COMS 4771 Machine Learning (Spring 2018)
Problem Set #1

Yufei Zhao - yz3170@columbia.edu
Minghao Li - ml4025@columbia.edu
Zhuoran Xiong - zx2214@columbia.edu

February 13, 2018

Problem 1

(i)

$$\int_{-\infty}^{+\infty} p(x|\theta)\,dx = 1 \Rightarrow c\cdot\int_{0}^{+\infty} x^2 e^{-\frac{x}{\theta}}\,dx = 1, \text{where c is a constant}$$

$$\begin{aligned}
\int_0^{+\infty} x^2 e^{-\frac{x}{\theta}}\,dx &= -\theta\int_0^{+\infty} x^2\,d(e^{-\frac{x}{\theta}}) \\
&= -\theta\cdot(x^2 e^{-\frac{x}{\theta}}\big|_0^{+\infty} - 2\int_0^{+\infty} x e^{-\frac{x}{\theta}}\,dx) \\
&= -2\theta^2\int_0^{+\infty} x\,d(e^{-\frac{x}{\theta}}) \\
&= 2\theta^2\int_0^{+\infty} e^{-\frac{x}{\theta}}\,dx \\
&= 2\theta^3
\end{aligned}$$

$$\Rightarrow p(x|\theta) = \begin{cases} \frac{1}{2\theta^3}x^2 e^{-\frac{x}{\theta}} & ,x \geqslant 0 \\ \\ 0, & ,\text{otherwise} \end{cases}$$

$$\begin{aligned}
\Rightarrow \theta_{ML} &= \arg\max_\theta \prod_{i=1}^{n} \frac{1}{2\theta^3}x^2 e^{-\frac{x}{\theta}} \\
&= \arg\max_\theta \sum_{i=1}^{n} -3\log\theta - \frac{x}{\theta}
\end{aligned}$$

By finding the critical point, we get

$$\theta_{ML} = \frac{x_1 + x_2 + \ldots + x_n}{3n}$$

(ii)

$$p(x|\theta) = \begin{cases} \frac{1}{2\theta} & ,-\theta \leqslant x \leqslant \theta \\ \\ 0 & \text{,otherwise} \end{cases}$$

$$\theta < \max_i \{|x_i|\} \Rightarrow \prod p(x|\theta) = 0$$
$$\theta \geqslant \max_i \{|x_i|\} \Rightarrow \theta_{ML} = \arg\max_\theta \frac{1}{2^n \theta^n}$$
$$\Rightarrow \theta_{ML} = \max_i \{|x_i|\}$$

(iii)

for Gaussian Distribution (μ is unknown):

$$\mu_{ML} = \frac{1}{n} \sum_i x_i$$

$$\sigma^2_{ML} = \frac{1}{n} \sum_i (x_i - \mu_{ML})^2$$

$$\begin{aligned} \mathbb{E}[\mu_{ML}] &= \frac{1}{n} \sum_i \mathbb{E}[x_i] = \frac{1}{n} n \cdot \mu = \mu \\ \mathbb{E}[\sigma^2_{ML}] &= \frac{1}{n} \sum_i \mathbb{E}[x_i^2] - 2\mathbb{E}[\mu_{ML} \cdot \frac{1}{n} \sum_i x_i] + \mathbb{E}[\mu^2_{ML}] \\ &= \mathbb{E}[x^2] - \mathbb{E}[\mu^2_{ML}] \\ &= \sigma^2 - \sigma^2_{\mu_{ML}} \end{aligned}$$

$$\text{in which } \sigma^2_{\mu_{ML}} = \mathbb{D}\big[\frac{1}{n} \sum_i x_i\big] = \frac{1}{n^2} \sum_i \mathbb{D}[x_i] = \frac{1}{n} \sigma^2$$

$$\text{so } \mathbb{E}[\sigma^2_{ML}] = \frac{n-1}{n} \sigma^2 \text{ is biased}$$

$$\hat{\sigma^2} = \frac{1}{n-1} \sum_i \big(x_i - \mu_{ML}\big)^2 \text{ is an unbiased estimation}$$

(iv)

$$\theta_{ML} = \arg\max_\theta \mathcal{L}(\theta|x) = \arg\max_\theta \mathbf{P}(x|\theta)$$

