clear all;

clc;

noSamples = 1000;

feature1 = generateFeature1Samples(noSamples);

feature2 = generateFeature2Samples(noSamples);

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

% FIGURE 1

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

figure(1);

clf;

plot(feature1(1,:),feature1(2,:), 'y\*');

hold on;

plot(feature2(1,:),feature2(2,:), 'r\*');

xlabel('Feature 1');

ylabel('Feature 2');

title('Spatial Distribution of Feature 1 and Feature 2');

axis([-150 150 -150 150])

grid on;

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

% FIGURE 2

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

noSamples = 1000;

samples = zeros(2,noSamples);

targets = zeros(1,noSamples);

for i = 1:noSamples

if rand() < 0.5

samples(:,i) = generateFeature1Samples(1);

targets(i) = 1;

else

samples(:,i) = generateFeature2Samples(1);

targets(i) = 2;

end

end

figure(2);

clf;

plot(samples(1,targets==1),samples(2,targets==1), 'y\*');

hold on;

plot(samples(1,targets==2),samples(2,targets==2), 'r\*');

xlabel('Feature 1');

ylabel('Feature 2');

title('Spatial Distribution of Feature 1 and Feature 2');

axis([-150 150 -150 150])

grid on;

W = [1;1;1];

W = updatePerceptronWeights(samples, targets, W);

% plot the perceptron line using the perceptron weights.

x = -150:0.1:150;

y = -(W(1) + W(2) \* x) / W(3);

plot(x, y, '-b');

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

% FIGURE 3

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

%% SIMULATION PARAMETERS

variance = [0, 10, 100, 1000, 10000];

meanValues = [-10, -10, 0, 10, 10;

-10, 10, 0, -10, 10];

expectedLearningRate = 0.35;

for varIndex = 1:numel(variance)

for meanIndex = 1:size(meanValues,2)

noiseVariance1 = variance(varIndex);

noiseSTD1 = sqrt(noiseVariance1);

noiseMean1 = meanValues(:,meanIndex);

noiseVariance2 = variance(varIndex);

noiseSTD2 = sqrt(noiseVariance2);

noiseMean2 = meanValues(:,meanIndex);

% The initial value of weight. It is chosen randomly.

W = [1; 1; 1];

epochIndex = 0;

while(true)

epochIndex = epochIndex + 1;

%% TRANING SAMPLES

traningNoSamples = 7000; % NUMBER OF TRANING SAMPLES

traningSamples = zeros(2,traningNoSamples);

traningTargets = zeros(1,traningNoSamples);

for i = 1:traningNoSamples

if rand() < 0.5

traningSamples(:,i) = generateFeature1Samples(1) + ...

noiseSTD1 \* randn(2,1) + noiseMean1;

traningTargets(i) = 1;

else

traningSamples(:,i) = generateFeature2Samples(1) + ...

noiseSTD2 \* randn(2,1) + noiseMean2;

traningTargets(i) = 2;

end

end

%% VALIDATION SAMPLES

validationNoSamples = 150; % NUMBER OF VALIDATION SAMPLES

validationSamples = zeros(2,validationNoSamples);

validationTargets = zeros(1,validationNoSamples);

for i = 1:validationNoSamples

if rand() < 0.5

validationSamples(:,i) = generateFeature1Samples(1) + ...

noiseSTD1 \* randn(2,1) + noiseMean1;

validationTargets(i) = 1;

else

validationSamples(:,i) = generateFeature2Samples(1) + ...

noiseSTD2 \* randn(2,1) + noiseMean2;

validationTargets(i) = 2;

end

end

%% TEST SAMPLES

testNoSamples = 150; % NUMBER OF TEST SAMPLES

testSamples = zeros(2,testNoSamples);

testTargets = zeros(1,testNoSamples);

for i = 1:testNoSamples

if rand() < 0.5

testSamples(:,i) = generateFeature1Samples(1) + ...

noiseSTD1 \* randn(2,1) + noiseMean1;

testTargets(i) = 1;

else

testSamples(:,i) = generateFeature2Samples(1) + ...

noiseSTD2 \* randn(2,1) + noiseMean2;

testTargets(i) = 2;

end

end

%% UPDATE WEIGHTS

% Plotting the traning samples and finding the weights of the perceptron.

W = updatePerceptronWeights(traningSamples, traningTargets, W);

%% DECISIONS USING TEST SAMPLES

decision = zeros(1,testNoSamples);

for i = 1:testNoSamples

S = [1; testSamples(1,i); testSamples(2,i)];

predicate = sign(W'\*S);

if (predicate <= 0)

decision(i) = 1;

else

decision(i) = 2;

end

end

%% STATISTICS

totalDecisionError = sum(decision ~= testTargets);

learningRate = totalDecisionError/testNoSamples;

if learningRate > expectedLearningRate

continue

end

%% PLOT STATISTICS FOR INFORMATION

fprintf('------------------------------\n');

fprintf('Variance = %d\n', variance(varIndex));

fprintf('Mean = [%d; %d]\n', meanValues(1,meanIndex), meanValues(2,meanIndex));

fprintf('Epoch = %d\n', epochIndex);

fprintf('Total decision error = %d\n', totalDecisionError);

fprintf('Learning rate = %f\n', learningRate);

break;

end

end

end

%%

% This function finds the values of perceptron weights from feature groups.

% An example illustration is given as

% <code>

% </code>

function [W] = updatePerceptronWeights(mixedFeatures, targets, prevW)

if size(mixedFeatures,2) ~= numel(targets)

error('The size of the targets and the number of samples does not matches!');

end

W = prevW;

for i = 1:numel(targets)

S = [1; mixedFeatures(1,i); mixedFeatures(2,i)];

target = targets(i);

if target == 1

% For the first group

if sign(W'\*S) >= 0

% Update the weights using Feature 1.

W = W - S;

end

elseif target == 2

% For the second group

if sign(W'\*S) <= 0

% Update the weights using Feature 2.

W = W + S;

end

else

error('Unknown target!')

end

end

end

%%

% This function spatially generates the samples that belong to Feature 1.

% An example usage is illustrated below.

% <code>

% noSamples = 1000;

% [xVal, yVal] = generateFeature1Samples(noSamples);

% </code>

function [samples] = generateFeature1Samples(noSamples)

width = 100;

% The first point on which the line passes

x0 = -20.55;

y0 = -0.2771;

% The second point on which the line passes

x1 = 0.6928;

y1 = 10.35;

% We generate the samples in the following. If the generated sample is over

% the line then the sample is accepted. otherwise, we will try to find the

% value of y untill the point is over the line.

samples = zeros(2,noSamples);

for index = 1:noSamples

x = width\*(2\*rand()-1);

y = width\*(2\*rand()-1);

yhat = (y1-y0)/(x1-x0)\*(x-x0)+y0;

while y < yhat

% The point is not over the line so we generate y again.

y = width\*(2\*rand()-1);

end

samples(1,index) = x;

samples(2,index) = y;

end

end

%%

% This function spatially generates the samples that belong to Feature 2.

% An example usage is illustrated below.

% <code>

% noSamples = 1000;

% [xVal, yVal] = generateFeature2Samples(noSamples);

% </code>

function [samples] = generateFeature2Samples(noSamples)

width = 100;

% The first point on which the line passes

x0 = -0.692;

y0 = -10.35;

% The second point on which the line passes

x1 = 20.55;

y1 = 0.2771;

% We generate the samples in the following. If the generated sample is

% lower the line then the sample is accepted. otherwise, we will try to find the

% value of y untill the point is lower the line.

samples = zeros(2,noSamples);

for index = 1:noSamples

x = width\*(2\*rand()-1);

y = width\*(2\*rand()-1);

yhat = (y1-y0)/(x1-x0)\*(x-x0)+y0;

while y > yhat

% The point is not lower the line so we generate y again.

y = width\*(2\*rand()-1);

end

samples(1,index) = x;

samples(2,index) = y;

end

end