Stats 315a: Statistical Learning
Problem Set 1

Dong-Bang Tsai*

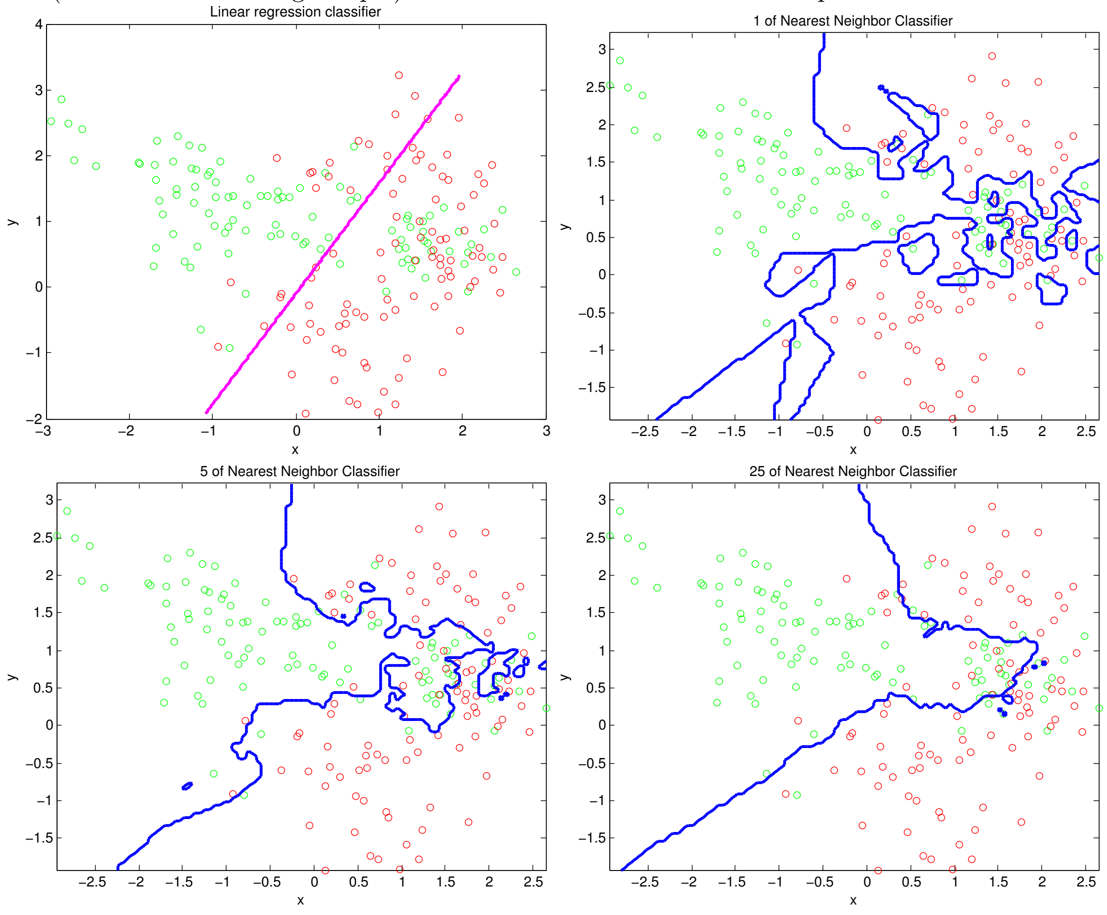
Department of Applied Physics, Stanford University, Stanford, California 94305, USA

(Dated: January 25, 2012)

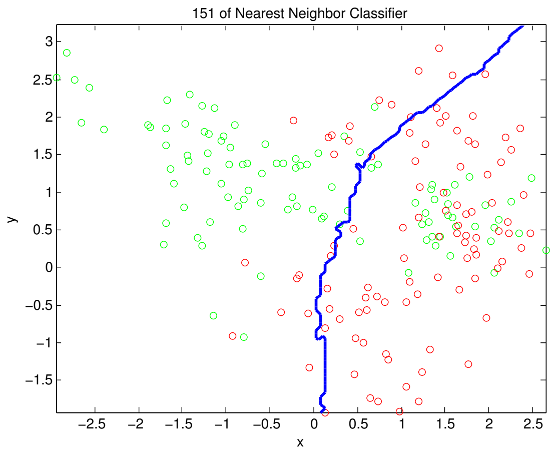
Problem 1

(a)

MATLAB is used for this homework instead of R; I already asked professor if we can use MATLAB, and he said it's okay. My teammates are Jin Chen, and Wenqiong Guo. Basically, the code will generate a training sample of size 100 for each class, as well as a test sample of 5,000 per class. Then, we use linear regression and K-nearest neighbour (trained with training sample) to do the classification on test sample.



