Report

15307130283 Pingxuan Huang

I. Dimensionality reduction

(I) PCA

（related code：I_2.m）

- **Answer to the experimental question**： The PCA that is carried out after the centralization (removal of the mean) is the PCA that meets the requirements, as shown below:

Let Q be the projection matrix, then the vector obtained by back-projecting x_i after projecting it to a low dimension is：

$$QQ^T(x_i - b) + b$$

Loss function：

$$f = \sum_{k=1}^{K} \|x_i - QQ^T(x_i - b) + b\|_Z^2 \quad s.t.\ Q^TQ = I$$

According objective function：

$$min\, f\ s.t. Q^TQ = I$$

$$令 \frac{\partial f}{\partial b} = 0$$

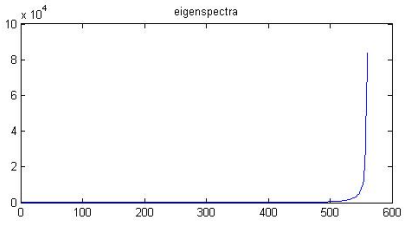
$$2\Sigma_{i=1}^{m}(I - QQ^T)(I - QQ^T)(x_i - b) = 0$$

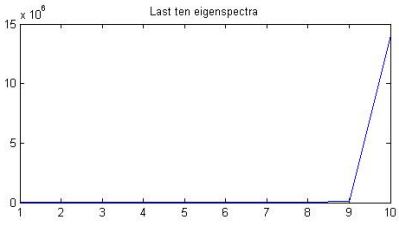
$$(I - QQ^T)\sum_{i=1}^{m}(x_i - b) = 0$$

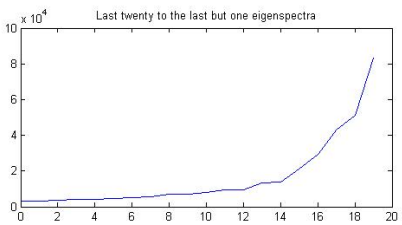
$$b = \frac{1}{m}\Sigma_{i=1}^{m}x_i \ \ i.e\ b\ is\ the\ mean\ of\ x_i$$

∴ PCA that after the centralization is the authentic PCA

1、







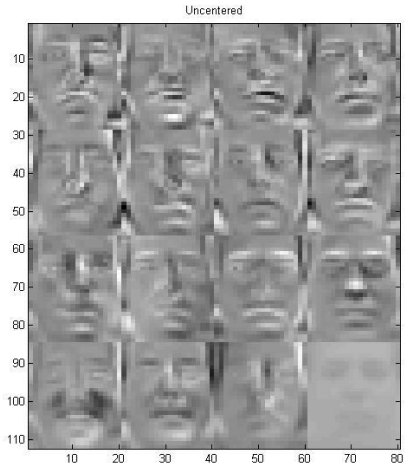
- The “all eigenvalues”, the “maximum 10 eigenvalues”, eigenvalues of Vctr and the

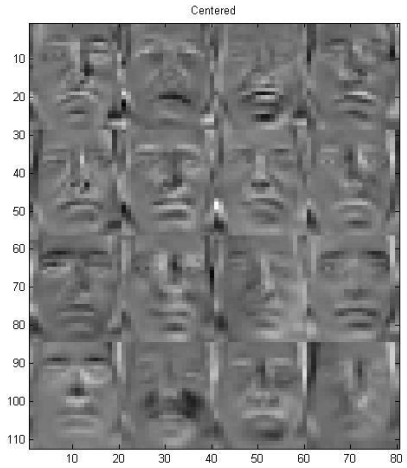
“maximum twenty eigenvalues except the largest eigenvalues” are plotted.

Data analysis：

The last feature value is much larger than other feature values, so the corresponding feature vector has a great influence on the projection result. The choice of k is difficult, and different k values can be selected for different requirements of the similarity between the projection data and the original data.

