**EE 5841 Machine Learning**

**Project 4**

**Kernel Logistic Regression**

**Derivation of the dual formulation for linear logistic regression:**

Let (,) denote the training set where ϵ and ϵ (-1,1)

Taking max log likelihood and adding the regularization term

Where

Taking

Taking Lagrangian

To obtain the equation in terms of α

Taking Partial derivative of the Lagrangian equation w.r.t. w

Taking Partial derivative of the Lagrangian equation w.r.t. b

Taking Partial derivative of the Lagrangian equation w.r.t.

From equation (5)

Taking

Taking partial derivative of Equation (9) w.r.t.

Substituting equation (8)

Substituting equation (8)

is a convex function, therefore is also a convex function

Therefore,

From equation (4)

Substituting equation (12)

From equation (9)

Substituting equation (10)

Substituting equation (7)

Substituting equation (6)

To maximize the above equation, we have to minimize the function of α

Therefore, the Dual formulation of the Linear Logistic Regression will be

Subject to

**Kernel Version of the Dual formation of logistic regression:**

From equation (15)

Substituting equation (6)

Subject to

Where is the Kernel Term

**Kernel Version of Pr(y|x) used the obtained dual variables:**

From equation (1)

Substituting equation (6)

Subject to

Where is the Kernel Term

Code:

clear all;

data = load('heartstatlog\_trainSet.txt');

labels = load('heartstatlog\_trainLabels.txt');

dataTest = load('heartstatlog\_testSet.txt');

labelTest = load('heartstatlog\_testLabels.txt');

data = bsxfun(@rdivide,bsxfun(@minus,data,mean(data)),std(data));

labels = 2\*(labels - 1.5);

labelTest = 2\*(labelTest - 1.5);

%Formulating Linear functions

kernel = data\*data';

kernelTest = data\*dataTest';

C = [0.01 0.1 0.25 1 5 25 100];

for j = 1:size(C,2)

k = 5;

Ntrain=length(data);

ind = randperm(Ntrain);

testInd=ind(1:floor(Ntrain/k))';

trainData = data;

trainLabels = labels;

trainData(testInd,:) = [];

trainLabels(testInd,:) = [];

testData = data(testInd,:);

testLabels = labels(testInd,:);

kernelCVTrain = kernel;

kernelCVTrain(testInd,:) = [];

kernelCVTrain(:,testInd) = [];

kernelCVTest = kernel(1:size(trainData,1),testInd);

kernelCVVal = kernel(1:size(testData,1),testInd);

kernelTrain = data(1:size(trainData,1),:)\*data(1:size(trainData,1),:)';

kernelTestCV = kernelTest(1:size(trainData,1),:);

alpha = ones(size(kernelCVTrain,1),1)\*(0.5)\*(1/C(j));

fun3 = @(alpha)objFun3(alpha,trainLabels,kernelCVTrain,C(j));

A = [];

b = [];

Aeq = trainLabels';

beq = 0;

lb = zeros(size(kernelCVTrain,1),1);

ub = C(j)\*ones(size(trainLabels,1),1);

opAlpha = fmincon(fun3,alpha,A,b,Aeq,beq,lb,ub);

supporters = find(opAlpha > 1e-5);

b = 0;

bias = @(b)objBias3(b,opAlpha,trainLabels,kernelCVTrain,C(j));

opB(j) = fminunc(bias,b);

predVal = sign((((opAlpha.\*trainLabels)/C(j))'\*kernelCVTest)' + opB(j));

errorsCVValid(:,j) = sum(predVal ~= testLabels)/length(testLabels);

predTrain = sign((((opAlpha.\*trainLabels)/C(j))'\*kernelTrain)' + opB(j));

errorsCVTrain(:,j) = sum(predTrain ~= labels(1:length(predTrain)))/length(predTrain);

predTest = sign((((opAlpha.\*trainLabels)/C(j))'\*kernelTestCV)' + opB(j));

errorsTestCV(:,j) = sum(predTest ~= labelTest)/length(labelTest);

end

[minErrorCV,lambdaInd] = min(errorsCVValid);

bestLambda = C(lambdaInd);

bestB = opB(lambdaInd);

%running on entire training data

A = [];

b = [];

Aeq = labels';

beq = 0;

lb = zeros(size(kernel,1),1);

ub = bestLambda\*ones(size(labels,1),1);

alpha = ones(size(kernel,1),1)\*(0.5)\*(1/bestLambda);

fun3 = @(alpha)objFun3(alpha,labels,kernel,bestLambda);

trainAlpha = fmincon(fun3,alpha,A,b,Aeq,beq,lb,ub);

supporters = find(trainAlpha > 1e-5);

b = 0;

bias = @(b)objBias3(b,trainAlpha,labels,kernel,bestLambda);

trainB = fminunc(bias,b);

pred = sign((((trainAlpha.\*labels)/bestLambda)'\*kernel)' + trainB);

errorsTrain = sum(pred ~= labels)/length(labels);

%running on testing data

pred = sign((((trainAlpha.\*labels)/bestLambda)'\*kernelTest)' + bestB);

errorsTest = sum(pred ~= labelTest)/length(labelTest);

figure

plot(log10(C),errorsCVTrain,'-x');

hold on

plot(log10(C),errorsCVValid,'-o');

hold on

plot(log10(C),errorsTestCV,'--gs','LineWidth',2,'MarkerSize',10,'MarkerEdgeColor','b');

legend('Train','Validation','Test');

xlabel('Lambda');

ylabel('Error');

Plot:

