# COMS 4771 Homework 1

Rui Ding (rd2622), Xiaochun Ma (xm2203), Seungmin Lee (sl3254)

Sep 25, 2017

## 1 Problem 1

### 1.1 (i)

$p(x|\theta) = \theta e^{-\theta x}$ for $x >= 0$

Given n iid observations $x_1$ to $x_n$, the Likelihood function:

$$L(\theta|x_1, x_2...x_n) = p(x_1|\theta)p(x_2|\theta)....p(x_n|\theta)$$

$$L = \theta^n e^{-\theta(\sum x_i)}$$

The log likelikhood is l = log L :

$$l = nlog(\theta) - \theta(\sum x_i)$$

We want to choose a $\hat{\theta}$ such that l is maximized. First order condition:

$$\frac{dl}{d\theta} = 0$$

$$\frac{n}{\hat{\theta}} - \sum x_i = 0$$

$$\hat{\theta} = \frac{n}{\sum x_i}$$

This gives the MLE of $\theta$.

### 1.2 (ii)

$p(x|\theta) = \frac{1}{\theta}$ for $0 <= x <= \theta$

Given n iid observations $x_1$ to $x_n$, the Likelihood function:

$$L(\theta|x_1, x_2...x_n) = p(x_1|\theta)p(x_2|\theta)....p(x_n|\theta)$$

Under the condition that all $x_i$ are in $[0, \theta]$:

$$L = \frac{1}{\theta^n}$$

Otherwise this would just be zero due to the probability distribution. This condition is met when $\theta >= max(x_i)$ assuming all $x_i$ positive.(Otherwise any parameter will return a likelihood of zero due to the negative observation.) Under the condition that $\theta >= max(x_i)$, $L(x|\theta) = \frac{1}{\theta^n} <= \frac{1}{max(x_i)^n}$. The MLE is the $\hat{\theta}$ that maximizes L. So $\hat{\theta} = max(x_i)$ is the MLE of $\theta$.

## 1.3 (iii)

$p(x|\mu, \sigma^2)$ is given for all x.

Given n iid observations $x_1$ to $x_n$, the Likelihood function:

$$L(\mu, \sigma^2|x_1, x_2...x_n) = (2\pi\sigma^2)^{-\frac{n}{2}} e^{-\frac{\sum (x_i-\mu)^2}{2\sigma^2}}$$

The log likelikhood is l = log L :

$$l = -\frac{n}{2}(log(2\pi) + log(\sigma^2)) - \frac{\sum (x_i - \mu)^2}{2\sigma^2}$$

Since $\mu$ is unknown, we need to estimate as well. We want to choose a $\hat{\sigma^2}$ and a $\hat{\mu}$ such that l is maximized. First order condition:

$$\frac{dl}{d\sigma^2} = 0$$

$$\frac{dl}{d\mu} = 0$$

$$\hat{\mu} = \frac{\sum x_i}{n} = \bar{x}$$

$$-\frac{n}{2\hat{\sigma^2}} + \frac{\sum (x_i - \mu)^2}{2\hat{\sigma^2}^2} = 0$$

$$\hat{\sigma^2} = \frac{\sum (x_i - \bar{x})^2}{n}$$

This gives the MLE of $\sigma^2$, next we find its expectation, utilizing iid assumption:

$$E[\hat{\sigma^2}] = E[(x_i - \bar{x})^2] = E[x_i^2 + \bar{x}^2 - 2x_i\bar{x}] = Var(x_i) + E[x_i]^2 + Var(\bar{x}) + E[\bar{x}]^2 - 2E[x_i]E[\bar{x}]$$

$$E[\hat{\sigma^2}] = \sigma^2 + \mu^2 + \frac{\sigma^2}{n} + \mu^2 - 2\mu^2 = (1 + \frac{1}{n})\sigma^2$$

This result suggests that $\hat{\sigma^2}$ is a biased estimator of $\sigma^2$.

A modification to make the MLE consistent would be to know the mean $\mu$ ahead of time, which would be normalizing your dataset and have a mean zero.

