AHSANULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

CSE 4213 : PATERN RECOGNITION LAB
EXPERIMENT 02

Peceptron Gradient Decent

Author:
Naimul Haque

Roll:
14.02.04.080 (B1)

Submission date: June 08, 2018

Naimul Haque, *14.02.04.080* ,
Ahsanullah University of Science and Technology

1 Perceptron Algorithm

The Perceptron is inspired by the information processing of a single neural cell called a neuron.

A neuron accepts input signals via its dendrites, which pass the electrical signal down to the cell body.

In a similar way, the Perceptron receives input signals from examples of training data that we weight and combined in a linear equation called the activation.

Mathematically,

$$g(x) = w^t x_a$$

where w is the weight vector and x_a is the augmented feature vector.

2 Gradient Descent

Gradient Descent is the process of minimizing a function by following the gradients of the cost function.

This involves knowing the form of the cost as well as the derivative so that from a given point you know the gradient and can move in that direction, e.g. downhill towards the minimum value.

For the Perceptron algorithm, each iteration the weights (w) are updated using the equation

$$w_{t+1} = w_t + \mu \sum_{x \in M} x_a$$

where **M** is the set of miss-classified features.

3 Problem set up

Suppose,we have following sample in our two class problem :

```
The feature vector:

m =

      1    1     1
      1   -1     1
      4    5     1
      2    2.5  -1
      0    2    -1
      2    3    -1
```

where **x = m(:,1:2)** and **classes = m(:,3)**

We now we generate the high dimensional sample points **y**, as discussed in the class. We used the following formula:

$$y = [x1^2, x2^2, x1 * x2, x1, x2, 1]$$

We get the values of **y** as :

```
The feature vector:
m =

      1    1     1
      1   -1     1
      4    5     1
      2    2.5  -1
      0    2    -1
```

```
2    3    -1


The tranformed feature vector:


y =

    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
    1.0000    1.0000   -1.0000    1.0000   -1.0000    1.0000
   16.0000   25.0000   20.0000    4.0000    5.0000    1.0000
   -4.0000   -6.2500   -5.0000   -2.0000   -2.5000   -1.0000
         0   -4.0000         0         0   -2.0000   -1.0000
   -4.0000   -9.0000   -6.0000   -2.0000   -3.0000   -1.0000
```

The value of **w** is initialized with all zeroes. $w = [111111]$

4 Training

The set the learning rate $\mu = 0.1$ and start our **gradient decent algorithm** with

The step by step numerical analysis for 2 epoch is shown below:

```
Epoc = 1

Learning rate = 0.1

The weight vector:


w =

     1     1     1     1     1     1


The Feature fector:

y =

    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
    1.0000    1.0000   -1.0000    1.0000   -1.0000    1.0000
   16.0000   25.0000   20.0000    4.0000    5.0000    1.0000
   -4.0000   -6.2500   -5.0000   -2.0000   -2.5000   -1.0000
         0   -4.0000         0         0   -2.0000   -1.0000
   -4.0000   -9.0000   -6.0000   -2.0000   -3.0000   -1.0000


The updated weight vector:

w =

  0.2000   -0.9250   -0.1000    0.6000    0.2500    0.7000

The g(x) =

    0.7250
    0.4250
  -17.5750
    2.9563
    2.5000
    5.4750

Number of misclasifications = 1

Epoc = 2

Learning rate = 0.1

The weight vector:


w =

    0.2000   -0.9250   -0.1000    0.6000    0.2500    0.7000

The updated weight vector:

w =

    7.3000    6.1750    7.0000    7.7000    7.3500    7.8000

The feature vector:
y =

    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
    1.0000    1.0000   -1.0000    1.0000   -1.0000    1.0000
   16.0000   25.0000   20.0000    4.0000    5.0000    1.0000
   -4.0000   -6.2500   -5.0000   -2.0000   -2.5000   -1.0000
         0   -4.0000         0         0   -2.0000   -1.0000
   -4.0000   -9.0000   -6.0000   -2.0000   -3.0000   -1.0000


The g(x) =

   43.3250
   14.6250
  486.5250
 -144.3688
  -47.2000
 -172.0250


Number of misclasifications = 3
```

5 Code

The algorithm is implemented using **Matlab** and the code is presented below:

```
clc;
clear all;
%x1 %x2 %classes
m=[

1   1   1
1   -1  1
4   5   1
2   2.5    -1
0   2    -1
2   3   -1

];

x1 = m(:,1);
x2 = m(:, 2);
classes = m(:,3);


plot(x1(classes == 1),
   x2(classes == 1),'b*');
hold on;
```

```
plot(x1(classes == -1),
   x2(classes == -1),'r*');

title('Distribution of
   datapoints of
   classes(1,-1)');
xlabel('x1');
ylabel('x2');
xlim([-6 9]);
ylim([-4 8]);
grid on;

y = [x1(classes == 1).^2
   x2(classes == 1).^2
   x1(classes ==
   1).*x2(classes == 1)
   x1(classes == 1)
   x2(classes == 1)    [1 1 1]'
    ;
    -x1(classes == -1).^2
       -x2(classes == -1).^2
       -x1(classes ==
       -1).*x2(classes == -1)
          -x1(classes == -1)
       -x2(classes == -1)
       -[1 1 1]'];

w = [0 0 0 0 0 0];

epoch = 2000;
alpha = 0.1;

for i = 1:epoch
    w = w +
       alpha*sum(y(y*w'<=0,:));
end

syms x1 x2;

s =
   sym(w(1,1)*x1*x1+w(1,2)*x2*x2+w(1,3)*x1*x2+w(1,4)*
   x1+w(1,5)*x2+w(1,6));
s2=solve(s,x2);
xvals1=[-10:0.01:10];
xvals2(1,:) =
   subs(s2(1),x1,xvals1);
plot(xvals1,xvals2(1,:),'k');
```

6 Output Graph

The decision boundary is drawn after training the weighting vector after **2000** epochs.

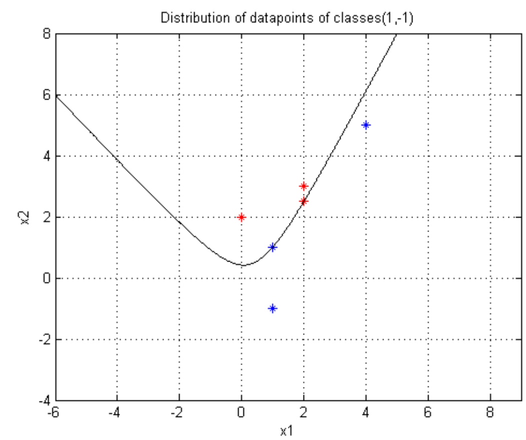

Figure 1. Decision Boundary