Chapter 1.2: Gradient Descent with Math.

Madhu Sanjeevi (Mady) Follow
Sep 26, 2017 · 4 min read

This story I wanna talk about a famous machine learning algorithm called *Gradient Descent* which is used for optimizing the machine leaning algorithms and how it works including the math.

From chapter 1 we know that we need to update *m* and *b* values, we call them **weights** in machine learning. Lets alias *b* and *m* as – *θ0* and *θ1* (theta 0 and theta 1) respectively.

First time we take random values for *θ0* and *θ1*, and we calculate y

y = θ0+θ1*X
In machine learning we say hypothesis so **h(X) = θ0+θ1*X**

h(X)=y but this y is not actual value in our data-set, this is predicted y from our hypothesis.

For example lets say our data-set is something like below and we take random values which are **1** and **0.5** for **θ0** and **θ1** respectively.

```
X          y              h(x)= θ0+θ1*X          predicted y
10         5                    1+0.5*10              6
12         6.6
3          1
...        ...    { Actual y value is 5 and predicted y value is 6 }
```

From this we calculate the error which is

```
error = (h(x)-y)² --> (Predicted - Actual)²
error = (6-5)² = 1

² is to get rid of negative values (what if Actual y=6 and Py=5)
```

we just calculated the error for one data point in our data-set , we need to repeat this for all data points in our data set and sum up the all errors to one error which is called ***Cost Function 'J(θ)'*** in machine learning.

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Cost Function.

m The number of training examples

m The number of training examples

$x^{(i)}$ The input vector for the i^{th} training example

$y^{(i)}$ The class label for the i^{th} training example

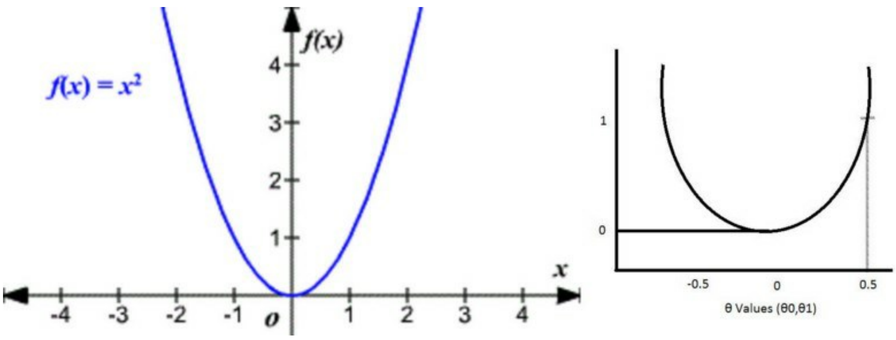
Θ The chosen parameter values or "weights" (Θ_0, Θ_1, Θ_2)

$h_\theta(x^{(i)})$ The algorithm's prediction for the i^{th} training example using the parameters Θ.

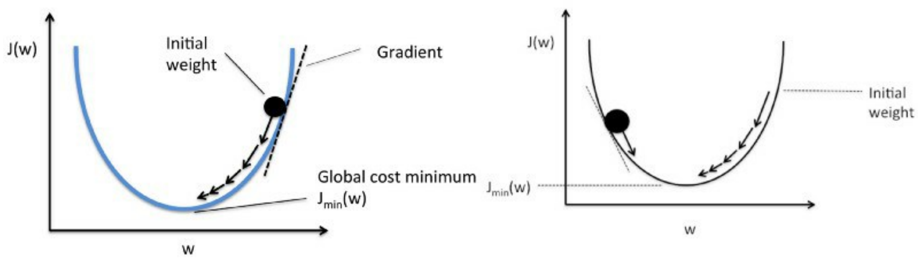
Our goal is to minimize the cost function (error) **we want our error close to zero Period.**

we have the error **1** for first data-point so lets treat that as whole error and reduce to zero for sake of understanding.

for $(h(x)-y)^2$ function we get always positive values and graph will look like this(Left) and lets plot the error graph.



Here is the gradient descent work comes into the picture.



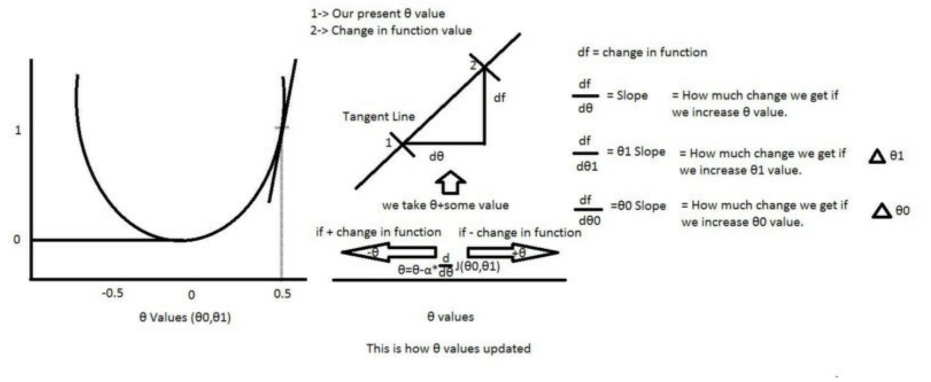
+θ values (Left), -θ values(Right)

By taking the little steps down to reach the minimum value (bottom of the curve) and changing the **θ** values in the process.

