***1) Hong-Lee Clustering Method***

function [MemberMat, FuzzyMemFunc] = Hong\_Lee\_Clustering(Training, alpha, C)

%sort input training values by output values

[B, IX] = sort(Training(:,end));

ManyTrain = length(B);

Training = Training(IX,:);

%find differences between adjacent, sorted training values

Diff = diff(Training(:,end));

%find STD of differences

DiffSTD = std(Diff);

%find similarity between adjacent values

S = Diff <= C\*DiffSTD\*ones(size(Diff));

S = (1-(Diff/(C\*DiffSTD))).\*S;

%create membership matrix == [fuzzy member value, fuzzy group asigned to]

MemberMat = zeros(ManyTrain,2);

MemberMat(1,2) = 1;

%find initial groups each training value placed in

for ii=2:ManyTrain

if S(ii-1)<alpha

MemberMat(ii,2)=max(MemberMat(:,2))+1;

else

MemberMat(ii,2)=MemberMat(ii-1,2);

end

end

FuzzyGroupNum = max(MemberMat(:,2));

FuzzyMemFunc = zeros(FuzzyGroupNum,3);

for ii = 1:FuzzyGroupNum

iiMembers = find(MemberMat(:,2)==ii);

tempYs = Training(iiMembers, end);

tempSs = S(iiMembers(1:end-1));

if isempty(tempSs)

tempSs=[1,1];

else

tempSs = [tempSs(1), tempSs(1:end)', tempSs(end)]';

end

%find central vertex of triangle membership func

b\_j\_n = 0; b\_j\_d = 0;

for jj = 1:length(iiMembers)

sweight = (tempSs(jj)+tempSs(jj+1))/2;

b\_j\_n = b\_j\_n + tempYs(jj)\*sweight;

b\_j\_d = b\_j\_d + sweight;

end

%set b

FuzzyMemFunc(ii,2) = b\_j\_n / b\_j\_d;

mintempSs = min(tempSs);

MemberMat(iiMembers(1),1) = mintempSs;

MemberMat(iiMembers(end),1) = mintempSs;

%set a

FuzzyMemFunc(ii,1) = FuzzyMemFunc(ii,2) -...

(FuzzyMemFunc(ii,2) - tempYs(1))/(1-mintempSs);

%set c

FuzzyMemFunc(ii,3) = FuzzyMemFunc(ii,2) +...

(tempYs(end) - FuzzyMemFunc(ii,2))/(1-mintempSs);

end

MemberMat(MemberMat(:,1)==0,1)=1;

***3) Fuzzy c-mean Clustering***

function [U, V, z] = Find\_U\_V(X, c, m, tol, MaxItt)

[p,n] = size(X);

% create initial U randomly

U = zeros(c,n);

U(1,:) = rand(1,n);

for ii = 2:c-1

U(ii,:) = (ones(1,n)-sum(U(1:ii-1,:),1)).\*rand(1,n);

end

U(end,:) = ones(1,n)-sum(U(1:end-1,:),1);

% create initial V from U

V = ones(p,1)\*(1./ sum(U.^m,2))';

for ii = 1:p

V(ii,:) = V(ii,:).\* sum(U.^m.\*(ones(c,1)\*X(ii,:)),2)';

end

%U, V, pause

RefChange = 999;

itt=0;

%while RefChange > tol && itt < MaxItt

while itt < MaxItt

% Save old values for RefChange

OldU=U; OldV=V;

% Create New Values

% New U from V

temp = 0;

for jj = 1:c

XV = X - V(:,jj)\*ones(1,n);

% for G == I

XV = sum(XV.^2,1);

XV = 1./(XV.^(1/(m-1)));

U(jj,:) = XV;

temp = temp + XV;

end

U = U./(ones(c,1)\*temp);

% New V from U

V = ones(p,1)\*(1./ sum(U.^m,2))';

for ii = 1:p

V(ii,:) = V(ii,:).\* sum(U.^m.\*(ones(c,1)\*X(ii,:)),2)';

end

% Compute Ref Change

RefChange = mean(mean(abs((V-OldV)./OldV),2));

itt=itt+1;

end

% Calculate variance of method

z = 0;

for jj = 1:c

XV = X - V(:,jj)\*ones(1,n);

% for G == I

XV = sum(XV.^2,1);

XV = sum(XV.\*(U(jj,:).^m));

z = z + XV;

end