%% Read data and convert to matrix form

%fid = fopen('Querylevelnorm.txt','rt');

fid = fopen('Querylevelnorm.txt','rt');

%Read data row-wise

wholerowread = textscan(fid,'%s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s',69623);

% Read relevance

Y = wholerowread{1,1};

%Read features

parameters = wholerowread(:,3:48);

%Convert features to vector

X = [parameters{:}];

%%

%%Create matrix and initialize to all zeros

X\_new = zeros(69623,46);

[rowSize,columnSize] = size(X);

for k=1:rowSize

for j=1:columnSize

%For X\_new enter matrix values from position 3 to 48(features)

strings = strsplit(X{k,j},':');

X\_new(k,j) = str2double(strings(2));

end

end

Y\_new = str2double(Y);

%%Divide data into random training set and apply random permutation to

% XVectors

%% Divide data into training, test and validation

noTrainDocs = floor(0.8 \* length(X\_new));

noValidationDocs = floor(0.9 \* length(X\_new));

X\_rand = randperm(length(X\_new));

X\_rand = (X\_rand).';

trainingIndexes = X\_rand(1:noTrainDocs);

X\_training = X\_new(trainingIndexes,:);

Y\_training = Y\_new(trainingIndexes,:);

%% create va;idation data

validationIndexes = X\_rand(noTrainDocs+1:noValidationDocs);

X\_validation = X\_new(validationIndexes,:);

Y\_validation = Y\_new(validationIndexes,:);

%% create test data

testIndexes = X\_rand(noValidationDocs+1:end);

X\_test = X\_new(testIndexes,:);

Y\_test = Y\_new(testIndexes,:);

trainInd1 = trainingIndexes;

validInd1 = validationIndexes;

noValidationDocs = noValidationDocs-noTrainDocs;

%%

fclose(fid);

%%calculate mu1

M1= 5;

mu1 = zeros(46,M1);

Xforclosed = randperm(noTrainDocs,M1);

Xforclosed = (Xforclosed).';

% populate mu1 matrix

for i= 1 : M1

mu1(:,i) = X\_training(Xforclosed(i),:);

end

%%

%plot(mu1);

%%

% populate sigma

sigma = zeros(46:46);

sigma = var(X\_training)\*0.1;

for i=1:46

if sigma(1,i) < 0.0001

sigma(1,i) = 0.01;

end

end

sigma = diag(sigma);

D= 46;

Sigma1= zeros(D,D);

for i= 1:M1

Sigma1(:,:,i) = sigma;

end

%%

%% Calculate design matrix for training set

phi = zeros(noTrainDocs,M1);

phi(:,1)= 1;

for j= 2 : M1

for i = 1 : noTrainDocs

a= inv(Sigma1(:,:,j));

b= (X\_training(i,:).'-mu1(:,j)).';

c= (X\_training(i,:).'-mu1(:,j));

d= -0.5 \* b \* a \* c;

phi(i,j) = exp(d);

end

end

%%

lambda1= 0.2;

w1 = inv( lambda1\*eye(M1,M1)+ phi.'\*phi)\*phi.'\*Y\_training;

%%

%plot(w1,lambda1);

%% Root mean square error for training set

Err= 0.5 \* ((Y\_training-(phi\*w1)).')\*(Y\_training-(phi\*w1));

trainPer1 = sqrt((2\*Err)/noTrainDocs);

%% For validation set

%calculate design matrix

phi\_valid = zeros(noValidationDocs,M1);

phi\_valid(:,1)= 1;

for j= 2 : M1

for i = 1 : noValidationDocs

a= inv(Sigma1(:,:,j));

b= (X\_validation(i,:).'-mu1(:,j)).';

c= (X\_validation(i,:).'-mu1(:,j));

d= -0.5 \* b \* a \* c;

phi\_valid(i,j) = exp(d);

end

end

%%

lambda1= 0.2;

%w1\_valid = inv( lambda1\*eye(M1,M1)+ phi\_valid.'\*phi\_valid)\*phi\_valid.'\*Y\_validation;

%% Root mean square error for training set

Err\_valid= 0.5 \* ((Y\_validation-(phi\_valid\*w1)).')\*(Y\_validation-(phi\_valid\*w1));

validPer1 = sqrt((2\*Err\_valid)/noValidationDocs);

%%

%plot(lambda1,validPer1);

%%

w02 = w02.';

%%%%

%Create matrix and initialize to all zeros

load('synthetic.mat');

noTrainSyn = floor(0.9 \* length(x));

noValidationSyn= size(x,2)- noTrainSyn;

