MATLAB m-flies for –

**SYSTEMS ANALYTICS: Adaptive Machine Learning WORKBOOK**

m-files in this folder are referenced in my book.

**Disclaimer:**

* Code is provided “as is”. NO support for debugging or further explanation.
* Code is written in a “linear” fashion (= “lame”! ☺) so that it is self-explanatory!
* Write out the steps in the text of the book and use m-file contents as a “companion” to deepen your understanding of the text!

If you create improved versions of my m-files, kindly share it with me (pgmadbiz@gmail.com) so that I can include them in the 2nd edition of the book. Many thanks!

Enjoy!

PG

July 2016

**m-files follow . . .**

% NAIVE BAYES CLASSIFIER

clear

tic

disp('--- start ---')

distr='normal';

%distr='kernel';

%

A=xlsread('Data');

B=A(:,[1,2,3,4]);

Bm=mean(B);

Bs=std(B);

%

B=[B(:,1)-Bm(1),B(:,2)-Bm(2),B(:,3)-Bm(3),B(:,4)-Bm(4)];

%

B=[B(:,1)/Bs(1),B(:,2)/Bs(2),B(:,3)/Bs(3),B(:,4)/Bs(4)];

%

Clabel=A(:,5);

parallelcoords(B,'Group',Clabel)

%

X=B';

%

R=X\*X'/(size(X,2));

%

[Q ev]=eig(R);

ev=fliplr(diag(ev)');

figure(1); plot(ev); title('Eigenvalue');

Q=fliplr(Q);

%

K=3

%K=input('Signal subspace dim, "K"? ');

%

F1=X'\*Q(:,1);

F2=X'\*Q(:,2);

F3=X'\*Q(:,3);

%F4=X'\*Q(:,4);

F=[F1,F2,F3];

RF=(F'\*F)/(size(F,1));

%

% Training & Test Sets - FEATURES

%

Train=zeros(90,4);

TestL=zeros(60,4);

Test=zeros(60,3);

%

for i=1:150

 if i>=1 & i<=30 Train(i,:)=[F(i,:),Clabel(i)];end;

 if i>=51 & i<=80 Train(i-20,:)=[F(i,:),Clabel(i)];end;

 if i>=101 & i<=130 Train(i-40,:)=[F(i,:),Clabel(i)];end;

%

 if i>=31 & i<=50 TestL(i-30,:)=[F(i,:),Clabel(i)];end;

 if i>=81 & i<=100 TestL(i-60,:)=[F(i,:),Clabel(i)];end;

 if i>=131 & i<=150 TestL(i-90,:)=[F(i,:),Clabel(i)];end;

end;

%

Test=TestL(:,1:3);

%

%

% Create a training set

x = Train(:,1:3);

y = Train(:,4);

% test set

u=Test;

v=TestL(:,4);

yu=unique(y);

nc=length(yu); % number of classes

ni=size(x,2); % independent variables

ns=length(v); % test set

% compute class probability

for i=1:nc

 fy(i)=sum(double(y==yu(i)))/length(y);

end

switch distr

 case 'normal'

 % normal distribution

 % parameters from training set

 for i=1:nc

 xi=x((y==yu(i)),:);

 mu(i,:)=mean(xi,1);

 sigma(i,:)=std(xi,1);

 end

 % probability for test set

 for j=1:ns

 fu=normcdf(ones(nc,1)\*u(j,:),mu,sigma);

 P(j,:)=fy.\*prod(fu,2)';

 end

 case 'kernel'