Assume that

$$g(\theta)_{ML} = g(\theta^*) \neq g(\theta_{ML})$$

then we have

$$\mathbf{P}(x|\theta^*) > \mathbf{P}(x|\theta_{ML})$$

which is contradictory with

$$\theta_{ML} = \arg\max_{\theta} \mathbf{P}(x|\theta)$$

So

$$g(\theta)_{ML} = g(\theta_{ML})$$

Let

$$\sigma = \sqrt{\frac{1}{n-1}\sum_i (x_i - \mu)^2}$$

then

$$\sigma_{ML} = g(\mu_{ML}) = \sqrt{\frac{1}{n-1}\sum_i (x_i - \mu_{ML})^2}$$

Problem 2

(i) We are using mathematic induction method to prove the affirm.

For the easiest situation, which means $D = 1$, there can only be two kinds of input as 1 or 0. So it is very simple when designing the decision tree classifier: only one node with threshold of 0.5 is needed, let's call it n_1, and its topology is like below:

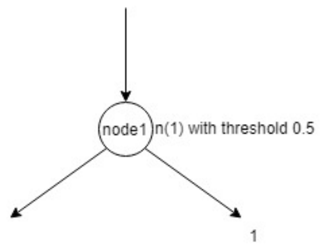

Figure 1: one node decision tree

By the one-node classification, the input is divided into two parts, 1 or 0. Next we can give the result based on the output.

Next, suppose we have successfully classified $D = N - 1$ dimension input binary data. That is shown in the picture below:

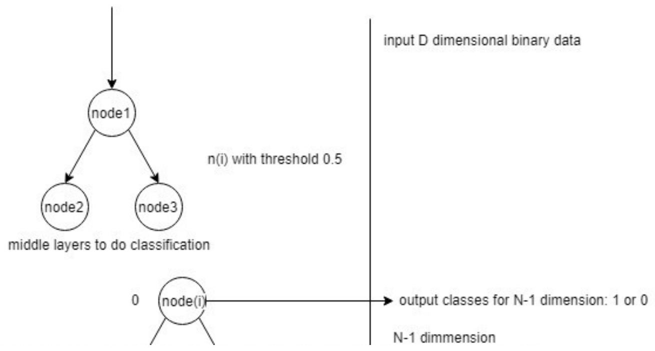
input D dimensional binary data
node1
n(i) with threshold 0.5
node2
node3
middle layers to do classification
0
node(i)
output classes for N-1 dimension: 1 or 0
N-1 dimmension

Figure 2: N-1 dimension decision tree

Some explanation: 0 beside the node(i) is the number in the $n - 1$ bit of the input $N - 1$ dimensional binary array. It means that when the last bit is 0, say the whole array can be something like :

$$XX...0 \Rightarrow 1\,\text{or}\,0$$

where there's $N - 1$ bits in this array.

Then, for $D = N$, we have the following topology:

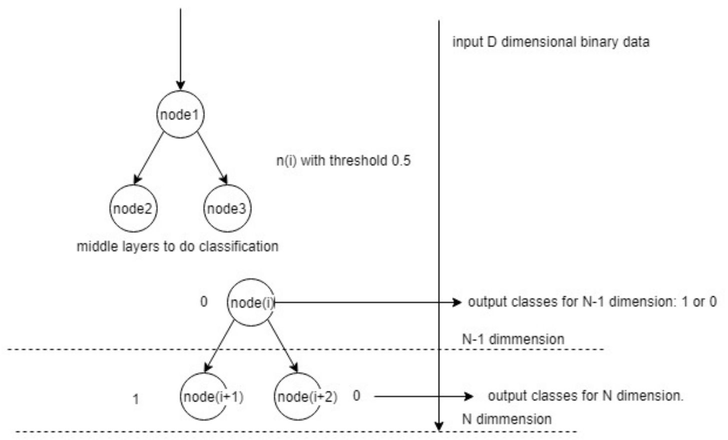
input D dimensional binary data
node1
n(i) with threshold 0.5
node2
node3
middle layers to do classification
0
node(i)
output classes for N-1 dimension: 1 or 0
N-1 dimmension
1
node(i+1)
node(i+2)
0
output classes for N dimension.
N dimmension

