Pattern Recognition

Programming Project 1

Yongqing Zheng, Financial Math

I. Executive Summary

Three of methods of pattern classification: maximum-likelihood estimation, Parzen Window and k-nearest neighbor method. Then using these classifier to classify dataset.

II. Statement of the Problem

Using training dataset to be the sample, then classify the test dataset and verify the accuracy of classifier. For Parzen window and Knn methods, using leave-one-out method and training dataset to choose proper parameters. For maximum-likelihood and Parzen window, using Bayesian decision theory to classify three datasets.

III. Description of the Statistics

1. Bayesian classifier based on Maximum-Likelihood estimation

We assume that data approximately satisfy Gaussian distribution. Thus, according to the multi-dimension Gaussian distribution:

$$p(x) = \frac{1}{(2\pi^{\frac{d}{2}}|\sum|^{\frac{1}{2}})} \exp\left[-\frac{1}{2}(x-\mu)^t \sum^{-1}(x-\mu)\right]$$

We have to obtain the two parameters- mean (μ) and covariance ($\sum$).

The maximum-likelihood estimation of mean and covariance for multivariate Gaussian function is:

$$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} x_k$$

$$\hat{\sum} = \frac{1}{n}\sum_{k=1}^{n}(x_k - \hat{\mu})\,(x_k - \hat{\mu})^t$$

2. Bayesian classifier based on Parzen window estimation

Parzen-window is a classic and performance-driven non-parameter method for estimating densities. The basic idea is to use the average density of points within a hypercube to interpret the overall density function. General speaking, if we assume x is an arbitrary point in the space, N is the total sample number, to predict the $p_N(x)$, which is the density at the location of x, we made a hypercube with length of h, and the volume is $V_N = h_N^d$, we can compute the

samples that fall into the hypercube and then get the predicted density $p_N(x)$. To be more specific, the steps are as following:

a) window function

We choose Gaussian function to be the kernel function, because it makes the predicted function more smooth, and only have one parameter:

$$\varphi(u) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2} u^2)$$

and the window function would be:

$$\varphi(\frac{x - x_i}{h_N}) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x - x_i}{h_N}\right)^2\right]$$

h is the width of windows, we need some experiment to choose the optimal h value.

b) condition probability

$$p_n(x \mid w_i) = \frac{1}{n_i} \sum_{j=1}^{n_i} \frac{1}{V_n} \varphi\left(\frac{x - x_i}{h}\right)$$

3. Basic Knn rule

Letting the dataset $D^n = \{x_1, \ldots, x_n\}$ denote a set of n labeled classes and letting $x' \in D^n$ be the class nearest to the test point **x**. Then the *nearest-neighbor rule* for classifying x is to assign it the label associated with x'.

4. Bayesian decision classification theory

Classify the test sample according to Bayesian criterion, the choose the largest $p_n(\omega_i|x)$

$$p_n(\omega_i \mid x) = \frac{P(x \mid \omega_i) \cdot P(\omega_i)}{\sum_{i=1}^{n} P(x \mid \omega_i) \cdot P(\omega_i)}$$

IV. Description of the Algorithm and Implementation

1. Bayesian classifier based on Maximum-Likelihood estimation

a) Calculating the mean and variation matrix of training dataset by using function mean() and cov().

b) Then using mvnpdf() function to calculate the multi-dimension probability density function of test dataset.
c) Choose the maximum pdf of each class as their class.
d) For handwritten dataset, the covariance matrix is invertible, so we should calculate the

$$\boldsymbol{\Sigma}(\beta) = (1-\beta)\boldsymbol{\Sigma} + \beta\mathbf{I}$$

2. Bayesian classifier based on Parzen window estimation
a) Using normpdf() function to calculate window function of each data of dataset
b) Using sum() to calculate the sum of window function.
c) Calculating the condition probability
d) Using leave-one-out to find the optimal h_N value.
e) Using test dataset to verify.

3. Basic Knn rule
a) Calculating the Euclidean distance of each x and x_i, by using norm() function.
b) Sort the distance, by using sort() function.
c) Choosing the k minimum number of distance, find their label
d) Using leave-one-out method to find the optimal n value.
e) Using test dataset to verify.