*Electronic address: `dbtsai@stanford.edu`



```
% HW1 Q1
% Dong-Bang Tsai,Jin Chen, Wenqiong Guo
% ========================================================================
clear all;

M = 10;
mu_green = [0 1];
mu_red = [1 0];
sigma_green = [1 0; 0 1];
sigma_red = [1 0; 0 1];

% Generate M centroids from a bivariate Gaussian distribution
green_centroids = mvnrnd(mu_green,sigma_green, M); % multivariate normal distribution
red_centroids = mvnrnd(mu_red,sigma_red, M);

% Pick an index at random with probability 1/10, and then generate N(m_k,
% I/5)
n = 100; %number of points for each class in training data
m = 5000; % # of points for each class in test data

% Generate the training data
train_green = [];
train_red = [];
for i = 1:n
    rand_index_g = randi(M);% randomly pick a centroid
    rand_index_r = randi(M);
    center_green = green_centroids(rand_index_g,:);
    center_red = red_centroids(rand_index_r,:);
    train_green(i,:) = mvnrnd(center_green,sigma_green/5, 1);% multivariate normal dist
    train_red(i,:) = mvnrnd(center_red,sigma_red/5, 1);
end

% Generate the test data
test_green = [];
test_red = [];
for j = 1:m
    rand_index_g = randi(M);% randomly pick a centroid
    rand_index_r = randi(M);
    center_green = green_centroids(rand_index_g,:);
    center_red = red_centroids(rand_index_r,:);
    test_green(j,:) = mvnrnd(center_green,sigma_green/5, 1);% multivariate normal dist
    test_red(j,:) = mvnrnd(center_red,sigma_red/5, 1);
end
```

```
% Save the variables
save Data;

% Plot data points
figure(1); plot(train_green(:,1), train_green(:,2),'go'); hold on;
plot(train_red(:,1), train_red(:,2),'ro');
% Linear regression for training data
min_x = min([train_green(:,1); train_red(:,1)]);
max_x = max([train_green(:,1); train_red(:,1)]);
min_y = min([train_green(:,2); train_red(:,2)]);
max_y = max([train_green(:,2); train_red(:,2)]);
[X,Y] = meshgrid(linspace(min_x, max_x),linspace(min_y, max_y)); % just to make data in correct format for
X = X(:); Y = Y(:);
sample = [X Y];
training = [train_green; train_red];
group = [ones(n,1); zeros(n,1)];% green = 1, red = 0
[class,err,POSTERIOR,logp,coeff_linear] = classify(sample, training, group); % linear classifier
K = coeff_linear(1,2).const;
L = coeff_linear(1,2).linear;
f = @(x,y) K + [x y]*L; % linear function from classifier
h = ezplot(f,[min_x max_x min_y max_y]);
title('Linear regression classifier')
set(h, 'Color', 'm', 'linewidth', 2); xlabel('x'); ylabel('y');
hold off;

figure(2); plot(test_green(:,1), test_green(:,2),'go'); hold on;
plot(test_red(:,1), test_red(:,2),'ro'); xlabel('x'); ylabel('y');

hold off;

% Plot KNN classifier for training data
for i = 1:4
    k = [1 5 25 151];
    figure(i+2); plot(train_green(:,1), train_green(:,2),'go'); hold on;
    plot(train_red(:,1), train_red(:,2),'ro');
    title(sprintf('%d of Nearest Neighbor Classifier',k(i)));
    % Knn classifier for training data (lacks error)
    class_knn = knnclassify(sample, training, group, k(i)); % knn classifier
    contour(linspace(min_x, max_x),linspace(min_y, max_y), reshape(class_knn, 100, 100), 1, 'b', 'linewidth
    axis([min_x max_x min_y max_y]); xlabel('x'); ylabel('y');
    hold off;
end
```

(b)

```
% HW1 Q1-b
% ======================================================================
clear all;
load Data;

sample = [test_green; test_red];
training = [train_green; train_red];
group = [ones(n,1); zeros(n,1)];
k = [1 3 5 9 15 25 45 83 151];

% knn error for test data
```