 % kernel distribution

 % probability of test set estimated from training set

 for i=1:nc

 for k=1:ni

 xi=x(y==yu(i),k);

 ui=u(:,k);

 fuStruct(i,k).f=ksdensity(xi,ui);

 end

 end

 % re-structure

 for i=1:ns

 for j=1:nc

 for k=1:ni

 fu(j,k)=fuStruct(j,k).f(i);

 end

 end

 P(i,:)=fy.\*prod(fu,2)';

 end

 otherwise

 disp('invalid distribution stated')

 return

end

% get predicted output for test set

[pv0,id]=max(P,[],2);

for i=1:length(id)

 pv(i,1)=yu(id(i));

end

% compare predicted output with actual output from test data

confMat=myconfusionmat(v,pv);

disp('confusion matrix:')

disp(confMat)

conf=sum(pv==v)/length(pv);

disp(['accuracy = ',num2str(conf\*100),'%'])

%

figure; plot(pv)

err=TestL(:,4)-pv(:);

testmse=(err'\*err)/size(err,1)

%

figure

gscatter(Test(:,1),Test(:,2),TestL(:,4));

h = gca;

xylim = [h.XLim h.YLim]; % Get current axis limits

hold on

%

for i=1:nc

ezcontour(@(x1,x2)mvnpdf([x1,x2],mu(i,1:2),sigma(i,1:2)),xylim+0.5\*[-1,1,-1,1],60)

end;

%

title('Naive Bayes')

xlabel('Feature #1')

ylabel('Feature #2')

hold off

%

Toc

%

% iris\_SB.m

% Simple Boost

%

Cplr\_MLR=xlsread('MLRtr');

Cp\_MLR=xlsread('MLRts');

Cplr\_DT=xlsread('DTtr');

Cp\_DT=xlsread('DTts');

%

subplot(2,2,1)

plot(Cplr\_MLR);

title('MLR Training')

%

subplot(2,2,2)

plot(Cp\_MLR);

title('MLR Test')

%

subplot(2,2,3)

plot(Cplr\_DT);

title('DT Training')

%

subplot(2,2,4)

plot(Cp\_DT);

title('DT Test')

%

Cl\_tr= xlsread('CLtr');

Cl\_ts= xlsread('CLts');

%

Tr=[ones(90,1),Cplr\_MLR,Cplr\_DT];

Tt=[ones(60,1),Cp\_MLR,Cp\_DT];

%

% Pseudo-Inverse

\*

w=inv(Tr'\*Tr)\*Tr'\*Cl\_tr;

%

Cptr=Tr\*w;

%

figure;

subplot(1,2,1)

plot(Cptr)

title('SB Train')

err=Cl\_tr-Cptr(:);

trainmse=(err'\*err)/size(err,1)

%

Cpts=Tt\*w;

%

subplot(1,2,2)

plot(Cpts)

title('SB Test')

err=Cl\_ts-Cpts(:);

testmse=(err'\*err)/size(err,1)

%

%

% iris\_RLS.m

% includes Kernel method

%

A=xlsread('Data');

B=A(:,[1,2,3,4]);

Bm=mean(B);

Bs=std(B);

%

B=[B(:,1)-Bm(1),B(:,2)-Bm(2),B(:,3)-Bm(3),B(:,4)-Bm(4)];

%

B=[B(:,1)/Bs(1),B(:,2)/Bs(2),B(:,3)/Bs(3),B(:,4)/Bs(4)];

%

Clabel=A(:,5)';

%

% Data Matrix is MxN where M is the # FEATURES!

%

X=B';

X3=X(1:3,:);

%figure(1);scatter(X3(1,:),X3(2,:));

%

%

R3=X3\*X3'/(size(X3,2));

%

[Q ev] = eig(R3);

ev=fliplr(diag(ev)')

Q=fliplr(Q);

%

Qp=Q(:,1:2);

P=Qp\*inv(Qp'\*Qp)\*Qp';

%

f3=P\*X3;

%figure(2); scatter(f3(1,:),f3(2,:));

%

% Training & Test Sets

%

F=f3;

%

Train=zeros(4,90);

TestL=zeros(4,60);

Test=zeros(3,60);