## 1.4 (iv)

$\hat{\theta}$ is the MLE for $\theta$, so given $x_i$ data points, it is the most likely $\theta$ associated with the observations' underlying distribution. It maximizes:

$$L(\theta|x_1, x_2, ..., x_n)$$

Since $g(\theta)$ is a well-formed function of $\theta$, by definition the induced likelihood of $\gamma = g(\theta)$ is:

$$L^\star(\gamma|x_1, x_2, ..., x_n) = sup_{\gamma=g(\theta)} L(\theta|x_1, x_2, ..., x_n)$$

This suprema is taken when $\hat{\theta}$ satisfies $\hat{\gamma} = g(\hat{\theta})$. Then the maximized induced likelihood is:

$$L^\star(\hat{\gamma}|x_1, x_2, ..., x_n) = L(\hat{\theta}|x_1, x_2, ..., x_n)$$

This shows that $\hat{\gamma}_{mle} = g(\hat{\theta}_{mle})$ Therefore, in (iii) we have $\hat{\sigma^2} = \frac{\sum (x_i-\bar{x})^2}{n}$, applying function $\hat{\sigma} = \sqrt{\hat{\sigma^2}}$, we get $\hat{\sigma} = \sqrt{\frac{\sum (x_i-\bar{x})^2}{n}}$, which is the MLE of the standard deviation.

# 2 Problem 2

## 2.1 (i)

Given this is a binary classification: Error rate E = $P[f_t(X) = y_1, Y = y_2] + P[f_t(X) = y_2, Y = y_1] = P[X > t, Y = y_2] + P[X <= t, Y = y_1]$

## 2.2 (ii)

At any threshold t, we can compute

$$\frac{dE}{dt} = P[X = t, Y = y_1] - P[X = t, Y = y_2]$$

(since $\frac{d}{dt}P[X <= t] = P[X = t]$) Therefore if $\frac{dE}{dt}$ is not zero, then there is a modification we can make to t such that this reduces the error rate E. Thus, the minimized error rate appears at a optimal threshold value t which satisfies:

$$P[X = t, Y = y_1] = P[X = t, Y = y_2]$$

$$P[X = t|Y = y_1]P[Y = y_1] = P[X = t|Y = y_2]P[Y = y_2]$$

## 2.3 (iii)

First calculate Error Rate when given the distributions $P[X|Y = y_1], P[X|Y = y_2]$ are two gaussians, and that $P[Y = y_1] = P[Y = y_2] = \frac{1}{2}$. We write the first condition as:

$$P[X <= t|Y = y_1] = \Phi_1(t), P[X <= t|Y = y_2] = \Phi_2(t)$$

So error rate E(as a function of t) is :

$$E(t) = \frac{1}{2}(1 - \Phi_2(t)) + \frac{1}{2}\Phi_1(t)$$

Now consider a Naive Bayes classifier on this binary classification problem. We write the pdf of $X|Y = y_1$ as $g_1(X)$ and pdf of $X|Y = y_2$ as $g_2(X)$. Bayes Error = $P[Y = y_2, g_1(X) > g_2(X)] + P[Y = y_1, g_2(X) > g_1(X)]$, which is an averaged probability for all the observations $(x, y)$ Considering the fact that at a given x, either $P[g_1(x) > g_2(x)] = 1$ or $P[g_2(x) > g_1(x)] = 1$, so the Bayes error for each case is : $P[Y = y_2|g_1(X) > g_2(X)]$ or $P[Y = y_1|g_2(x) > g_1(x)]$, which by the contruction of naive Bayes, we know are both smaller than 0.5. Therefore a weighted average of such Bayes error rates over all n observations is smaller than 0.5.

$$BE < \frac{1}{2}$$

To achieve this rate BE using our threshold classifier $f_t(X)$, we would need:

$$\frac{1}{2}(1 - \Phi_2(t)) + \frac{1}{2}\Phi_1(t) = BE < \frac{1}{2}$$

In a setting such that the gaussian distribution $g_2(x)$ at a far left side of the real line and $g_1(x)$ is at a far right side (so they basically does not intersect), we can easily achieve E(t) = BE by picking some value t in the far left side such that $\frac{1}{2}(1 - \Phi_2(t)) \approx BE$ while $\Phi_1(t) \approx 0$.

On the other hand, if we switch the position of these two guassians, than $E(t)$ would be always greater than $\frac{1}{2}$, thus failing to achieve BE.