Figure 3: N dimension decision tree

That is to say, for each last bit in $N-1$ dimension array, we only need to add one more bit. And this one more bit will be classified using the decision tree we have discussed in the first step. Under this circumstances, we have two cases to discuss.

case 1: even after classification, the last bit 1 or 0 won't affect the result since the number

$XX...0X \Rightarrow$ the output of $XX...0$.

case 2: after the classification, the last bit does affect the outputs. For example:

$XX...01 \Rightarrow 1$ while $XX...00 => 0$

The two cases have nothing to do with the demonstration because whichever the case is, we have got the result that the N dimension array can be classified by decision tree, given it works well for $N-1$ dimension tree. I empathize the two cases to clarify that there maybe several tree structures in this scenario, it could be a full tree or not corresponding to case 2 and 1, which will help analyze the second question.

Continuing the induction process. Finally, by proving that from $D = 1$, to $D = N$, the decision tree can functionally replace the binary tree: by giving a threshold between 0 and 1,

$$g(x) = f(x) \text{ for all } x \in [0,1]^D$$

(ii) for the maximum decision tree scenario, the binary number needs the last bit to classify between each other. That is to say,

$XX...0 => 1$ while $XX...1 => 0$ In this case, the maximum height is D since we need D layers nodes to do the classification.

The function g is something like I mentioned in the last paragraph: it needs the last bit to distinguish between each other for the different label.

Problem 3

In this problem, we need to prove the optimality in continuous space $\mathcal{Y}$.

First we are given the predictor as $f(x) := \mathbb{E}[Y|X = x]$, and we also define $Q(g)$ for any predictor $g : \mathcal{X} \to \mathbb{R}$ as $Q(g) := \mathbb{E}\left[(g(x) - y)^2\right]$. Since the optimal predictor we want has the lowest $Q(g)$, we need to prove that for any g, we have $Q(f) \leq Q(g)$.

In order to prove this, we can prove that $Q(g) - Q(f) \geq 0$ for any $g(x)$ and $f(x) := \mathbb{E}[Y|X = x]$, we can prove this step by step

$$
\begin{aligned}
Q(g) - Q(f) &= \mathbb{E}_{x,y}[(g(x) - y)^2] - \mathbb{E}_{x,y}[(f(x) - y)^2] && (1)\\
&= \mathbb{E}_{x,y}[(g(x) - f(x) + f(x) - y)^2] - \mathbb{E}_{x,y}[(f(x) - y)^2] && (2)
\end{aligned}
$$

we have

$$\mathbb{E}_{x,y}[(g(x)-f(x)+f(x)-y)^2] = \mathbb{E}_{x,y}[[g(x)-f(x)]^2+2(f(x)-y)[g(x)-f(x)]]+\mathbb{E}_{x,y}[(f(x)-y)^2].$$

So,

$$Q(g) - Q(f) = \mathbb{E}_{x,y}[[g(x) - f(x)]^2 + 2(f(x) - y)[g(x) - f(x)]. \quad (3)$$

According to $\mathbb{P}(x, y) = \mathbb{P}(y|x)\mathbb{P}(x)$, we can write $\mathbb{E}_{x,y}$ as$\mathbb{E}_(x, y) = \mathbb{E}_x[\mathbb{E}_y(y|X = x)]$. Therefore, we can write it in this way

$$Q(g) - Q(f) = \mathbb{E}_x[\mathbb{E}_y[[g(x) - f(x)]^2 + 2(f(x) - y)[g(x) - f(x)]]|X = x] \quad (4)$$

If we can prove for each fixed x we have $\mathbb{E}_y[[g(x) - f(x)]^2 + 2(f(x) - y)[g(x) - f(x)]] \geq 0$, then the expect value of this must greater or equal to zero.

$$\mathbb{E}_y[[g(x) - f(x)]^2 + 2(f(x) - y)[(g(x) - f(x)]] = [g(x) - f(x)]^2 + 2\mathbb{E}_y[(f(x) - y)[g(x) - f(x)]]$$

