CMPE 297 – Machine Learning

Spring 2017

Homework #1

Name: Ritvick Paliwal

SJSU ID: 010733970

Quesition 1:
Express specifying the input space X, output space Y, target function f : X -> Y , and the specifics of the data set that you can learn from for a problem of interest to you in the framework of learning from data by which there is no analytic solution, but you have data from which to construct an empirical solution. (20%)

The data I am considering here is to predict the health of the water in California Bay Area's water bodies. The water health (health from perspective of positive or negative impact on fish's health) is dependent on various factors viz pH, dissolved CO2, dissolved O2, Amount of Sodium etc.. The health of the water is here classified as Good (+1) or Bad (-1) which forms our output space.

For simplicity the input space is classified with two attributes pH and dissolved CO2.

Formalization:

Input x -> Attributes in water bodies (pH, CO2)
Output y -> Good / Bad health ? (+1 / -1)
Target Function: f: X->Y (Ideal formula to decide water health)
Data: (x1,y1), (x2,y2,) .. , (xn,yn) (Historical Records from bay-keepers)
Hypothesis: g: X->Y

Sample Historical Data

X	X1 (pH)	8.1	9	8.5	8.4	7.8	6	7.7	6.2
	X2 (CO2)	0.5	9.8	0.6	0.3	0.2	2.1	0.4	2.3
y		+1	+1	+1	+1	+1	-1	+1	-1

The good / bad health is determined by a threshold value. If the value calculated by the Target function is > threshold, then the health is good otherwise bad.

Ideal target function will be summation of $w_i.x_i$ for all i = 0 to n where w is weight for each attribute and x is the attribute value.

Question 2:
Create a target function f and the data set D to see how the perceptron learning algorithm works. Take d = 2 for visualization and choose a random line in the plane as your target function, where map one side as +1 and the other side as -1. Choose the inputs xn of the data set as random points in the plane and evaluate the target function on each xn to get the corresponding output yn.

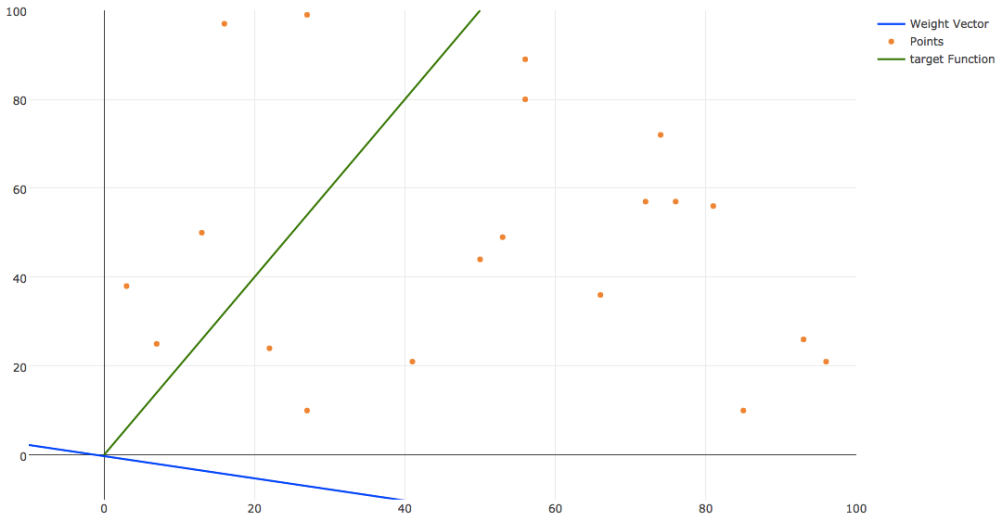
Generate a data set of size 20 (e.g., Gaussian random). Try the PLA on your data set to see how long (how many updates) it takes to converge and how well the final hypothesis g matches your target function f (by plots). (50%)

Taking the ideal equation as 2x – y = 0;
We have a random function to generate 20 points. (see Code)

Starting weights are = [25,25,100]

Iteration 1 :

