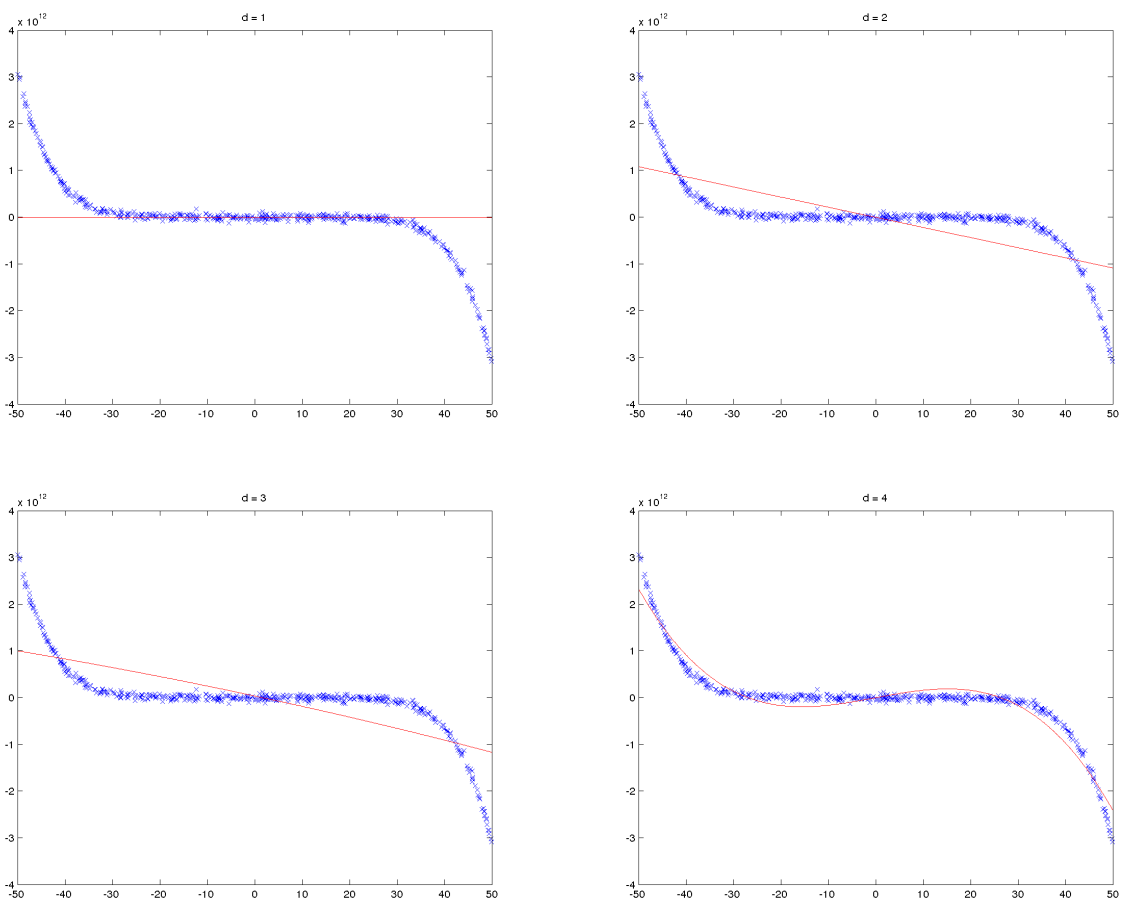
COMS4771 Machine Learning 2014: Homework 1 Solution

