Lab 2

Gabriel Casciano, 500744076

Aleksandar Nikolovski, 500572171

Khayyam Butt, 500501040

The subject of study for this lab is Linear Discrimnant functions.

The training set being used is the iris training set.

Question 1

Construct a new data set from **A** and **B** by setting aside 30% of the sample from those original sets (this will be used for *training purposes*). Construct a second set from the remaining 70% of data (this will be used for *testing*). Use the 30% set (training set) to compute the weight vector $\eta(k) = \eta(\bullet) = 0.01, \Theta = 0,$ and an initial values of $a(0) = [0, 0, 1]$. Limit your maximum iterations to 300 (i.e even if you can not achieve $\theta = 0$).

```
clear
setup

a = [0,0,1];
eta = 0.01;
iter = 300;
theta  = 0;

weightVector1 = gradientDescent(a , theta, ABtraining, eta, iter, 1)
```

```
weightVector1 = 1×3
    0.0800    0.6430   -0.7020
```

Questions 2

Use the 70% set (test samples) and the weight vector computer in the previous step to calculate the classification accuracy of the classifier.

```
for i  = 1 : length(ABtesting)
    class = ABtesting(i,3);
    temp = [1, ABtesting(i, 1:2)];
    if(classifier1(weightVector1, temp) ~= class)
        err = err + 1;
    end
end
percentError1 = err/length(ABtesting) * 100
```

```
percentError1 = 0
```

Question 3

Repeat the above steps by changing the data split to 70% (training) and 30% (testing).

```
clear
setup

a = [0,0,1];
eta = 0.01;
iter = 300;
theta  = 0;

weightVector2 = gradientDescent(a , theta, ABtesting, eta, iter, 1)
```

```
weightVector2 = 1×3
    0.0800    0.7310   -0.8490
```

```
for i  = 1 : length(ABtraining)
    class = ABtraining(i,3);
    temp = [1, ABtraining(i, 1:2)];
    if(classifier1(weightVector2, temp) ~= class)
        err = err + 1;
    end
end

percentError2 = err/length(ABtraining) * 100
```

```
percentError2 = 0
```

Question 4

Repeat all the above for test and training sets constructed from the combination of data set **B** and **C** (i.e *Iris Versicolor* vs, *Iris Virginica*)

Question 4 - 1

```
clear
setup

a = [0,0,1];
eta = 0.01;
iter = 300;
theta  = 0;

weightVector3 = gradientDescent(a , theta, BCtraining, eta, iter, 2)
```

```
weightVector3 = 1×3
    3.4000    3.7810   -3.3250
```

Question 4 - 2

```
for i  = 1 : length(BCtesting)
```

```
    class = BCtesting(i,3);
    temp = [1, BCtesting(i, 1:2)];
    if(classifier1(weightVector3, temp) ~= class)
        err = err + 1;
    end
end
percentError3 = err/length(BCtesting) * 100
```

```
percentError3 = 50
```

Question 4 - 3

```
clear
setup

a = [0,0,1];
eta = 0.01;
iter = 300;
theta  = 0;

weightVector4 = gradientDescent(a , theta, BCtesting, eta, iter, 2)
```

```
weightVector4 = 1×3
    5.2100    5.9920   -4.5750
```

```
for i  = 1 : length(BCtraining)
    class = BCtraining(i,3);
    temp = [1, BCtraining(i, 1:2)];
    if(classifier1(weightVector4, temp) ~= class)
        err = err + 1;
    end
end

percentError4 = err/length(BCtraining) * 100
```

```
percentError4 = 50
```

Question 5

Compute the weight vextors and study the gradient descent algorithms for two different values of $\eta(k)$, and the initial values of **a**.

```
% Compare to question 2 values. training with ABtesting and testing with
% ABtraining.
clear
setup

a = [1,1,1]; % different a
eta = 0.01;
iter = 300;
theta  = 0;
```

```
weightVector5 = gradientDescent(a , theta, ABtesting, eta, iter, 1)
```

```
weightVector5 = 1×3
    0.8300    0.8850   -1.2110
```

```
for i  = 1 : length(ABtraining)
    class = ABtraining(i,3);
    temp = [1, ABtraining(i, 1:2)];
    if(classifier1(weightVector5, temp) ~= class)
        err = err + 1;
    end
end

percentError5 = err/length(ABtraining) * 100
```

```
percentError5 = 0
```

```
clear
setup

a = [2,2,2]; % different a
eta = 0.01;
iter = 300;
theta  = 0;

weightVector6 = gradientDescent(a , theta, ABtesting, eta, iter, 1)
```

```
weightVector6 = 1×3
    1.5400    1.0760   -1.4310
```

```
for i  = 1 : length(ABtraining)
    class = ABtraining(i,3);
    temp = [1, ABtraining(i, 1:2)];
    if(classifier1(weightVector6, temp) ~= class)
        err = err + 1;
    end
end

percentError6 = err/length(ABtraining) * 100
```

```
percentError6 = 23.3333
```

```
clear
setup

a = [0,0,1];
eta = 0.1; % different eta, more coarse
```

```
iter = 300;
theta  = 0;

weightVector7 = gradientDescent(a , theta, ABtesting, eta, iter, 1)
```

```
weightVector7 = 1×3
    3.3000   16.2500  -14.3500
```

```
for i  = 1 : length(ABtraining)
    class = ABtraining(i,3);
    temp = [1, ABtraining(i, 1:2)];
    if(classifier1(weightVector7, temp) ~= class)
        err = err + 1;
    end
end

percentError7 = err/length(ABtraining) * 100
```

```
percentError7 = 0
```

```
clear
setup

a = [0,0,1];
eta = 0.001; % different eta, more fine
iter = 300;
theta  = 0;

weightVector8 = gradientDescent(a , theta, ABtesting, eta, iter, 1)
```

```
weightVector8 = 1×3
   -0.0310    0.1509   -0.1333
```

```
for i  = 1 : length(ABtraining)
    class = ABtraining(i,3);
    temp = [1, ABtraining(i, 1:2)];
    if(classifier1(weightVector8, temp) ~= class)
        err = err + 1;
    end
end

percentError8 = err/length(ABtraining) * 100
```

```
percentError8 = 3.3333
```

Question 6

In each of the above steps, plot _the training data on_ feature space, the decision boundary for training set, the perceptron criterion

function ove the iterations and the number of iterations required for convergence.