> *How does it know how much value it should go down???*

The answer is in Math.

1. It draws the line(Tangent) from the point.
2. It finds the slope of that line.
3. It identifies how much change is required by taking the partial derivative of the function with respective to **θ**
4. The change value will be multiplied with a variable called **alpha**(learning rate) *we provide the value for alpha usually 0.01*
5. It subtracts this change value from the earlier **θ** value to get new **θ** value .



From above picture we can define our **θ0 and θ1.**

And alpha here is a learning rate usually we give 0.01 but it depends, it tells how big the step-size is towards reaching the minimum value.

$$\theta_0 := \theta_0 - \alpha \frac{d}{d\theta_0} J(\theta_0, \theta_1)$$
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_0, \theta_1)$$

Derivatives:

$$\frac{d}{d\theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$$
$$\frac{d}{d\theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

Repeat until convergence

{

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

}

θ0 and θ1 values(Left),more than two θ's (Right)

Again we know our J(**θ0,θ1)** so if we apply this to above equations for **θ0** and **θ1**, we get our new **θ0 and θ1** values**.**

> *How to calculate the derivatives???*

For example f(x) =x² → df/dx=2x How ???

$$x^n = n*x^{n-1}$$

> *How to calculate the partial derivatives???*

its same as calculating derivatives but here we calculate the derivative with respective to that value , others are constants (so d/dx(constant)=0)

$$\frac{d}{d\theta_1} J(\theta_1, \theta_2) = \frac{d}{d\theta_1} \theta_1^{\,2} + \overset{0}{\cancel{\frac{d}{d\theta_1} \theta_2^{\,2}}} = 2\theta_1$$

$$\frac{d}{d\theta_2} J(\theta_1, \theta_2) = \underset{0}{\cancel{\frac{d}{d\theta_2} \theta_1^{\,2}}} + \frac{d}{d\theta_2} \theta_2^{\,2} = 2\theta_2$$

The same thing we can apply for calculating partial derivative with respective to **θ0** and **θ1**.

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$\frac{d}{d\theta_0}J(\theta_0, \theta_1) = \frac{d}{d\theta_0}\left(\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2\right)$$

$$= \frac{1}{2m}\sum_{i=1}^{m}\frac{d}{d\theta_0}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Product Rule

$$= \frac{1}{2m}\sum_{i=1}^{m}2\left(h_\theta(x^{(i)}) - y^{(i)}\right)\boxed{\frac{d}{d\theta_0}\left(h_\theta(x^{(i)}) - y^{(i)}\right)}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

How come that box drawn disappeared in the next step above? just wait and see.

For calculating partial derivative with respective to **θ1** is also same as above except one little part is added

$$\frac{d}{d\theta_0}\left(h_\theta(x^{(i)}) - y^{(i)}\right) = \frac{d}{d\theta_0}\left(\overset{1}{\theta_0} + \overset{0}{\theta_1 x^{(i)}} - \overset{0}{y^{(i)}}\right) = 1$$

$$\frac{d}{d\theta_1}\left(h_\theta(x^{(i)}) - y^{(i)}\right) = \frac{d}{d\theta_1}\left(\overset{0}{\theta_0} + \theta_1 x^{(i)} - \overset{0}{y^{(i)}}\right) = x^{(i)}$$

θ0 box disappeared because value is 1 (Top)

So Final picture is

Derivatives:

$$\frac{d}{d\theta_0}J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

$$\frac{d}{d\theta_1}J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)\cdot x^{(i)}$$

Final **θ0** and **θ1** values

Hope its not confusing , and I know its little bit hard to grasp in the beginning but I am sure that this will make sense as you go through again and again.

So That’s it for this story , In the next story I will cover another interesting topic in machine learning so See ya!

Update : Code for Gradient Descent and linear regression

Machine Learning | Gradient Descent | Supervised Learning | Regression | Classification

1.1K claps

8

Madhu Sanjeevi (Mady)

Follow

Writes about Technology (AI, ML, DL) | Writes about Human Mind and Computer Mind. interested in ||Programming || Science || Psychology || NeuroScience || Math

Deep Math Machine learning.ai

Following

This is all about machine learning and deep learning (Topics cover Math,Theory and Programming)

Related reads

Implementation of Gradient Descent in Python

Deepak Battini
Sep 27, 2018 · 5 min r

470

Related reads

Delta Learning Rule & Gradient Descent | Neural Networks

Random Nerd
Apr 19, 2018 · 6 min r

409

Related reads

Step-by-Step Tutorial on Linear Regression with Stochastic...

Raimi Karim
Dec 11, 2018 · 10 min

166

Responses

Write a response…

Show all responses