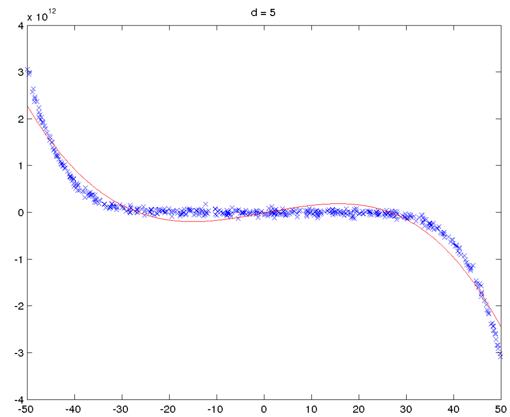
Robert Ying

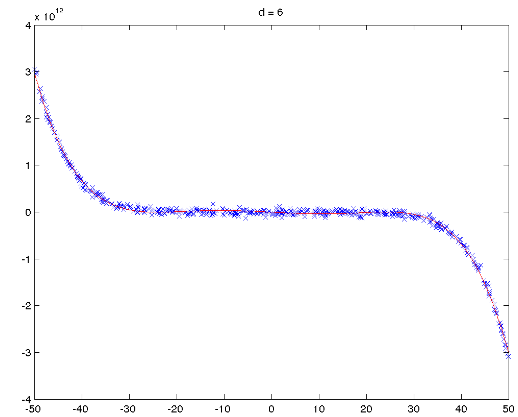
September 16, 2015

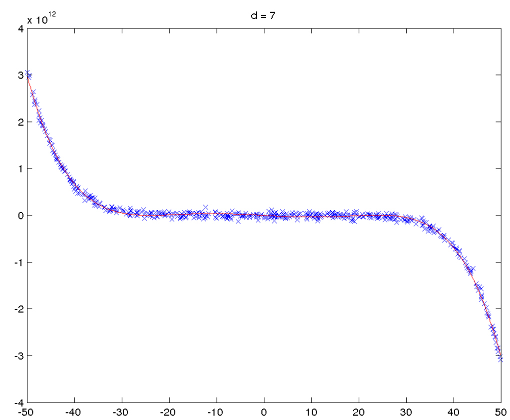
1 Problem 1

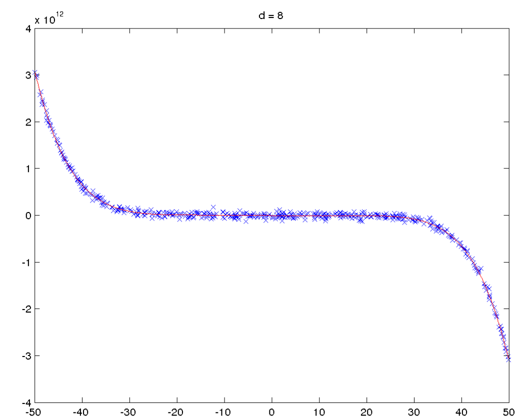
The plots for different choices of d are given below:

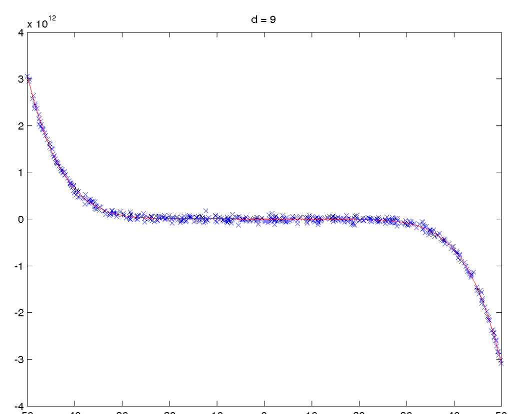


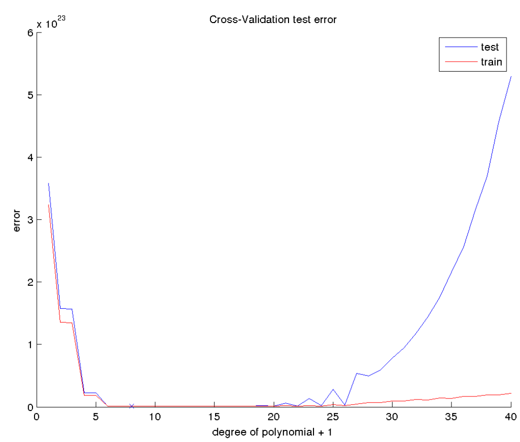












As can be seen in the cross-validation plot, the error is minimized for testing at $d = 8$. Higher values of d begin to overfit the data, as seen by the rapid increase in testing error.