ABtraining

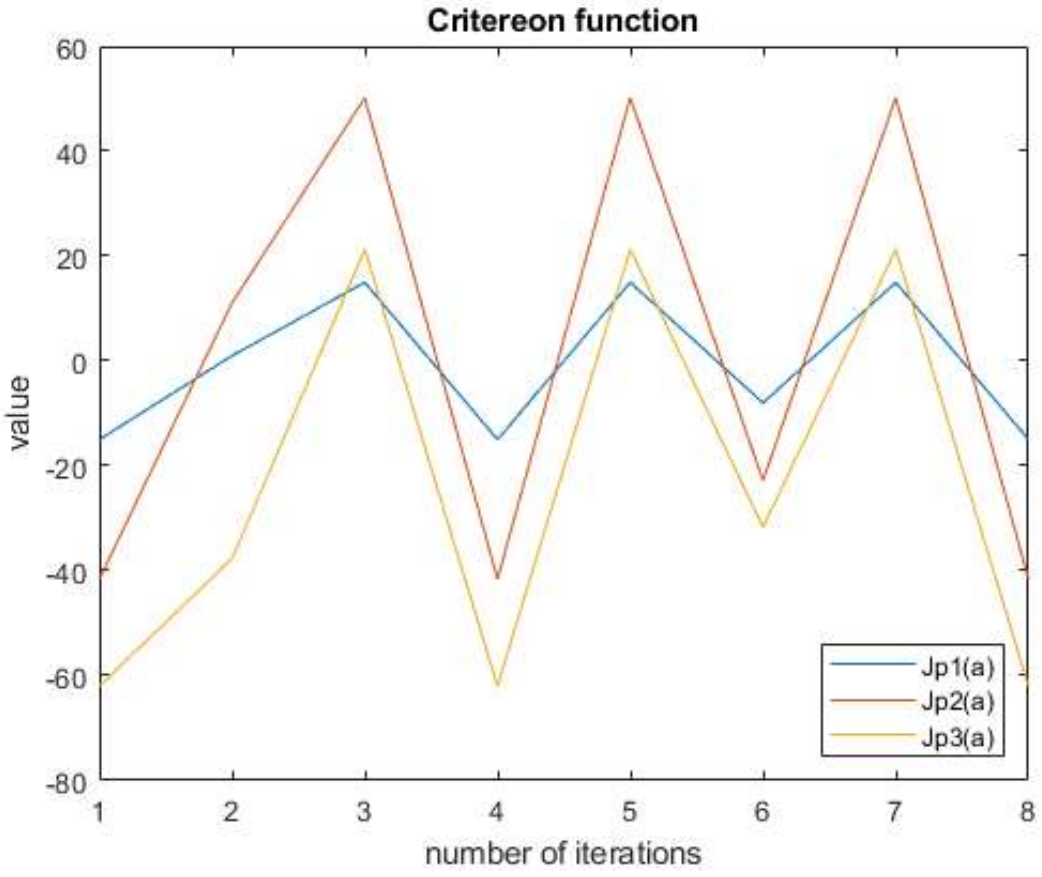
```
clear
setup

a = [0,0,1];
eta = 0.01;
iter = 300;
theta  = 0;

[w1, Jp1, conv1, fin1] = gradientDescent2(a , theta, ABtraining, eta, iter, 1);
[b11, b12] = decisionBoundary(fin1);
scale = 1:.1:3;
slope1 = b12/b11;
bound1 = slope1 .* scale + b12;

display(w1)
```

```
w1 = 1×3
    0.0800    0.6430   -0.7020
```

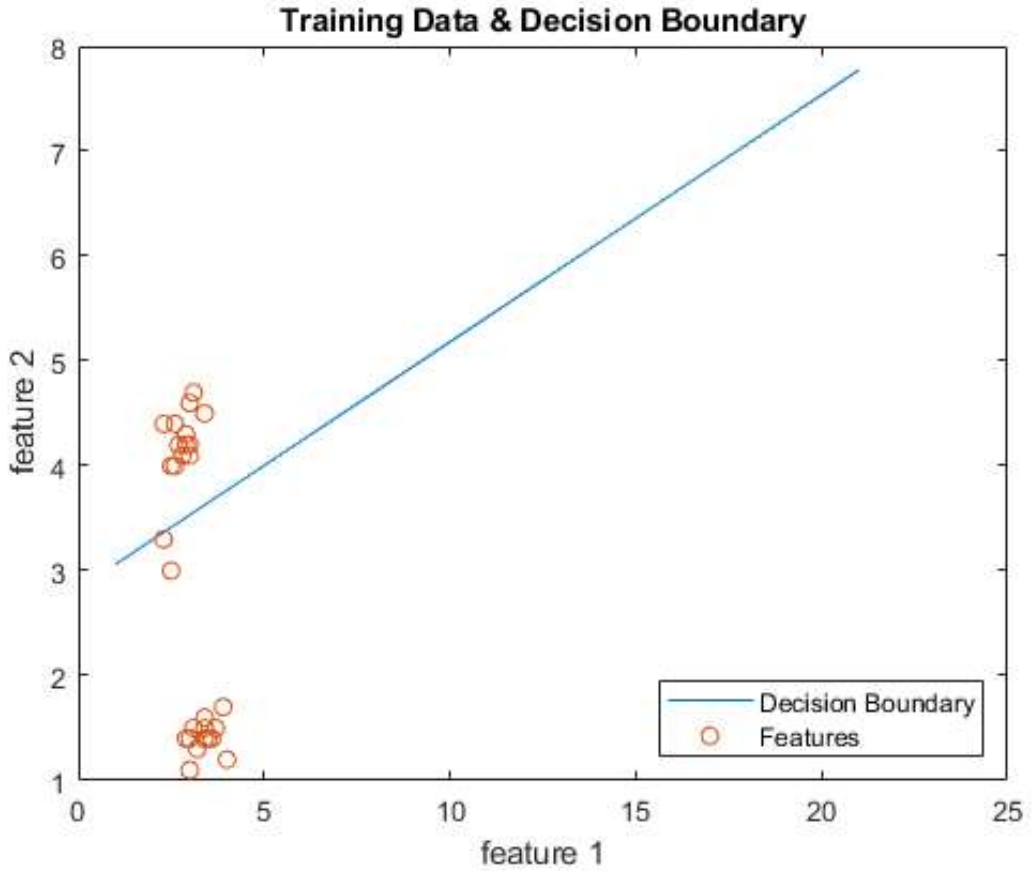
```
display(conv1)
```

```
conv1 = 8
```

```


figure
plot(Jp1)
title("Critereon function")
xlabel("number of iterations")
ylabel("value")
legend({"Jp1(a)", "Jp2(a)", "Jp3(a)"}, "location", "southeast")
```




```
figure
plot(bound1)
hold on
scatter(ABtraining(:,1),ABtraining(:,2))
title("Training Data & Decision Boundary")
xlabel("feature 1")
ylabel("feature 2")
legend({"Decision Boundary", "Features"}, "location", "southeast")
hold off
```



ABtesting

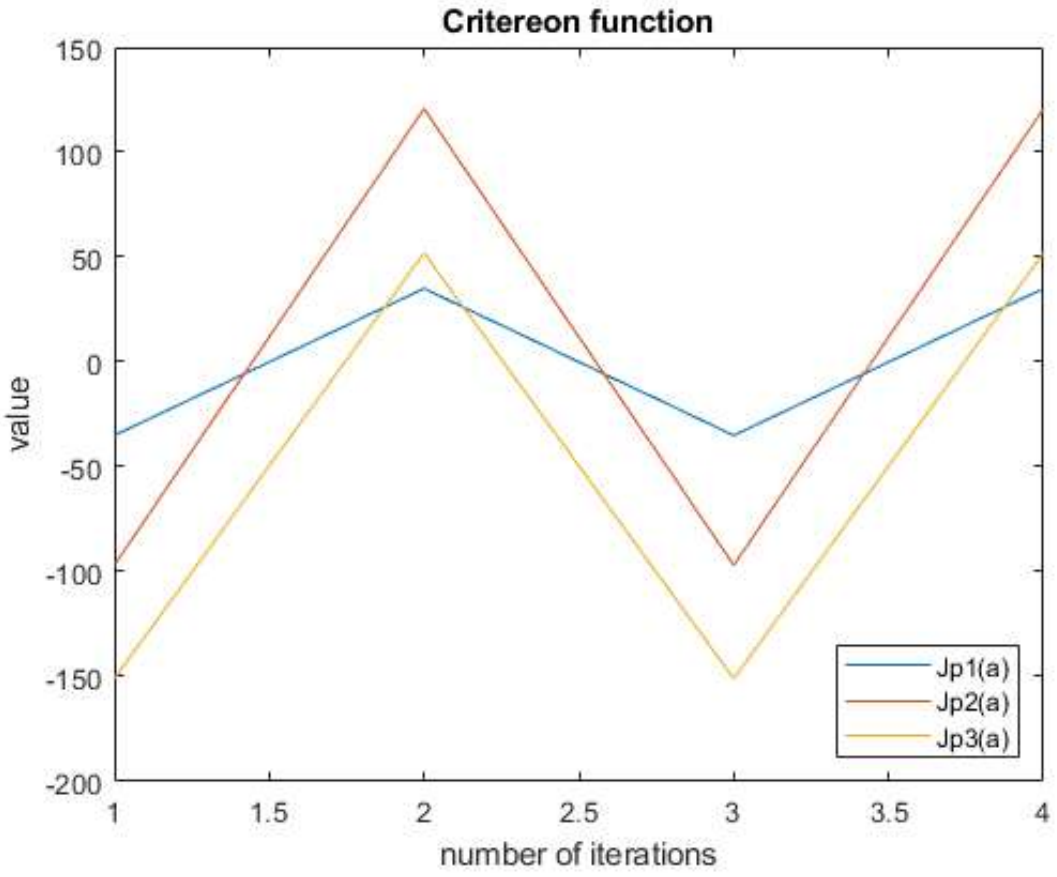
```
clear
setup

a = [0,0,1];
eta = 0.01;
iter = 300;
theta  = 0;

[w2, Jp2, conv2, fin2] = gradientDescent2(a , theta, ABtesting, eta, iter, 1);
[b21, b22] = decisionBoundary(fin2);
scale = 1:.1:3;
slope2 = b21/b22;
bound2 = slope2 .* scale + b21;

display(w2)
```

```
w2 = 1×3
    0.0800    0.7310   -0.8490
```

```
display(conv2)
```

```
conv2 = 4
```

```
figure
plot(Jp2)
title("Critereon function")
xlabel("number of iterations")
ylabel("value")
legend({"Jp1(a)", "Jp2(a)", "Jp3(a)"}, "location", "southeast")
```




```
figure
plot(bound2)
hold on
scatter(ABtesting(:,1),ABtesting(:,2))
title("Training Data & Decision Boundary")
xlabel("feature 1")
ylabel("feature 2")
legend({"Decision Boundary", "Features"}, "location", "southeast")
hold off
```