X = x.';

Y = t;

%%

plot(X,Y);

%% Divide data into training, test and validation

X\_rand\_syn = randperm(length(X));

X\_rand\_syn = (X\_rand\_syn).';

trainingIndexes\_syn = X\_rand\_syn(1:noTrainSyn);

validationIndexes\_syn = X\_rand\_syn(noTrainSyn+1:end);

%%

X\_training\_syn = X(trainingIndexes\_syn,:);

Y\_training\_syn = Y(trainingIndexes\_syn,:);

trainInd2 = trainingIndexes\_syn;

%%

M2= 5;

mu2 = zeros(10,M2);

Xforclosed\_syn = randperm(noTrainSyn,M2);

Xforclosed\_syn = (Xforclosed\_syn).';

% populate mu2 matrix

for i= 1 : M2

mu2(:,i) = X\_training\_syn(Xforclosed\_syn(i),:);

end

%% populate sigma\_syn

sigma\_syn = zeros(10:10);

sigma\_syn = var(X\_training\_syn)\*0.1;

for i=1:10

if sigma\_syn(1,i) < 0.0001

sigma\_syn(1,i) = 0.01;

end

end

sigma\_syn = diag(sigma\_syn);

D= 10;

Sigma2= zeros(D,D);

for i= 1:M2

Sigma2(:,:,i) = sigma\_syn;

end

%% Calculate design matrix for training set

phi\_syn = zeros(noTrainSyn,M2);

phi\_syn(:,1)= 1;

for j= 2 : M2

for i = 1 : noTrainSyn

a1= inv(Sigma2(:,:,j));

b1= (X\_training\_syn(i,:).'-mu2(:,j)).';

c1= (X\_training\_syn(i,:).'-mu2(:,j));

d1= -0.5 \* b1 \* a1 \* c1;

phi\_syn(i,j) = exp(d1);

end

end

%%

lambda2= 0.2;

w2 = inv( lambda2\*eye(M2,M2)+ phi\_syn.'\*phi\_syn)\*phi\_syn.'\*Y\_training\_syn;

%% Root mean square error for training set

Err2= 0.5 \* ((Y\_training\_syn-(phi\_syn\*w2)).')\*(Y\_training\_syn-(phi\_syn\*w2));

trainPer2 = sqrt((2\*Err2)/noTrainSyn);

%%

%Validation data calculations

X\_validation\_syn = X(validationIndexes\_syn,:);

Y\_validation\_syn = Y(validationIndexes\_syn,:);

validInd2 = validationIndexes\_syn;

%% Calculate design matrix for training set

phi\_syn\_valid = zeros(noValidationSyn,M2);

phi\_syn\_valid(:,1)= 1;

for j= 2 : M2

for i = 1 : noValidationSyn

a1= inv(Sigma2(:,:,j));

b1= (X\_training\_syn(i,:).'-mu2(:,j)).';

c1= (X\_training\_syn(i,:).'-mu2(:,j));

d1= -0.5 \* b1 \* a1 \* c1;

phi\_syn\_valid(i,j) = exp(d1);

end

end

%% Root mean square error for validation set

Err2syn= 0.5 \* ((Y\_validation\_syn-(phi\_syn\_valid\*w2)).')\*(Y\_validation\_syn-(phi\_syn\_valid\*w2));

%%

validPer2 = sqrt((2\*Err2syn)/noValidationSyn);

%%

%%

% Stochastic gradient models

eta1= zeros(1,noTrainDocs);

eta1(1,:)=1;

dw1=zeros(M1,noTrainDocs);

w01= 100\*rand(1,M1);

%%

prevw= w01;

%%

for i= 1: noTrainDocs

deltaed = -(Y\_training(i,:)- (prevw)\*phi(i,:).')\*phi(i,:);

deltae = deltaed + (lambda1\* prevw);

deltaw = (-1\*eta1(1,i)) \* deltae;

dw1(:,i) = deltaw;

prevw = prevw+ deltaw;

end

%%

w01= w01.';

%%

%synthetic data

% Stochastic gradient models

eta2= zeros(1,noTrainSyn);

eta2(1,:)=1;

dw2=zeros(M2,noTrainSyn);

w02= 100\*rand(1,M2);

%%

prevw2= w02;