2、



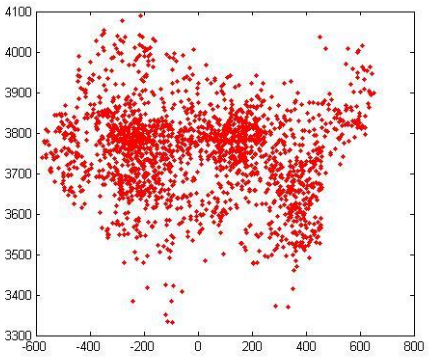


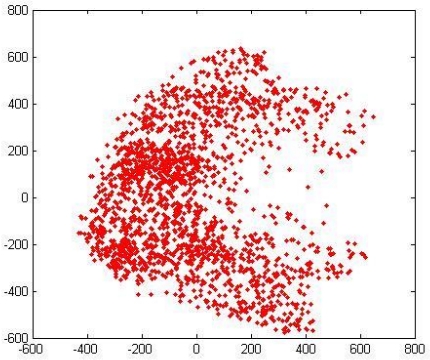
Non-centralized centralized

Analysis：

1) Overall trend: As the eigenvalue decreases, the according facial features (feature vector) become blurred. This is because the ability to express facial features is gradually weakened, and it can be noted that different feature vectors represent different feature points: for example, the first face focuses on extracting the basic facial features of the person; the second and thee third focuses on extracting the mouth features of the person.
2) Difference: Figure 2 is darker than Figure 1, because the corresponding feature vector of Figure 2 is centered, and the value is closer to 0 (this can be seen in the third question).
3) The contour of the face in the centralized picture (Fig. 2) is clearer than that of the first image, especially when the feature value is small. This also shows that **centralization is a reasonable operation**.
4) The fourth last eigenvector of the first graph is the third last eigenvector of the second graph, which indicates that the centralization operation has an influence on the importance of the eigenvector.

3、



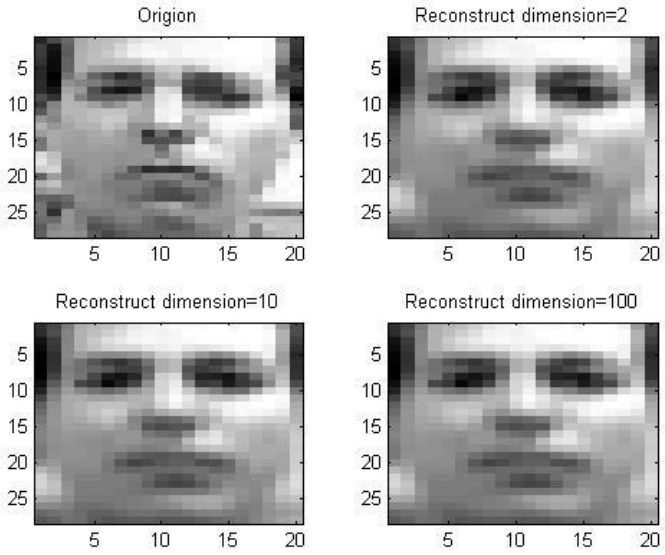


Non-centralized data centralized data

Analysis：

1) From the overall distribution of the data after projection, the centralized data is concentrated near 0 (the x\y axis is concentrated near 0), which is the result of centralization;
2) The centralized data is mainly distributed in two mutually perpendicular directions, because the projection operation projects the data into the two dimensions with the largest variance (two directions), and the characteristics of the non-centralized data are not obvious (only Reflecting a concentrated distribution in one dimension)
3) In the experiment, the adjacent projection points are selected for observation, and the similarity of the facial expressions between the adjacent points (the centralized data) is higher (the mouth, the eyes and the .et shows similar features) than that of non-centralized data; this shows that the centralized data projection can better maintain the "distance" characteristics.

4、



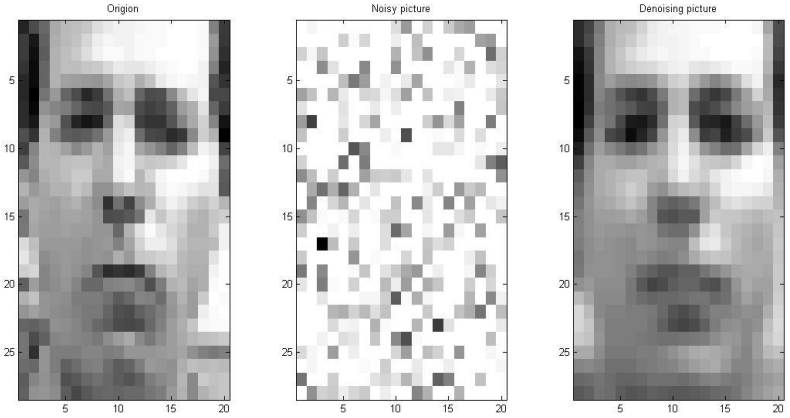
- Display the original image and the reconstructed data points after reducing it to 2, 10,

100 dimensions

Analysis：

1) The reconstructed image maintains the basic facial features of the original image;
2) The differences between reconstructed images after reducing to different dimensions are trivial, also considering that reducing the dimension to 2 dimensions can greatly reduce the complexity, it is reasonable to reduce to 2 dimensions;

5、



The figure above shows the original image, the Gaussian noise graph, and the noise reduction processing (dimensionality reduction & reconstruction). It can be clearly seen that extracting principal component has a good noise reduction effect. This is one of the main uses of the PCA.