%

for j=1:150

 if j>=1 & j<=30 Train(:,j)=[F(:,j);Clabel(j)];end;

 if j>=51 & j<=80 Train(:,j-20)=[F(:,j);Clabel(j)];end;

 if j>=101 & j<=130 Train(:,j-40)=[F(:,j);Clabel(j)];end;

%

 if j>=31 & j<=50 TestL(:,j-30)=[F(:,j);Clabel(j)];end;

 if j>=81 & j<=100 TestL(:,j-60)=[F(:,j);Clabel(j)];end;

 if j>=131 & j<=150 TestL(:,j-90)=[F(:,j);Clabel(j)];end;

end;

%

Test=TestL(1:3,:);

% Randomize Training Set

T=size(Train,2); k = randperm(T);

Train=Train(:,k(1:T));

Cptr=zeros(1,90);

Cp=zeros(1,60);

%

% RLS

%

XX=Train(1:3,1:T);

M=size(XX,1);

wo=zeros(M,1); wn=zeros(M,1);

Po=zeros(M,M);Pn=zeros(M,M);

lambda=0.1;

Po=(1/lambda)\*eye;

%

for n=1:T

 xx=XX(:,n)\*XX(:,n)';

 Nu=Po\*xx\*Po;

 Q=XX(:,n)'\*Po\*XX(:,n);

 De=1+Q;

 Pn=Po - (Nu/De);

 %

 g=Pn\*XX(:,n);

 al=Train(4,n) - wo'\*XX(:,n);

 wn=wo + g\*al;

 % Update old

 Po=Pn; wo=wn;

 % Predict

 Cptr(n)=wn'\*XX(:,n);

 %

 errTR(n)=Train(4,n)- Cptr(n);

end;

%

figure(10);plot(errTR)

trainmse=(errTR\*errTR')/size(errTR,2)

%

% Predict using regression coefficients on the Test Set

%

for n=1:60

Cp(n)=wn'\*Test(:,n);

errTS(n)=TestL(4,n)-Cp(n);

end;

%

figure(11);plot(Cp+2)

testmse=(errTS\*errTS')/size(errTS,2)

%

% KERNEL Method

%

m=3; % Feature vector dim

h=1.5;

for n=1:60

 num=0;

 den=0;

 for i=1:T

 nor=(Test(:,n)-Train(1:3,i))'\*(Test(:,n)-Train(1:3,i));

 term=(1/(h^m\*sqrt(2\*pi)))\*exp(-0.5\*(nor/h^2));

 num=num+Train(4,i)\*term;

 den=den+term;

 end;

 Cpk(n)=num/den;

end;

figure(12);plot(Cpk); hold; plot(TestL(4,:));hold;

errTK=TestL(4,:)-Cpk;

testmseK=(errTK\*errTK')/size(errTK,2)

%

% "EXTREME" Learning Machine

%

elmTR=fliplr(Train');

elmTS=fliplr(TestL');

%

[TrainingTime, TestingTime, TrainingAccuracy, TestingAccuracy]=elm(TrainingData\_File, TestingData\_File, Elm\_Type, NumberofHiddenNeurons, ActivationFunction);

%

%

% iris\_RPML.m

%

A=xlsread('Data');

B=A(:,[1,2,3,4]);

Bm=mean(B);

Bs=std(B);

%

B=[B(:,1)-Bm(1),B(:,2)-Bm(2),B(:,3)-Bm(3),B(:,4)-Bm(4)];

%

B=[B(:,1)/Bs(1),B(:,2)/Bs(2),B(:,3)/Bs(3),B(:,4)/Bs(4)];

%

Clabel=A(:,5)';

%

% Data Matrix is MxN where M is the # FEATURES!

