over-fit a particular data set. That is, $P(Y|\theta_{\text{final}}) > P(Y|\theta_{\text{true}})$. The algorithm also does **not** guarantee a global maximum.

Refrence : Wikipedia

2 Observations

2.1 K-Nearest-Neighbors

For speech dataset knn=5, 10, 30, 90, 100, 210, 230

knn =5

$$\begin{bmatrix} 8 & 13 & 6 \\ 2 & 17 & 5 \\ 0 & 5 & 9 \end{bmatrix}$$

Accuracy : 52.3%

Table 1: Results

	Precision	Recall	F-measure
Class 1	0.296	0.8	0.432
Class 2	0.708	0.485	0.575
Class 3	0.642	0.45	0.529
Average	0.548	0.578	0.512

knn=10

$$\begin{bmatrix} 10 & 12 & 5 \\ 3 & 17 & 4 \\ 1 & 8 & 5 \end{bmatrix}$$

Accuracy : 49.2%

Table 2: Results

	Precision	Recall	F-measure
Class 1	0.37	0.714	0.487
Class 2	0.708	0.459	0.556
Class 3	0.357	0.357	0.357
Average	0.478	0.51	0.466

knn=30

$$\begin{bmatrix} 8 & 18 & 1 \\ 3 & 21 & 0 \\ 0 & 12 & 2 \end{bmatrix}$$

Accuracy : 47.69%

Table 3: Result

	Precision	Recall	F-measure
Class 1	0.296	0.727	0.42
Class 2	0.875	0.411	0.559
Class 3	0.142	0.667	0.234
Average	0.437	0.601	0.404

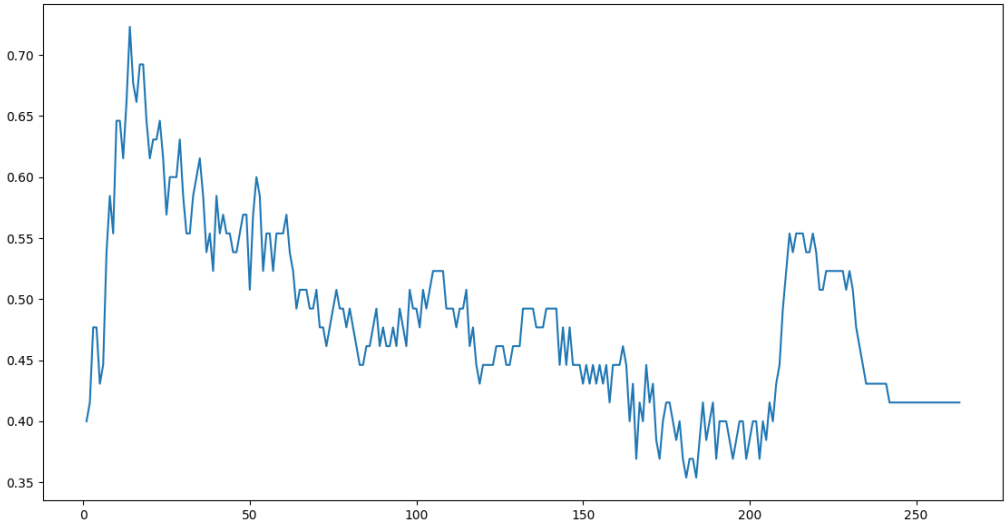
knn=90

$$\begin{bmatrix} 9 & 18 & 0 \\ 1 & 23 & 0 \\ 0 & 14 & 0 \end{bmatrix}$$

Accuracy : 49.2%

Table 4: Results

	Precision	Recall	F-measure
Class 1	0.333	0.9	0.486
Class 2	0.958	0.418	0.582
Class 3	0	not def	not def
Average	0.43	not def	not def



For image dataset knn=30,80,130

knn = 30 Accuracy = 44%

$$\begin{bmatrix} 44 & 6 & 0 \\ 34 & 15 & 1 \\ 37 & 6 & 7 \end{bmatrix}$$

Table 5: Results

	Precision	Recall	F-measure
Class 1	0.88	0.581	0.699
Class 2	0.30	0.382	0.336
Class 3	0.14	0.555	0.224
Average	0.44	0.506	0.409

