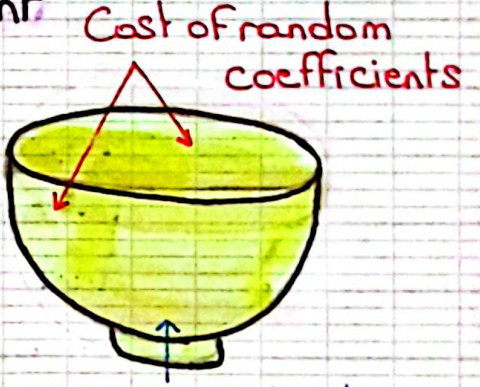
Gradient Descent

- Simple optimization algorithm that can be used with any machine learning algorithm
- Gradient descent is simply used to find the values of a function's parameters (coefficients) that minimize a cost function as much as possible

1) Intuition of Gradient Descent



- It's like a large bowl where we store some fruit
- This bowl represent the plot of the cost function (f).
- A randomn position on the surface of the bowl is the cost of the current values of the coefficients.
- The bottom of the bowl is the cost of the best set of coefficients, that give the minimum of the cost function

- This is an iterative method, so we should move from a random point, to reach the minimum point.
- As we moving we're varying the coefficients of the line until we reach the best line (or when we reach the minimum error)

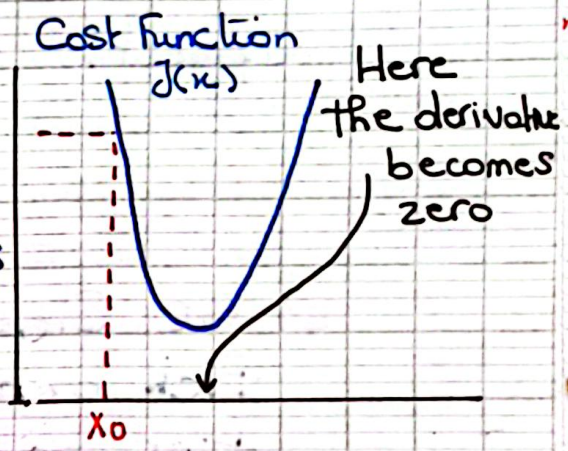
2) How does Gradient Descent Work?

. Iterative process, where we start at a coefficient's initial point (x_0) and we move step by step until we reach a minimum

. The update rule of our position is given by the formula:

$$x_{t+1} = x_t - \alpha \frac{dJ}{dx}\Big|_{x_t}$$

- x_{t+1}: parameter at step $t+1$
- x_t: parameter at previous step t
- α: Learning rate
- $\frac{dJ}{dx}\Big|_{x_t}$: derivative of the cost function with $x = x_t$

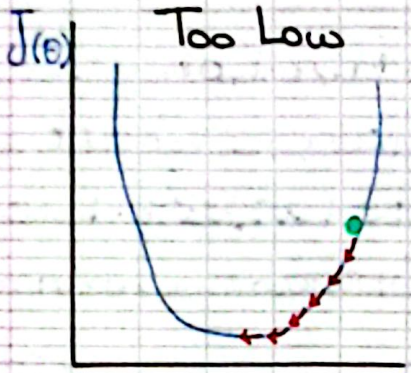


. We stop when the values of x don't change a lot : $x_{t+1} \approx x_t$

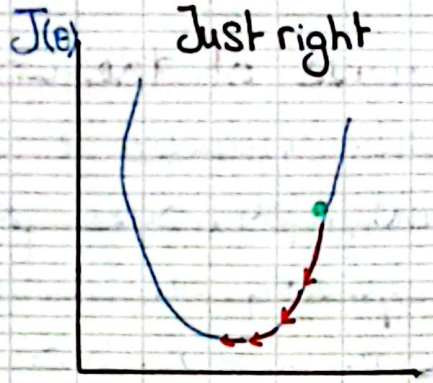
. Since it's an iterative process, we can stop when we reach the maximum nb of iterations