(II) Swiss Roll

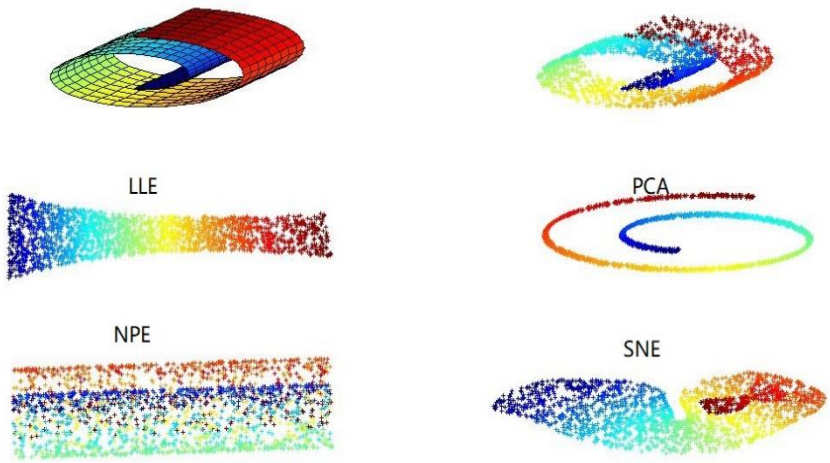
（related code: I_3.m）

- Dimensionality reduction with principal component analysis (PCA), Neighborhood preserving embedding (NPE), and distributed Stochastic Neighbor Embedding (SNE) algorithms

```
%% 1.3
% PCA
Z = X;
N = size(Z,2);
Zctr = Z - repmat(mean(Z, 2), 1, N);   %centralization
[Vctr, Dctr] = eig(Zctr*Zctr'/N);
[lambda_ctr, order] = sort(diag(Dctr));
Vctr = Vctr(:, order); %PCA


Z =   Vctr(:,end-1:end)'* Zctr; %projection
```

```
% NPE    SNE
tem = compute_mapping(X', 'NPE');
Z1 = tem';
tem = compute_mapping(X', 'SNE');
Z2 = tem';
```

Result：



Analysis：

1) Both LLE, NPE, and SNE are specialized Manifold learning algorithms, but from the results, the LLE and SNE can better reflect the distance information of source data, and SNE could further reflects the manifold characteristics (LLE can only reflect Distance characteristics);
2) In comparison, the learning result of NPE is very poor: this algorithm only makes a plane projection of the source data;
3) PCA records the spiral structure characteristics of the source data, which is the characteristic of principal component analysis (extracting the most significant features of the data), but the drawback is that the relationship between the data (distance relationship) is not really mined.

II. Mixed Gaussian model

- **Formula derivation**:

$$P_{(x)} = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma)$$

Set a latent variable Z: a k-dimensional vector

$$Z_k \in \{0,1\} \;\&\; \Sigma_k Z_k = 1$$

Therefore:

$$\begin{aligned} p_{(Z)} &= p_{(z_1 \cdots z_k)} \\ &= p_{(z_1)} \cdots p_{(z_k)} \\ &= \Pi_{k=1}^{K} \pi_k^{z_k} \end{aligned}$$

And:

$$P_{(x|Z_k=1)} = N(x|\mu_k, \Sigma)$$

Therefore：

$$P_{(x|Z)} = \prod_{k=1}^{K} N(x|\mu_k, \Sigma)^{Z_k}$$

Let：

$$r(Z_k) = P_{(z_k=1|x)}$$

so $r(Z_k)$ is the Posterior probability

E-step:

$$\begin{aligned} r_{(z_k)} &= \frac{P(x, z_k = 1)}{\sum_{j=1}^{K} p(x, z_k = 1)} \\ &= \frac{P(Z_k = 1) \cdot P(x|Z_k = 1)}{\sum_{j=1}^{k} P(Z_j = 1) \cdot P(x|Z_j = 1)} \\ &= \frac{\pi_K N(x|u_k, \Sigma)}{\sum_{j=1}^{K} \pi_j N(x|u_{j,} \Sigma)} \end{aligned}$$

Suppose $X_1 \ldots X_n$ are sampling points，then the likelihood is：

$$\begin{aligned} I &= \log \Pi_{n=1}^{N} P(x_n|\pi, \mu, \Sigma) \\ &= \sum_{n=1}^{N} \log\Big(\sum_{k=1}^{K} \pi_k \, N(x_n|\mu_k, \varepsilon)\Big) \end{aligned}$$

$$\text{where}(I : I(\mu, \Sigma, \pi))$$

M-step:

(1)

$$\begin{aligned} \frac{\partial I}{\partial(u_k)} &= -\sum_{n=1}^{N} \frac{\pi_k N(x_n|\mu_k, \Sigma)}{\sum_{j=1}^{K} \pi_j N(\pi_n|\mu_j, \Sigma)} \cdot \Sigma^{-1}(x_n - \mu_k) \\ &= -\sum_{n=1}^{N} r(z_{n_k}) \cdot \Sigma^{-1}(x_n - \mu_k) \end{aligned}$$

$let \;\; \frac{\partial I}{\partial(u_k)} = 0$:

$$\therefore \mu_k = \frac{1}{N_k}\sum_{n=1}^{N} r(Z_{nk}) \cdot X_n$$

$$\text{where } N_k = \sum_{n=1}^{N} r(z_{nk})$$

(2)

$$\frac{\partial I}{\partial \Sigma} = \sum_{k=1}^{K}\sum_{n=1}^{N} r(z_{nk})\left[-\frac{1}{2}\Sigma^{-1} + \frac{1}{2}\Sigma^{-1}(x_n-\mu_k)(x_n-\mu_k)^T\Sigma^{-1}\right]$$

令 $\frac{\partial I}{\partial(\Sigma)} = 0$：

$$\Sigma = \frac{1}{N}\sum_{k=1}^{K}\sum_{n=1}^{N} r(z_{nk})[(x_n-\mu_k)(x_n-\mu_k)^T]$$

(3)

$\because \sum_{k=1}^{K} \pi_k = 1$

$\therefore$ Introduce the Laplacian Operator

$$L(\pi_k) = I + \lambda\left(\sum_{k=1}^{K}\pi_k - 1\right)$$

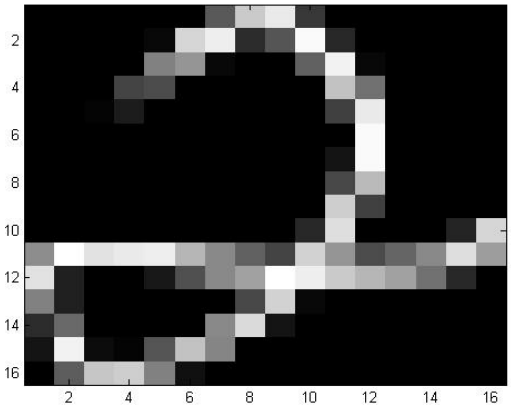
$$\frac{\partial L(\pi_k)}{\partial \pi_k} = \pi_k \cdot \sum_{n=1}^{N}\frac{N(x_n|\mu_k,\Sigma)}{\sum_{j=1}^{k}\pi_j N(x_n|\mu_j,\Sigma)} - N$$

let $\frac{\partial L(\pi_k)}{\partial \pi_k} = 0$

$$\therefore \pi_k = \frac{N_k}{N}$$

2.3

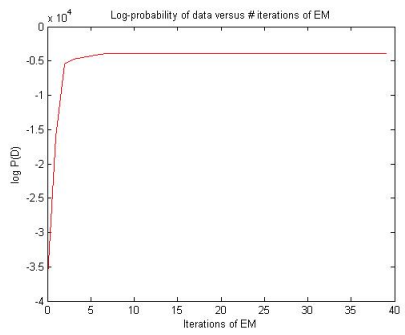
The visualization result of'2s'：

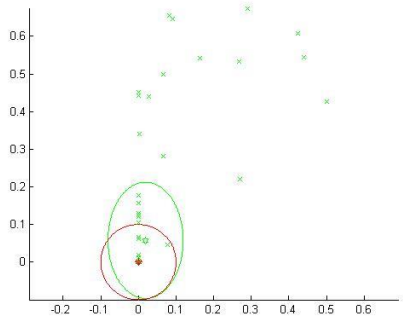


- Train the model on the 2 and 3 training sets before adjusting the parameters (40

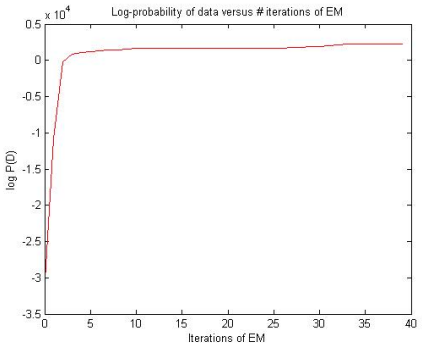
iterations)

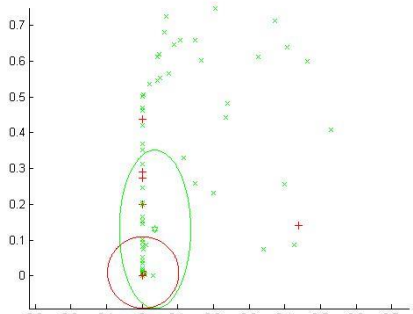
Training results on the dataset 2（log-Probability = -3905.1）





Training result on dataset3（log-Probability = 2355.1）:





Analysis：

1) From the cluster visualization map on the right side, the training results on the 2 and 3 data sets are not very good (one of the Gaussian distributions only works for the generation of very few data, and the differences between these two distributions are trivial, which render the model feckless);
2) Because the E-M step is fixed, the classification result is mainly affected by the initialization of parameters. Try different initialization methods as below:

Parameter initialization:

- Initialization of mean(μ):

For the convenience of testing, add *randConst* to the function's parameters (set the default value as 1)

```
function [p,mu,vary,logProbX] = mogEM_I(x,K,iters,minVary,randConst,plotFlag)
if nargin==3 minVary = 0; plotFlag = 0; randConst = 1;end;
```

- Analysis of the original code shows that the two main influence parameters are randConst and minVary:

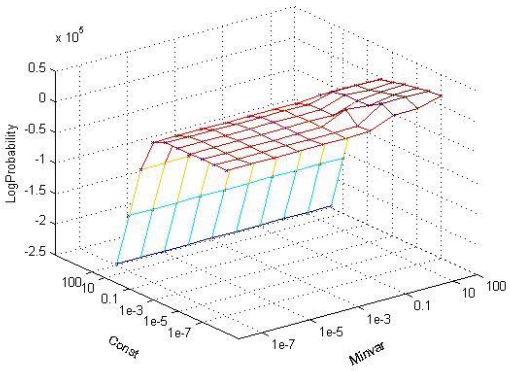
1) The original algorithm initializes each dimension of the model mean as "sample mean + sample variance" / randConst, so the larger the 'randConst' is, the smaller the influence of the "sample variance" is, and the initialization of the mean is closer to the sample mean;
2) When initialize the "probability of occurrence" of each cluster, if 'randConst' is large, the difference between the initial occurrence probability of each cluster is small

(close to 1/*number of cluster*). Otherwise, if 'randConst' ' is small, the initial probability value is random and discrete;

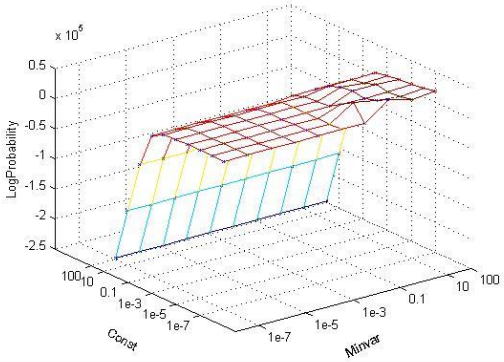
3) minVary determines the minimum variance of the model. When the dispersion of the data itself is not high, this parameter couldn't be set too large;

- Modify code to find the best parameters of minvar(1e-7 -- 100) and randConst(1e-7 -- 100)on a larger scale (run_III.m):

```
% search area
minvar= [1e-7,1e-6,1e-5,1e-4,1e-3,0.01,0.1,1,10,100];
const = [1e-7,1e-6,1e-5,1e-4,1e-3,0.01,0.1,1,10,100];


result=zeros(length(minvar),length(const));
for x = 1:length(minvar)     % the area of Minvar and Randconst
    for y = 1:length(const)
        [Pi,mu,vary,logProbX] = mogEM_I(train2,2,40,minvar(x),const(y),0);
        result(x,y) = logProbX(end);
    end
end


% plot the result
[M,C]=meshgrid(log10(minvar),log10(const));
mesh(M,C,result);
x1 = xlabel('Minvar');
x2 = ylabel('Const');
x3 = zlabel('LogProbability');
set(x1,'Rotation',27);
set(x2,'Rotation',-30);
hold on
plot3(M,C,result,'x','MarkerSize',3);
set(gca,'xtick',log10([1e-7,1e-5,1e-3,0.1,10,100]),'xticklabel',{'1e-7','1e-5','1e-3','0.1','10','100'})
set(gca,'ytick',log10([1e-7,1e-5,1e-3,0.1,10,100]),'yticklabel',{'1e-7','1e-5','1e-3','0.1','10','100'})


% find the best parametr
[rows,cols] = find(result==max(max(result)));
minvar(rows)
const(cols)
```



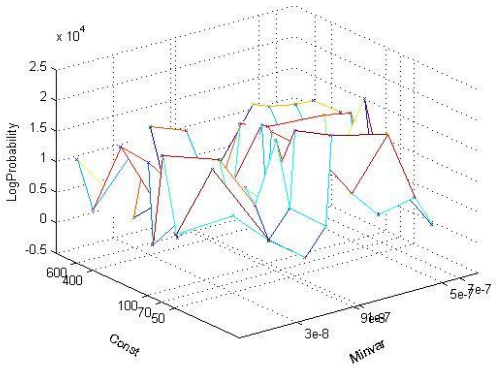
2s result of Large-scale search

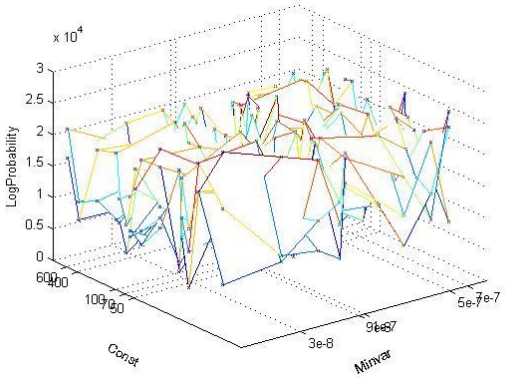


3s result of Large-scale search

	2s	3s
Best minVary	1.0000e-07	1.0000e-07
Best RandConst	100	10
Best log probability	18276	22871

- Change the search scope to the neighborhood of the optimal parameters for detailed search:





	2s	3s
Best minVary	1.0000e-08	1.0000e-08
Best RandConst	200	2
Best log probability	22916	29637

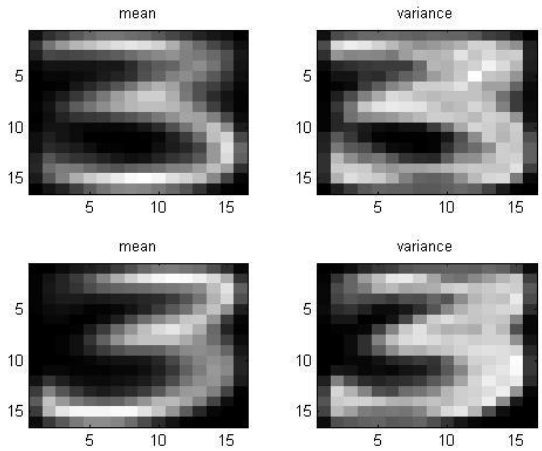
Analysis:

1) The optimal value of minVary always tends to be 0. Generally speaking, the smaller the value, the better the result. Considering that training is performed on the set of same number (2, 3), the data itself has a high similarity. Therefore, a better training model can be obtained when the model variance is smaller; but this also leads to the overfitting and a reduction of generalization ability;
2) The optimal RandConst is very volatile. RandConst is often different (and the difference is large) in different trainings. It can also be seen from the pre-experimental image that RandConst has little effect on the results. Since the

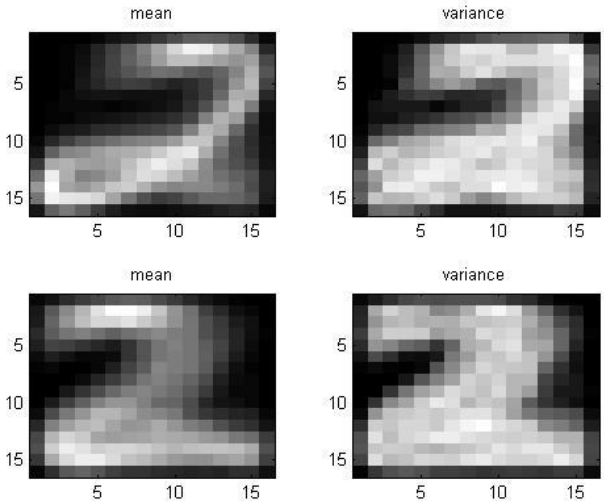
RandConst mainly influences the mean value and the occurrence probability of each cluster, also take the original training results into consideration, we can conclude that no matter how should we initialize the mean value and the occurrence probability , the final difference between the two distributions will not be too large;

- Visualization result（asststant_III.m）

Try to initialize the probability of each class (two types) as 0.5. From the EM algorithm, this operation is equivalent to setting randConst as a larger value:



	Model I	Model II
Proportion	0.3633	0.6367



	Model I	Model II
Proportion	0.3800	0.6200

Final training result as told above

Analysis：

1) From the mean vectors, both the two models of each set (2/3) can reflect the characteristics of the clustered numbers. This phenomenon explains why the mean vectors of the two models are closer to each other. But there are some

certain differences in the visualization results of the two models' mean vectors, such as the inclination, stroke width, etc.;

2) The overall form of the Variance vector is similar to the corresponding mean vector. Put another way, data in the same model concentrate to the model mean;
3) The recognition ability of the mean /variance matrix of 3 is higher than that of 2. To explain, for one thing, the training of 2 uses a larger minvar; for another thing, the data characteristics also attribute to this phenomenon. This also explains why the clustering result of 3 is better than that of 2.