4. Data normalization

For data base of iris dataset and UCI wine dataset, we can find that the variance between each feature is very large. For this circumstance, we need to normalize the dataset and let the difference of each feature become less. For small hand written digit dataset, the feature value all in [-1,1], this is a good region for classifying, so we don't need to classify this dataset.

We use Z-score normalization to preprocess both training dataset and testing dataset, which standardizes the original dataset making mean to be 0 and variance to be 1 for each dimension:

$$z = \frac{x-\mu}{\sigma}$$

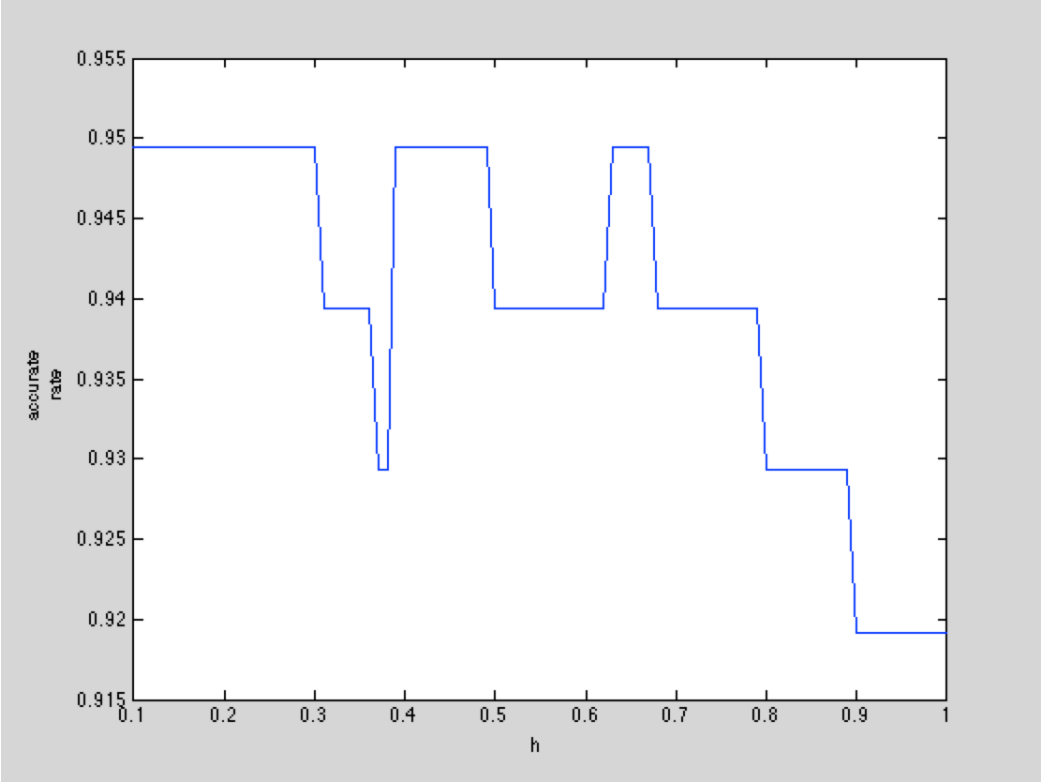
V. Description of the Experimental Design and Results

1. Bayesian classifier based on Maximum-Likelihood estimation

dataset	iris dataset	UCI wine dataset	small handwritten digit dataset
accurate rate	94.1176%	95.5056%	82.5000%

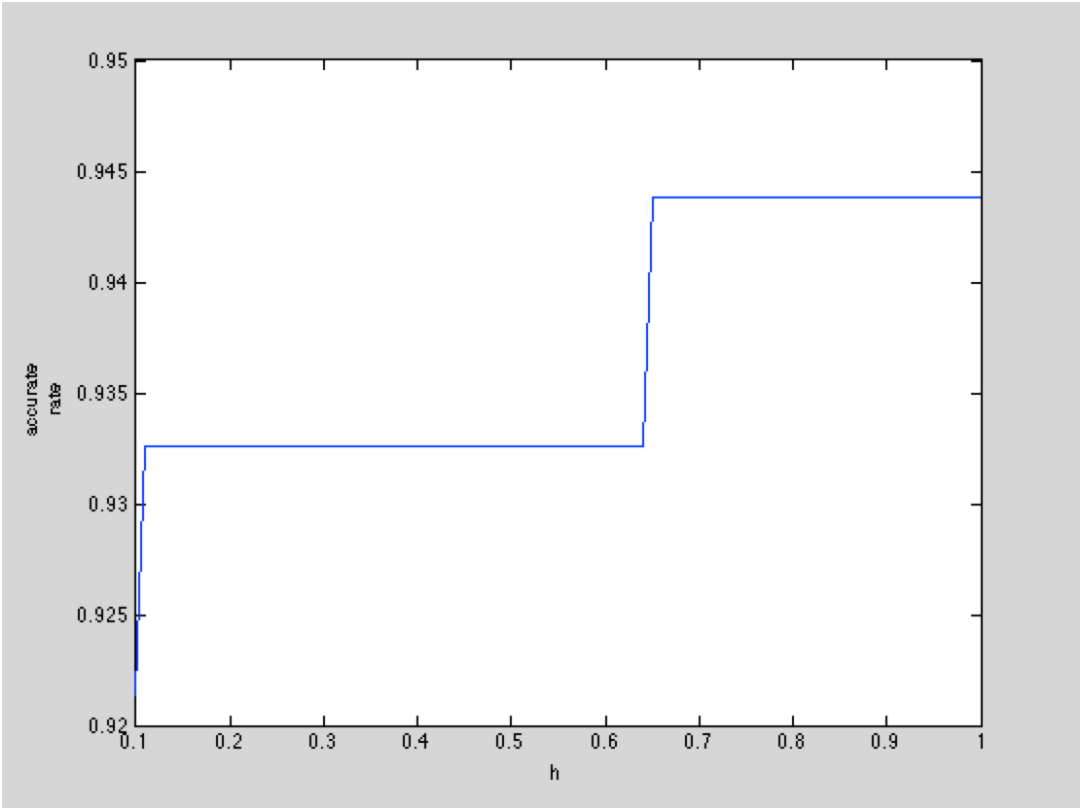
2. Bayesian classifier based on Parzen window estimation
a) choosing h

iris dataset



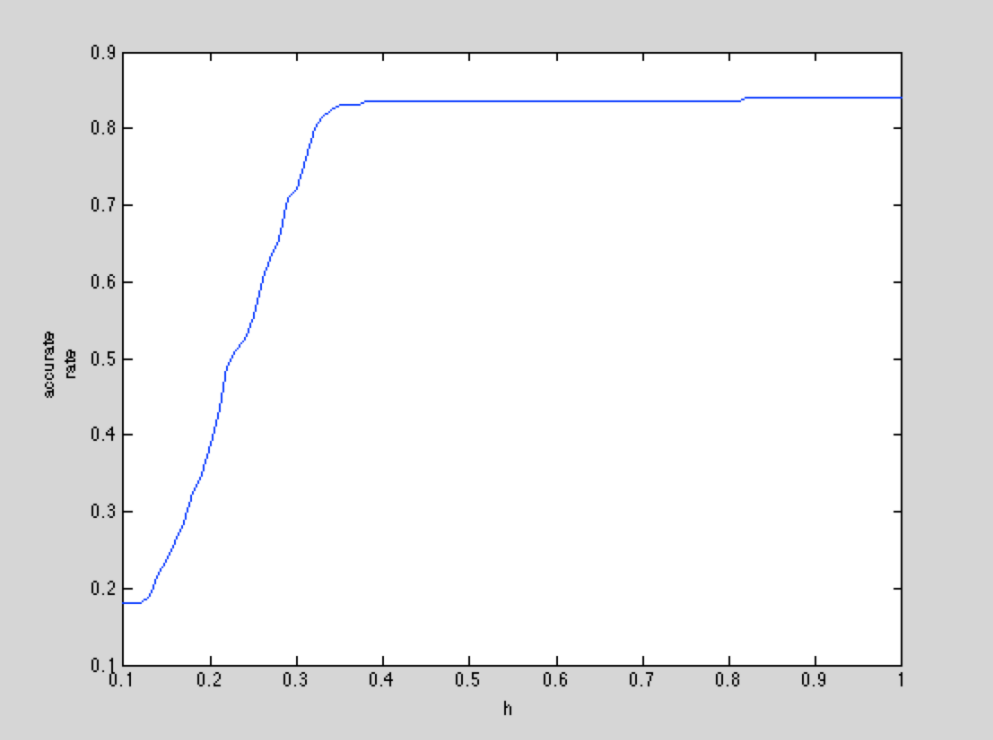
we choose h=0.4 as our window width.