The first part $[g(x) - f(x)]^2$ must ≥ 0, so what we need to compute is the value of $\mathbb{E}_y[(f(x) - y)[g(x) - f(x)]]$. Here we have $f(x) := \mathbb{E}[Y|X = x]$ and x is what ever value we can choose, so $f(x) := \mathbb{E}[Y]$. Plug in this into $\mathbb{E}_y[(f(x) - y)[g(x) - f(x)]]$ and we can get

$$
\begin{aligned}
\mathbb{E}_y[(f(x) - y)[g(x) - f(x)]] &= \mathbb{E}_y[(\mathbb{E}[Y] - y)[g(x) - f(x)]]\\
&= [g(x) - f(x)](\mathbb{E}[Y] - E_y[y])\\
&= [g(x) - f(x)](\mathbb{E}[Y] - E[Y])\\
&= 0
\end{aligned}
$$

So we prove that $\mathbb{E}_y[[g(x) - f(x)]^2 + 2(f(x) - y)[(g(x) - f(x)]] \geq 0$, therefore $Q(g) - Q(f)$. So $f(x) := \mathbb{E}[Y|X = x]$ is the optimal predictor with respect to Q for continuous output space.

Problem 4

(i) Firstly, to prove the matrix M is symmetric:

$$M^T = AA^T = A^T A = M$$

So M is symmetric.

Secondly, assign a vector $x \in R^{n\times d}$, then using the definition proof of positive semi-definite matrix:

$$x^T A^T Ax = (Ax)^T Ax = ||Ax||^2 \geqslant 0$$

So we know that M is symmetric positive semi-definite.

(ii) Firstly, when N=1, which is the initial point,

$$\beta^{(1)} = \beta^{(0)} + \eta\, A^T(b - A\beta^{(0)}) = \eta\, A^T b$$

Secondly, assume that it works well when N=i, that is to say:

$$\beta^{(i)} = \eta \sum_{k=0}^{i-1} (I - \eta\, M)^k v$$

So next let's calculate when $N = i + 1$:

(Considering $\beta^{(k)} := \beta^{(k-1)} + \eta\, A^T(b - A\beta^{(k-1)})$)

$$\begin{aligned}\beta^{(i+1)} &= \beta^{(i)} + \eta\, A^T(b - A\beta^{(i)}) \\ &= \eta\, A^T b + (I - \eta\, A^T A)\beta^{(i)}\end{aligned}$$

we have already known what the $\beta^{(i)}$ is according to the second step. So we replace $\beta^{(i)}$ with its expression.

$$\begin{aligned}\beta^{(i+1)} &= \eta\, v + (I - \eta\, A^T A)\,\eta \sum_{k=0}^{i-1}(I - \eta\, M)^k v \\ &= \eta\, v + \eta \sum_{k=1}^{i}(I - \eta\, M)^k v \\ &= \eta\,(1 - \eta\, M)^0 v + \eta \sum_{k=1}^{i}(I - \eta\, M)^k v \\ &= \eta \sum_{k=0}^{i}(I - \eta\, M)^k v\end{aligned}$$

Meanwhile, when calculating $\beta^{(i+1)}$ using

$$\beta^{(i+1)} = \eta \sum_{k=0}^{i} (I - \eta\, M)^k v$$

we could also get the same result.

Finally, for any given $N = i$, $N = i + 1$ works fine. Since the first number 1 is fit, then the expression satisfies any given N.

(iii) To solve this problem, we applied an important property of eigenvalue:

If λ is the eigenvalue of matrix A, then $f(\lambda)$ is the eigenvalue of $f(A)$.

Here,

$$f(M) = \eta \sum_{k=0}^{N-1} (I - \eta\, M)^k$$

Thus, given any eigenvalue λ_i of M, we can calculate $f(\lambda_i)$ as below:

$$\begin{aligned} f(\lambda_i) &= \eta \sum_{k=0}^{N-1} (1 - \eta\, \lambda_i)^k \\ &= \eta \frac{1 - (1 - \eta\lambda_i)^N}{\eta\lambda_i} \\ &= \frac{1 - (1 - \eta\lambda_i)^N}{\lambda_i} \end{aligned}$$

So the eigenvalue of $f(M)$ is $\frac{1-(1-\eta\lambda_i)^N}{\lambda_i}$ for $i \in [1, d]$ where d is defined as the number of eigenvalues of M.