knn = 80 Accuracy = 34%

$$\begin{bmatrix} 1 & 14 & 35 \\ 2 & 18 & 30 \\ 2 & 16 & 32 \end{bmatrix}$$

Table 6: Results

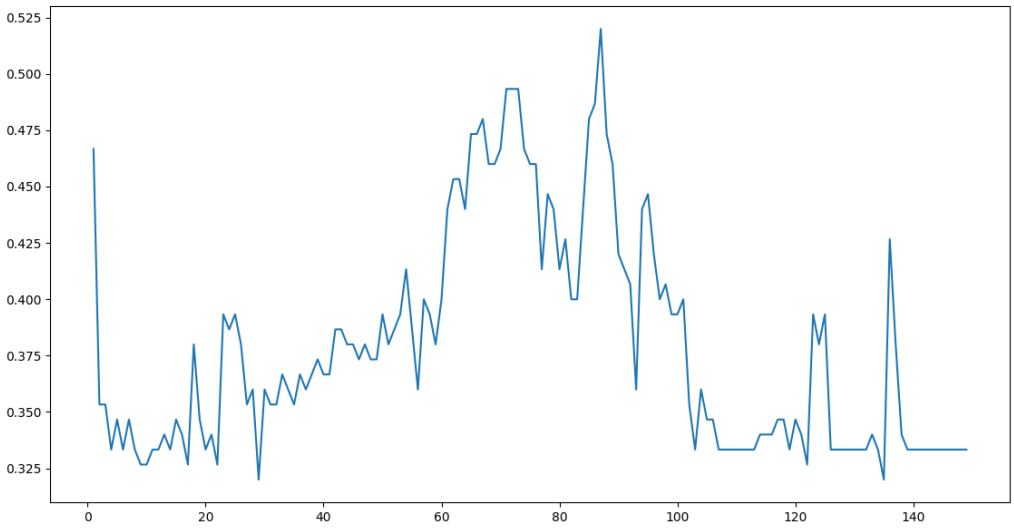
	Precision	Recall	F-measure
Class 1	0.02	0.2	0.036
Class 2	0.36	0.375	0.367
Class 3	0.64	0.329	0.434
Average	0.34	0.301	0.279

knn = 130 Accuracy = 41.33%

$$\begin{bmatrix} 42 & 8 & 0 \\ 30 & 20 & 0 \\ 32 & 18 & 0 \end{bmatrix}$$

Table 7: Results

	Precision	Recall	F-measure
Class 1	0.84	0.403	0.544
Class 2	0.40	0.434	0.416
Class 3	0	not def	not def
Average	0.413	not def	not def



2.1.1 Observations

Here, we can see that the accuracy is not following an order. k=1 has the highest whereas k=5 has the lowest. Hence, this method will not give the best possible k.

For scene image dataset

2.2 Baum-Welch EM Method

For speech dataset

Baum - welch iterations : 1

$$\begin{bmatrix} 18 & 8 & 1 \\ 13 & 8 & 3 \\ 3 & 1 & 10 \end{bmatrix}$$

Accuracy : 55.3%

Table 8: Results

	Precision	Recall	F-measure
Class 1	0.666	0.581	0.621
Class 2	0.25	0.471	0.326
Class 3	0.714	0.714	0.714
Average	0.543	0.588	0.554

baum-welch iterations : 5

$$\begin{bmatrix} 20 & 6 & 1 \\ 14 & 7 & 3 \\ 2 & 2 & 10 \end{bmatrix}$$

Accuracy: 56.9%

Table 9: My caption

	Precision	Recall	F-measure
Class 1	0.741	0.555	0.634
Class 2	0.291	0.466	0.358
Class 3	0.714	0.714	0.714
Average	0.582	0.578	0.569

Baum-Welch iterations : 10

$$\begin{bmatrix} 18 & 7 & 2 \\ 9 & 11 & 4 \\ 0 & 2 & 12 \end{bmatrix}$$

Accuracy:63.08%

Table 10: Results

	Precision	Recall	F-measure
Class 1	0.667	0.667	0.667
Class 2	0.458	0.55	0.5
Class 3	0.857	0.667	0.75
Average	0.661	0.628	0.639

2.2.1 Observations

For speech dataset
No of points in class 1 :109
No of points in class2 : 98
No of points in class3 : 57
The precision for class 2 is low when compared to the other classes.
This is because class 2 is 'dI' which has similarity wih both class pronounciations 'di' and 'DI'.
As there is no pronounciation similarity between 'di' and 'DI', the points of 'di' do not go into 'DI' and vice-versa.
The accuracy in the Baum Welch increases with the number of iterations.

