CSE881 HW5

Nan Cao, A52871775

Oct 08th, 2016

Problem 1

(a)

Tree 1		Predicted	
		+	-
Actual	+	20	10
	-	10	20

Tree 2		Predicted	
		+	-
Actual	+	25	5
	-	20	10

(b)

$$error_1 = \frac{10+10}{60} = \frac{1}{3}$$
$$error_2 = \frac{5+20}{60} = \frac{5}{12}$$

Tree 1 has a lower training error.

(c)

$$\begin{aligned} Cost(&= Cost(Tata|tree) + Cost(Tree) \\ Cost_1 &= 3[log_2 4] + 4[log_2 2] + 20[log_2 60] \\ &= 6 + 4 + 120 = 130bits \\ Cost_2 &= 2[log_2 4] + 3[log_2 2]25[log_2 60] \\ &= 4 + 3 + 180 = 157bits \end{aligned}$$

So Tree 1 is better.

Problem 2

(a) Yes

$$\begin{aligned} \phi_1(\boldsymbol{x}) &= x_1 x_2 \\ \phi_2(\boldsymbol{x}) &= x_3 x_4 \\ \boldsymbol{w} &= (1, -1) \\ f(\boldsymbol{x}) &= x_1 x_2 - x_3 x_4 \end{aligned}$$

(b) No.
(c) No
(d) Yes

$$\begin{aligned} \phi_1(\boldsymbol{x}) &= x_1 x_2 \\ \boldsymbol{w} &= -1 \\ f(\boldsymbol{x}) &= -x_1 x_2 \end{aligned}$$

Problem 3

Linear support vector machine

		Predicted	
		+	-
Actual	+	2473	222
	-	315	1591

$$Error\ Rate = \frac{222+315}{4601} = 0.1167$$

(b)

Nonlinear support vector machine

		Predicted	
		+	-
Actual	+	2542	191
	-	174	1694

$$Error\ Rate = \frac{191+174}{4601} = 0.07933$$

Nonlinear support vector machine is more effective, because it has a smaller error rate.

```
% NAN CAO CSE881 HW5
clear;
% set dir in nan's win lap
% cd C:\Users\nan66\Dropbox\CSE881\HW5\;
% set dir in nan's linux lap
cd /home/nan/Dropbox/CSE881/HW5;
% set dir in remote server
% cd /CSE881/HW5;
A = load('spambase.data');
N = size(A, 1);
seed = 52871775; % seed for random number generator
rng(seed); % for repeatability of your experiment
A = A(randperm(N),:); % this will reshuffle the rows in matrix A
X=A(:,1:57);
Y=A(:,58);
%You need to compare the performance of the linear and non-linear support
%vector machine classifiers on this data set using nested cross-validation.
%Use k=10 for the outer loop (classifier evaluation) and k=5 for the inner
%loop (model selection) of the nested cross-validation procedure
N=length(A);
indices = crossvalind('Kfold',N,10);%outer loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Linear
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
lambda = logspace(-4,3,11); % create a set of candidate lambda values
cm3a=zeros(2,2); %create a bin to store the confusion matrix
bestLambda=0;%initialize the best lambda
for k=1:10
Lam=0
```

```
testID = (indices == k); %id of test data in kth round
trainID = ~testID; %id of train data in kth round
Xtrain = X(trainID,:);
Ytrain = Y(trainID);
Xtest = X(testID,:);
Ytest = Y(testID);
% tune the hyper-parameter lambda for linear SVM.
model = fitclinear(Xtrain, Ytrain, 'Kfold', 5, 'Learner', 'svm', 'Lambda',
    lambda);
foldNumber = 3; % to examine the model created for the 3rd fold
model.Trained{foldNumber}
ce = kfoldLoss(model) % to examine the classification error for each lambda
Lam=lambda(ce==min(ce));
bestLambda(k)=Lam(1);
SVMmodel = fitclinear(Xtrain, Ytrain, 'Learner', 'svm', 'Lambda',
    bestLambda(k));
pred = predict(SVMmodel, Xtest);
cp = classperf(Ytest);
classperf(cp,pred);
cp.DiagnosticTable % to show the confusion matrix
cm3a=cm3a+cp.DiagnosticTable;% sum the confusion matrix
cp.ErrorRate % to show the classification error
end;
cm3a
errorrate3a=trace(rot90(cm3a))/sum(cm3a(:))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Non-Linear
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
    %%%%--------------------------------------------------------------------
% for testing use only
%dlmwrite(Lam3b.txt,bestLambda);
%dlmwrite(Sig3b.txt,bestSigma);
%dlmwrite(indices.txt,indices);
%dlmwrite(A.txt,A);
%%%%%%%%%%%%%%%%%%%%%%%%%
% indices=load('indices.txt')
% bestLambda=load('Lam3b.txt')
% bestSigma=load('Sig3b.txt')
%
    %%%%--------------------------------------------------------------------
lambda = logspace(-3,3,11); % create a set of lambda values
sigma = logspace(-3,3,11); % create a set of kernel scale values
cvloss = zeros(11,11); % stores the CV error for each (lambda,sigma) pair
cm3b=zeros(2,2); %create a bin to store the confusion matrix
bestLambda=0;%initialize the bestlamda
for k=1:10
testID = (indices == k); %id of test data in kth round
trainID = ~testID; %id of train data in kth round
Xtrain = X(trainID,:);
Ytrain = Y(trainID);
Xtest = X(testID,:);
```

```
Ytest = Y(testID);
for i=1:11
for j=1:11
SVMmodel = fitcsvm(Xtrain, Ytrain, 'KernelFunction','RBF','KernelScale',
    sigma(j),'BoxConstraint',lambda(i),'Kfold', 5);
cvloss(i,j)=kfoldLoss(SVMmodel);
end;
end;
[a,b]=find(cvloss==min(cvloss(:)));
i1=a(1);% just in case if there are two or more min values
j1=b(1);
bestLambda(k)=lambda(i1);
bestSigma(k)=sigma(j1);
SVMmodel = fitcsvm(Xtrain, Ytrain,'KernelFunction','RBF','KernelScale',
    bestSigma(k),'BoxConstraint',bestLambda(k));
pred = predict(SVMmodel, Xtest);
cp = classperf(Ytest);
classperf(cp,pred);
cp.DiagnosticTable % to show the confusion matrix
cm3b=cm3b+cp.DiagnosticTable;% sum the confusion matrix
cp.ErrorRate % to show the classification error
end;
cm3b % confusion matrix
errorrate3b=trace(rot90(cm3b))/sum(cm3b(:)) % errorrate
```