UCI wine dataset



so, we choose h=0.8.

Small handwritten digit dataset



So, we choose h=0.9

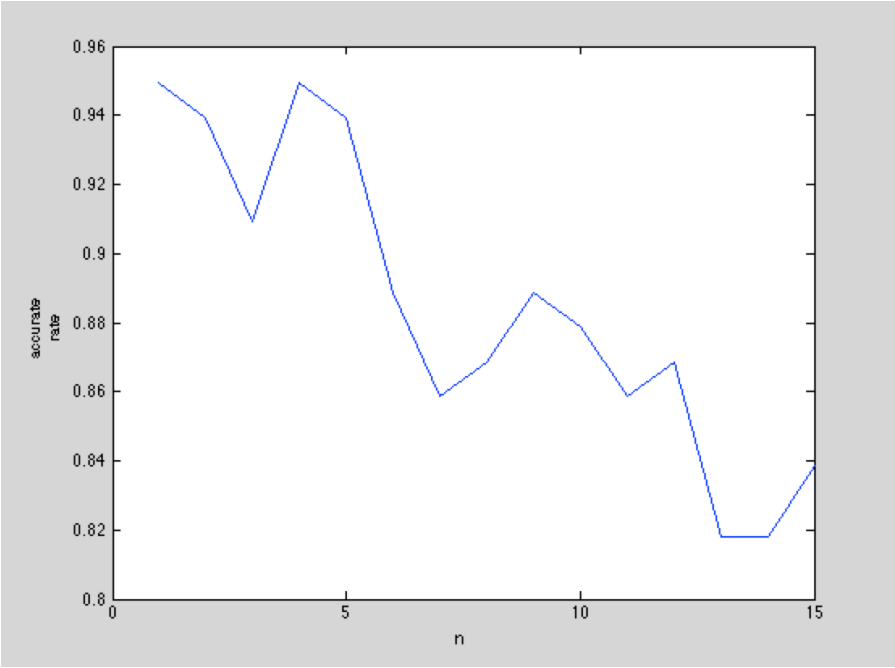
b) Test dataset result

dataset	iris dataset	UCI wine dataset	small handwritten digit dataset
accurate rate	96.0784%	94.3820%	80.5000%