BCtraining

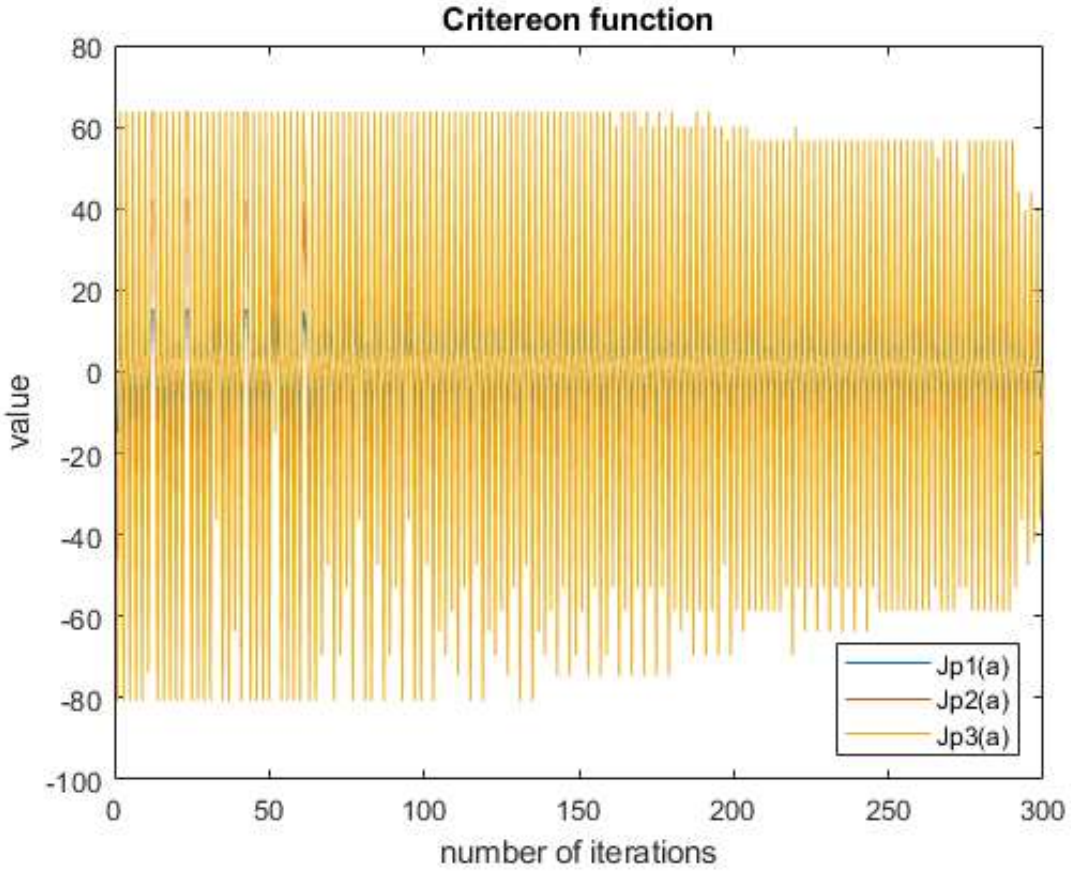
```
clear
setup

a = [0,0,1];
eta = 0.01;
iter = 300;
theta  = 0;

[w3, Jp3, conv3, fin3] = gradientDescent2(a , theta, BCtraining, eta, iter, 2);
[b31, b32] = decisionBoundary(fin3);
scale = 1:.1:3;
slope3 = b31/b32;
bound3 = slope3 .* scale + b31;

display(w3)
```

```
w3 = 1×3
    3.4000    3.7810   -3.3250
```

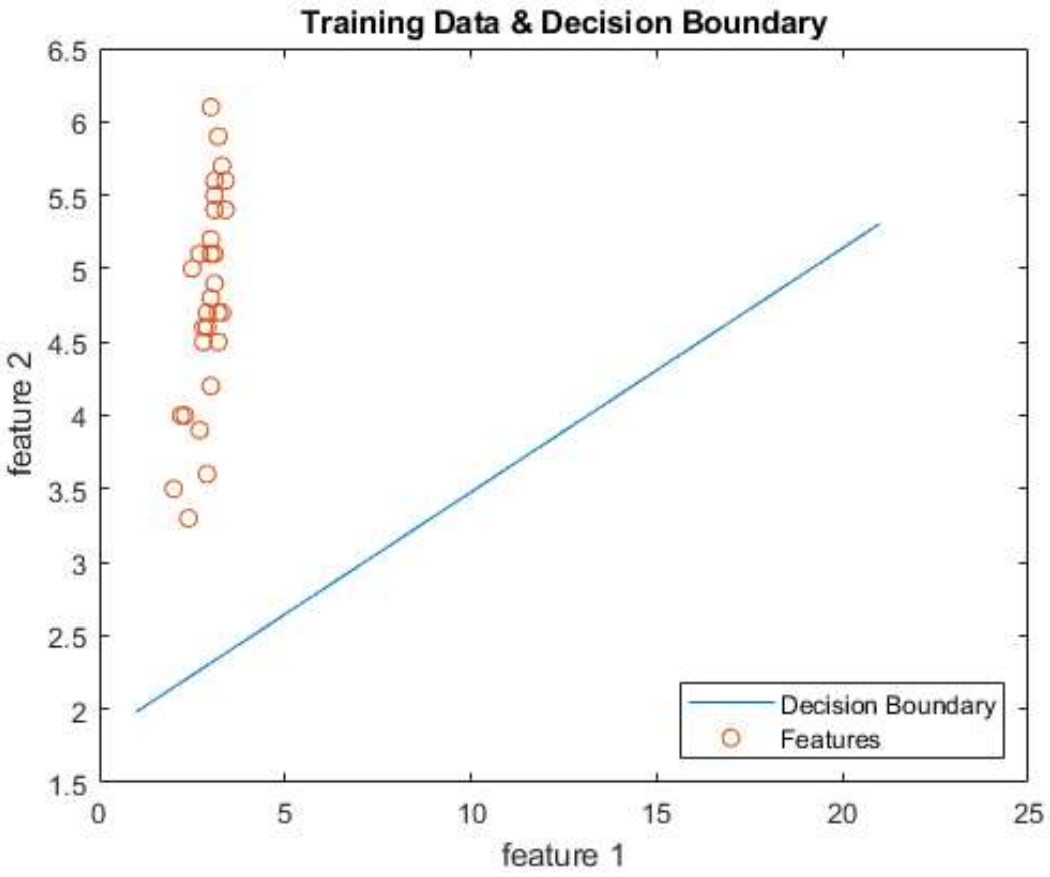
```
display(conv3)
```

```
conv3 = 300
```

```
figure
plot(Jp3)
title("Critereon function")
xlabel("number of iterations")
ylabel("value")
legend({"Jp1(a)", "Jp2(a)", "Jp3(a)"}, "location", "southeast")
```




```
figure
plot(bound3)
hold on
scatter(BCtraining(:,1),BCtraining(:,2))
title("Training Data & Decision Boundary")
xlabel("feature 1")
ylabel("feature 2")
legend({"Decision Boundary", "Features"}, "location", "southeast")
hold off
```