%%

for i= 1: noTrainSyn

deltaed = -(Y\_training\_syn(i,:)- (prevw2)\*phi\_syn(i,:).')\*phi\_syn(i,:);

deltae = deltaed + (lambda2\* prevw2);

deltaw = (-1\*eta2(1,i)) \* deltae;

dw2(:,i) = deltaw;

prevw2 = prevw2+ deltaw;

end

%%

yplot=Y;

xplot1=X(:,1);

polyval(xplot1,yplot,M2);

xplot2=X(:,2);

xplot3=X(:,3);

xplot4=X(:,4);

xplot5=X(:,5);

xplot6=X(:,6);

xplot7=X(:,7);

xplot8=X(:,8);

xplot9=X(:,9);

yplottemp=smooth(xplot1,yplot);

plot(xplot1,yplottemp,'\*'); grid on;

% wplot1=w(:,1);

cftool;

%% hyper tuning

%%Create matrix and initialize to all zeros

load('synthetic.mat');

noTrainSyn = floor(0.9 \* length(x));

noValidationSyn= size(x,2)- noTrainSyn;

X = x.';

Y = t;

minvalidper=1000;

%%

%plot(X,Y);

%% Divide data into training, test and validation

X\_rand\_syn = randperm(length(X));

X\_rand\_syn = (X\_rand\_syn).';

trainingIndexes\_syn = X\_rand\_syn(1:noTrainSyn);

validationIndexes\_syn = X\_rand\_syn(noTrainSyn+1:end);

%%

X\_training\_syn = X(trainingIndexes\_syn,:);

Y\_training\_syn = Y(trainingIndexes\_syn,:);

trainInd2 = trainingIndexes\_syn;

%%

for M2 1:9

mu2 = zeros(10,M2);

Xforclosed\_syn = randperm(noTrainSyn,M2);

Xforclosed\_syn = (Xforclosed\_syn).';

% populate mu2 matrix

for i= 1 : M2

mu2(:,i) = X\_training\_syn(Xforclosed\_syn(i),:);

end

%% populate sigma\_syn

sigma\_syn = zeros(10:10);

sigma\_syn = var(X\_training\_syn)\*0.1;

for i=1:10

if sigma\_syn(1,i) < 0.0001

sigma\_syn(1,i) = 0.01;

end

end

sigma\_syn = diag(sigma\_syn);

D= 10;

Sigma2= zeros(D,D);

for i= 1:M2

Sigma2(:,:,i) = sigma\_syn;

end

%% Calculate design matrix for training set

phi\_syn = zeros(noTrainSyn,M2);

phi\_syn(:,1)= 1;

for j= 2 : M2

for i = 1 : noTrainSyn

a1= inv(Sigma2(:,:,j));

b1= (X\_training\_syn(i,:).'-mu2(:,j)).';

c1= (X\_training\_syn(i,:).'-mu2(:,j));

d1= -0.5 \* b1 \* a1 \* c1;

phi\_syn(i,j) = exp(d1);

end

end

%%

%lambda2= 0.2;

for lambda = 0.2:0.9

w2 = inv( lambda2\*eye(M2,M2)+ phi\_syn.'\*phi\_syn)\*phi\_syn.'\*Y\_training\_syn;

end

%% Root mean square error for training set

Err2= 0.5 \* ((Y\_training\_syn-(phi\_syn\*w2)).')\*(Y\_training\_syn-(phi\_syn\*w2));

trainPer2 = sqrt((2\*Err2)/noTrainSyn);

%%

%Validation data calculations

X\_validation\_syn = X(validationIndexes\_syn,:);

Y\_validation\_syn = Y(validationIndexes\_syn,:);

validInd2 = validationIndexes\_syn;

%% Calculate design matrix for training set

phi\_syn\_valid = zeros(noValidationSyn,M2);

phi\_syn\_valid(:,1)= 1;

for j= 2 : M2

for i = 1 : noValidationSyn

a1= inv(Sigma2(:,:,j));

b1= (X\_training\_syn(i,:).'-mu2(:,j)).';

c1= (X\_training\_syn(i,:).'-mu2(:,j));

d1= -0.5 \* b1 \* a1 \* c1;

phi\_syn\_valid(i,j) = exp(d1);

end

end

%% Root mean square error for validation set

Err2syn= 0.5 \* ((Y\_validation\_syn-(phi\_syn\_valid\*w2)).')\*(Y\_validation\_syn-(phi\_syn\_valid\*w2));

%%

validPer2 = sqrt((2\*Err2syn)/noValidationSyn);

if minvalidper > validper2

minmu2= mu2;

minSigma2= Sigma2;

minphi2=phi\_syn\_valid;

end

end

%%