2.4

- Modify mogEm.m to mogEm_II.m

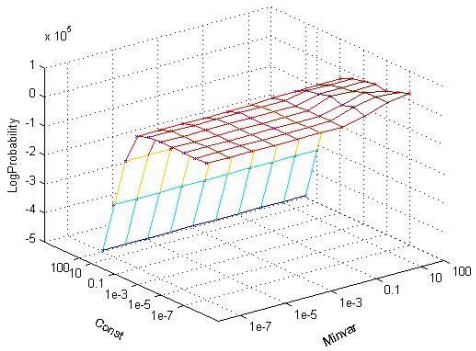
```
%-------------------- Modify the code here -------------------------------
% Change the random initialization with k-means algorithm, use 5
% iterations.
mu = kmeans(x, K, 5);
%-------------------------------------------------------------------------
```

- First use the original initialization method for training, and the parameter search method is similar to 2.3
- Result of the large-scale search

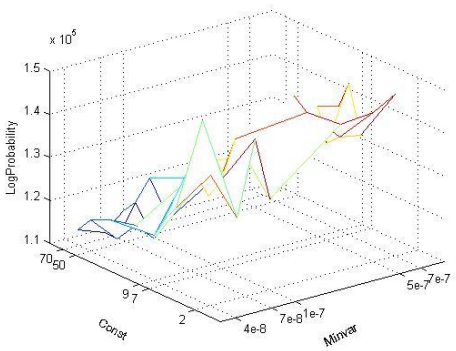
	Train
Best minVary	1.0000e-07
Best RandConst	10
Best log probability	138330
Time Consuming	606.0560s

Detailed search:

	Train
Best minVary	6.0000e-08
Best RandConst	5
Best log probability	148100
Time Consuming	237.7930



Result of large-scale search



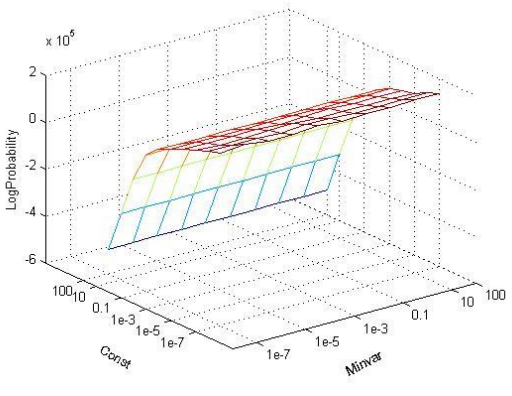
Detailed search

- Use k-means for initialization
- Result of large-scale search

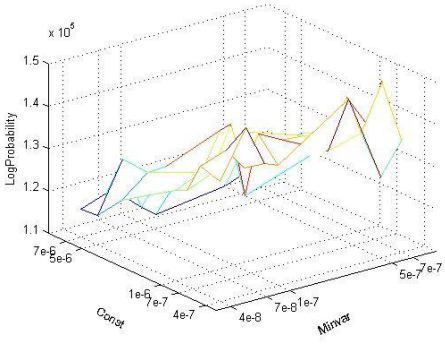
	Train
Best minVary	1.0000e-07
Best RandConst	1.0000e-06
Best log probability	143410
Time Consuming	639.8660

Detailed search:

	Train
Best minVary	4.0000e-08
Best RandConst	3.0000e-06
Best log probability	149600
Time Consuming	260.34

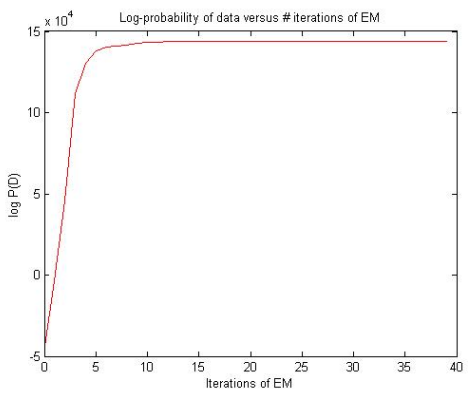


Result of large-scale search

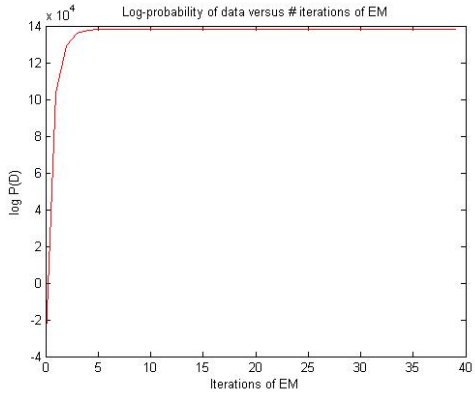


Detailed search

- Compare the convergence speed of these two methods



Original method



k-means

Analysis:

1) Since the iterations of k-means initialization is only 5 times, and the k-means

method is only used in the initialization phase, the difference in time (speed) between the two methods is not large, but the k-means method is slightly more time-consuming; However, it can be expected that as the number of iterations of the k-means method increases, the time cost of k-means initialization will definitely increase;