BCtesting

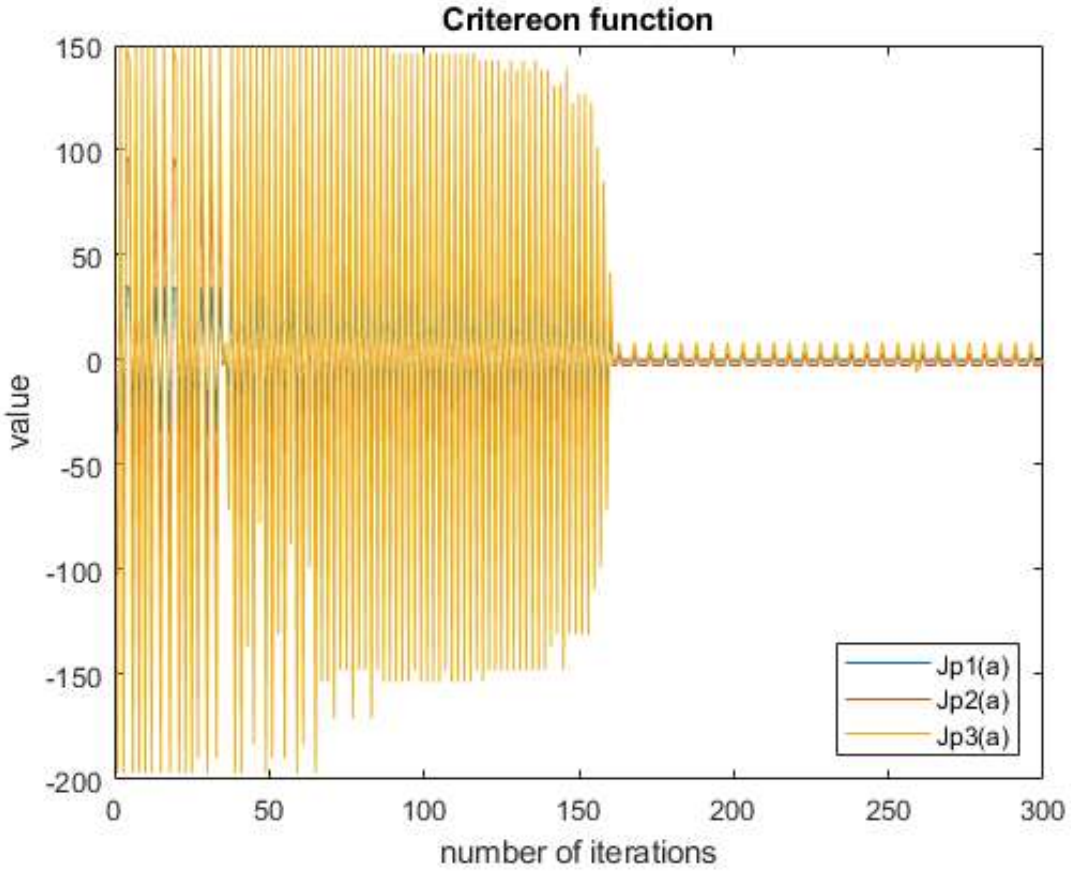
```
clear
setup

a = [0,0,1];
eta = 0.01;
iter = 300;
theta  = 0;

[w4, Jp4, conv4, fin4] = gradientDescent2(a , theta, BCtesting, eta, iter, 2);
[b41, b42] = decisionBoundary(fin4);
scale = 1:.1:3;
slope4 = b41/b42;
bound4 = slope4 .* scale + b41;

display(w4)
```

```
w4 = 1×3
    5.2100    5.9920   -4.5750
```

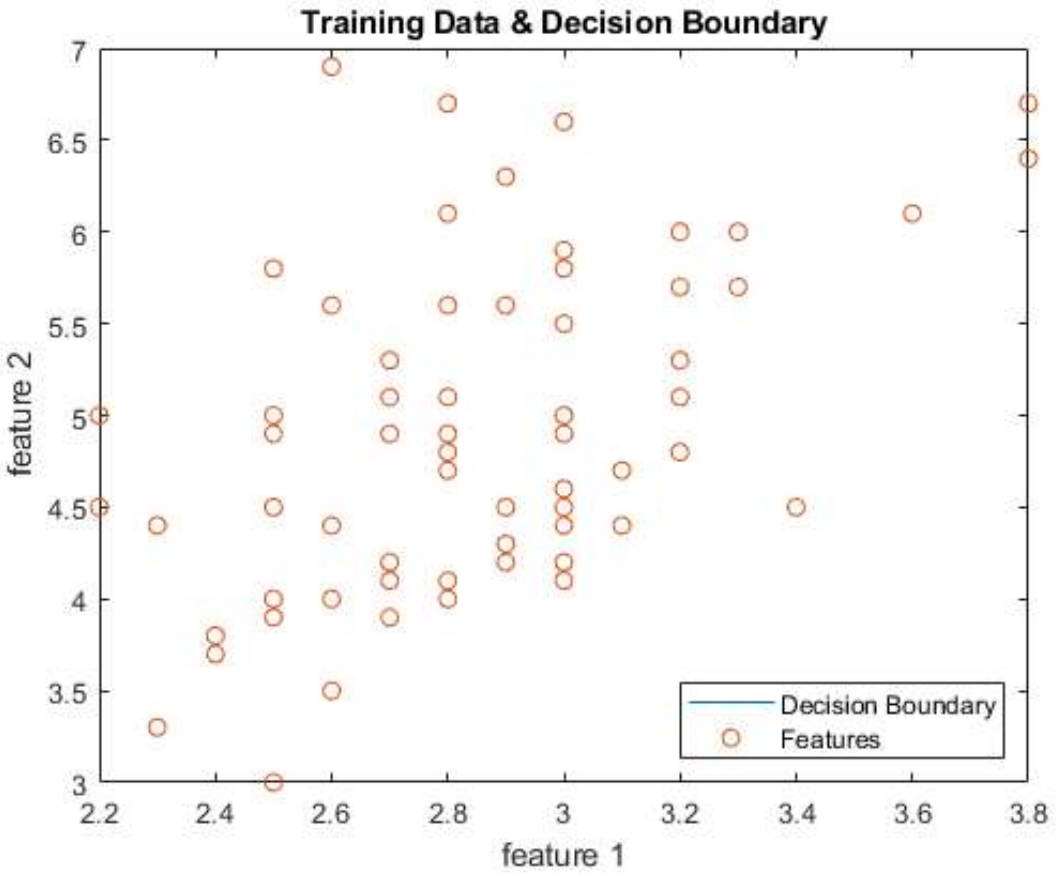
```
display(conv4)
```

```
conv4 = 300
```

```
figure
plot(Jp4)
title("Critereon function")
xlabel("number of iterations")
ylabel("value")
legend({"Jp1(a)", "Jp2(a)", "Jp3(a)"}, "location", "southeast")
```




```
figure
plot(bound4)
hold on
scatter(BCtesting(:,1),BCtesting(:,2))
title("Training Data & Decision Boundary")
xlabel("feature 1")
ylabel("feature 2")
legend({"Decision Boundary", "Features"}, "location", "southeast")
hold off
```