3) The importance of the Learning Rate (α)

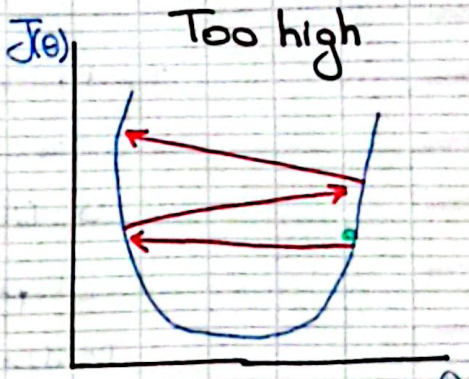
How big of steps a gradient descent algorithm takes into the direction of the local minimum is determined by the learning rate wich determines the rate (how fast or slow) we will move towards the optimal coefficients/parameters



A small learning rate requires many updates before reaching the minimum point (very cost)



The optimal learning rate swiftly reaches the minimum point.



Too large of a learning rate causes drastic updates which lead to divergent behaviors.
(risk overshooting min)

4) GD Steps

- Pick an initial value for learning rate α
- Pick an initial value for the parameter x_0
- Evaluate the derivative over x_t
- Update the parameter x_{t+1} using the GD update formula
- Repeat until we reach convergence (i.e. minimum cost)

$$x_{t+1} = x_t - \alpha \left.\frac{dJ}{dx}\right|_{x_t}$$

Derivative of the Cost Function

- The cost function in a linear regression problem is mainly the mean squared error (MSE)
- In the MSE, we compute the derivatives as follows in general form.

$$\text{Cost} = \frac{1}{n}\sum_{i=1}^{n}(y - y_{pred})^2 = \frac{1}{n}\sum_{i=1}^{n}(y - (a_0 + a_1 x))^2$$

- Our task is to find a_0 and a_1 that minimize the cost, but how we do this?
  1) We should take the partial derivatives of the cost with respect to a_0 and a_1
  2) Set the partial derivatives equal to 0
  3) Solve for a_0 and a_1

1) Partial Derivatives

- Let's start taking the partial derivative with respect to a_0

$$\frac{d\,cost}{da_0} = \frac{d}{da_0}\frac{1}{n}\sum_{i=1}^{n}(y_i - (a_0 + a_1 x))^2$$

derivative of sum is the sum of derivatives

$$= \frac{1}{n}\sum_{i=1}^{n}\frac{d}{da_0}(y - (a_0 + a_1 x))^2$$

$u^n = n u^{n-1} \cdot u'$

as we take the derivative with respect to a_0 we treat $y_i, a_1 x_i$ as constants

$$= \frac{1}{n}\sum_{i=1}^{n} 2(y_i - (a_0 + a_1 x)) \cdot \frac{d}{da_0}(y - (a_0 + a_1 x))$$

$y' = -1$

$$= \frac{1}{n}\sum_{i=1}^{n} 2(y - (a_0 + a_1 x)) \cdot (-1)$$

$$= -\frac{2}{n}\sum_{i=1}^{n}(y - (a_0 + a_1 x))$$

. Now with respect to a_1

$$\frac{d\,cost}{da_1} = \frac{d}{da_1}\frac{1}{n}\sum_{i=1}^{n}(y-(a_0+a_1x))^2$$

The derivatives of sum is the sum of the derivatives

$$= \frac{1}{n}\sum_{i=1}^{n}\frac{d}{da_1}(y-(a_0+a_1x))^2$$

$u^n = nu^{n-1}\cdot u'$

as we take the derivative in respect to a_1, we treat y_i, a_0 as constants.