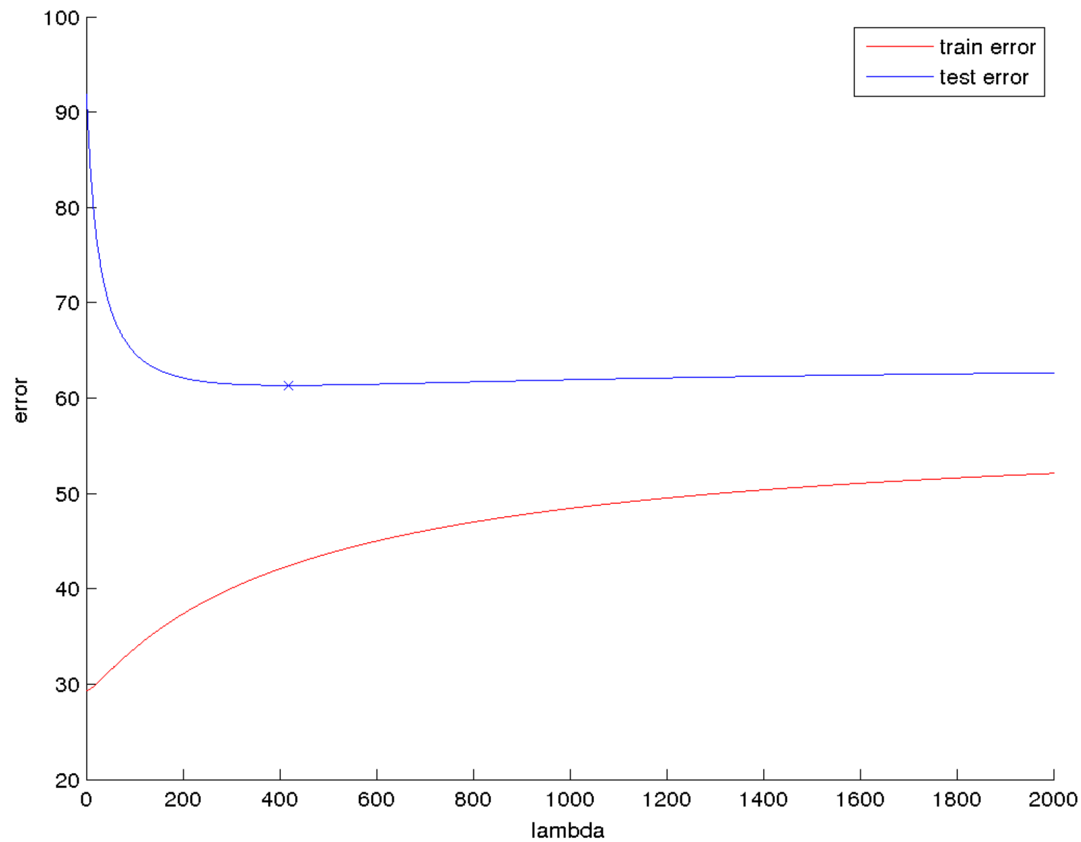
$\boldsymbol{\theta}$ values for various d:

- $d = 1$: [109021069510]

- $d = 2$: [-22973948397, 58291727749]
- $d = 3$: [110026131, -22476073416, -30832651874]
- $d = 4$: [-26334804, 6374177, 17727357428, -1884434220]
- $d = 5$: [33856, -26295917, -67685290, 17734650681, 16041141394]
- $d = 6$: [-16399, 5013, 19384276, -10680485, -5940688633, -50787725]
- $d = 7$: [12, -61342, -36743, 19236962, 22026113, -5880102685, -3554614876]
- $d = 8$: [-3.79, -3.77, -1169, 10115, 2073264, -3752974, -1022283784, -1137535780]
- ...

Note that these values are based on how the testing and training data are split, and so may vary in your particular case.

2 Problem 2



As can be seen in the plot above, increasing λ rapidly decreases the testing error (while increasing training error). The minimal λ value in this case was ≈ 419, though this is again dependent on how the data set is split.