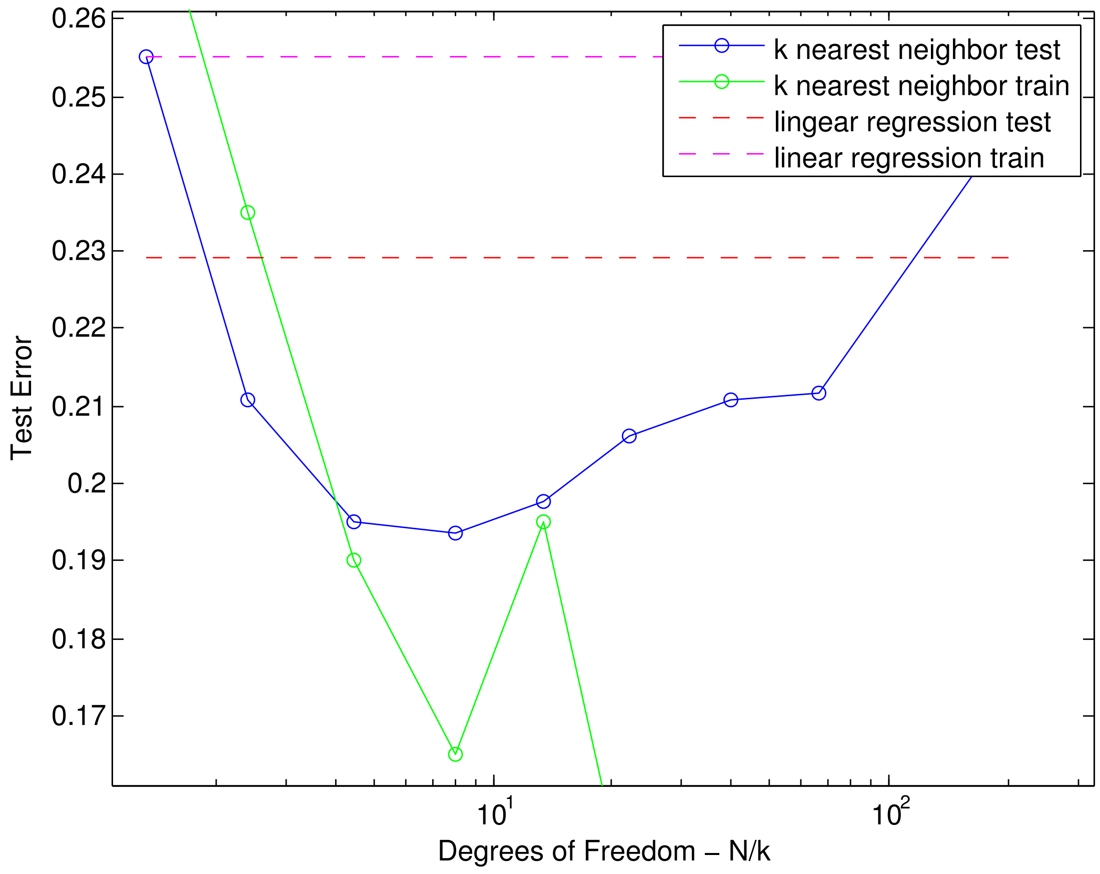
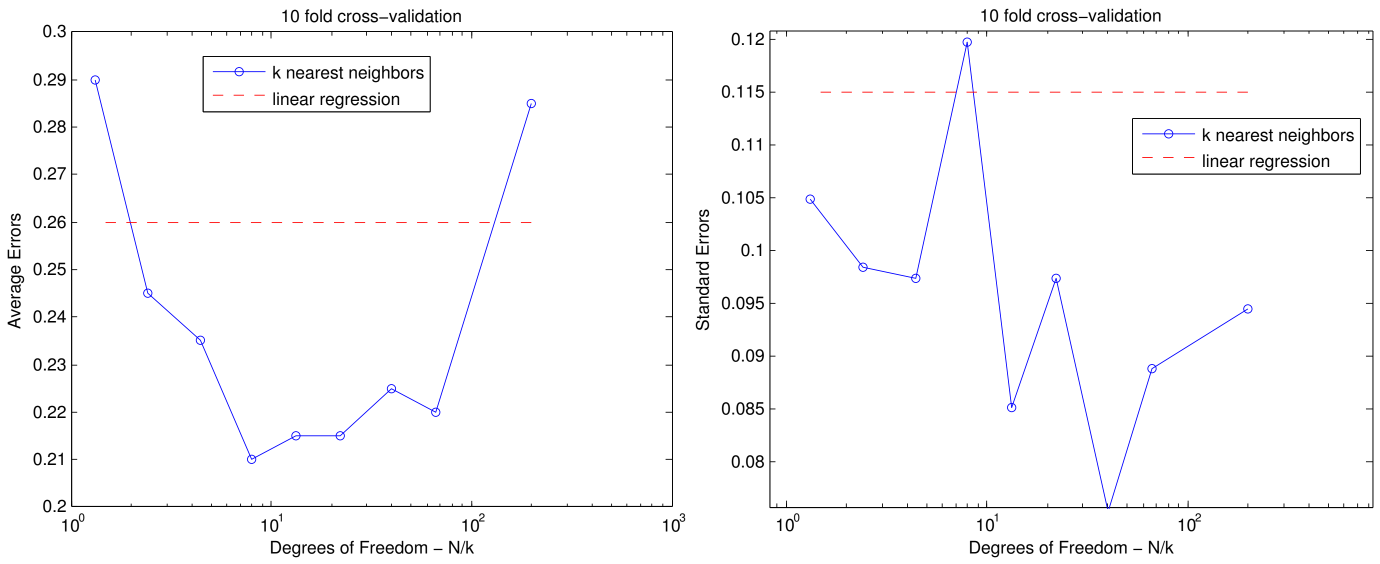