2) From the final log-prob analysis, the results of the two methods is not much different, but since the use of k-means initialization is equivalent to introducing additional information about the data distribution to the model, the final result of k-means is better;
3) From another perspective, the convergence speed of k-means initialization is significantly better than that of the original model. This is because the k-means method will introduce information about the data (information of the model mean) to the model, which is equivalent to Pre-training;

2.5 Classification with MoGs

According to the Bayesian formula:

$$P(i|x) = \frac{P(i,x)}{P(x)} = \frac{P(i) \cdot P(i,x)}{\Sigma_j P(j) \cdot P(x|j)}$$

Because sampling amount of 2s and 3s dataset are the same:

$$P(i|x) = \frac{P(i,x)}{\Sigma_j P(x|j)}$$

- Implement the code according to the formula (initialize with k-means method) (run_q4.m)

Taking the over-fitting problem and the previous experimental results into account, I decided to set minvar, RandConst as:

	Minvar	RandConst
2s	1e-7	1e-6
3s	1e-7	1e-4

The average error rate after training and classifying each model 20 times:

```
% Train a MoG model with K components for digit 2
  [Pi,mu2,vary2,logProbX] = mogEM_II(train2,K,30,1e-7,1e-6,0);
```

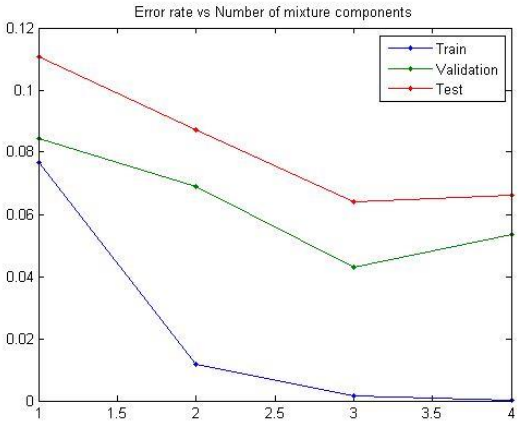
```
% Train a MoG model with K components for digit 3
[Pi,mu3,vary3,logProbX] = mogEM_II(train3,K,30,1e-7,1e-4,0);
```

```
% Caculate the probability P(d=1|x) and P(d=2|x),
        % classify examples, and compute the error rate
        % Hints: you may want to use mogLogProb function
        %-------------------- Add your code here --------------------------------
         % Do classification on train
        for i = 1:size(train,2)
```

```
            logProb2 = mogLogProb(P2,mu2,vary2,train(:,i));
            logProb3 = mogLogProb(P3,mu3,vary3,train(:,i));
            predict(i) = logProb3 > logProb2;    % do classification
        end
        errorTrain(k) = errorTrain(k) + sum(predict~= target_train)/size(train,2);


        % Do classification on validation
        predict = [];
        for i = 1:size(valid,2)
            logProb2 = mogLogProb(P2,mu2,vary2,valid(:,i));
            logProb3 = mogLogProb(P3,mu3,vary3,valid(:,i));
            predict(i) = logProb3 > logProb2;    % do classification
        end
        errorValidation(k) = errorValidation(k) + sum(predict~= target_valid)/size(valid,2);


        predict = [];
        % Do classification on test
        for i = 1:size(test,2)
            logProb2 = mogLogProb(P2,mu2,vary2,test(:,i));
            logProb3 = mogLogProb(P3,mu3,vary3,test(:,i));
            predict(i) = logProb3 > logProb2;    % do classification
        end
        errorTest(k) = errorTest(k) + sum(predict~= target_test)/size(test,2);


    end
end
```

```
% Plot the error rate
%-------------------- Add your code here --------------------------------
 % Calculate average error rate
errorTrain = errorTrain/20;
errorValidation = errorValidation/20;
errorTest = errorTest/20;


plot(1:4,errorTrain,'.-',1:4,errorValidation,'.-',1:4,errorTest,'.-');
legend('Train','Validation','Test')
title('Error rate vs Number of mixture components')
```

Classification result:



Answering questions：

1) As mentioned in the title, the training error has a decreasing trend when the number of clusters increases. This is because the increase of the cluster introduces more gaussian components, which introduces more parameters and increases the complexity of the model, resulting in the fitting result becomes better;
2) At the start stage of Cluster increasing, the classification results on both validation and test set are also better than before. However, when the number of Cluster is too large, the over-fitting phenomenon occurs, and classification results on both validation and test sets became worse;
3) Because the classification task needs to be performed on the unknown data set, it is necessary to refer to the classification result on the validation and test. From the above experiments, to maintain the model accuracy as well as to prevent over-fitting, we can set the number of clusters as 15.