ABtraining, different a1

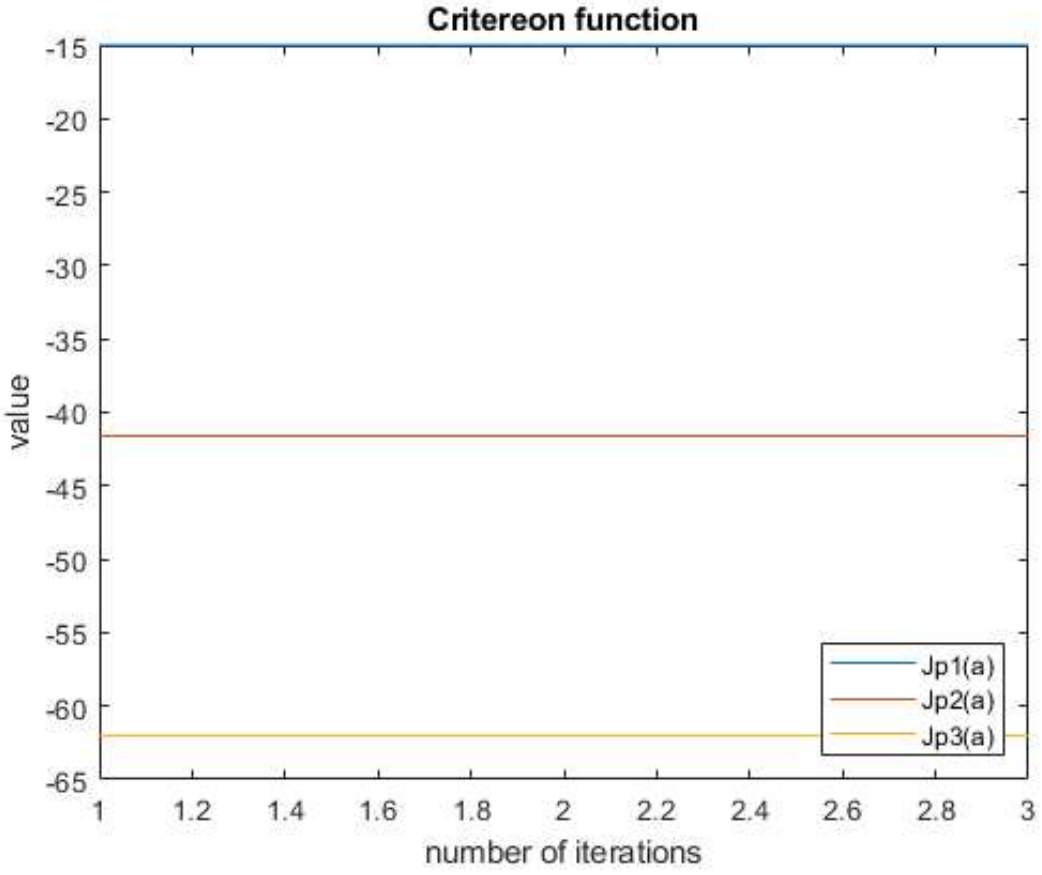
```
clear
setup

a = [1,1,1];
eta = 0.01;
iter = 300;
theta  = 0;

[w5, Jp5, conv5, fin5] = gradientDescent2(a , theta, ABtraining, eta, iter, 1);
[b51, b52] = decisionBoundary(fin5);
scale = 1:.1:3;
slope5 = b52/b51;
bound5 = slope5 .* scale + b52;

display(w5)
```

```
w5 = 1×3
    0.7000    0.2540   -0.6470
```

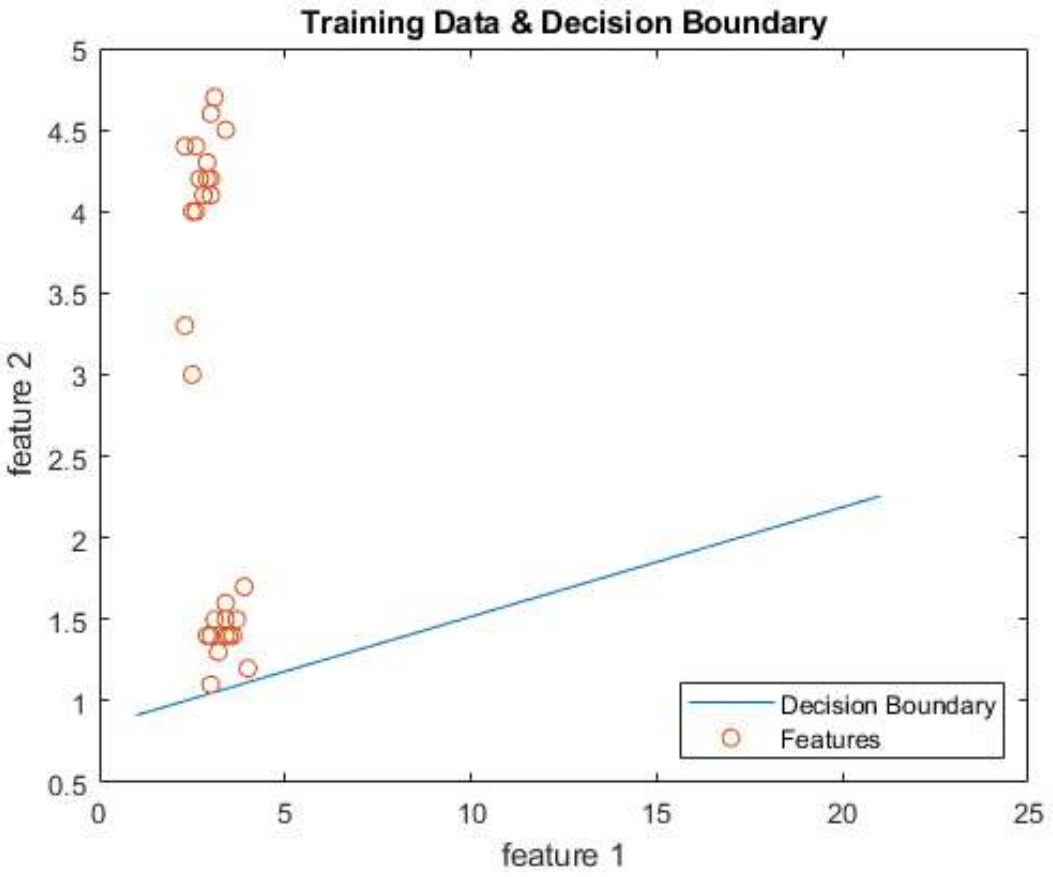
```
display(conv5)
```

```
conv5 = 3
```

```
figure
plot(Jp5)
title("Critereon function")
xlabel("number of iterations")
ylabel("value")
legend({"Jp1(a)", "Jp2(a)", "Jp3(a)"}, "location", "southeast")
```




```
figure
plot(bound5)
hold on
scatter(ABtraining(:,1),ABtraining(:,2))
title("Training Data & Decision Boundary")
xlabel("feature 1")
ylabel("feature 2")
legend({"Decision Boundary", "Features"}, "location", "southeast")
hold off
```