(iv)

$$\begin{aligned} ||\beta^{(N)} - \hat{\beta}||_2^2 &= ||\beta^{(N-1)} + \eta A^T(b - A\beta^{(N-1)}) - \hat{\beta}||_2^2 \\ &= ||(I - \eta\, M)\beta^{(N-1)} + \eta v - \hat{\beta}||_2^2 \\ &= ||(I - \eta M)\beta^{(N-1)} + (\eta M - I)\hat{\beta}||_2^2 \\ &= ||(I - \eta M)(\beta^{(N-1)} - \hat{\beta})||_2^2 \end{aligned}$$

Then we can view the expression $\beta^{(N)} - \hat{\beta}$ as $A^{(N)}$, the conclusion we got above can be simplified as:

$$A^{(N)} = (I - \eta M)A^{(N-1)}$$

by calculating it recursively from N all the way to 1, we can get the following equation:

$$A^{(N)} = (I - \eta M)^N(-\hat{\beta})$$

Using the definition and properties, continue the calculation as below:

$$\begin{aligned} ||\beta^{(N)} - \hat{\beta}||_2^2 &= ||(I - \eta M)^N(-\hat{\beta})||_2^2 \\ &\leqslant ||(I - \eta M)^N||_2^2 \, ||\hat{\beta}||_2^2 \end{aligned}$$

Here we got the $||\hat{\beta}||_2^2$, besides, we can simplify the first norm with the following thoughts: view the $(I - \eta M)^N$ as $(I - \eta M)(I - \eta M)$... and dispart the first norm using a lesequal relation:

$$||\beta^{(N)} - \hat{\beta}||_2^2 \leqslant ||(I - \eta M)||_2^2||(I - \eta M)||_2^2...||\hat{\beta}||_2^2$$

Through the relationship of 2-norm and eigenvalues of a matrix:

$$norm(M) \geqslant det(\lambda_{min})$$

$$\begin{aligned} ||\beta^{(N)} - \hat{\beta}||_2^2 &\leqslant (1 - \eta\lambda_{min})^2(1 - \eta\lambda_{min})^2...||\hat{\beta}||_2^2 \\ &\leqslant e^{-2\eta\lambda_{min}N}||\hat{\beta}||_2^2 \end{aligned}$$

which is the final conclusion.

Problem 5

(i)

To fit training data into Multiple Gaussian model and achieve acceptable accuracy and performance, it is necessary to make sure the covariance matrix Σ to be invertible.
After look into the training data, it is obvious that a large portion of pixels(out of 784) remain zero for all 10000 samples. These features, which mostly located at corners and margins of the images, are not informational and can cause Σ to be singular.
To get rid of these features, we followed two steps to preproccess the original data:

1. Treat each feature to be independent from others, compute its variance and pick top 200 features with largest variance.
2. For each sample, apply $x'_{ij} = \frac{x_{ij}-\mu_j}{\sigma_j}$ to normalize each feature to a standard Gaussian distribution.

Then, with the preprocessed data(10000$\times$200), we can train the classifier with MLE.

1. Split 10000 samples into training and test set.
2. Go through all training samples, count number and compute mean vector and covariance matrix for each class.
3. For each test sample, fit it into models of all classes, and predict its class with which maximum $\mathbf{P}(X|Y)\mathbf{P}(Y)$.

(ii)

To implement a k-NN classifier without k-d tree is pretty simple:

1. For each test sample, compute distances between it and all training samples. Pick k training samples with smallest distance.
2. Predict the class of test sample with the majority label among the k training samples previously picked.

This naive implementation is quite slow, comparing with the previous classifier. With a split of (9000, 1000) of training and test set, it takes up to 80 seconds to classify all test sample(around 80ms for each sample).

(iii)

The probabilistic classifier has higher accuracy over k-NN classifier, both on different training size and on different portion of training data. The following two figures shows how accuracy of these two classifier changes in terms of size and portion of training set, perspectively.