%

X=B';

X3=X(1:3,:);

%figure(1);scatter(X3(1,:),X3(2,:));

%

%

R3=X3\*X3'/(size(X3,2));

%

[Q ev] = eig(R3);

ev=fliplr(diag(ev)')

Q=fliplr(Q);

%

Qp=Q(:,1:2);

P=Qp\*inv(Qp'\*Qp)\*Qp';

%

f3=P\*X3;

%figure(2); scatter(f3(1,:),f3(2,:));

%

% Training & Test Sets

%

F=f3;

%

Train=zeros(4,90);

TestL=zeros(4,60);

Test=zeros(3,60);

%

for j=1:150

 if j>=1 & j<=30 Train(:,j)=[F(:,j);Clabel(j)];end;

 if j>=51 & j<=80 Train(:,j-20)=[F(:,j);Clabel(j)];end;

 if j>=101 & j<=130 Train(:,j-40)=[F(:,j);Clabel(j)];end;

%

 if j>=31 & j<=50 TestL(:,j-30)=[F(:,j);Clabel(j)];end;

 if j>=81 & j<=100 TestL(:,j-60)=[F(:,j);Clabel(j)];end;

 if j>=131 & j<=150 TestL(:,j-90)=[F(:,j);Clabel(j)];end;

end;

%

Test=TestL(1:3,:);

% Randomize Training Set

T=size(Train,2); k = randperm(T);

Train=Train(:,k(1:T));

Cptr=zeros(1,90);

Cp=zeros(1,60);

%

% RPML Method

%

m=3; % Feature vector dim

M=10; % # hidden nodes

wi=rand(m,M);

b=rand(M,1);

%

D1=zeros(M,90);

for j=1:90

 D1(:,j)=b+wi'\*Train(1:3,j);

end;

D1=D1';

%

D=zeros(90,M);

for i=1:90

 for j=1:M

 D(i,j)=1/(1+exp(-D1(i,j)));

 end;

end;

%

wo=inv(D'\*D)\*D'\*Train(4,:)';

%

D1T=zeros(M,60);

for j=1:60

 D1T(:,j)=b+wi'\*Test(:,j);

end;

D1T=D1T';

%

DT=zeros(60,M);

for i=1:60

 for j=1:M

 DT(i,j)=1/(1+exp(-D1T(i,j)));

 end;

end;

Cpk=DT\*wo;

figure(12);plot(Cpk); hold; plot(TestL(4,:));hold;

errTK=TestL(4,:)-Cpk';

testmseK=(errTK\*errTK')/size(errTK,2)

% dbmmc.m

%

% data=dbmoon(N,d,r,w)

% N: # of samples each class

% d: seperation of two class, negative value means overlapping (default=1)

% r: radius (default=10),

% w: width of ring (default=6)

%

N=200;

dbd=dbmoon(N/2,-5,10,2);

dbd=dbd';

%

x1=dbd(1:2,:);d1=dbd(3,:);

data=[x1;d1];

M=size(data,1);

%

% Markov Chain

%

mu=[1 0]; % initial distribution

P=[[.6 .4]; [.4 .6];]; % transition matrix

n=N-1; % number of time steps to take

x=zeros(1,n+1); % clear out any old values

t=0:n; % time indices

x(1)=rando(mu); % generate first x value (time 0, not time 1)

for i=1:n,

 x(i+1) = rando(P(x(i),:));

end

%

%plot(t, x, '\*');

%axis([0 n 0 (length(mu)+1)]);

%

L=zeros(N/2,2);

%

L(:,1)=linspace(1,N/2,N/2);

L(:,2)=linspace((N/2)+1,N,N/2);

%

P=1;Q=1;

kr=x;

Ind=zeros(N,1);

%

for n=1:N;

%

K=kr(n);

if P>N/2;P=1;end; if Q>N/2;Q=1;end;

if K==1; Ind(n)=L(P,K);P=P+1; else Ind(n)=L(Q,K);Q=Q+1; end;

%

end;

%

%

%

%Ind=linspace(1,2000,2000)'; % NO sort

% Sort

dats=data(:,Ind);

%

for i=1:M; dats(i,:)=dats(i,:)-mean(dats(i,:));end;