FIG. 1: Misclassification curves for the training and testing samples using k nearest neighbors method and linear regression method. This figure pretty much matches Figure 2.4 in textbook.

```
knn_error_test = [];
for i = 1:9
% knnclassify is a matlab function
    class_knn_test = knnclassify(sample, training, group, k(i));
% count the number of points that are misclassified
    num_misclassified = sum(class_knn_test ~= [ones(m,1); zeros(m,1)]);
% calculate the test error by dividing points misclassified by total points
    knn_error_test = [knn_error_test num_misclassified/(2*m)];
end
figure(7); semilogx(200./k, knn_error_test, 'b-o'); hold on;

% knn error for training data
knn_error_train = [];
for i = 1:9
    class_knn_train = knnclassify(training, training, group, k(i));
    num_misclassified = sum(class_knn_train ~= [ones(n,1); zeros(n,1)]);
    knn_error_train = [knn_error_train num_misclassified/(2*n)];
end
semilogx(200./k, knn_error_train, 'g-o'); hold on;

% linear regression error
% classify is a matlab function for linear regression
class_linear_test = classify(sample, training, group);
class_linear_train = classify(training, training, group);
```

```
% count the number of points that are misclassified
num_misclassified_test = sum(class_linear_test ~= [ones(m,1); zeros(m,1)]);
num_misclassified_train = sum(class_linear_train ~= [ones(n,1); zeros(n,1)]);
% calculate the test error by dividing points misclassified by total points
linear_error_test =  num_misclassified_test/(2*m);
linear_error_train =  num_misclassified_train/(2*n);
% linear regression test data
semilogx(200./k, linear_error_test*ones(9,1), 'r--');hold on;

% linear regression train data
semilogx(200./k, linear_error_train*ones(9,1), 'm--');  hold on;
legend('k nearest neighbor test','k nearest neighbor train','lingear regression test','linear regression tr

xlabel('Degrees of Freedom - N/k'); ylabel('Test Error');
axis([1 300 0.15 0.25]);
hold off;
```

(c)



```
% H1 Q1-c
% ====================================================================
clear all;
load Data;

[n, m] = size(train_green);
training = [train_green; train_red];
k = [1 3 5 9 15 25 45 83 151]; % k values for knn classification

knn_avg_error = []; % initialize the errors
linear_avg_error = [];
for i = 1:10 % loop over each 10% chunk of the data for 10-fold cross validation
    training_10 = [train_green(int8(0.1*(i-1)*n+1):int8(0.1*i*n),:); train_red(int8(0.1*(i-1)*n+1):int8(0.1
    training_g_90 = train_green;
    training_r_90 = train_red;
    training_g_90(int8(0.1*(i-1)*n+1):int8(0.1*i*n),:) = [];
    training_r_90(int8(0.1*(i-1)*n+1):int8(0.1*i*n),:) = [];
    training_90 = [training_g_90; training_r_90]; % the rest 90% of the data
    group = [ones(0.9*n,1); zeros(0.9*n,1)];

    % KNN classification
    knn_error = [];
    for i = 1:9
        class_knn = knnclassify(training_10, training_90, group, k(i));
        num_misclassified = sum(class_knn ~= [ones(0.1*n,1); zeros(0.1*n,1)]);
        knn_error = [knn_error num_misclassified/(2*0.1*n)];
    end
    knn_avg_error = [knn_avg_error; knn_error]; % keep a running vector of all the errors

    % Linear regression
    linear_error = [];
    class_linear = classify(training_10, training_90, group);
    num_misclassified = sum(class_linear ~= [ones(0.1*n,1); zeros(0.1*n,1)]);
    linear_error =  num_misclassified/(2*0.1*n);
    linear_avg_error = [linear_avg_error linear_error]; % keep a running vector of all the errors
end

knn_sd = std(knn_avg_error);
linear_sd = std(linear_avg_error);
knn_avg_error = mean(knn_avg_error); % average of all the errors for each fold
linear_avg_error = mean(linear_avg_error); % average of all the errors for each fold

figure(8); semilogx(200./k, knn_avg_error, 'b-o'); hold on;
plot(200./k, linear_avg_error*ones(9,1), 'r--');
xlabel('Degrees of Freedom - N/k'); ylabel('Average Errors');
title('10 fold cross-validation');
legend('k nearest neighbors','linear regression')
hold off;

figure(9); semilogx(200./k, knn_sd, 'b-o'); hold on;
plot(200./k, linear_sd*ones(9,1), 'r--');
xlabel('Degrees of Freedom - N/k'); ylabel('Standard Errors');
title('10 fold cross-validation');
legend('k nearest neighbors','linear regression')
hold off;
```

(d)

Since the training data is a mixture of several low-variance Gaussian distributions, with careful chosen k, KNN classifier will always give better performance compared with linear regression method; and this property is observed in our simulation.

As you can see, in KNN method, the error for test sample will start to decrease from 0.255 to 0.195 when $N/k = 1$ to 10, and if we try to increase the degrees of freedom, the error will increase for test sample, and it's basically over-fitted to training data; therefore, in this case, the error of predicting training data will be decrease. Ultimately, when $k = 1$, the KNN method will give zero for predicting training data.

Since in the real world, most of time, we just have small data set, we don't want to waste the data by splitting them into training data and testing data. Therefore, we can use 10-fold cross-validation to estimate and chose the parameters of the model. In this case, assuming that we only have one training data, we can use 10fold cross-validation to find that the best degrees of freedom is around $N/k = 10$, and this is agreed with the result in (a).