3 Problem 3

3.1 Part 1

Proof. Prove that $g(-z) = 1 - g(z)$ when $g(z) = \frac{1}{1+e^{-z}}$

$$\begin{aligned} g(-z) &= \frac{1}{1+e^{z}} \\ &= \frac{1}{\frac{1}{e^{-z}} + \frac{e^{-z}}{e^{-z}}} \\ &= \frac{e^{-z}}{1+e^{-z}} \\ &= \frac{(1+e^{-z})-1}{1+e^{-z}} \\ &= 1 - \frac{1}{1+e^{-z}} \\ &= 1 - g(z) \end{aligned}$$

□

3.2 Part 2

Proof. Given: $y = g(z) = \frac{1}{1+e^{-z}}$

Assume: $g^{-1}(y) = \ln\left(\frac{y}{1-y}\right)$

Then

$$\begin{aligned} \ln\frac{y}{1-y} &= \ln(y) - \ln(1-y) \\ &= \ln\left(\frac{1}{1+e^{-z}}\right) - \ln\left(1 - \frac{1}{1+e^{-z}}\right) \\ &= \ln\left(\frac{1}{1+e^{-z}}\right) - \ln\left(\frac{e^{-z}}{1+e^{-z}}\right) \\ &= \ln 1 - \ln(1-e^{-z}) - \ln e^{-z} + \ln(1+e^{-z}) \\ &= z \end{aligned}$$

Therefore, $g^{-1}(g(z)) = z$. □

4 Problem 4

First find the gradient of the risk:

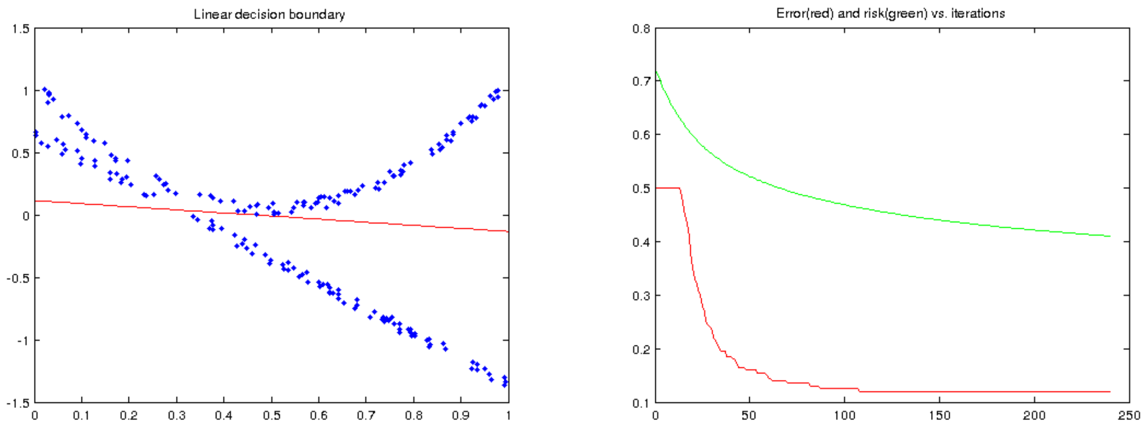
Let $f = f(x, \theta)$, given

$$\begin{aligned} f &= \frac{1}{1 + e^{-\theta^T \mathbf{x}}} \\ R &= \frac{1}{N} \sum_{i=1}^{N} (y_i - 1) \log(1 - f_i) - y_i \log(f_i) \end{aligned}$$