3 Problems Faced

3.1 Scaling

In order to understand why scalling is required for inplementing the reestimation procedure of HMM consider the defination of $\alpha_t(i)$ .It can be seen that $\alpha_t(i)$ consists of the sum of a large number of terms each of the form

$$\prod_{s=1}^{t-1} a_{q_s q_{s+1}} \prod_{s=1}^{t} b_{qs}(O_s)$$

with q_t=S_i .Since each a and b term is less than 1 ,it can be seen that as t starts to get big each term of $\alpha_t(i)$ starts to head exponentially to zero.For sufficiently large t (eg, 100 or more) the dynamic range of the $\alpha_t(i)$ computation will exceed the precision range of essentially any machine. Hence the only reasonable way of performing the computation is by incorporating a scaling pocedure.
The basic pocedure is used to multiply $\alpha_t(i)$ by a scaling coefficient that is independent of i ,with the goal of keeping the scaled $\alpha_t(i)$ within the dynamic range of the computer for 1¡=t¡=T.A similar scaling is done to the $\beta_t(i)$coefficient and then at the end of the computation the scaling coefficients are cancelled out excatly.
To understand this scaling procedure better ,consider the reestimation formula for the state transition coefficients .If we write the reestimation formula directly in terms of the forward and backward variables we get

$$\bar{a}_{ij} = c$$

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \alpha_t(i)\, a_{ij} b_j(O_{t+1})\, \beta_{t+1}(j)}{\sum_{t=1}^{T} \sum_{j=1}^{N} \alpha_t(i)\, a_{ij} b_j(O_{t+1})\, \beta_{t+1}(j)}.$$

Consider the computation of $\alpha_t(i)$ for each t we first compute $\alpha_t(i)$ according to the induction formula

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(O_{t+1}), \qquad 1 \le t \le T-1$$
$$1 \le j \le N.$$

and then we multiply it by a scaling coefficient c_t where

$$C_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)}$$

Thus for a fixed t we first compute

$$\alpha_t(i) = \sum_{i=1}^{N} \hat{\alpha}_{t-1}(j) a_{ij} b_j(O_t)$$

Then the scaled coefficient set is computed as

$$\hat{\alpha}_t(i) = \frac{\sum_{j=1}^{N} \hat{\alpha}_{t-1}(j)\, a_{ij} b_j(O_t)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \hat{\alpha}_{t-1}(j)\, a_{ij} b_j(O_t)}.$$

By induction we can write as

$$\hat{\alpha}_{t-1}(j) = (\prod_{\tau=1}^{t-1} C_\tau)\alpha_{t-\tau}(j)$$

Thus we can write as

$$\hat{\alpha}_t(i) = \frac{\sum_{j=1}^{N} \alpha_{t-1}(j)\left(\prod_{\tau=1}^{t-1} c_\tau\right) a_{ij} b_j(O_t)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_{t-1}(j)\left(\prod_{\tau=1}^{t-1} c_\tau\right) a_{ij} b_j(O_t)} = \frac{\alpha_t(i)}{\sum_{i=1}^{N} \alpha_t(i)}$$

ie each is efficiently scaled by the sum over all states of
Next we compute the terms from the backward recursion.The only difference here is that we use the same scale factors for each time t for the betas as we use the same scale factors for each time t for the betas as we used for the alphas.Hence the scalled are of the form

$$\hat{\beta}_t(i) = c_t \beta_t(i)$$

Since each scale factor effictively restores the magnitude of the and terms are compared using the same scaling factors on the as was used on the is an effective way of keeping the computation within resonable bounds.Furthermore in terms of the scaled variables we see that the reestimation equation becomes

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \hat{\alpha}_t(i)\, a_{ij} b_j(O_{t+1})\, \hat{\beta}_{t+1}(j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \hat{\alpha}_t(i)\, a_{ij} b_j(O_{t+1})\, \hat{\beta}_{t+1}(j)}$$

but each can be written as

$$\hat{\alpha}_t(i) = [\prod_{s=1}^{t} c_s]\alpha_t(i) = C_t\alpha_t(i)$$

and each can be written as

$$\hat{\beta}_{t+1}(j) = \left[\prod_{s=t+1}^{T} c_s\right]\beta_{t+1}(j) = D_{t+1}\beta_{t+1}(j).$$

Thus can be written as

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} C_t\alpha_t(i)\, a_{ij} b_j(O_{t+1})\, D_{t+1}\beta_{t+1}(j)}{\sum_{t=1}^{T-1}\sum_{j=1}^{N} C_t\alpha_t(i)\, a_{ij} b_j(O_{t+1})\, D_{t+1}\beta_{t+1}(j)}.$$

FInally the term can be seen to be of the form

$$C_t D_{t+1} = \prod_{s=1}^{t} c_s \prod_{s=t+1}^{T} c_s = \prod_{s=1}^{T} c_s = C_T$$

independent of t.Hence the terms cancle outof both the numerator and denominator of and the exact reestimation equation is therefore relized.

