clear all;

clc;

PI=pi;

phiqdata=xlsread('J:\Universita di Bologna\sv sir files 28 jan\trainingskipjumpHMMsetsepoxyvoidfile15phiqmax5deg.xlsx');

symb=21;

Nsamp=30;

for j=1:Nsamp

k=1;

for i=1:2:143

PHIp(j,k)=phiqdata(j,i);

Qp(j,k)=phiqdata(j,i+1);

Np(j,k)=(phiqdata(j,i+1))/2;

k=k+1;

end

end

zz=k-1;

for j=1:Nsamp

for k=1:zz

qsum=sum(Qp(j,k))/72;

if qsum~=0

Qpmod(j,k)=Qp(j,k)/qsum; %Normalizing Q As suggested by Dr.Cavallini

end

end

end

wind=72;

%----------------------------%

tot=wind\*Nsamp; % total no of phi, q sets

z=0;

ex = 18;

data=phiqdata;

testdata(1:Nsamp,:) = phiqdata(1:Nsamp,:);

w(1,:) = data(1,:);

w(2,:)= data(3,:);

w(3,:) = data(5,:);

w(4,:)= data(6,:);

w(5,:) = data(8,:);

w(6,:) = data(10,:);

w(7,:)= data(11,:);

w(8,:)= data(13,:);

w(9,:) = data(15,:);

w(10,:) = data(16,:);

w(11,:) = data(18,:);

w(12,:) = data(20,:);

w(13,:) = data(21,:);

w(14,:) = data(23,:);

w(15,:) = data(25,:);

w(16,:) = data(26,:);

w(17,:) = data(28,:);

w(18,:) = data(30,:);

w\_o = zeros(size(w)); % create a zero matrix of same dimentions of weight

dif = w - w\_o; % initialise difference.

count = 1; % initialise iteration count.

while sum(sum(dif)) ~= 0 && count ~= 500

w\_o = w; % remember the weights of previous iterations.

for ii = 1:Nsamp

for jj = 1:ex

eq\_dist(jj) = ((testdata(ii,:)-w(jj,:)) \* ((testdata(ii,:)-w(jj,:))')) ; % equiledian distance

end

[temp,near\_class(ii)] = min(eq\_dist); % find the cluster which is in minimum distance from the training exempler.

end

for ii = 1:ex

[a,b] = find(near\_class == ii);

temp\_sum = 0;

for jj = b

temp\_sum = temp\_sum + testdata(jj,:);

end

if sum(a) == 0

count;

ii;

end

w(ii,:) = temp\_sum / sum(a);

end

dif = abs(w - w\_o);

count = count+1;

%------min dist----------%

end

clear g

for i=1:ex

for j=1:Nsamp

eq\_dist(j)=sqrt((w(i,:)-testdata(j,:))\*((w(i,:)-testdata(j,:))'));

end

[temp,j1]=min(eq\_dist);

g(i,:)=testdata(j1,:);

gg=g;% store the sample which is at a minimum distance

end

% Normalizing the trained values of Q and generating Observations from the 18 % trained samples (To use in HMM)🡪 Observations are generated by discretizing % Qpmod(=Qp/Qsum) into 21 symbols 0 to 1 in intervals of 0.05 so that we will % have an observation sequence which has only 21 symbols.

for j=1:ex

k=1;

for i=1:2:143

PHIp(j,k)=g(j,i);

Qp(j,k)=g(j,i+1);

%Np(j,k)=(phiqdata(j,i+1))/2;

k=k+1;

end

end

zz=k-1;

for j=1:ex

qsum=sum(Qp(j,:));

for i=1:72

if qsum~=0

Qpmod(j,:)=Qp(j,:)/qsum;

zz=0;

for k=0:0.05:1

zz=zz+1;

if Qpmod(j,i)>=k

Observ(j,i)=zz;

end

end

else

Observ(j,i)=1;

end

end

end

for i=1:ex

l=1;d=1;

for j=1:144

g1(i,l,d)=g(i,j);

d=d+1;

if d==3

d=1;

l=l+1;

end

end

end

wind=72;

ex=18;

for k=1:ex

clear g;

g(:,:)=g1(k,:,:);

st=4;

for i=1:st

c(i,:)=g(i,:);

end

cl=1;

while cl==1

cl=1;

for i=1:st

for j=1:wind

dm(i,j)=sqrt((c(i,:)-g(j,:))\*((c(i,:)-g(j,:))'));

end

end

sm=zeros(st,wind);