Problem 2 (ESL 2.4)

The squared distance from any sample point to the origin has a χ^2_p distribution with mean p; therefore, since a prediction point x_0 is drawn from this distribution, it will have a expected squared distance p from the origin.

Because $z_i = a^T x_i$, and a is independent from x_i, we can conclude that $z_i \sim N$, normal distribution. Now, let's calculate the expectation value of z_i. We know that for a p dimensional vector a

$$E(z) = E(a^T x) = a^T E(x) = 0 \tag{1}$$

The co-variant can be given by

$$Cov(a^T x) = a^T a = 1 \tag{2}$$

As a result, we get $z_i \sim N(0,1)$, and the expected squared distance will be $E(z^2) = 1$.

(b)

Problem 3 (ESL 2.7)

(a)

For linear regression, the exact solution can be given by

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{3}$$

therefore, the predicted value for a given input will be

$$\hat{f}(x_0) = x_0^T\hat{\beta} \tag{4}$$

Compare with the desired form of the estimator,

$$\hat{f}(x_0) = \sum_{i=1}^{N} l_i(x_0; \mathbf{X}) y_i \tag{5}$$

it's found that

$$l_i(x_0; \mathbf{X}) = x_0^T(\mathbf{X}^T\mathbf{X})^{-1}x_i^T \tag{6}$$

Therefore, linear regression is the member of this class of estimators.

For k-nearest-neighbour regression, the estimator is

$$\hat{f}(x_0) = \frac{1}{k}\sum_{x_i \in N_k(x_0)} y_i \tag{7}$$

where $N_k(x_0)$ is the neighbourhood of x_0 defined by the k closest points x_i in the training sample. Now, compare with the desired estimator form, it's found that

$$l_i(x_0; \mathbf{X}) = \begin{cases} 0 & x_i \notin N_k(x_0) \\ \frac{1}{k} & x_i \in N_k(x_0) \end{cases} \tag{8}$$

Therefore, k-nearest-neighbour regression is the member of this class of estimators.

(b)

Using Eq. (2.25) in the textbook, the conditional mean squared error can be rewritten as

$$\begin{aligned} E_{\mathbf{y}|\mathbf{x}}(f(x_0) - \hat{f}(x_0))^2 &= E_{\mathbf{y}|\mathbf{x}}(\hat{f}(x_0) - E_{\mathbf{y}|\mathbf{x}}\hat{f}(x_0))^2 + (E_{\mathbf{y}|\mathbf{x}}\hat{f}(x_0) - f(x_0))^2 \\ &= Var_{\mathbf{y}|\mathbf{x}}(\hat{f}(x_0)) + Bias^2_{\mathbf{y}|\mathbf{x}}(\hat{f}(x_0)) \end{aligned} \tag{9}$$

Substitute Eq. (5) into Eq. (9), we have

$$\begin{aligned}
Var_{\mathbf{y}|\mathbf{x}}(\hat{f}(x_0)) &= Var_{\mathbf{y}|\mathbf{x}}\left(\sum_{i=1}^{N} l_i(x_0;\mathbf{X})y_i\right) \\
&= \sum_{i=1}^{N} l_i(x_0;\mathbf{X}) Var_{\mathbf{y}|\mathbf{x}}\left(y_i\right) \\
&= \sum_{i=1}^{N} l_i(x_0;\mathbf{X})\sigma^2 \qquad (10)
\end{aligned}$$

and

$$\begin{aligned}
Bias_{\mathbf{y}|\mathbf{x}}(\hat{f}(x_0)) &= Bias_{\mathbf{y}|\mathbf{x}}\left(\sum_{i=1}^{N} l_i(x_0;\mathbf{X})y_i\right) \\
&= E_{\mathbf{y}|\mathbf{x}}\sum_{i=1}^{N} l_i(x_0;\mathbf{X})y_i - f(x_0) \\
&= \sum_{i=1}^{N} l_i(x_0;\mathbf{X})f(x_i) - f(x_0) \qquad (11)
\end{aligned}$$

(c)

For the unconditional mean squared error, we have

$$E_{\mathbf{y},\mathbf{x}}(f(x_0)-\hat{f}(x_0))^2 = Var_{\mathbf{y},\mathbf{x}}(\hat{f}(x_0)) + Bias^2_{\mathbf{y},\mathbf{x}}(\hat{f}(x_0)) \qquad (12)$$