It should be obvious that the above scaling procedure applies equally well to reestimation of the or coefficiets.It should also be obvious that the scaling procedure of need not be applied at every time instant t but can be performed whenever desired or necessary.If scalling is not performed at some instant t ,the scalling coefficients are set to 1 at that time and all the conditions discussed above are then met.
The only real change to the HMM procedure because of scaling is the procedure for computing .We can not merely sum up the terms since these are scaled already.
However we can use the property that

$$\prod_{t=1}^{T} C_t \sum_{i=1}^{N} \alpha_T(i) = C_T \sum_{i=1}^{N} \alpha_T(i) = 1$$

Thus we have

$$\prod_{t=1}^{T} c_t \cdot P(O|\lambda) = 1 \tag{101}$$

or

$$P(O|\lambda) = \frac{1}{\prod_{t=1}^{T} c_t} \tag{102}$$

or

$$\log [P(O|\lambda)] = -\sum_{t=1}^{T} \log c_t. \tag{103}$$

Thus the log of P can be computed ,but not since it would be out of the dynamic range of the machine anyway.

3.1.1 Baum-Welch

For image dataset

Baum - welch iterations : 1
N = [2, 2, 2]
M = 20

$$\begin{bmatrix} 18 & 17 & 15 \\ 23 & 16 & 11 \\ 24 & 17 & 9 \end{bmatrix}$$

Accuracy : 28.66%

Table 11: Results

	Precision	Recall	F-measure
Class 1	0.36	0.277	0.313
Class 2	0.32	0.32	0.32
Class 3	0.18	0.257	0.211
Average	0.286	0.284	0.281

Baum - welch iterations : 1
N = [5, 5, 5]
M = 20

$$\begin{bmatrix} 19 & 17 & 14 \\ 22 & 15 & 13 \\ 23 & 17 & 10 \end{bmatrix}$$

Accuracy : 29.33%

Table 12: Results

	Precision	Recall	F-measure
Class 1	0.38	0.296	0.332
Class 2	0.30	0.306	0.302
Class 3	0.20	0.27	0.229
Average	0.293	0.29	0.287

baum-welch iterations : 3
N = [2, 2, 2]
M = 20

$$\begin{bmatrix} 19 & 16 & 15 \\ 23 & 17 & 10 \\ 22 & 18 & 10 \end{bmatrix}$$

Accuracy: 30.66%

Table 13: My caption

	Precision	Recall	F-measure
Class 1	0.38	0.297	0.334
Class 2	0.34	0.333	0.336
Class 3	0.20	0.286	0.235
Average	0.306	0.305	0.302

Baum-Welch iterations : 5
N = [2, 2, 2]
M = 20

$$\begin{bmatrix} 19 & 16 & 15 \\ 20 & 18 & 12 \\ 24 & 16 & 10 \end{bmatrix}$$

Accuracy:31.33%

Table 14: Results

	Precision	Recall	F-measure
Class 1	0.38	0.301	0.336
Class 2	0.36	0.36	0.36
Class 3	0.20	0.27	0.229
Average	0.313	0.310	0.308

Baum-Welch iterations : 5
N = [5, 5, 5]

M = 20

$$\begin{bmatrix} 19 & 17 & 14 \\ 20 & 18 & 12 \\ 23 & 17 & 10 \end{bmatrix}$$

Accuracy : 31.33%

Table 15: Results

	Precision	Recall	F-measure
Class 1	0.38	0.306	0.339
Class 2	0.36	0.346	0.352
Class 3	0.2	0.277	0.232
Average	0.313	0.309	0.307

Baum-Welch iterations : 10
N = [2, 2, 2]
M = 20

$$\begin{bmatrix} 19 & 16 & 15 \\ 23 & 17 & 10 \\ 22 & 18 & 10 \end{bmatrix}$$

Accuracy:30.66%

Table 16: Results

	Precision	Recall	F-measure
Class 1	0.38	0.296	0.332
Class 2	0.34	0.333	0.336
Class 3	0.20	0.285	0.235
Average	0.661	0.628	0.301

Baum-Welch iterations : 10
N = [5, 5, 5]
M = 20

$$\begin{bmatrix} 19 & 16 & 15 \\ 21 & 18 & 11 \\ 23 & 17 & 10 \end{bmatrix}$$

Accuracy:31.33

Table 17: Results

	Precision	Recall	F-measure
Class 1	0.38	0.301	0.336
Class 2	0.36	0.352	0.356
Class 3	0.20	0.277	0.232
Average	0.313	0.31	0.308

3.1.2 Observations

Our observarions seen by me the updation of takes in single iteration and later the change is very less near 10^{-12}
The accuracy here we get is the least case of GMM
From this HMM is not good to clasify the natural images with RBG color histogram repersentation because ther are not much colurs in natural data and when taken in on 64*64 patch all the pixels are almost same and no wide spread of data.

4 Conclusions

GMM is better than HMM for natural images.