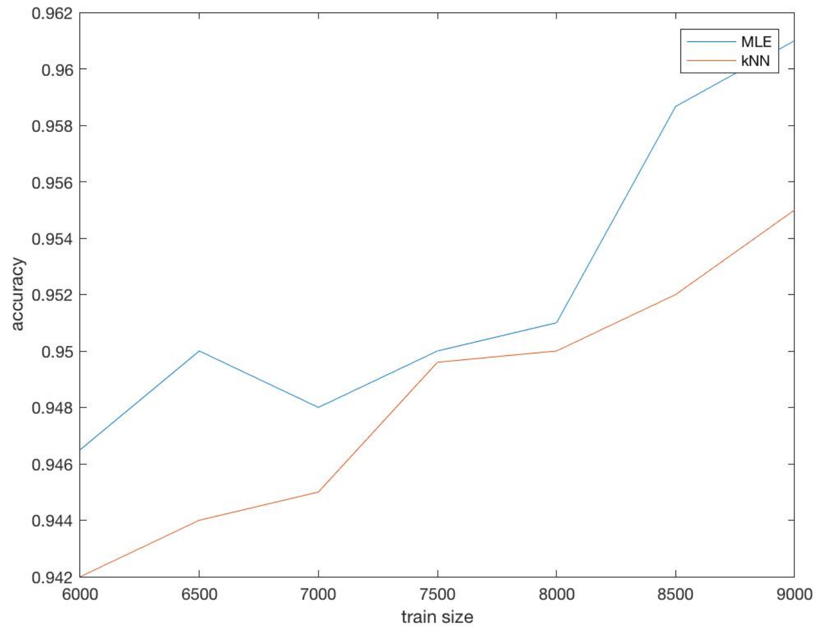

Figure 4: accuracy of MLE and kNN with different training size

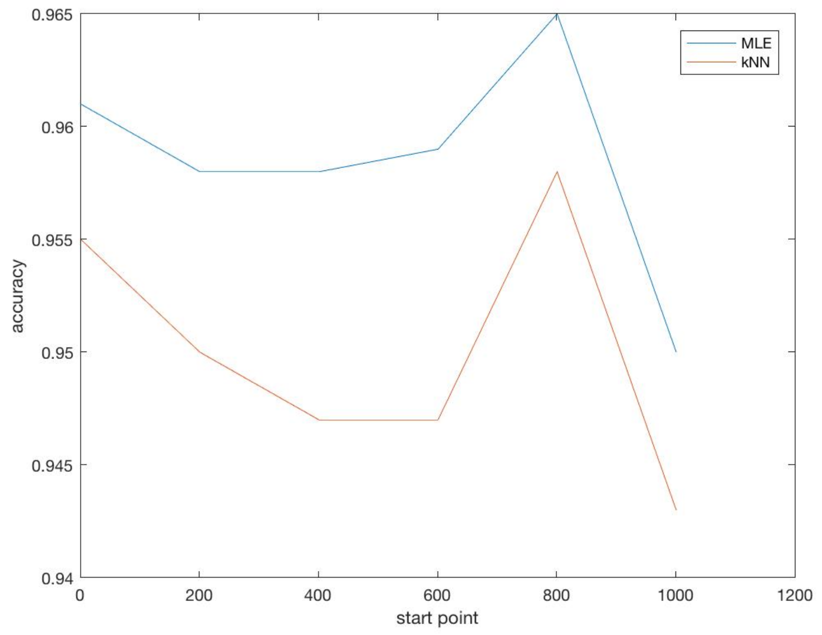

Figure 5: accuracy of MLE and kNN with different part of training data

From the perspective of speed, the probabilistic classifier is definitely better than k-NN classifier. Because for each test sample, it only computes probability for each class, while k-NN have to go through every training sample to compute their distance(when k-d tree is not implemented).

(iv)

After testing k-NN with L1, L2 and $L\infty$ on 9000 rows of training data, with k from 1 to 7, it is clear that L1 and L2 have similar performance, which is much better than $L\infty$.