# 3 Problem 3

## 3.1 (i)

Notice that given a state x, f(x) is a random variable denoting the choice to be made. We have: $E[R(X, f(X))] = \int E[R(x, f(x))]p(x)dx$ For every given x, expectation is a weighted average over all possible choices of actions a, where is reward is R(x,a): $E[R(x, f(x))] = \sum R(x, a)p(a|x)$

$$E[R(X, f(X))] = \int (\sum R(x, a)p(a|x))p(x)dx$$

## 3.2 (ii)

Notice that $\sum p(a|x) = 1$ for any given x. For any x, there must be a maximum reward in that state, call it $R^\star(x)$, such that: $R(x, a) <= R^\star(x)$ for all a in A, and the equality is taken with some optimal action $a = \hat{a}$ Therefore:

$$E[R(X, f(X))] = \int (\sum R(x, a)p(a|x))p(x)dx <= \int R^\star(x)(\sum p(a|x))p(x)dx = \int R^\star(x)p(x)dx$$

This maximum expected reward is achieved only by selecting the optimal action $\hat{a}$ in every state x.

## 3.3 (iii)

No. In a suboptimal rule where the best choice $\hat{a}$ in a state x is not chosen, then among the remaining choices there will be a choice that returns a second largest reward $R_2$. Then by randomizing between the remaining choices, the expected reward will not exceed this $R_2$ values. Thus, if in a suboptimal situation, randomizing will not give you higher benefit/reward than deterministically choosing the second optimal action.

# 4 Problem 4

## 4.1 (i)

First prove that f has a bounded second derivative. For any x in R, let it be in between a infinitely small interval [a,b]. We know that for some z in [a,b] and a fixed number L:

$$|f'(a) - f'(b)| = |f''(z)(a - b)| <= L|a - b|$$

In the limit that a-¿b, —f"(z)— = —f"(x)—¡=L. Now consider $x - \hat{\eta f'(x)}$ , $f(\hat{x}) = f(x) - f'(x)\eta f'(x) + \frac{1}{2}f''(z)\eta^2 f'(x)^2$ for some z in between x and $\hat{x}$.

$$f(x) - f(\hat{x}) = \eta f'(x)^2 - \frac{1}{2}f''(z)\eta^2 f'(x)^2 >= (\eta - \frac{L}{2}\eta^2)f'(x)^2$$

The result uses Taylor reminder theorem and the fact that f" is bounded by L. If we choose $0 < \eta < \frac{2}{L}$, then $f(x) - f(\hat{x}) >= 0$ is guaranteed. In particular, if $\eta = \frac{1}{L}$, then this difference is a maximal decrease. This equality is taken only when $f'(x) = 0$.

## 4.2 (ii)

Pseudocode to return minimum value: start at some $x = x_0$, given the functional form of f, f', f" as input. While $f'(x) \neq 0$: $L = |f''(x)|$ $\eta = \frac{1}{L}$ $x = x - nf'(x)$ return f(x)

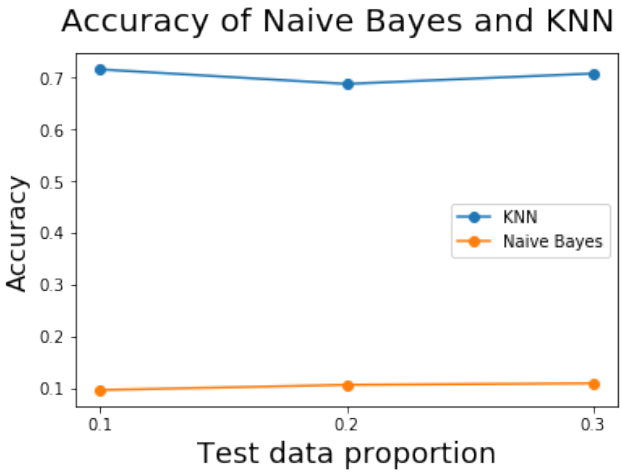


Figure 1: 5.3 Accuracy of Naive Bayes and KNN of different test data proportion

### 4.3 (iii)

See attached python code. The result for the given function is: Minimum of f appears at x = 1.25175793139, the minimum values is f(x) = 6.55283446767 First derivative at minimum point: 0.0

## 5 Problem 5

### 5.1 (i)

We implement a naive bayes classifier by setting standard deviation threshold on each column, normalize them and adding a small identity matrix to the covariance matrix($\lambda = 0.3$). See code hw5.1.py The best accuracy rate on test run with 9:1 split is 0.096. The best accuracy rate on test run with 8:2 split is 0.106. The best accuracy rate on test run with 7:3 split is 0.109. Error rate large no matter the split.