ABtraining, different a2

```
clear
setup

a = [2,2,2];
eta = 0.01;
iter = 300;
theta  = 0;

[w6, Jp6, conv6, fin6] = gradientDescent2(a , theta, ABtraining, eta, iter, 1);
[b61, b62] = decisionBoundary(fin6);
scale = 1:.1:3;
slope6= b62/b61;
bound6 = slope6 .* scale + b62;
display(w6)
```

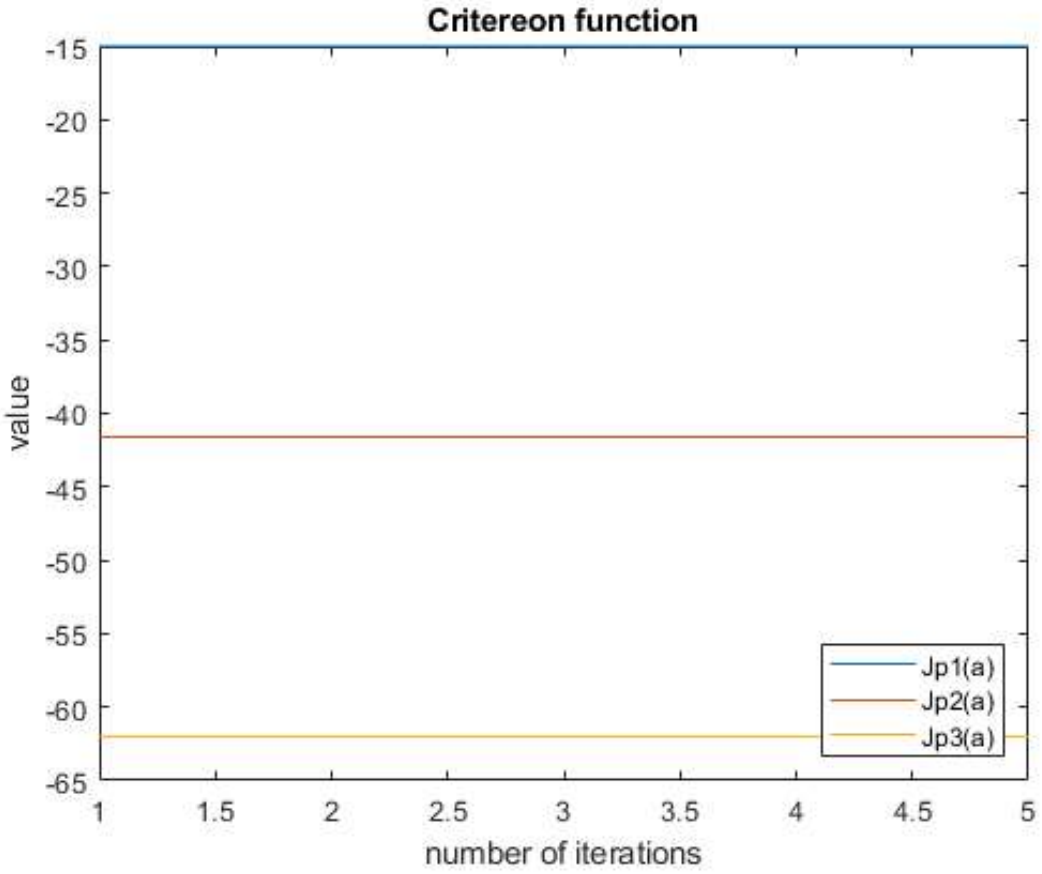
```
w6 = 1×3
    1.4000    0.4220   -0.8870
```

```
display(conv6)
```

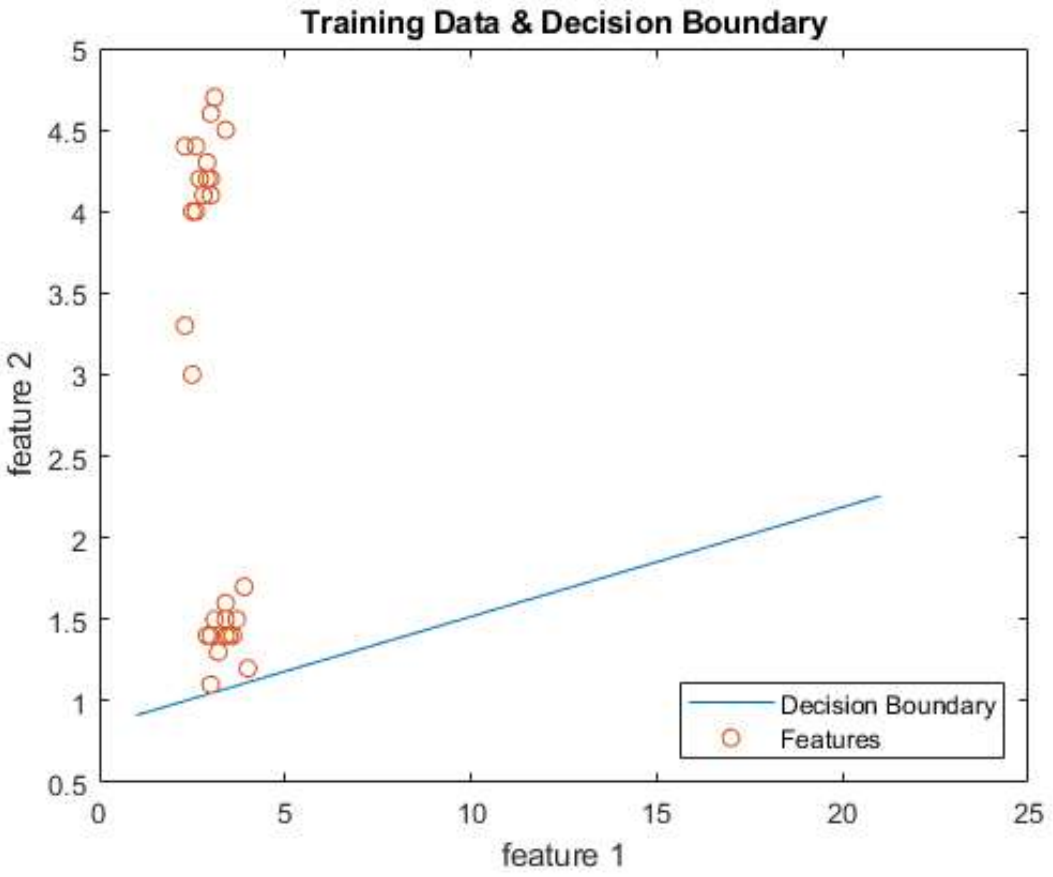
```
conv6 = 5
```

```
figure
```

```
plot(Jp6)
title("Critereon function")
xlabel("number of iterations")
ylabel("value")
legend({"Jp1(a)", "Jp2(a)", "Jp3(a)"}, "location", "southeast")
```




```
figure
plot(bound6)
hold on
scatter(ABtraining(:,1),ABtraining(:,2))
title("Training Data & Decision Boundary")
xlabel("feature 1")
ylabel("feature 2")
legend({"Decision Boundary", "Features"}, "location", "southeast")
hold off
```



ABtraining, coarse $\eta(k)$

```
clear
setup

a = [0,0,1];
eta = 0.1;
iter = 300;
theta  = 0;

[w7, Jp7, conv7, fin7] = gradientDescent2(a , theta, ABtraining, eta, iter, 1);
[b71, b72] = decisionBoundary(fin7);
scale = 1:.1:3;
slope7= b72/b71;
bound7 = slope7 .* scale + b72;
display(w7)
```