Therefore, the variance will be

$$\begin{aligned}
Var_{\mathbf{y},\mathbf{x}}(\hat{f}(x_0)) &= E_{\mathbf{y},\mathbf{x}}(\hat{f}(x_0) - E_{\mathbf{y},\mathbf{x}}\hat{f}(x_0))^2 \\
&= E_x E_{\mathbf{y}|\mathbf{x}}(\hat{f}(x_0) - E_x E_{\mathbf{y}|\mathbf{x}}\hat{f}(x_0))^2 \\
&= E_x E_{\mathbf{y}|\mathbf{x}}\left(\sum_{i=1}^{N} l_i(x_0;\mathbf{X})y_i - E_x\left(\sum_{i=1}^{N} l_i(x_0;\mathbf{X})f(x_i)\right)\right)^2 \\
&= E_x\left[E_{\mathbf{y}|\mathbf{x}}\left(\sum_{i=1}^{N} l_i(x_0;\mathbf{X})y_i\right)^2 - 2E_{\mathbf{y}|\mathbf{x}}\left(\sum_{i=1}^{N} l_i(x_0;\mathbf{X})y_i\right)\left(\sum_{i=1}^{N} l_i(x_0;\mathbf{X})f(x_i)\right)\right. \\
&\quad \left. + E_x\left(\sum_{i=1}^{N} l_i(x_0;\mathbf{X})f(x_i)\right)\left(\sum_{i=1}^{N} l_i(x_0;\mathbf{X})f(x_i)\right)\right] \\
&= E_x E_{y|x}\left(\sum_{i=1}^{N} l_i(x_0;\mathbf{X})y_i - \sum_{i=1}^{N} l_i(x_0;\mathbf{X})f(x_i)\right)^2 + E_x\left(E_x(\sum_{i=1}^{N} l_i(x_0;\mathbf{X})f(x_i)) - \sum_{i=1}^{N} l_i(x_0;\mathbf{X})f(x_i)\right)^2 \\
&= E_x Var_{y|x}(\hat{f}(x_0)) + Var_x\left(\sum_{i=1}^{N} l_i(x_0;\mathbf{X})f(x_i)\right) \\
&= E_x\left(\sum_{i=1}^{N} l_i(x_0;\mathbf{X})\sigma^2\right) + Var_x\left(\sum_{i=1}^{N} l_i(x_0;\mathbf{X})f(x_i)\right) \qquad (13)
\end{aligned}$$

The Bias will be

$$\begin{aligned}Bias_{y,x}(\hat{f}(x_0) &= E_{\mathbf{y},\mathbf{x}}\hat{f}(x_0) - E_x f(x_0) \\ &= E_x(E_{y|x}\hat{f}(x_0) - f(x_0)) \\ &= E_x(\sum_{i=1}^{N} l_i(x_0;\mathbf{X})f(x_i) - f(x_0)) \\ &= E_x Bias_{y|x}(\hat{f}(x_0)) \end{aligned} \tag{14}$$

(d)

Compare the result in (b), and (c), we find that the unconditional squared bias is the expected value respected to X of conditional squared bias. $E_x Bias_{y|x}(\hat{f}(x_0)) = Bias_{x,y}(\hat{f}(x_0))$

The unconditional variance will be the expected value respected to X of conditional variance with extra variance term.

$$Var_{y,x}(\hat{f}(x_0)) = E_x Var_{y|x}(\hat{f}(x_0)) + Var_x\left(\sum_{i=1}^{N} l_i(x_0;\mathbf{X})f(x_i)\right)$$

Problem 4

(a)

Let $\hat{\beta}$ be the least squares estimation over a set of training data $(x_1, y_1), ..., (x_n, y_n)$ using

$$R_{tr}(\hat{\beta}) = \frac{1}{N}\sum_{1}^{N}(y_i - \hat{\beta}^T x_i)^2 \tag{15}$$

, while $\tilde{\beta}$ is the least squares estimation over another set of training data $(\tilde{x}_1, \tilde{y}_1), ..., (\tilde{x}_n, \tilde{y}_n)$ drawn at random from the same population as the first training data using

$$R_{te}(\tilde{\beta}) = \frac{1}{M}\sum_{1}^{M}(\tilde{y}_i - \tilde{\beta}^T \tilde{x}_i)^2 \tag{16}$$

Since both training data are drawn from the same population, if they are randomly picked up, we can say

$$E[R_{tr}(\hat{\beta})] = E[R_{te}(\tilde{\beta})] \tag{17}$$

, which can be formally proven by

$$\begin{aligned}E[R_{tr}(\hat{\beta})] &= E[\frac{1}{N}\sum_{1}^{N}(y_i - \hat{\beta}^T x_i)^2] \\ &= \frac{1}{N}E[\sum_{1}^{N}(y_i - \hat{\beta}^T x_i)^2] \\ &= \frac{1}{M}E[\sum_{1}^{M}(y_i - \hat{\beta}^T x_i)^2] \\ &= \frac{1}{M}E[\sum_{1}^{M}(\tilde{y}_i - \tilde{\beta}^T \tilde{x}_i)^2] \\ &= E[R_{te}(\tilde{\beta})] \end{aligned} \tag{18}$$

Also, $\tilde{\beta}$ is the least squares estimation over $(\tilde{x}_1, \tilde{y}_1), ..., (\tilde{x}_n, \tilde{y}_n)$, so

$$\frac{1}{M}\sum_1^M(\tilde{y}_i - \tilde{\beta}^T\tilde{x}_i)^2 \le \frac{1}{M}\sum_1^M(\tilde{y}_i - \hat{\beta}^T\tilde{x}_i)^2$$
$$\Rightarrow R_{te}(\tilde{\beta}) \le R_{te}(\hat{\beta})$$
$$\therefore E[R_{te}(\tilde{\beta})] \le E[R_{te}(\hat{\beta})] \tag{19}$$