%

x=dats(1:2,:);d=dats(3,:);

%

figure(1);

subplot(3,1,1);plot(x');title('MC-randomized Inputs');

subplot(3,1,2); plot(d);title('Desired Signal');

subplot(3,1,3); scatter(x(1,:),x(2,:));title('Double Moon');

%

%

% kpkf.m

% Random Proj with KALMAN Filter & Smoother

%

%

% State-space model:

%

% s[n] = A s[n-1] + q[n-1]

% y[n] = H[n] s[n] + r[n]

%

% Number of States = M = Attribute/Feature vector dimension

%

% Random Projection method

%

tic

N=size(x,2);

M=20; % # hidden nodes

%

H=zeros(N,M);

ypr=zeros(1,N);

ysm=zeros(1,N);

A=eye(M,M);

k1=0.1;

Q=k1\*eye(M);

k2=100;

r=k2;

k3=0.01;

Pf=zeros(M,M,N);

Pf(:,:,1)=(1/k3)\*eye(M);

Pp=zeros(M,M,N);

sf=zeros(M,N);

sp=zeros(M,N);

v=zeros(1,N);

K=zeros(M,1);

%

PP=zeros(M,M,N);

PF=Pf;

PS=zeros(M,M,N);

sP=zeros(M,N);

sF=sf;

sS=zeros(M,N);

G=zeros(M,M);

%

%

%

D1=zeros(M,N);

D=zeros(N,M);

%

del=2; % Lagged Desired Response

tch=zeros(del,N);

m=size(x,1)+del; % Feature vector dim

k=1;

wi=k\*rand(m,M);

b=k\*rand(M,1);

%

for t=1:N;

 %

 if t > (del); tch(:,t)=[-d(t-1);-d(t-2)];end;

 xi=[x; tch];

 %xi=x; % NO lagged Teacher Forcing

 m=size(xi,1); % Feature vector dim

%

 D1(:,t)=b+wi'\*xi(:,t);

%

 for j=1:M;D(t,j)=1/(1+exp(-D1(j,t)));end;

%

 H(t,:)=D(t,:);

%

% UPDATES

%

 for n=2:t;

 sp(:,n)=A\*sf(:,n-1);

 Pp(:,:,n)=A\*Pf(:,:,n-1)\*A'+Q;

 yprd(n)=H(n,:)\*sp(:,n);

 %

 v(n)=d(n)-H(n,:)\*sp(:,n-1);

 E=H(n,:)\*Pp(:,:,n)\*H(n,:)'+r;

 K=Pp(:,:,n)\*H(n,:)'\*inv(E);

 sf(:,n)=sp(:,n)+K\*v(n);

 Pf(:,:,n)=Pp(:,:,n)-K\*E\*K';

 yflt(n)=H(n,:)\*sf(:,n);

 end;

%

% Smoothing

%

 PF=Pf; sF=sf;

 sS(:,t)=sF(:,t); PS(:,:,t)=PF(:,:,t);

%

 for n=t-1:-1:1;

 %

 sP(:,n+1)=A\*sF(:,n);

 PP(:,:,n+1)=A\*PF(:,:,n)\*A' + Q;

 G=PF(:,:,n)\*A'\*inv(PP(:,:,n+1));

 %

 sS(:,n)=sF(:,n)+G\*(sS(:,n+1)-sP(:,n+1));

 PS(:,:,n)=PF(:,:,n)+G\*(PS(:,:,n+1)-PP(:,:,n+1))\*G';

 ysm(n)=H(n,:)\*sS(:,n);

 %

 end;

%

end;

%

yclass=yprd;

%

Toc

%

% kptv.m

% Random Proj with Time-Varying KALMAN Filter & Smoother

% --> Young KF&S

%

% State-space model:

%

% s[n] = A s[n-1] + q[n-1]

% y[n] = H[n] s[n] + r[n]

%

% Number of States = M = Attribute/Feature vector dimension

%

% Random Projection method

%

tic

N=size(x,2);