3. Basic Knn rule

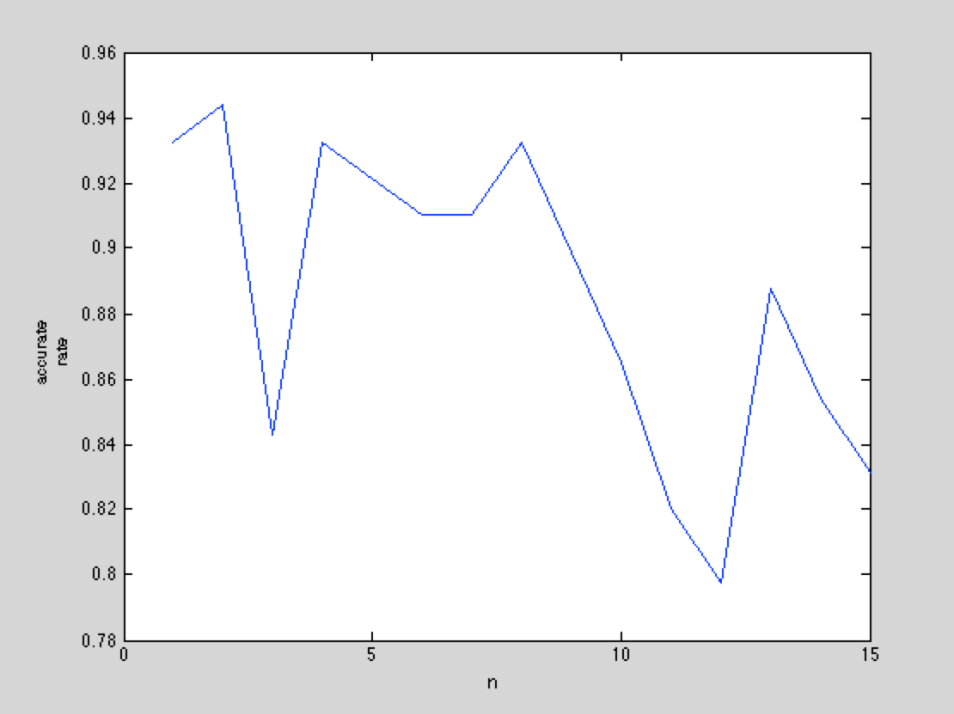
a) Choosing n

iris dataset



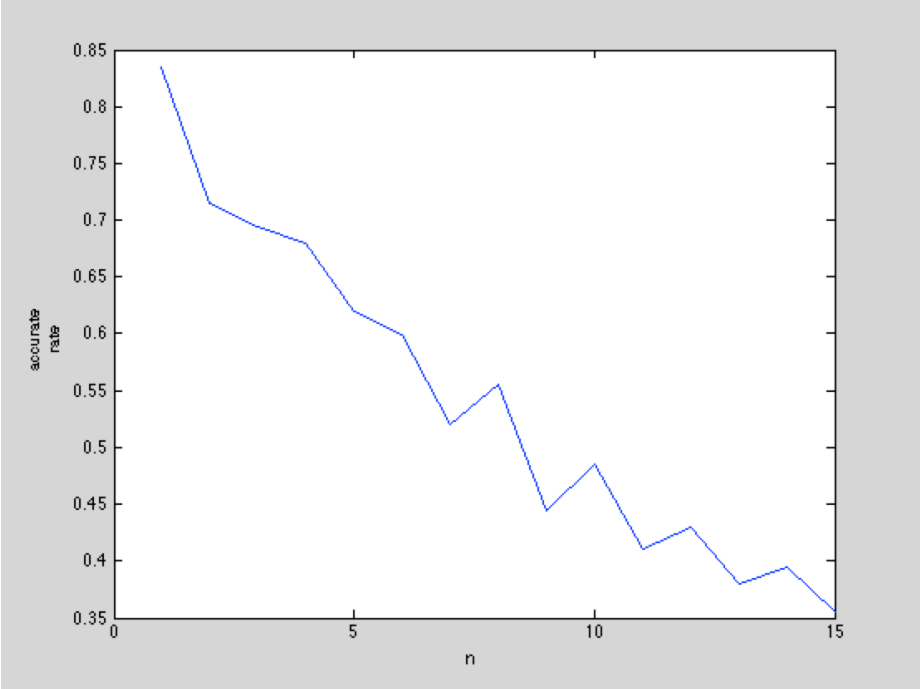
k=1 or 4, by plugging k into test data, we find that k=1 has more accurate rate.

UCI wine dataset



So we choose k=2.

small handwritten digit dataset



So, we choose k=1.

b) Test dataset result

dataset	iris dataset	UCI wine dataset	small handwritten digit dataset
accurate rate	92.1569%	92.1348%	80.5000%

VI. Conclusions

dataset	**iris dataset**	**UCI wine dataset**	**small handwritten digit dataset**
maximum likelihood	94.1176%	95.5056%	82.5000%
parzen window estimation	96.0784%	94.3820%	80.5000%
Basic Knn rule	92.1569%	92.1348%	80.5000%
optimal result	parzen	MLE	MLE

From the diagram above, we can get some conclusion.

First, we compare each column:

For iris dataset, we can find that the parzen window has the best performance. The maximum likelihood method, didn't have the best performance. That means the probability density function of this dataset, gaussian distribution, is not very appropriate. parzen window better than knn means the parzen function is selected properly.
For UCI dataset the MLE is better. The pdf is more appropriate than non parameter estimation. parzen window better than knn means the parzen function is selected properly.
For digit dataset the MLE is better. The pdf is more appropriate than non parameter estimation. parzen window better than knn means the parzen function is selected properly.