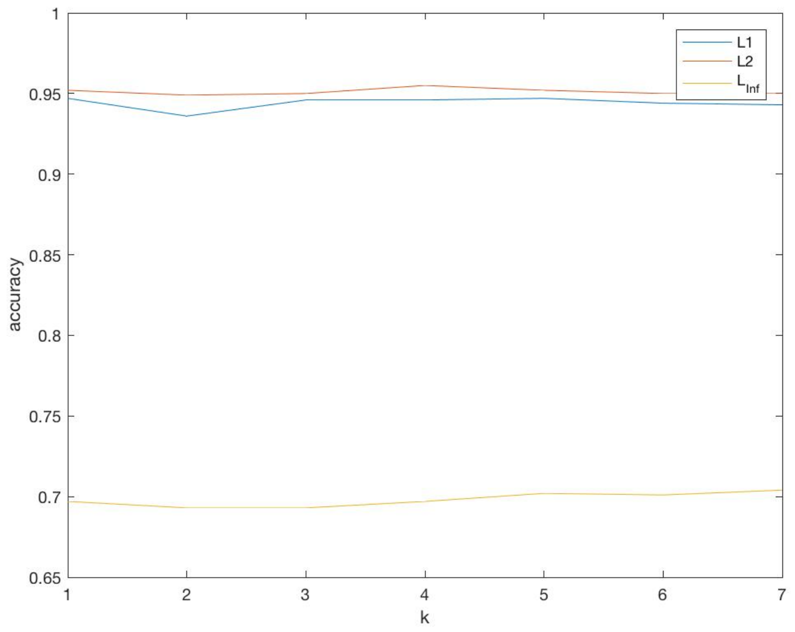

Figure 6: accuracy of different distance measurements

Problem 6

(i)

Design procedure:

1. Process the data:

(a) Load the data from the .mat file. In order to the speed up the data, this program only picks the pixels with the biggest variance;

(b) Split the data. First needs to shuffle the data. Then, this program use the idea of stratified split to make sure that the ratio of each classes in training set is the same as the ratio in the test set.

2. Write algorithm to evaluate the uncertainty. This program choose to use Gini Index as measurement. This measurement of uncertainty is used to evaluate which kind of splits is better and find the best threshold.

3. In the tree algorithm, the tree first split the whole training set and do it recursively until there is only one data or the depth exceed the maximum depth K.

4. According to the decision tree built by the training data, do predict according to the data and get the predict result.

5. In the end, compute the error rate of train and test set and plot the line.

(ii)

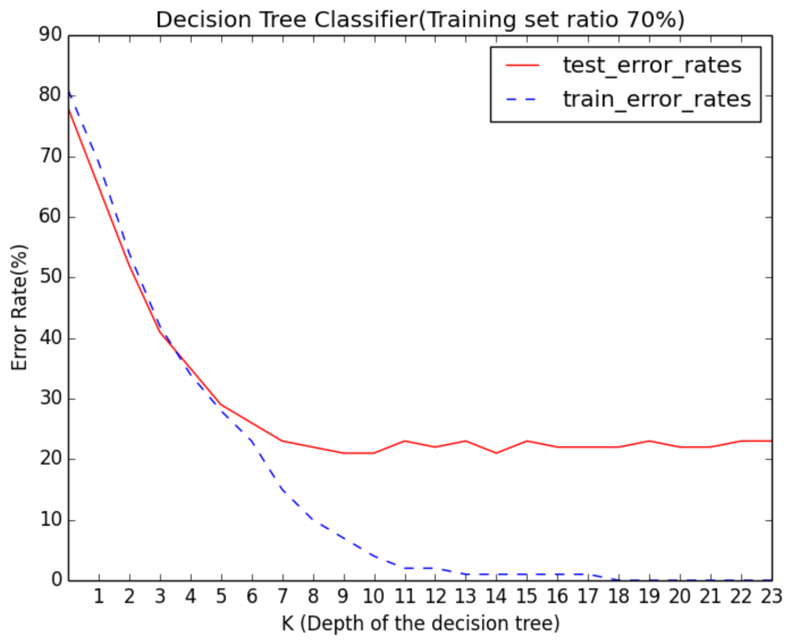

Figure 7: Trends of error rate of training set and test set(split rate=0.7)

(iii)

According to the graph, trends of error rate change a little. Because the trends of error 50% train set and 30% training set nearly have the same training error rate, just as Figure 7.2 and Figure 7.3 show. However, since we have fewer training data, the final test error is higher when the train set is smaller.