### 5.2 (ii)

We implement a naive knn classifier with k=20. The squared distance calculation between a training data point $x_i$ and test point $x$ is $||x_i||^2 + ||x||^2 - 2 < x, x_i >$ where when comparing between training points with a fixed test point we can essentially ignore $||x||^2$ part. The mode of the 20 nearest neighbor's class is returned as the predicted class label. See code hw5.2.py The accuracy rate on test run with 9:1 split is 0.715. The accuracy rate on test run with 8:2 split is 0.678. The accuracy rate on test run with 7:3 split is 0.707.

### 5.3 (iii)

According to fig.1, the accuracy of KNN is a lot better than Naive Bayes Classifier. This is because for this particular problem, probability distribution is not obvious and to fit Bayes model, Gaussian distribution is used, but is only a approximation. On the contrary, KNN is suitable for this problem because it is supposed to use a split of space for pixels to classify handwriting digits.

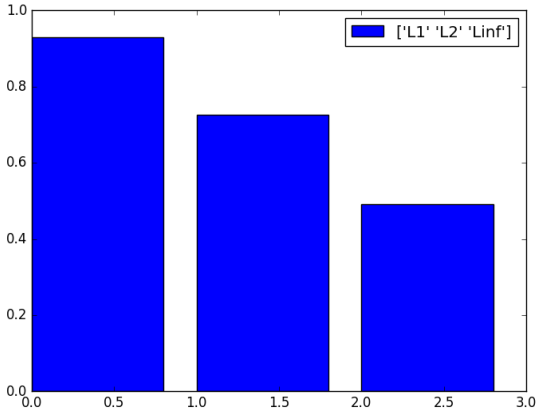


Figure 2: 5.4 Distance and Accuracy

### 5.4 (iv)

Implementing three norms on the naive knn and we tested their performance on a 9:1 train test split. The accuracy of L1 norm is 0.93, of L2 is 0.727 and of $L_\infty$ is 0.492 as the plot below demonstrates. The best metric is L1.

## 6 Problem 6

### 6.1 (i)

From the definition of f, we can see that it is essentially:

$$f = \sum_{k=1,2,..,10} \sum_{y_i=k} \sum_{d} \frac{(x_d^i - \theta_k)^2}{2}$$

For all k = 1,2,...,10:

$$\frac{\partial}{\partial \theta_k} f = \sum_{y_i=k} (D\theta_k - \sum_{d} x_d^i)$$

The gradient vector would just be:

$$\nabla_\theta f = (\frac{\partial}{\partial \theta_1} f, \frac{\partial}{\partial \theta_2} f, ....., \frac{\partial}{\partial \theta_{10}} f)$$

where all partial derivatives take the form above.

### 6.2 (ii)

See code hw6.2.py The number of iterations needed depends on the step size $\eta$(also the initial guess which we choose the zero vector). Taking $\eta$ = 1e-6,1e-7,1e-8,1e-9 needs iterations 22,358,3692,37039 respectively. The minimum value that each run converges to is f = 24297968346.4. A plot of f values for each step size is shown below.

The plot below summarizes the relation between step size and iterations to convergence.

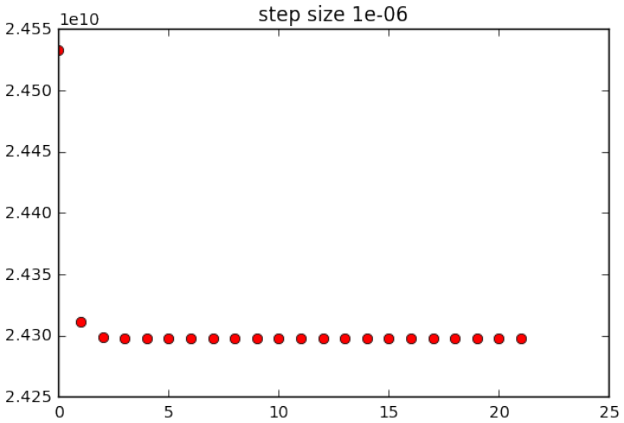


Figure 3: 6.2.1 Step size

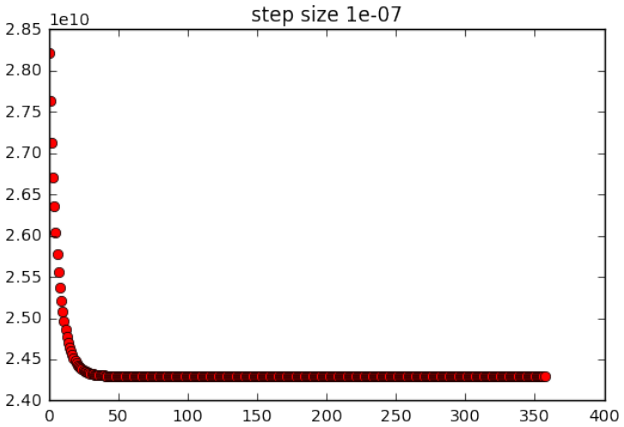


Figure 4: 6.2.2 Step size

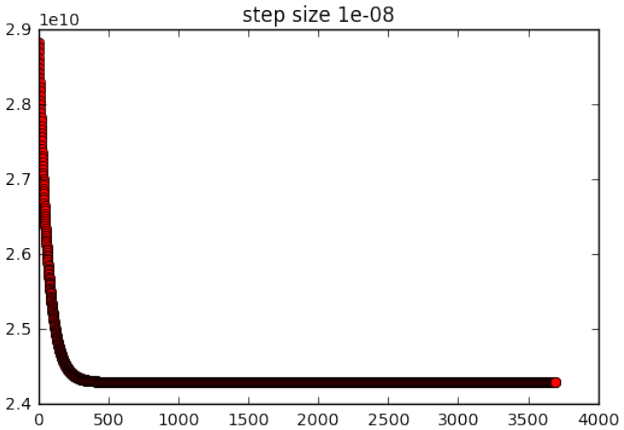


Figure 5: 6.2.3 Step size

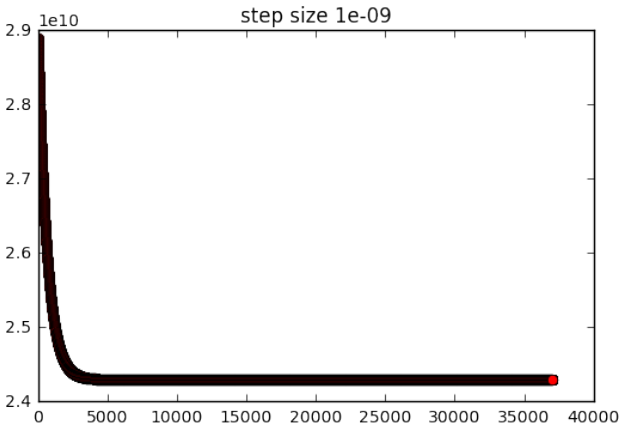


Figure 6: 6.2.4 Step size

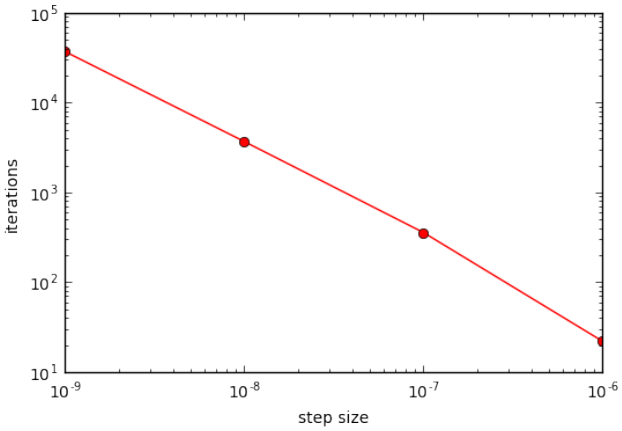


Figure 7: 6.2.5 Step size

### 6.3 (iii)

Given a particular sample $(x_i, y_i)$, let $k_i = y_i$

$$f_i = \sum_d \frac{(x_d^i - \theta_{k_i})^2}{2}$$

The first partial derivatives of $f_i$ are all zero except:

$$\frac{\partial}{\partial \theta_{k_i}} f_i = D\theta_{k_i} - \sum_d x_d^i$$

Therefore the gradient of $f_i$ has only one nonzero entry at the $k_i$ th index with value shown above.

### 6.4 (iv)

For each shuffle, we use the 10000 data points as 10000 steps of gradient descent. The convergence is very slow so we have to choose very small $\eta$ and large number of iterations to achieve convergence to the minimum f values which using the calculation in part (ii) we know is 24297968346.4. See code q6.4.py

### 6.5 (v)

The single direction gradient values are mostly centered around zero at the end of our fast algorithm. The total gradient, however, is still significantly different from zero.