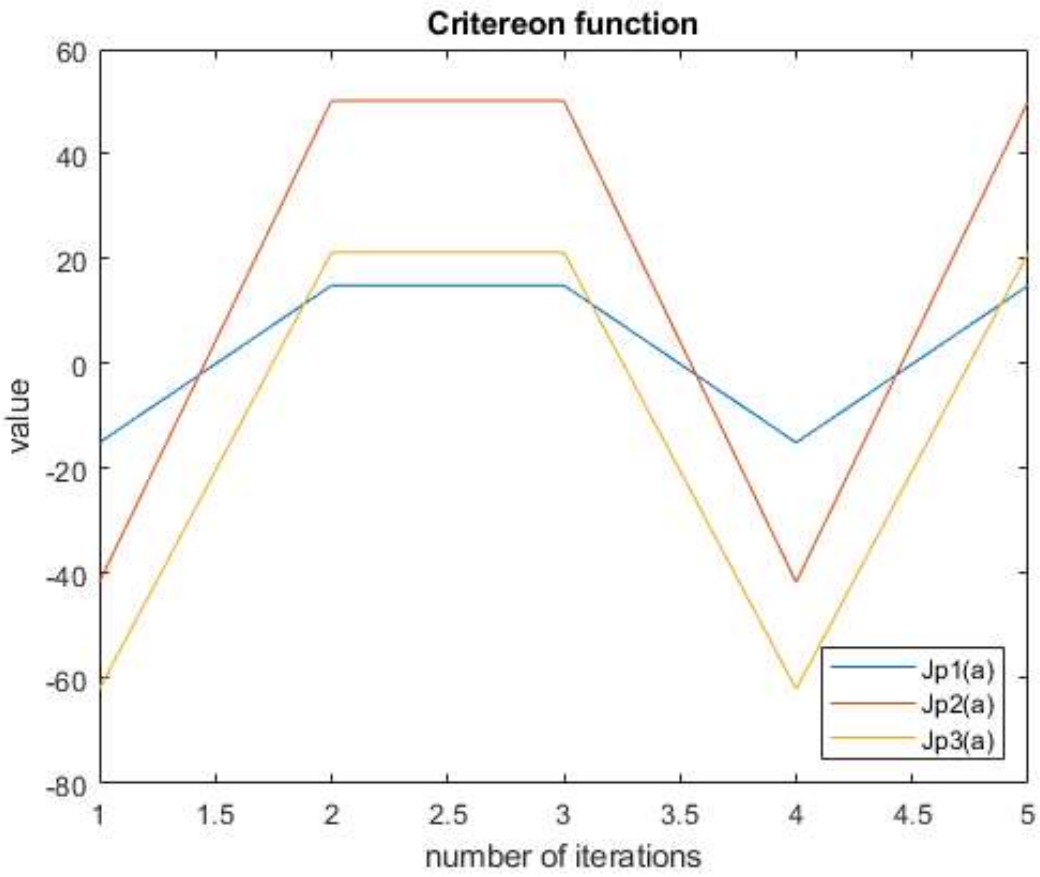
```
w7 = 1×3
    0.9000    5.0300   -7.3400
```

```
display(conv7)
```

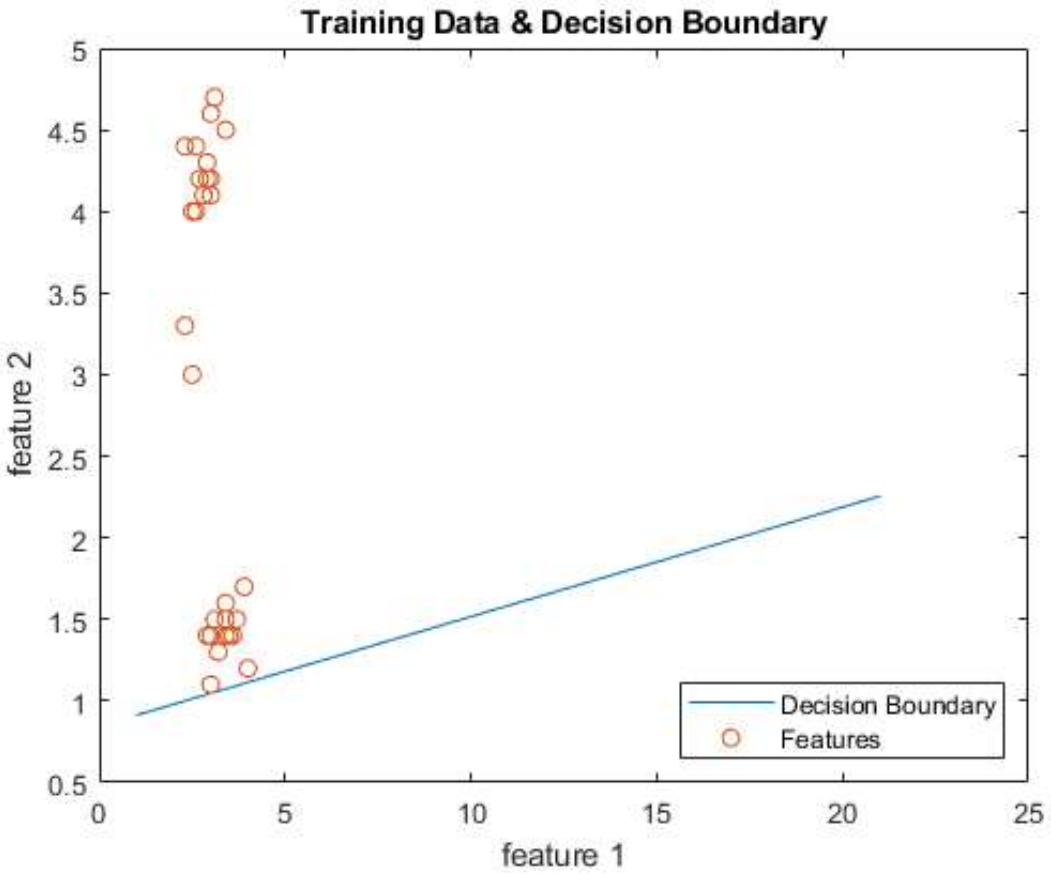
```
conv7 = 5
```

```
figure
```

```
plot(Jp7)
title("Critereon function")
xlabel("number of iterations")
ylabel("value")
legend({"Jp1(a)", "Jp2(a)", "Jp3(a)"}, "location", "southeast")
```




```
figure
plot(bound7)
hold on
scatter(ABtraining(:,1),ABtraining(:,2))
title("Training Data & Decision Boundary")
xlabel("feature 1")
ylabel("feature 2")
legend({"Decision Boundary", "Features"}, "location", "southeast")
hold off
```



ABtraining, fine $\eta(k)$

```
clear
setup

a = [0,0,1];
eta = 0.001;
iter = 300;
theta  = 0;

[w8, Jp8, conv8, fin8] = gradientDescent2(a , theta, ABtraining, eta, iter, 1);
[b81, b82] = decisionBoundary(fin8);
scale = 1:.1:3;
slope8= b82/b81;
bound8 = slope8 .* scale + b82;
display(w8)
```