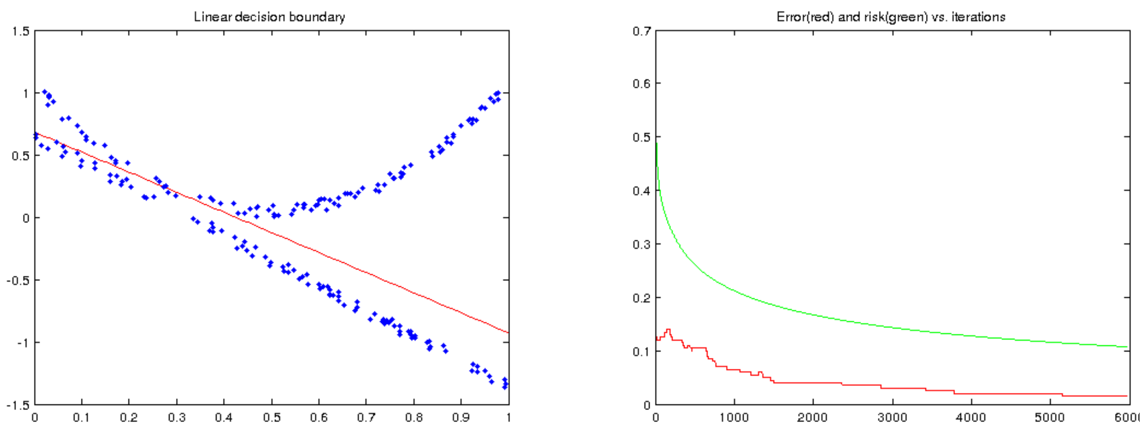
then

$$\begin{aligned} \nabla_\theta R &= \nabla_\theta \left(\frac{1}{N} \sum_{i=1}^{N} (y_i - 1) \log(1 - f_i) - y_i \log(f_i) \right) \\ &= \frac{1}{N} \sum_{i=1}^{N} (y_i - 1) \frac{d}{d\theta} \log(1 - f_i) - y_i \frac{d}{d\theta} \log(f_i) \\ &= \frac{1}{N} \sum_{i=1}^{N} (y_i - 1) \frac{d}{d\theta} \log \left(1 - \frac{1}{1 + e^{-\theta^T \mathbf{x}_i}} \right) - y_i \frac{d}{d\theta} \log \left(\frac{1}{1 + e^{-\theta^T \mathbf{x}_i}} \right) \\ &= \frac{1}{N} \sum_{i=1}^{N} (y_i - 1) \frac{d}{d\theta} \left(\log(e^{-\theta^T \mathbf{x}_i}) - \log(1 + e^{-\theta^T \mathbf{x}_i}) \right) - y_i \frac{d}{d\theta} \left(- \log(1 + e^{-\theta^T \mathbf{x}_i}) \right) \\ &= \frac{1}{N} \sum_{i=1}^{N} (y_i - 1) \left(-\mathbf{x}_i + \frac{\mathbf{x}_i e^{-\theta^T \mathbf{x}_i}}{1 + e^{-\theta^T \mathbf{x}_i}} \right) - y_i \left(\frac{\mathbf{x}_i e^{-\theta^T \mathbf{x}_i}}{1 + e^{-\theta^T \mathbf{x}_i}} \right) \\ &= \frac{1}{N} \sum_{i=1}^{N} (1 - y_i) \left(\mathbf{x}_i - \frac{\mathbf{x}_i e^{-\theta^T \mathbf{x}_i}}{1 + e^{-\theta^T \mathbf{x}_i}} \right) - y_i \left(\frac{\mathbf{x}_i e^{-\theta^T \mathbf{x}_i}}{1 + e^{-\theta^T \mathbf{x}_i}} \right) \end{aligned}$$

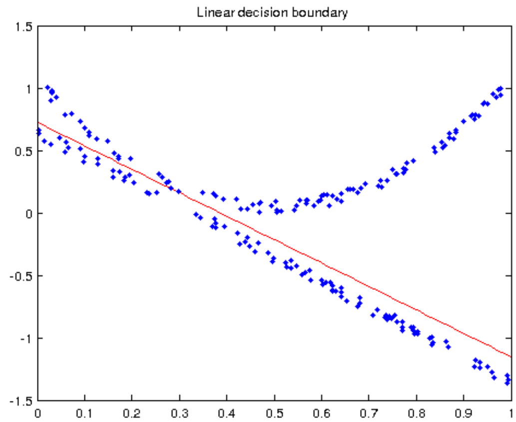
Using $\epsilon = 0.005$ and $\eta = 0.1$ in the gradient descent algorithm, we get the following linear decision boundary: ($\theta = [0.592, 2.390, -0.273]$). The binary classification error and the risk are given below:

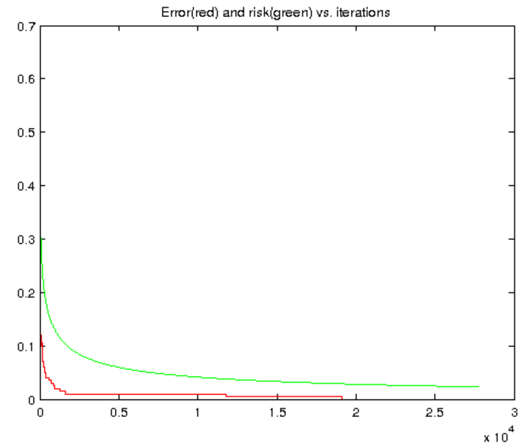


For $\epsilon = 0.002$ and $\eta = 0.5$ in the gradient descent algorithm, we get the following linear decision boundary: ($\theta = [20.825, 12.928, -8.805]$). The binary classification error and the risk are given below:



For $\epsilon = 0.001$ and $\eta = 2.0$ in the gradient descent algorithm, we get the following linear decision boundary: ($\theta = [68.623, 36.586, -26.412]$). The binary classification error and the risk are given below:





5 Matlab source

5.1 Problem 1

problem1.m

```
load('problem1.mat')

% finding a good value for $d$

full_dataset_errors = [];
full_dataset_models = {};

for d=1:9
    [err, model] = polyreg(x, y, d);
    title(sprintf('d = %d', d));
    full_dataset_errors(d) = err;
    full_dataset_models{d} = model;

    filename = sprintf('p1_d%d.png', d);
    print(filename, '-dpng');
end

close all;

% cross validation
cross_validation_train_errors = [];
cross_validation_test_errors = [];
cross_validation_models = {};

ind = crossvalind('Kfold', 500, 2);
xtrain = x(ind == 1);
ytrain = y(ind == 1);
xtest = x(ind == 2);
ytest = y(ind == 2);

for d=1:40
    [err, model, errT] = polyreg(xtrain, ytrain, d, xtest, ytest);
    title(sprintf('d = %d', d));
    cross_validation_train_errors(d) = err;
    cross_validation_test_errors(d) = errT;
    cross_validation_models{d} = model;
end

clf;
hold on;

plot(cross_validation_test_errors, 'b')
plot(cross_validation_train_errors, 'r')
[~, i] = min(cross_validation_test_errors);
plot(i, cross_validation_test_errors(i), 'bx');
xlabel('degree of polynomial + 1');
ylabel('error');
legend('test', 'train');
```

```
title('Cross-Validation test error');
print('p1_cv_plot.png', '-dpng');
```

polyreg.m

```
function [err,model,errT] = polyreg(x,y,D,xT,yT)
%
% Finds a D-1 order polynomial fit to the data
%
%    function [err,model,errT] = polyreg(x,y,D,xT,yT)
%
% x = vector of input scalars for training
% y = vector of output scalars for training
% D = the order plus one of the polynomial being fit
% xT = vector of input scalars for testing
% yT = vector of output scalars for testing
% err = average squared loss on training
% model = vector of polynomial parameter coefficients
% errT = average squared loss on testing
%
% Example Usage:
%
% x = 3*(rand(50,1)-0.5);
% y = x.*x.*x-x+rand(size(x));
% [err,model] = polyreg(x,y,4);
%

xx = zeros(length(x),D);
for i=1:D
  xx(:,i) = x.^(D-i);
end
model = pinv(xx)*y;
err   = (1/(2*length(x)))*sum((y-xx*model).^2);

if (nargin==5)
  xxT = zeros(length(xT),D);
  for i=1:D
    xxT(:,i) = xT.^(D-i);
  end
  errT  = (1/(2*length(xT)))*sum((yT-xxT*model).^2);
end

q  = (min(x):(max(x)/300):max(x))';
qq = zeros(length(q),D);
for i=1:D
  qq(:,i) = q.^(D-i);
end

clf
plot(x,y,'X');
hold on
if (nargin==5)
```

```
    plot(xT,yT,'cO');
end
plot(q,qq*model,'r')
```