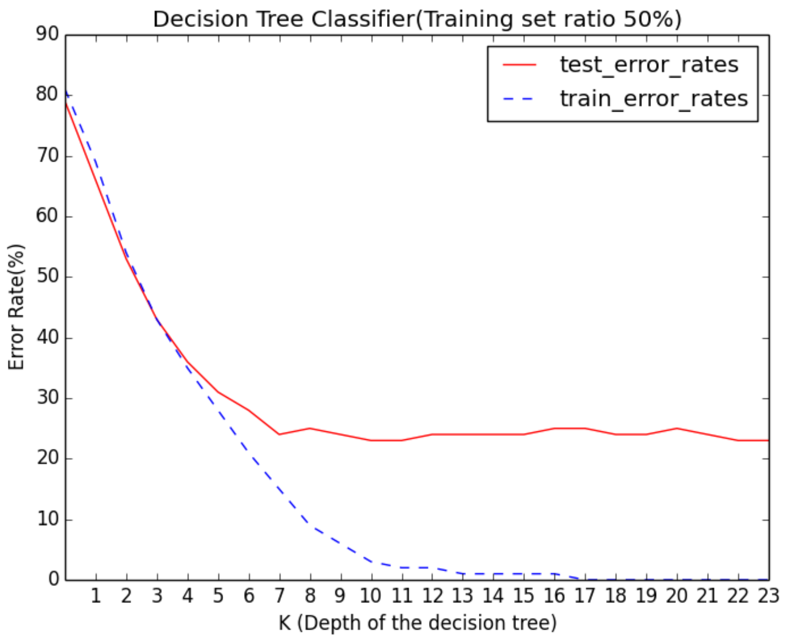

Figure 8: Trends of error rate of training set and test set(split rate=0.5)

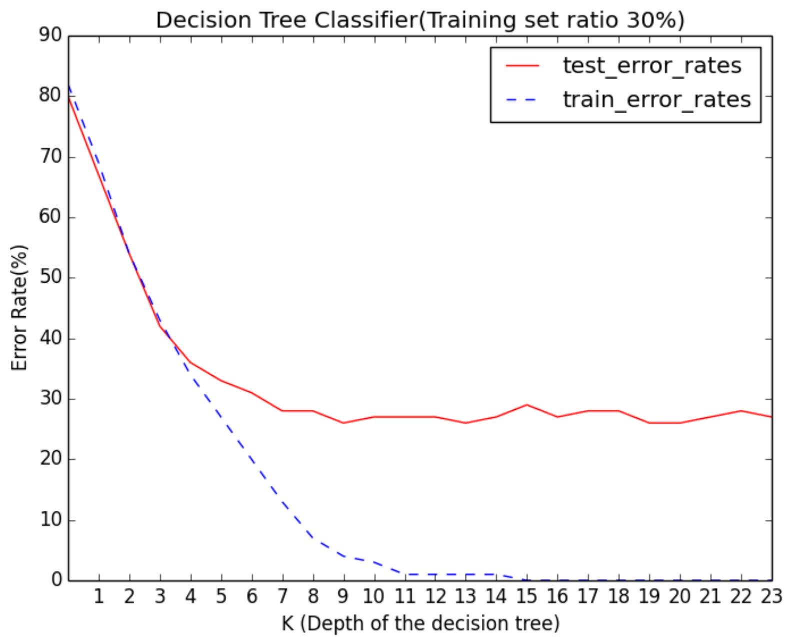

Figure 9: Trends of error rate of training set and test set(split rate=0.3)

(iv)

First, when the K is small, the distribution of the training set can not be easily found with such a shallow tree. So the model is too simple to fit the training dataset. Also, the test dataset is also be hard to fit with this distribution.
Then, then the K is growing, the tree have more modes, each nodes can separate the dataset with the best threshold right now. Then the distribution of the training set can be better represent by more complex model.
When the model become more complex, the distribution can be perfectly represent by the complex tree. However, this tree is only an representation of distribution of training dataset, the test set will have difference with then training data set, so the test set error will always contain the difference with training data set.
When the tree become too complex, which means the depth of the tree is bigger, it can perfectly fit the training set, maybe even one percent fit. Then, some random nose of the training set may also be included in the tree and this can lead to wrong prediction of the test set. Therefore, the test error may sometimes be higher when the depth of the tree become larger.

(v)

Based on the plot of the trends of testing error, in order to have lower test error rate and higher speed performance, K = 10 is a good setting for the depth of the decision tree when the ration of training data is 70%.