**IE-5300** *1001099696*

**Homework-2** *Praveen Suguru*

1. **Generalized Function of KNN classifier:**

function Lpred=myknn(Dtrain,Ltrain,Dtest,K,KNNopt,Dorder)

C = unique(Ltrain);

r=Dorder;

Lpred = [];

for iC=1:length(C)

cl=C(iC);

idx=find(Ltrain==cl);

eval(['dclass',num2str(cl),'=Dtrain(idx,:);']);

end

if KNNopt==1

for i=1:size(Dtest)

Ftest=Dtest(i,:);

DKave=[];

for iC=1:length(C)

cl=C(iC);

Feat=eval(['dclass',num2str(cl)]);

Ns = size(Feat,1);

dmat=abs(Feat-repmat(Ftest,Ns,1));

dlist=nthroot(sum(dmat.^r,2),r);

dsort=sort(dlist);

dKave=mean(dsort(1:K));

DKave=[DKave;dKave,cl];

end

[vmin,imin]=min(DKave(:,1));

Cpred=DKave(imin,2);

Lpred = [Lpred;Cpred];

end

end

if KNNopt==2

for i=1:size(Dtest)

Ftest=Dtest(i,:);

DKave=[];

for iC=1:length(C)

cl=C(iC);

Feat=eval(['dclass',num2str(cl)]);

Ns = size(Feat,1);

dmat=abs(Feat-repmat(Ftest,Ns,1));

dlist=nthroot(sum(dmat.^r,2),r);

dsort=sort(dlist);

dKave=mean(dsort(1:K));

DKave=[DKave;dKave,cl];

end

[vmin,imin]=min(DKave(:,1));

Cpred=DKave(imin,2);

Lpred = [Lpred;Cpred];

end

end

1. **N-fold Cross Validation Function:**

function data\_nfold = divide\_nfold\_data(feature, label, N)

C=unique(label);

for iC = 1:length(C)

cl = C(iC);

idx = find(label==cl);

data = feature(idx,:);

L = length(idx);

feat\_nfold = nfold\_set(data,N);

eval(['data\_nfold.class',num2str(cl), '=feat\_nfold;']);

end

function feat\_nfold = nfold\_set(feat,N)

L = size(feat,1);

n = floor (L/N);

rem = mod(L,N);

a = n\*ones(N,1);

if rem>0

b = nchoosek(1:N,rem);

c=ceil(rand\*size(b,1));

idx = b(c,:);

a(idx) = a(idx)+1;

end

nfoldpt=[0;cumsum(a)];

nint = [nfoldpt(1:end-1)+1,nfoldpt(2:end)];

for i = 1:N

dsub = feat(nint(i,1):nint(i,2),:);

eval(['feat\_nfold.fold',num2str(i),'=dsub;']);

end

1. **KNN classifier for IRIS dataset:**

**Input:**

N = 10;

C = unique(label);

ACC\_nfold = [];

data\_nfold = divide\_nfold\_data(feat, label, N);

k= [3,5,7,9];

KNNopt\_v = [1:2];

Dorder\_v = [1,2,10];

% fprintf(KNN\_res\_2,'%4s %8s %8s %12s %12s\n','K','Dorder', 'KNNopt', 'MeanACC', 'std\_ACC');

for K=k

for KNNopt= KNNopt\_v

for Dorder= Dorder\_v

for ifold = 1:N

% Formulate training and testing

idx\_test = ifold; idx\_train = setdiff(1:N, ifold);

Dtest = []; Ltest = [];

Dtrain= []; Ltrain = [];

for iC = 1:length(C);

cl = C(iC);

dtest = eval(['data\_nfold.class', num2str(iC), '.fold', num2str(ifold)]);

Dtest = [Dtest; dtest];

Ltest = [Ltest; cl\*ones(size(dtest, 1), 1)];

for itr=1:length(idx\_train)

idx = idx\_train(itr);

dtrain = eval(['data\_nfold.class', num2str(iC), '.fold', num2str(idx)]);

Dtrain = [Dtrain; dtrain];

Ltrain = [Ltrain; cl\*ones(size(dtrain, 1), 1)];

end

end

% KNN classification

Lpred = myknn(Dtrain, Ltrain, Dtest, K, KNNopt, Dorder);

acc = sum(Lpred==Ltest)/length(Ltest);

ACC\_nfold = [ACC\_nfold; acc];

end

ACC\_mean = mean(ACC\_nfold);