With Eq. (17) and Eq. (20), we get

$$\therefore E[R_{tr}(\hat{\beta})] \le E[R_{te}(\hat{\beta})] \tag{20}$$

Problem 5 (ESL 3.2)

Method 1

For each point x_0, forming a 95% confidence interval implies that we need to know the confidence interval for $\hat{\beta}$, and using linear function $a^T\hat{\beta}$ to construct the 95% confidence band. Since $\hat{\beta} \sim N(\beta, (\mathbf{X}^T\mathbf{X})^{-1}\sigma^2))$, we can obtain the $1-2\alpha$ confidence interval for β as

$$\left(\hat{\beta} - z^{(1-\alpha)}\sqrt{(\mathbf{X}^T\mathbf{X})^{-1}}\sigma, \hat{\beta} + z^{(1-\alpha)}\sqrt{(\mathbf{X}^T\mathbf{X})^{-1}}\sigma\right) \tag{21}$$

where $z^{(1-\alpha)}$ is the $1-\alpha$ percentile of the normal distribution. (Note that if we replace the α to known value $\hat{\alpha}$, the distribution will become t distribution, and it normally become negligible as the sample increases; therefore, we typically use the normal quartiles.) As a result, the confidence interval from each input point x_0 will be

$$\left(a^T\hat{\beta} - z^{(1-\alpha)}\sqrt{a^T(\mathbf{X}^T\mathbf{X})^{-1}a}\sigma, \hat{\beta} + z^{(1-\alpha)}\sqrt{a^T(\mathbf{X}^T\mathbf{X})^{-1}a}\sigma\right) \tag{22}$$

where $y_0 = a^T\hat{\beta} = \sum_{j=0}^3 \hat{\beta}_j x_0^j$. For 95%, α is 0.025.

Method 2

From Eq. (3.15) from textbook, the confidence set for β can be given by

$$C_\beta = \left\{\beta|(\hat{\beta}-\beta)^T\mathbf{X}^T\mathbf{X}(\hat{\beta}-\beta) \le \sigma^2\chi_4^{2^{(1-2\alpha)}}\right\} \tag{23}$$

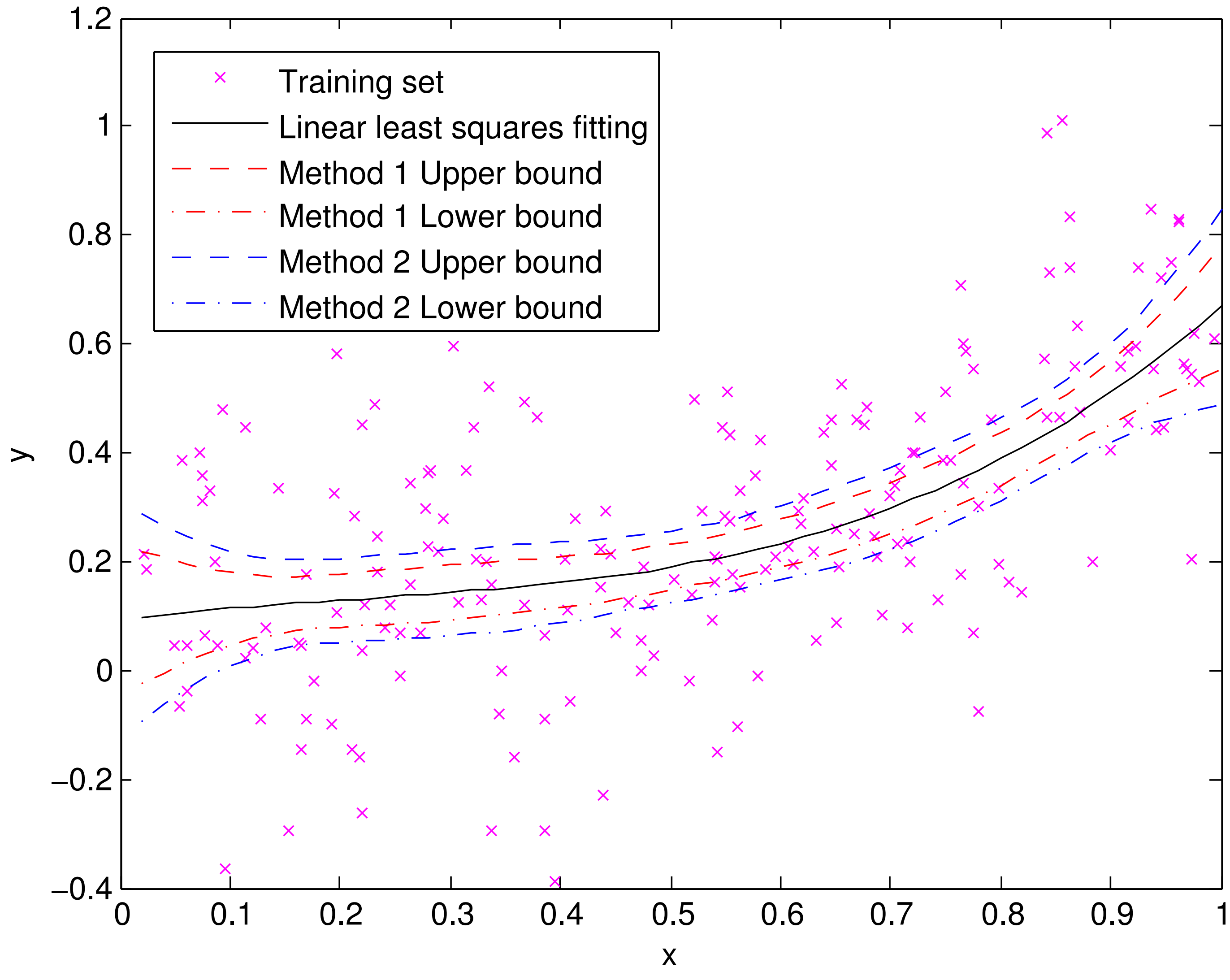
Therefore, the 95%confidence band from each input point x_0 will be