5.2 Problem 2

problem2.m

```
load('problem2.mat')

% cross validation
cross_validation_train_errors = [];
cross_validation_test_errors = [];
cross_validation_models = {};

ind = crossvalind('Kfold', 400, 2);
xtrain = x(ind == 1,:);
ytrain = y(ind == 1);
xtest = x(ind == 2,:);
ytest = y(ind == 2);

d = 1;
lambdas = 0:0.5:2000;
for lambda=lambdas
    [err, model, errT] = ridgereg(xtrain, ytrain, lambda, xtest, ytest);
    cross_validation_train_errors(d) = err;
    cross_validation_test_errors(d) = errT;
    cross_validation_models{d} = model;
    d = d + 1;
end
close all;

hold on;
plot(lambdas, cross_validation_train_errors, 'r');
plot(lambdas, cross_validation_test_errors, 'b');
[~, i] = min(cross_validation_test_errors);
plot(lambdas(i), cross_validation_test_errors(i), 'bx');
xlabel('lambda');
ylabel('error');
legend('train error', 'test error');
print('p2_cv_plot.png', '-dpng');
```

ridgereg.m

```
function [err,model,errT] = ridgereg(x,y,lambda,xT,yT)
%
% Performs a multivariate ridge regression with
%
%    function [err,model,errT] = ridgereg(x,y,D,xT,yT)
%
% x = vector of input scalars for training
% y = vector of output scalars for training
% lambda = the penalty parameter lambda
% xT = vector of input scalars for testing
% yT = vector of output scalars for testing
% err = average squared loss on training
```

```
% model = vector of polynomial parameter coefficients
% errT = average squared loss on testing
%
%

xtx = x' * x;
model = (x' * x + lambda * eye(size(xtx))) \ x' * y;
err    = (1/(2*size(x, 1)))*sum((y-x*model).^2);

if (nargin==5)
  errT   = (1/(2*size(xT, 1)))*sum((yT - xT*model).^2);
end
```

5.3 Problem 4

problem4.m

```
function problem4()
    close all;
    load('dataset4.mat');
    stepsize = 2;
    tol = 0.001;
    theta = rand(size(X,2),1);
    maxiter = 200000;
    curiter = 0;
    % For plotting stats
    risks = [];
    errs = [];
    prevtheta = theta+2*tol;
    while norm(theta - prevtheta) >= tol
        if curiter > maxiter
            break;
        end
        % Current stats
        r = risk(X, Y, theta);
        f = 1./(1+exp(-X*theta));
        f(f >= 0.5) = 1;
        f(f < 0.5) = 0;
        err = sum(f~=Y)/length(Y);
        fprintf('Iter:%d, error:%0.4f, risk:%0.4f\n', curiter,err,r);
        risks = cat(1, risks, r);
        errs = cat(1, errs, err);

        % Update theta
        prevtheta = theta;
        G = gradient(X, Y, theta);
        theta = theta - stepsize*G;
        curiter = curiter + 1;
    end

    figure, plot(1:curiter, errs, 'r', 1:curiter, risks, 'g');
    title('Error(red) and risk(green) vs. iterations');
    disp('theta');
    disp(theta)
    x=0:0.01:1;
    y = (-theta(3) - theta(1).*x)/theta(2);
    figure, plot(x, y, 'r'); hold on;
    plot(X(:,1), X(:,2), '.');
    title('Linear decision boundary');
end

function R = risk(x, y, theta)
    f = 1./(1+exp(-x*theta));
    r = (y-1).*log(1-f)-y.*log(f); r(isnan(r)) = 0;
    R = mean(r);
```

```
end

function g = gradient(x, y, theta)
    yy = repmat(y, 1, size(x,2));
    f = 1./(1+exp(-x*theta));
    ff = repmat(f, 1, size(x,2));
    d = x.*repmat(exp(-x*theta), 1, size(x,2));
    g = (1-yy).*(x - d.*ff) - yy.*d.*ff;
    g = sum(g);
    g = g/length(y);
    g = g';
end
```