### 6.6 (vi)

Our part (ii) algorithms uses precalculated values so it runs fast when calculating f values and gradient values. At 1e-9 it needs 37039 steps to converge. The fast algorithm in (iv) computes gradient faster but it have to run with small steps to achieve convergence. Even then it does not converge as closely to the true minimum as the original one. Our algorithm changes $\eta$ to be smaller as the steps run, and it takes about 50000 steps to get to a close result of about 24298000000. The algorithm cannot converge any more and it starts to wander around this value.

## 7 Problem 7

### 7.1 (i)

See python code. The hyperparameter K is maximum depth of the tree.

### 7.2 (ii)

The plots are shown in the next section. Random shuffle of the entire dataset at the beginning ensures random split.

### 7.3 (iii)

Split ratio: 7000:3000 Split ratio: 8000:2000 Split ratio: 9000:1000

The general trend is alike, with the training error decreasing almost linearly with the increasing depth of the decision tree. The test error decreases first and reaches a minimum error rate at some depth then turns upward. For three splitting between the training data and the test data using ratios 9:1,8:2,7:3, we find that the maximal depth is around 8 to 9,

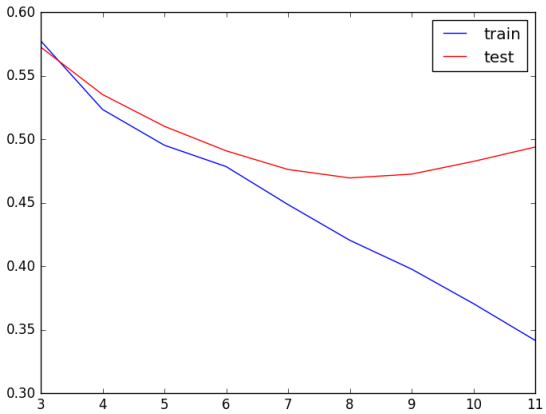


Figure 8: 7.7.3 Accuracy of train and test data

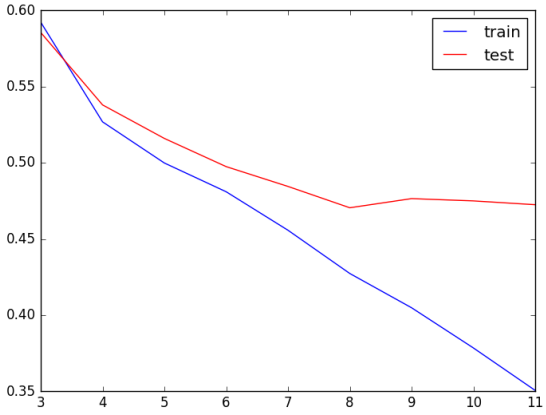


Figure 9: 7.8.2 Accuracy of train and test data

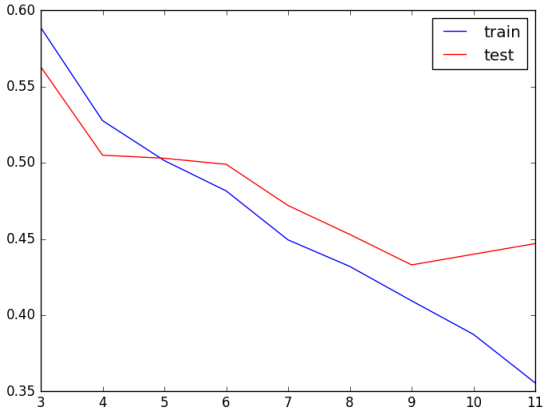


Figure 10: 7.9.1 Accuracy of train and test data

where the test error reaches a minimum value. The 9:1 split has a higher test error rate and it returns an optimal depth of 9 while the other two returns 8. The 7:3 split has the smoothest curve showing a clear minimal error rate at depth 8.

## 7.4 (iv)

The difference in training error and test error behavior with tree depth K is explained by the concept of overfitting. At the beginning as we increase K the model gets better so both errors should decrease. While the training error will necessarily decrease with K since we constructed the decision tree based on training data, the test error will reach a minimum value and no longer decrease. Because as K gets large, our model is too specified and may overfit the training data and perform less well against generalizations.

## 7.5 (v)

As shown before in section (iii), the error convergence tests show that a good max tree depth K is about 8.