$$\left(a^T\hat{\beta} - \sqrt{a^T(\mathbf{X}^T\mathbf{X})^{-1}a\chi_4^{2^{(1-2\alpha)}}}\sigma, \hat{\beta} + \sqrt{a^T(\mathbf{X}^T\mathbf{X})^{-1}a\chi_4^{2^{(1-2\alpha)}}}\sigma\right) \tag{24}$$

where α is 0.025, and $\chi_l^{2^{(1-\alpha)}}$ is the chi-squared distribution on l degrees of freedom. (Similar to the discussion in method 1, if we don't know the α, but we know $\hat{\alpha}$ which can be calculated from training set, the chi-squared distribution will be replaced by $F_{p_1-p_0, M-p_1-1}$ distribution.)

According to simulation result shown in the figure, it's observed that Method 2 has wider confidence band.

```
% HW1, Q5 (ESL3.2)
% Dong-Bang Tsai
clear all;

N = 200;             % N of training set
alpha = (1-0.95)/2; % 95% confidence interval
sigma = 0.2;         % Gaussian random noise
```



```
% The training set
beta = randn(4,1);
x_points = rand(N,1);
X = [ones(N,1),x_points,x_points.^2,x_points.^3];
y = X*beta + normrnd(0,sigma,N,1);
% Learned hypothesis
hat_beta = ((X'*X))\X'*y;

% Plot array
N_plot = 50;
x_plot_points = [1/N_plot:1/N_plot:1]';
a = [ones(N_plot,1),x_plot_points,x_plot_points.^2,x_plot_points.^3];

% Compute the confidence band
hat_sigma = sqrt( 1/(N-4)*sum( (X*beta-y).^2 ) );
delta1 = norminv(1-alpha)*sqrt(diag(a*((X'*X)\a')))*hat_sigma;
delta2 = sqrt(diag(a*((X'*X)\a'))*chi2inv(1-2*alpha,4))*hat_sigma;

plot(x_points,y(:,1),'mx');
hold on;
```

```
plot(x_plot_points,a*beta,'k-');
plot(x_plot_points,a*beta + delta1,'r--');
plot(x_plot_points,a*beta - delta1,'r-.');
plot(x_plot_points,a*beta + delta2,'b--');
plot(x_plot_points,a*beta - delta2,'b-.');
xlabel('x'); ylabel('y');
legend('Training set','Linear least squares fitting','Method 1 Upper bound', 'Method 1 Lower bound', 'Metho
hold off;
```

Problem 6

(a)

When $p \gg N$, the columns of X are not lineraly independent which implies X isn't of full rank. Therefore, the solution of least squares given by $\hat{\beta} = (X^TX)^{-1}X^Ty$ will not be uniquely defined since X^TX is singular, and the residuals of the solution will be 0. The solution is unique only if X has a full column rank, and X^TX is positive definite.

(b)

In the ridge regression approach, the equation we are trying to solve is

$$(X^TX + \lambda I)\beta = X^Ty \tag{25}$$

The solution can be given by $\hat{\beta}_\lambda = (X^TX + \lambda I)^{-1}X^Ty$. We add positive constant to diagonal of X^TX; therefore, $(X^TX + \lambda I)^{-1}$ is non-singular, even if X^TX is not of full rank. As a result, the solution always exists, and is unique.

(c)

As we show that the solution of ridge regression is $\hat{\beta}_\lambda = (X^TX + \lambda I)^{-1}X^Ty$, if we try to get the λ smaller, it will eventually become least squares solution which is $\hat{\beta}_\lambda = (X^TX)^{-1}X^Ty$.

(d)

Using $X = UDV^T$

$$\begin{aligned}
\hat{\beta}_\lambda &= (X^TX + \lambda I)^{-1}X^Ty \\
&= \left(VD^TU^TUDV^T + \lambda I\right)VDU^Ty \\
&= V\left(D^2 + \lambda I\right)V^TDU^Ty \\
&= V\left(D^2 + \lambda I\right)DU^Ty
\end{aligned} \tag{26}$$