K-means with random initialization:

Minimum within-cluster sum of squares: 222.365988

Mean within-cluster sum of squares: 246.628452

Standard deviation of within-cluster sum of squares: 62.104709

K-means with K-means++ initialization:

Minimum within-cluster sum of squares: 222.365988

Mean within-cluster sum of squares: 240.187166

Standard deviation of within-cluster sum of squares: 54.047144

 

 

 

%

% hw03 section 1 implementation for ml701 class.

% willie neiswanger, 11/2012.

%

function hw03()

dat = csvread('hw3-cluster.csv');

fprintf('\nK-means with random initialization:\n'); doPartsBandC(dat,'initrand');

fprintf('\nK-means with K-means++ initialization:\n'); doPartsBandC(dat,'init++');

fprintf('\nK-means with 5 clusters:\n'); doPartsEandF(dat);

dat2 = csvread('hw3-cluster2.csv');

fprintf('\nK-means with 3/9 clusters:\n'); doPartsEandF(dat2);

dat3 = csvread('hw3-cluster3.csv');

fprintf('\nK-means without normalization:\n'); doPartsGandH(dat3);

fprintf('\nK-means with normalization:\n'); doPartsGandH(normalizeData(dat3));

function doPartsBandC(data,initstring)

% carry out 200 iterations of kmeans

[meanscell,wcsosVec] = runMultipleKMeans(data,5,200,initstring);

% plot data and means for all iterations

figure,plot(data(:,1),data(:,2),'o','color',[0.5,0.5,0.5]);

for i=1:length(meanscell)

hold on; plot(meanscell{i}(:,1), meanscell{i}(:,2), 'o', 'color','k','MarkerFaceColor', 'k');

end

drawnow

% get min, mean, and st-dev of the within-cluster sum of squares

fprintf('Minimum within-cluster sum of squares: %f\n',min(wcsosVec));

fprintf('Mean within-cluster sum of squares: %f\n',mean(wcsosVec));

fprintf('Standard deviation of within-cluster sum of squares: %f\n',std(wcsosVec));

end

function doPartsEandF(data)

% find best K

initstring = 'initrand';

wcsosPerK = [];

for k=1:15

[meanscell,wcsosVec] = runMultipleKMeans(data,k,200,initstring);

wcsosPerK(k) = min(wcsosVec);

fprintf('Completed: K=%d\n',k);

end

figure

subplot(1,2,1), plot(1:15,wcsosPerK,'o'); title('Min-WCSOS vs. K');

subplot(1,2,2), plot(1:15,sqrt(wcsosPerK),'o'); title('sqrt(Min-WCSOS) vs. K');

drawnow

end

function doPartsGandH(data)

% experiment with normalization

initstring = 'init++';

[meanscell,wcsosVec] = runMultipleKMeans(data,2,500,initstring);

% plot data and means for all iterations

figure,plot(data(:,1),data(:,2),'o','color',[0.5,0.5,0.5]);

for i=1:length(meanscell)

hold on; plot(meanscell{i}(:,1), meanscell{i}(:,2), 'o', 'color','k','MarkerFaceColor', 'k');

end

drawnow

disp('Results shown in plot.')

end

function [meanscell,wcsosVec] = runMultipleKMeans(data,k,iter,initstring)

% run kmeans multiple times, get within-cluster sum of squares each time

meanscell = {}; wcsosVec = [];

for i=1:iter

[dataLabels,dataMeans] = kmeans(data,k,initstring);

meanscell{end+1} = dataMeans;

wcsosVec(end+1) = getWcSumOfSquares(data,dataLabels,dataMeans);

end

end

function [labels,means]=kmeans(data,k,whichInit)

if strcmpi(whichInit,'initrand')

means = initMeans\_rand(data,k);

elseif strcmpi(whichInit,'init++')

means = initMeans\_plusplus(data,k);

end

% kmeans algorithm

lastmeans = []; itercount = 0;

while (not(isequal(lastmeans,means)) & itercount<100)

itercount = itercount+1;

lastmeans=means;

% assign each example to closest mean in means vec

for k = 1 : size(means,1)

diffmat = data - repmat(means(k,:), size(data,1), 1);

distmat(:,k) = sqrt(sum(diffmat .\* diffmat, 2)); % distmat(i,k) holds distance of i\_th data to k\_th mean

end

for i = 1 : size(data,1)

[del, keep] = min(distmat(i,:));

labels(i) = keep;

end

% assign each mean in means vec to mean of assigned examples

for k = 1 : size(means,1)

assignedInd = find(labels == k);

means(k,:) = sum(data(assignedInd, :))/length(assignedInd); % compute mean

end

end

end

function means = initMeans\_rand(data,k)

% initialize clusters to random data points

means = data(randperm(size(data,1),k),:);

end

function means = initMeans\_plusplus(data,k)

% k-means++ initialization

means(1,:) = data(randi(size(data,1)),:);

for a=2:k

for b=1:size(means,1);

tmpmat = data - repmat(means(b,:), size(data,1), 1);

distmat(:,b) = sqrt(sum(tmpmat.\*tmpmat,2));

end

datadist = min(distmat,[],2);

datadist = datadist / sum(datadist);

nextInd = catrnd(datadist);

means(a,:) = data(nextInd,:);

end

end

function wcsos = getWcSumOfSquares(data,labels,means)

% get the within-cluster sum of squares

% sum, over all data points, the square of the norm of (the data point minus its assigned cluster center)

wcsos = 0;

for i=1:size(data,1)

wcsos = wcsos + norm(data(i,:) - means(labels(i),:))^2;

end

end

function ind = catrnd(p,n)

% get n samples from a categorical distribution with parameter p (a vector)

if nargin==1, n=1; end

k = length(p);

p = reshape(p,k,1);

ind = sum(repmat(rand(1,n),k,1) > repmat(cumsum(p)/sum(p),1,n),1)+1;

end

function data = normalizeData(data)

data = data - repmat(mean(data),size(data,1),1);

data = data ./ repmat(std(data),size(data,1),1);

end

end