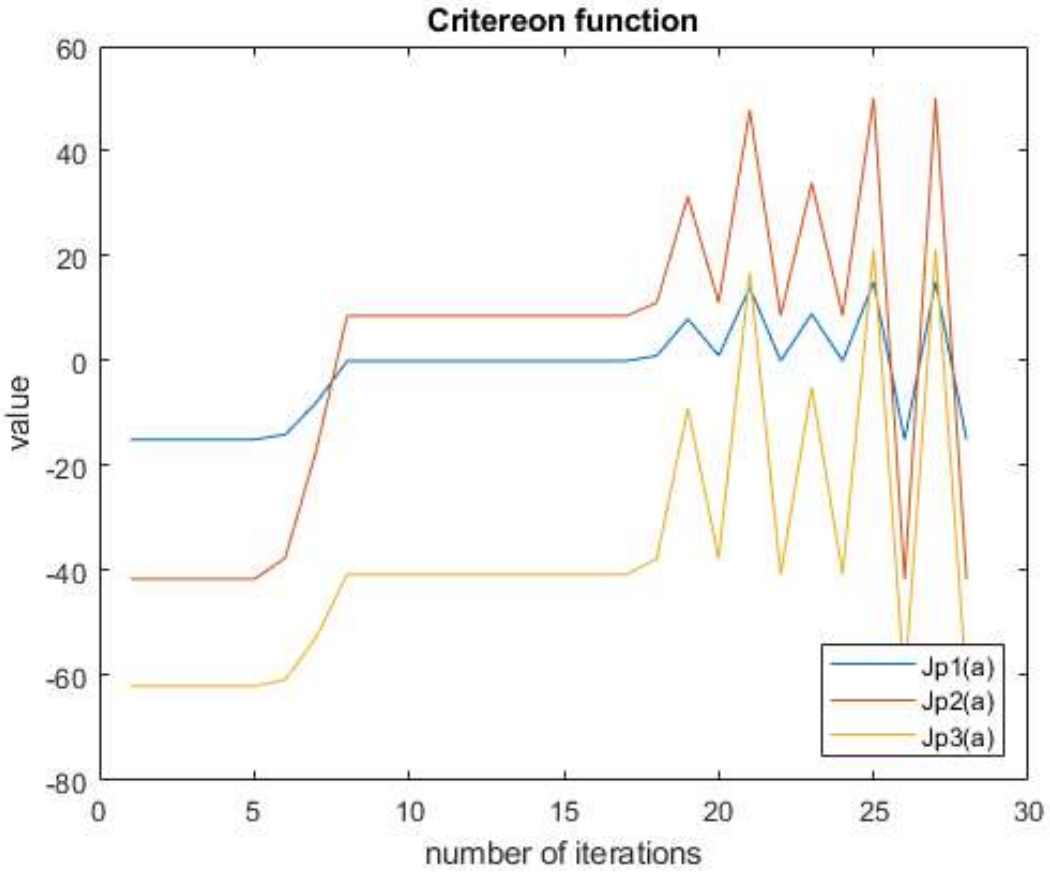
```
w8 = 1×3
   -0.0490    0.0433   -0.0445
```

```
display(conv8)
```

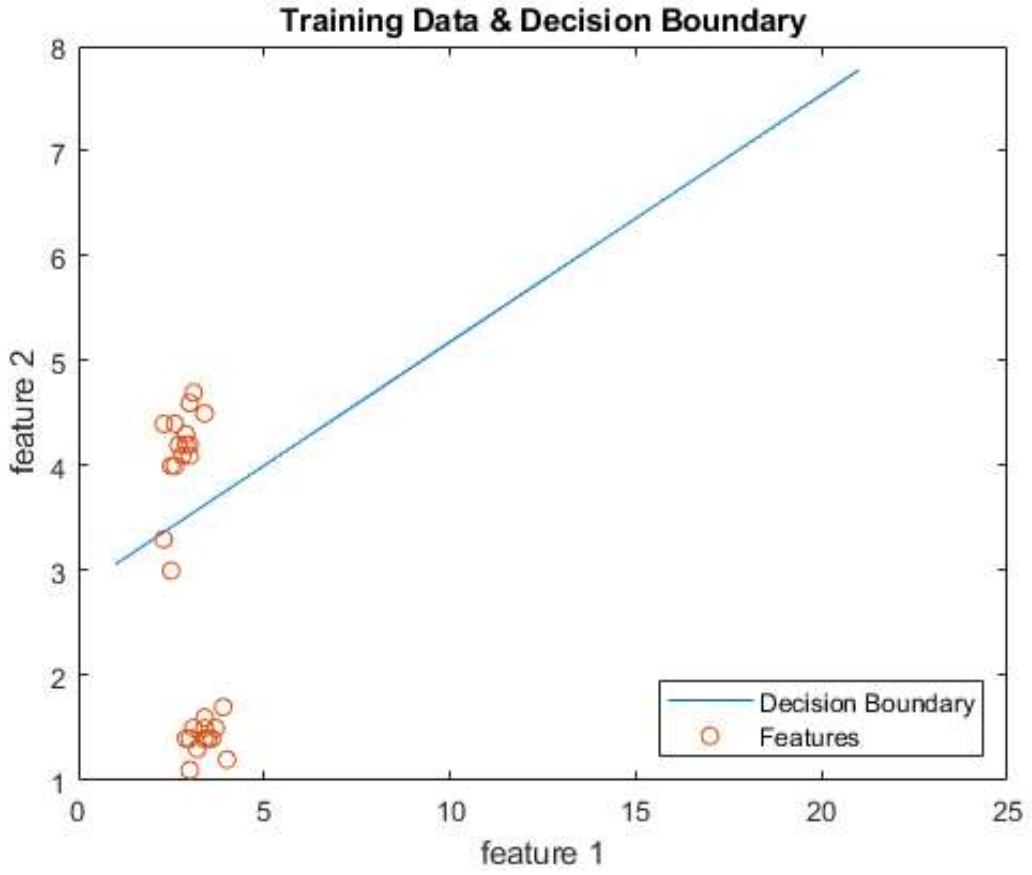
```
conv8 = 28
```

```
figure
```

```
plot(Jp8)
title("Critereon function")
xlabel("number of iterations")
ylabel("value")
legend({"Jp1(a)", "Jp2(a)", "Jp3(a)"}, "location", "southeast")
```




```
figure
plot(bound8)
hold on
scatter(ABtraining(:,1),ABtraining(:,2))
title("Training Data & Decision Boundary")
xlabel("feature 1")
ylabel("feature 2")
legend({"Decision Boundary", "Features"}, "location", "southeast")
hold off
```



Question 7

Discuss in your report the following: (i) the effect of using different sizes for training and test data, (ii) the effect of different learning rates, threstemp, initial weight values, (iii) the criterion function over the iterations, and (iv) comment on the achieved classification accuracy.

(i)

By utilizing a larger training set the gradient descent will converge in less iterations since it has more data to pass through. This is evident if you observe the time it took the ABtraining dataset to converge compared to the ABtraining dataset which is twice as large as its counterpart;

(ii)

Learning Rates: By utilizing a more fine learning rate you increase the time required to converge and sometimes it is possible to over fit the dataset thus producing error. By utilizing a more coarse learning rate you decrease the time required to converge and again you can under fit the dataset prodcuing error. In our case the finer learning rate produced more error, likely due to the lack of training data (only 30 sets) and the coarser learning rate produced zero error. Threshold: The threshold is essentially the acceptable amount of error, so by changing the threshold you are allowing a certain amount of error. Therefore, it is ideal to keep the error as low as possible, but with very large datasets this may not be possible due to the computation time required. Initial Weight Values: By changing the intial weight values it can have many effects, in some cases such as when we changed it to [1,1,1] it allows you produced

zero error and converge sooner. In other cases, such as when we changed it to [2,2,2] you may converge sooner, but there is more error.

(iii)

Since the criterion function is based the on the features that were miss classified it behaves like a bang-bang control system, where it will correct itself over time by overshooting, then undershooting until it converges correctly. You can see this in the plots from question 6. In some cases like when using dataset BC, the features are too similar thus creating a very noisy criterion function, in this case 300 iterations was not enough for it to converge properly resulting in miss classifying class 3 wrong.

(iv)

for datasetAB a perfect classification was achieved based on the initial values provided in questions 1 when both the training and testing sets where used to train. When using datasetBC, the features are too similar producing a 50% error, which is really a 100% error since everything is being classified as class 2, had the amount of allowable iterations been increased convergence may have been possible.