[temp,temp1]=min(dm);

for i=1:wind

sm(temp1(i),i)=1;

end

tsm=sum(sm,2);

for i=1:st

if tsm(i,1)>1

temp2=0;

for j=1:wind

if sm(i,j)==1

temp2=temp2+g(j,:);

end

end

c(i,:)=temp2/tsm(i,1);

end

end

for i=1:st

for j=1:wind

dm(i,j)=sqrt((c(i,:)-g(j,:))\*((c(i,:)-g(j,:))'));

end

end

sm1=zeros(st,wind);

[temp,temp1]=min(dm);

for i=1:wind

sm1(temp1(i),i)=1;

end

if sm==sm1

cl=0;

end

end

sumstate(:,k)=sum(sm1,2);

ss(k,:)=temp1(1,:);

end

S=ss;

% Initial states have been calculated as per your previous code

%-----------------------------------------------%

ex=18;

N=st;

wind=72;

for i=1:N

pii(i)=0;

pc(i)=0;

for z=1:ex

if(S(z,1)==i)

pii(i)=pii(i)+1;

end

end

Pi(i)=pii(i)/ex;

end

A=zeros(5);

AA=zeros(5);

for i=1:N

for j=1:N

AA(i,j)=0;

for y=1:ex

for x=1:wind-1

if((S(y,x)==i)&&(S(y,(x+1))==j))

AA(i,j)=AA(i,j)+1;

end

end

end

te=sum(AA,2);

end

A(i,:)=AA(i,:)/te(i);

end

% Initial value o f Bj(Ok) as an Uniform distribution..

for k=1:st

for j=1:symb

sum0(k,j)=0;

for i=1:ex

for m=1:72

if (S(i,m)==k && Observ(i,m)==j)

sum0(k,j)=sum0(k,j)+1;

end

end

end

end

end

b2=zeros(size(sum0));

sumstate1=sum(sumstate,2);

for i=1:st

b2(i,:)=sum0(i,:)/sumstate1(i);

end

Ob=Observ(10,:);

clear b;

b=b2;

a=A;

N = st;

K = symb;

sum0=0;

T=length(Ob);

Beta=zeros(T,N);

Alpha=zeros(T,N);

ZI=zeros(T,N,N);

Gamma=zeros(T,N);

E\_T=zeros(1,N);

E\_I\_J=zeros(1,N);

E\_Pi=zeros(1,N);

E\_A=zeros(N,N);

N\_E\_A=zeros(N,N);

E\_B=zeros(N,K);

sum1=zeros(K);

p\_v=zeros(N);

status=zeros(1,N);

%Fowward Algorithm

for i=1:N

Alpha(1,i)=Pi(i) \* b(i,Ob(1));

end

for t=1:T-1

for j=1:N

sum0=0;

for i=1:N

sum0= sum0+ Alpha(t,i)\*a(i,j);

end

Alpha(t+1,j)=sum0 \* b(j,Ob(t+1));

end

end

disp('The forward matrix is:');

for i=1:T

for j=1:N

fprintf('%.8f',Alpha(i,j));

fprintf(' ');

end

fprintf('\n');

end

fprintf('\n');

%Backward Algorithm

for i=1:N

Beta(T,i)=1;

end

for t=T-1:-1:1

for i=1:N

sum0=0;

for j=1:N

sum0=sum0+(a(i,j)\*Beta(t+1,j)\*b(j,Ob(t+1)));

end

Beta(t,i)=sum0;

end

end

disp('The backward matrix is:');

for i=1:T

for j=1:N

fprintf('%.8f',Beta(i,j));

fprintf(' ');

end

fprintf('\n');

end

fprintf('\n');

%Baum-Welch Algorithm

kk=0;

sum2=0;

%Calculation of ZI values

for t=1:T-1

for i=1:N

for j=1:N

nu=Alpha(t,i)\*b(j,Ob(t+1))\*Beta(t+1,j)\*a(i,j);

sum0=0;

for m=1:N

for n=1:N

sum0 = sum0 + (Alpha(t,m) \*a(m,n) \*b(n,Ob(t+1)) \*Beta(t+1,n));

end

end

ZI(t,i,j) = nu/sum0;

end

end

end

% disp('The ZI matrix is:');