M=20; % # hidden nodes

%

% Create A & D matrix

al=0.3;

bt=1;gm=1;et=1;dl=0;

Ai=[al bt; 0 gm];

%

tmp = repmat({Ai},M,1);

A = blkdiag(tmp{:});

%

Di=[dl 0; 0 et];

tmp = repmat({Di},M,1);

Dd = blkdiag(tmp{:});

%

H=zeros(N,2\*M);

ypr=zeros(1,N);

ysm=zeros(1,N);

k1=0.1;

Q=k1\*eye(2\*M);

k2=100;

r=k2;

k3=0.01;

Pf=zeros(2\*M,2\*M,N);

Pf(:,:,1)=(1/k3)\*eye(2\*M);

Pp=zeros(2\*M,2\*M,N);

sf=zeros(2\*M,N);

sp=zeros(2\*M,N);

v=zeros(1,N);

K=zeros(2\*M,1);

%

PP=zeros(2\*M,2\*M,N);

PF=Pf;

PS=zeros(2\*M,2\*M,N);

sP=zeros(2\*M,N);

sF=sf;

sS=zeros(2\*M,N);

L=zeros(2\*M,N);

%

%

%

D1=zeros(M,N);

D=zeros(N,M);

%

del=2; % Lagged Desired Response

tch=zeros(del,N);

m=size(x,1)+del; % Feature vector dim

k=1;

wi=k\*rand(m,M);

b=k\*rand(M,1);

%

for t=1:N;

 %

 if t > (del); tch(:,t)=[-d(t-1);-d(t-2)];end;

 xi=[tch; x];

 %xi=x; % NO lagged Teacher Forcing

 m=size(xi,1); % Feature vector dim

%

 D1(:,t)=b+wi'\*xi(:,t);

%

 for j=1:M;D(t,j)=1/(1+exp(-D1(j,t)));end;

%

 for j=1:M; J=(j-1)\*2+1; H(t,J:(J+1))=[D(t,j) 0]; end;

%

% UPDATES

%

 for n=2:t;

 sp(:,n)=A\*sf(:,n-1);

 Pp(:,:,n)=A\*Pf(:,:,n-1)\*A'+ Dd\*Q\*Dd';

 yprd(n)=H(n,:)\*sp(:,n);

 %

 E=Pp(:,:,n)\*H(n,:)'\*inv(1+H(n,:)\*Pp(:,:,n)\*H(n,:)');

 sf(:,n)=sp(:,n)+E\*(d(n)-H(n,:)\*sp(:,n));

 Pf(:,:,n)=Pp(:,:,n)-E\*H(n,:)\*Pp(:,:,n);

 yflt(n)=H(n,:)\*sf(:,n);

 end;

%

%

% Smoothing

%

 PF=Pf; sF=sf;

 sS(:,t)=sF(:,t); PS=PF;

%

 for n=t-1:-1:2;

 %

 sS(:,n)=sF(:,n) - PF(:,:,n)\*A'\*L(:,n);

 L(:,n-1)=(eye(2\*M,2\*M)-PF(:,:,n)\*H(n,:)'\*H(n,:))'\*(A'\*L(:,n) - H(n,:)'\*(d(n)-H(n,:)\*A\*sF(:,n-1)));

 %

 PP(:,:,n+1)=A\*PF(:,:,n)\*A' + Dd\*Q\*Dd';

 G=PF(:,:,n)\*A'\*inv(PP(:,:,n+1));

 PS(:,:,n)=PF(:,:,n)+G\*(PS(:,:,n+1)-PP(:,:,n+1))\*G';

 %

 ysm(n)=H(n,:)\*sS(:,n);

 %

 end;

%

sf(:,t)=sS(:,t); % OPTIMIZATION!!!!

Pf(:,:,t)=PS(:,:,t);

end;

%

yclass=yprd;

%

Toc