Perceptron Algorithm - Demo

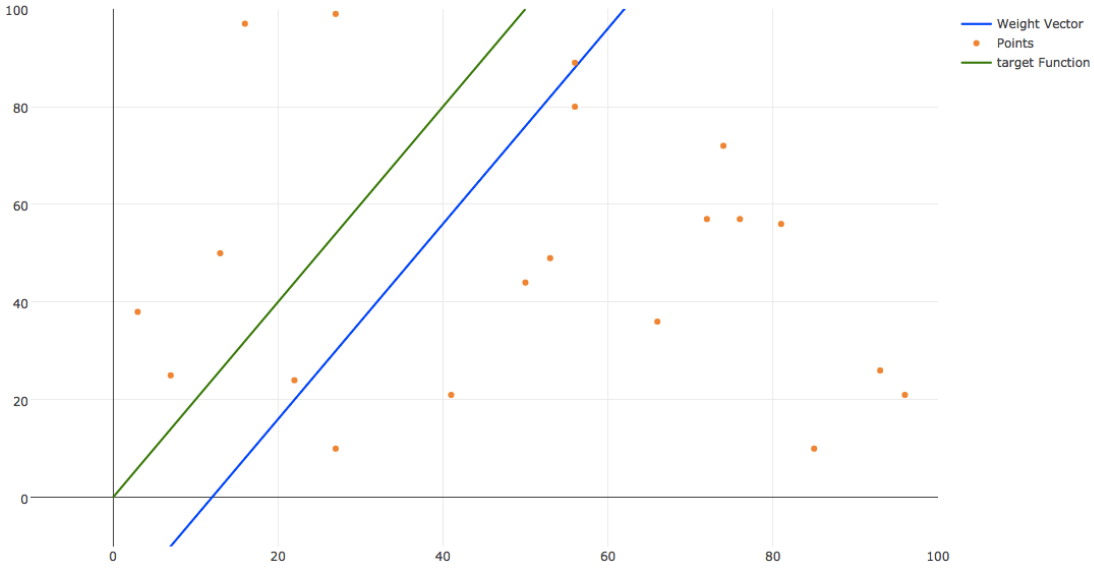


Itiration

Current Itiration = 0 weights = 25,25,100

Iteration 2 :

Perceptron Algorithm - Demo

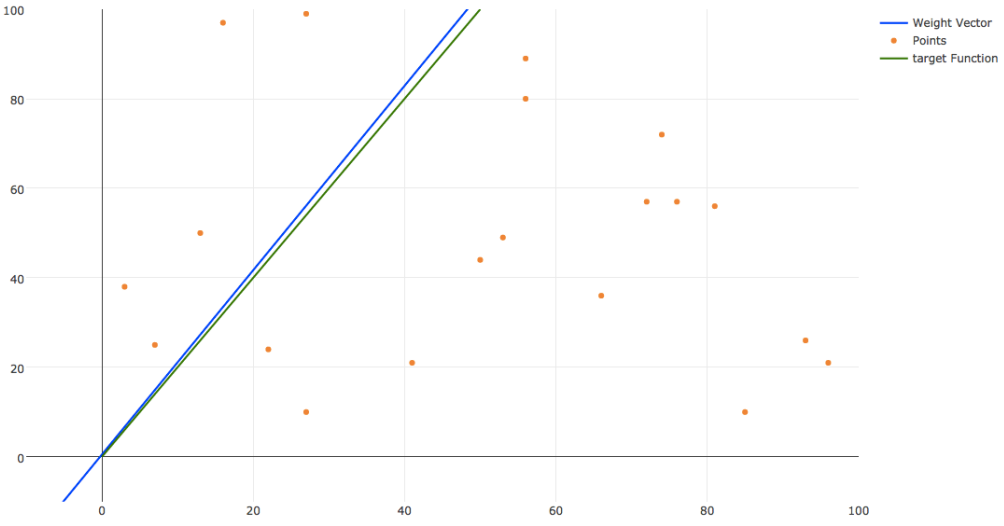


Itiration

Current Itiration = 1 weights = 24,-2,1

Iteration 6 :

Perceptron Algorithm - Demo



Itiration

Current Itiration = 6 weights = 25,103,-50

Code Link Here:
https://github.com/ritvick/perceptron-plotly

Demo Link:
http://ritvick.com/ms/ml/perceptron/

3. Use the linear regression algorithm to run the classification problem

The linear regression algorithm runs for real values and even +1, -1 can be considered as real values (in fact they are).

we will consider weight vector as = $\begin{pmatrix} w1 \\ w2 \end{pmatrix}$

The Data Matrix will be = $\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$ 20 in our case.

The Result vector is = $\begin{pmatrix} y^1 \\ y^2 \\ \vdots \\ y^n \end{pmatrix}$ +1/-1 for each case generated in code

We calculate. E_{in}

$$E_{in}(w) = \frac{1}{N} \sum_{n=1}^{N} (w^T x_n - y_n)^2$$

$$= \frac{1}{N} \|Xw - y\|^2$$

To Minimise we equate derivative to 0.

$$\nabla E_{in}(w) = \frac{2}{N} X^T(Xw - y) = 0$$

$$X^T X w = X^T y$$

$$w = (X^T X)^{-1} X^T y$$

This will be initial weights and it can be used as starting point for PLA or Pocket Algorithm.

Code to calculate w is below :⇒

```
<script type="text/javascript">

        // For Question 3

        var mathjs = math;
        console.log(mathjs);
        var X = [];
        for(var i =0; i < pointsX1.length; i++) {

            var point = [pointsX1[i],pointsX2[i]];
            X.push(point);

        }
        console.log(X.toString());
        var y = mathjs.transpose(resultY);
        console.log(y.toString());


        console.log();
        var transpose = mathjs.transpose(X);
        console.log(transpose);
        var mult = (mathjs.multiply(transpose,X));

        console.log(mult);
        var inv = mathjs.inv(mult);
         console.log(inv);

         var second = mathjs.multiply(transpose,y);
         console.log(second);
        console.log((mathjs.multiply(inv,second)));
```

```
1,1,1,1,1,-1,1,1,-1,1,-1,1,-1,1,1,1,-1,1,1,-1                    (index):233
▶ [Array[20], Array[20]]                                          (index):238
▶ [Array[2], Array[2]]                                            (index):241
▶ [Array[2], Array[2]]                                            (index):243
▶ [777, 181]                                                      (index):246
▶ [0.02094795748504346, -0.010673443726348677]                    (index):247
```

Code: Same index.html last script tag.
https://github.com/ritvick/perceptron-plotly/blob/master/index.html

Demo link: http://ritvick.com/ms/ml/perceptron/