% disp(ZI);

%Gamma computation

for t=1:T

for i=1:N

sum0=0;

for j=1:N

sum0 = sum0+ZI(t,i,j);

end

Gamma(t,i)=sum0;

end

end

disp('The Gamma matrix is:');

for i=1:T

for j=1:N

fprintf('%.8f',Gamma(i,j));

fprintf(' ');

end

fprintf('\n');

end

fprintf('\n');

%Expected number of transistions from state i

for i=1:N

sum0=0;

for t=1:T-1

sum0= sum0 + Gamma(t,i);

end

E\_T(i)=sum0;

end

disp('Expected no of transitions from the states:');

for i=1:N

fprintf('%.4f',E\_T(i));

fprintf('\n');

end

%Expected number of transitions from node i to node j

for i=1:N

for j=1:N

sum0=0;

for t=1:T-1

sum0= sum0+ZI(t,i,j);

end

E\_I\_J(i)=sum0;

fprintf('Expected no of transitions from the state %d to state %d:', i,j);

fprintf('%.4f \n',E\_I\_J(i));

end

end

%Computing estimated values for Pi ,A and B.

for i=1:N

E\_Pi(i)= Gamma(1,i);

end

for i=1:N

for j=1:N

sum0=0;

nu=0;

for t=1:T-1

sum0=sum0+ZI(t,i,j);

nu=nu+Gamma(t,i);

end

E\_A(i,j) = (sum0 / nu) ;

end

end

disp('The estimated state transition matrix is:');

for i=1:N

for j=1:N

fprintf('%.8f',E\_A(i,j));

fprintf(' ');

end

fprintf('\n');

end

fprintf('\n');

%Computing the matrix B

for j=1:N % number of states

sum2=0;

for kk=1:K

sum1(kk)=0;

end

for t=1:T %to traverse the ob servation sequence...

for kk=1:K

if(Ob(t) == kk)

sum2 = sum2+ Gamma(t,j); % overall sum ..........

sum1(kk)= sum1(kk) + Gamma(t,j);

break;

end

end

end

for kk=1:K

E\_B(j,kk) = (sum1(kk))/sum2;

end

end

disp('The estimated probability matrix is:');

for i=1:N

for j=1:K

fprintf('%.8f',E\_B(i,j));

fprintf(' ');

end

fprintf('\n');

end

fprintf('\n');

%probability of visit

sum0 = 0;

disp(' The probability of the node being visited during the training phase');

disp(N);

for i=1:N

if(i==1)

p\_v(i)=E\_Pi(i);

else

sum0=0;

for j=1:(i-1)

sum0= sum0 + p\_v(j)\*(E\_A(j,i)/(1-E\_A(j,j)) );

end

end

p\_v(i)= sum0 + (E\_Pi(i));

end

tt=1;

for i=1:N

if(p\_v(i)\*100 >= 40.0)

status(tt)=i;

tt= tt +1;

else

status(i)=0;

end

end

disp('The status during the transition is:');

disp(status);

fprintf('\n');

%Normalization

sum2 = 0;

sum3 = 0;

pp = 0;

pp1 = 0;

for i=1:N

if(i==status(pp+1)) % status(pp)

pp=pp+1;

for j=1:N

N\_E\_A(i,j)=E\_A(i,j);

end

else

sum3=0;

sum2=0;

pp1=0;

for j=1:N

if(j==status(pp1+1)) %status(pp1)

pp1=pp1+1;

sum3= sum3 + a(i,j);

else

sum2= sum2 + E\_A(i,j);

end

end

pp1=0;

for j=1:N

if(j~=status(pp1+1)) %status(pp1)

N\_E\_A(i,j)=(1-sum3)\*(E\_A(i,j)/sum2);

else

pp1=pp1+1;

N\_E\_A(i,j)=a(i,j);

end

end

end

end

disp('After Normalization:');

fprintf('\n');

disp('The estimated state transition matrix is:');

for i=1:N

for j=1:N

fprintf('%.8f',E\_A(i,j));

fprintf(' ');

end

fprintf('\n');

end

fprintf('\n');

disp('The estimated probability matrix is:');

for i=1:N

for j=1:K

fprintf('%.8f',E\_B(i,j));

fprintf(' ');

end

fprintf('\n');

end