ACC\_std= std(ACC\_nfold);

result\_combin= [K; Dorder; KNNopt; ACC\_mean; ACC\_std];

fprintf('%4.0f %8.0f %8.0f %12.2f %12.2f\n',result\_combin)

end

end

end

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| K | Dorder | KNNopt | ACC\_mean | ACC\_std |
| 3 | 1 | 1 | 0.96 | 0.05622 |
| 3 | 2 | 1 | 0.96333 | 0.05061 |
| 3 | 10 | 1 | 0.96667 | 0.04874 |
| 3 | 1 | 2 | 0.965 | 0.05004 |
| 3 | 2 | 2 | 0.96533 | 0.04901 |
| 3 | 10 | 2 | 0.96667 | 0.04832 |
| 5 | 1 | 1 | 0.96571 | 0.04781 |
| 5 | 2 | 1 | 0.96583 | 0.04743 |
| 5 | 10 | 1 | 0.96667 | 0.04714 |
| 5 | 1 | 2 | 0.966 | 0.0469 |
| 5 | 2 | 2 | 0.96606 | 0.0467 |
| 5 | 10 | 2 | 0.96667 | 0.04654 |
| 7 | 1 | 1 | 0.96615 | 0.0464 |
| 7 | 2 | 1 | 0.96619 | 0.04628 |
| 7 | 10 | 1 | 0.96711 | 0.04553 |
| 7 | 1 | 2 | 0.96667 | 0.04548 |
| 7 | 2 | 2 | 0.96667 | 0.04544 |
| 7 | 10 | 2 | 0.96741 | 0.04484 |
| 9 | 1 | 1 | 0.96737 | 0.04483 |
| 9 | 2 | 1 | 0.96733 | 0.04483 |
| 9 | 10 | 1 | 0.96762 | 0.04434 |
| 9 | 1 | 2 | 0.96758 | 0.04436 |
| 9 | 2 | 2 | 0.96754 | 0.04438 |
| 9 | 10 | 2 | 0.96778 | 0.04396 |

**Highest Classification Performance:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| K | Dorder | KNNopt | ACC\_mean | ACC\_std |
| 9 | **10** | **2** | **0.96778** | **0.04396** |

1. **Simple Decision Tree:**

**Input:**

function predL=dtree(used\_feat,x,y)

predL=[];

PetalL=used\_feat(:,1);

PetalW=used\_feat(:,2);

Ns=size(used\_feat,1);

for i = 1:Ns

PetalL = used\_feat(i,1);

PetalW = used\_feat(i,2);

if PetalL < 2.5

predL(i)=1;

else

if PetalW <= x

if PetalL > y

predL(i) = 3;

else

predL(i) = 2;

end

else

predL(i) = 3;

end

end

end

load iris.mat

gplotmatrix(feat(:,3),feat(:,4),label);

xlabel('Petal Length(cm)','fontsize',16,'fontweight','b');

ylabel('Petal Length(cm)','fontsize',16,'fontweight','b');

used\_feat = feat(:,3:4);

predL=dtree(used\_feat,1.70,5.3);

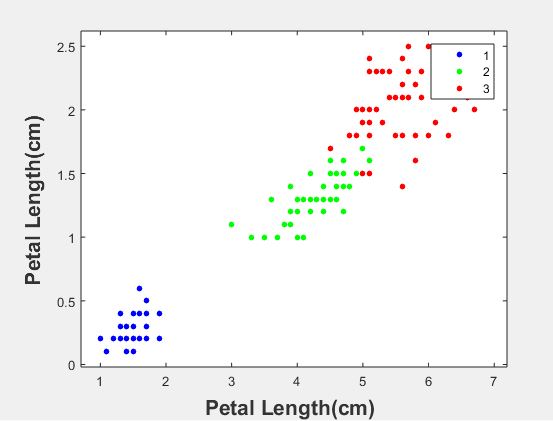
Lpred= (predL)'

ACC = sum(Lpred==label)/length(label);

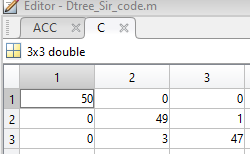
[C,order]=confusionmat(label,Lpred);

**Output:**

**Accuracy = 97.33%**

****

**Confusion Matrix:**



Interpretation:

Class 1= accuracy is **100%** - all the **50** samples have been segregated perfectly

Class 2= accuracy is **98%** - **49** samples are segregated perfectly and **1 is error**

Class 3= accuracy is **94%** - **47** samples are segregated perfectly and **3 are errors**

**Decision Tree:**