Then we compare each row:
For maximum likelihood method, UCI got the best performance. That means the distribution of UCI is most like Gaussian distribution. Digit dataset is the worst as gaussian distribution.
For Parzen window, iris dataset achieve the best performance. Because iris data set has the least feature(only 4 features) and it has more training data than UCI dataset. Digit data set has the worst performance, because it has too much 256 features only with 200 training data.
For knn method, iris dataset achieve the best performance. Because iris data set has the least feature(only 3 features) and it has more training data than UCI dataset. Digit data set has the worst performance, because it has too much 256 features only with 200 training data.

As conclusion, we can get some knowledge.
For maximum likelihood method, we should choose a good distribution of dataset.
For parzen window and Knn methods, we need more training data for more features. And the performance of parzen window method is better than Knn method, when parzen function is chosen properly.

VII. Program Listing

data pre process

%iris_process.m

```
m=3;
n=3;
d=4;
N=51;
T=99;

train=cell(1,n);
train{1,1}=iris_training(1:33,2:5);
train{1,2}=iris_training(34:66,2:5);
train{1,3}=iris_training(67:99,2:5);
train_lable=iris_training;
train_array=iris_training(:,2:5);

test=iris_test(:,2:5);
test_lable=iris_test;

norm_iris_training=(iris_training-repmat(mean(iris_training),size(iris_training,1),1))./
repmat(std(iris_training),size(iris_training,1),1);
norm_train{1,1}=norm_iris_training(1:33,2:5);
norm_train{1,2}=norm_iris_training(34:66,2:5);
norm_train{1,3}=norm_iris_training(67:99,2:5);
norm_train_array=norm_iris_training(:,2:5);

norm_iris_test=(iris_test-repmat(mean(iris_test),size(iris_test,1),1))./
repmat(std(iris_test),size(iris_test,1),1);
norm_test=norm_iris_test(:,2:5);
```

%uci_process.m

```
m=3;
n=3;
d=13;
N=89;
M=89;
T=89;

train={1:n};
train{1,1}=wine_uci_train(1:30,2:14);
train{1,2}=wine_uci_train(31:65,2:14);
train{1,3}=wine_uci_train(66:89,2:14);
train_array=wine_uci_train(:,2:14);
train_lable=wine_uci_train;

test=wine_uci_test(:,2:14);
test_lable=wine_uci_test;
```

```
norm_wine_uci_training=(wine_uci_train-repmat(mean(wine_uci_train),size(wine_uci_train,1),
1))./repmat(std(wine_uci_train),size(wine_uci_train,1),1);
norm_train{1,1}=norm_wine_uci_training(1:30,2:14);
norm_train{1,2}=norm_wine_uci_training(31:65,2:14);
norm_train{1,3}=norm_wine_uci_training(66:89,2:14);
norm_train_array=norm_wine_uci_training(:,2:14);

norm_wine_uci_test=(wine_uci_test-repmat(mean(wine_uci_test),size(wine_uci_test,1),1))./
repmat(std(wine_uci_test),size(wine_uci_test,1),1);
norm_test=norm_wine_uci_test(:,2:14);
```

%zip_process.m

```
m=10;
n=10;
d=256;
N=200;
T=200;

k=1;
train={1:n};
zip_train_small(:,1)=zip_train_small(:,1)+1;
zip_test_small(:,1)=zip_test_small(:,1)+1;
for i=1:10;
   train{1,i}=zip_train_small(k:k+19,2:257);
   k=k+20;
end
train_array=zip_train_small(:,2:257);
train_lable=zip_train_small;

test=zip_test_small(:,2:257);
test_lable=zip_test_small;

norm_zip_train_small=zip_train_small;
norm_train=cell(1,n);
k=1;
for i=1:10;
   norm_train{1,i}=norm_zip_train_small(k:k+19,2:257);
   k=k+20;
end
norm_train_array=norm_zip_train_small(:,2:257);

norm_zip_test_small=zip_test_small;
norm_test=norm_zip_test_small(:,2:257);
```

%maximum_likelihood.m

```
u=cell(1,n);
sig=cell(1,n);
ut=cell(1,n);
p=zeros(N,n);
```

```
for i=1:n
    u{1,i}=mean(train{1,i});
    sig{1,i}=cov(train{1,i});
    a=eye(d);
    sig{1,i}=0.5.*sig{1,i}+a;
end

count=0;
for i=1:N
    m=0;
    for j=1:n;
        p(i,j)=mvnpdf(test(i,:),u{1,j},sig{1,j});
    end
    [maxvalue index]=max(p(i,:));
    if index==test_lable(i,1)
        count=count+1;
    end
end
accurate=count/N;
error=1-accurate;
fprintf('the accuracy rate is %f \n.',accurate);
```

%parzen_window_parameter selection.m

```
u=zeros(1,d);
dx=zeros(1,1);
phi=zeros(1,1);
px=zeros(1,1);
count=zeros(0);
accurate=zeros(0);
m=0;

for h=0.1:0.01:1
    m=m+1;
    count(m)=0;
    accurate(m)=0;
    px=[];
    for i=1:n
        for j=1:size(norm_train{1,i},1)
            for k=1:n
                dx=repmat(norm_train{1,i}(j,:),size(norm_train{1,k},1),1)-norm_train{1,k};
                u=arrayfun(@(x)norm(dx(x,:)),1:size(dx,1))/h;
                phi=normpdf(u');
                if i==k
                    phi(j,:)=0;
                end
                px(k)=sum(phi)/(size(norm_train{1,k},1)*(h^d));
            end
            [maxvalue,w]=max(px);
```

```
            if w==i
                count(m)=count(m)+1;
            end
        end
    end
    accurate(m)=count(m)/T;
end
h=0.1:0.01:1;
plot(h,accurate);
for i=1:10
    fprintf('h=%f the accurat rate is %f. \n',h(i),accurate(i));
end
```

%parzen_window test.m

```
h=1.1;
count=0;
for i=1:N
    px=[];
    for j=1:n
        dx=repmat(norm_test(i,:),size(norm_train{1,j},1),1)-norm_train{1,j};
        u=arrayfun(@(x)norm(dx(x,:)),1:size(norm_train{1,j},1))/h;
        phi=normpdf(u',0,1);
        px(j)=sum(phi)/(size(norm_train{1,j},1)*(h^d));
    end
    [maxvalue,w]=max(px);
    if w==test_lable(i,1);
        count=count+1;
    end
end

accurate=count/N;
fprintf('the accuracy rate is %f \n.',accurate);
```

%knn_para selection.m

```
K=15;
count=zeros(1,K);

for k=1:K
    for i=1:T
        cnt=zeros(d,1);
        dx=repmat(norm_train_array(i,:),size(norm_train_array,1),1)-norm_train_array;
        dist=arrayfun(@(x)norm(dx(x,:)),1:size(norm_train_array,1));
        dist(i)=100;
        [minvalue index]=sort(dist);
        for j=1:k
            ind=train_lable(index(k),1);
            cnt(ind)=cnt(ind)+1;
        end
        [minvalue2 index2]=max(cnt);
```

```
        if index2==train_lable(i,1)
            count(k)=count(k)+1;
        end
    end
end
accurate=count/T;
error=1-accurate;
k=1:K;
plot(k,accurate)
for i=1:K
    fprintf('k= %f the accuracy rate is %f. \n',k(i),accurate(i));
end
```

%knn test.m

```
count=0;
k=1;

for i=1:N
    cnt=zeros(d,1);
    dx=repmat(norm_test(i,:),size(norm_train_array,1),1)-norm_train_array;
    dist=arrayfun(@(x)norm(dx(x,:)),1:size(norm_train_array,1));
    [minvalue index]=sort(dist);
    for j=1:k
        ind=train_lable(index(k),1);
        cnt(ind)=cnt(ind)+1;
    end
    [minvalue2 index2]=max(cnt);
    if index2==test_lable(i,1)
        count=count+1;
    end
end
accurate=count/N;
error=1-accurate;
fprintf('the accuracy rate is %f \n.',accurate);
```