$$= \frac{1}{n}\sum_{i=1}^{n}2(y-(a_0+a_1x))\cdot\underbrace{\frac{d}{da_1}(y-(a_0+a_1x))}_{=-x}$$

$$= \frac{1}{n}\sum_{i=1}^{n}2(y-(a_0+a_1x))\cdot(-x)$$

$$= -\frac{2}{n}\sum_{i=1}^{n}x\,(y-(a_0+a_1x))$$

2) Set partial derivatives equal to 0

a. $-2\sum_{i=1}^{n}(y-(a_0+a_1x)) = 0$ Solve this for a_0

b. $-2\sum_{i=1}^{n}x(y-(a_0+a_1x)) = 0$

↑ Substitute that here for a_1

a) $\sum(y-(a_0+a_1x)) = 0$

$[\sum y - \sum a_0 - \sum a_1x] = 0$ → with respect the summation a_0 and a_1 are constants so they can come outside

$[\Sigma y - na_0 - a_1\Sigma x] = 0$

$na_0 = \Sigma y - a_1\Sigma x$

$a_0 = \bar{y} - a_1\bar{x}$

b) $\sum x(y-(a_0+a_1x)) = 0$

$\sum x(y-(\bar{y}-a_1\bar{x}+a_1x)) = 0$

$\sum x(y-\bar{y}-a_1(x-\bar{x})) = 0$

$\sum x(y-\bar{y}) - \sum a_1x(x-\bar{x}) = 0$

$\sum x(y-\bar{y}) = a_1\sum x(x-\bar{x}) \Rightarrow a_1 = \frac{\sum x(y-\bar{y})}{\sum x(x-\bar{x})}$

Gradient Descent Example

. Given a set of data points representing the height and the corresponding weights of students we need to find a linear relation through Gradient Descent

Height	Weight
160	85
175	88
180	92
190	97

. First, we plot the data points, and pass a random line $y = a_0 + a_1 x$ to initialize the parameters (a_0: bias, a_1: Slope)

. Assume we choose $a_0\_initial = 0.2$ and $a_1\_initial = 0.2$, we can compute the predicted weight as $0.2 + 0.2 \times Height$

Height	Weight	W_pred
160	85	32.2
175	88	35.2
180	92	36.2
190	97	38.2

(we can see that the predicted values are not close at all to our real weights: So we need to work on that)

. Now we can calculate the squared error (MSE) for each example and then compute the function

Height	Weight	W_pred	error	Error squared
160	85	32.2	52.8	...
175	88	35.2	52.8	...
180	92	36.2	55.8	...
190	97	38.2	58.8	...

Cost = 1/4 * Sum (Error Squared)

$d\text{cost}/da_0 = -2/4 \sum (\text{error}) = -110.1$

$d\text{cost}/da_1 = -2/4 \sum \text{Height} (\text{error}) = -19452$

} derivatives of the cost function

. Now we perform the update equations using a learning rate of 0.01

$x_{t+1} = x_t - \alpha \, dJ/dx|_{x_t}$

$a_0 = a_{0\,initial} - 0.01 \times \frac{d\,cost}{da_0} = 0.2 - 0.01 \times -110.1$
$= 1.301$

$a_1 = a_{1\,initial} - 0.01 \times \frac{d\,cost}{da_1} = 0.2 + 0.01 \times 19452$
$= 194.72$

This is only 1 iteration of GD

→ Gradient Descent Tips

1) Take into consideration the cost versus the tim (by plotting cost vs time)
Analyzing the plot, we expect that a well performing rate in descent is going to decrease during each run / iteration.
If it doesn't decrease we should try reducing the learning rate

2) We should take a small value for the learning rate (0.1, 0.01, 0.001 or 0.0001)

3) The algorithm will reach the minimum cost faster if the shape of the cost function is not skewed and distorted.
We can achieve this by rescaling all of the input variables (x) to the same range (0 - 1) which is by performing Normalization

→ Gradient Descent for logistic regression

. The same concept of GD will be applied here, but the derivatives will change.

$$C(y_i, y_i') = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(\sigma(a_0 + a_1 x_i)) + (1 - y_i)\log(1 - (\sigma(a_0 + a_1 x_i))$$

. We need to calculate the derivatives of the cost function with respect to the different parameters involved in the prediction process

$$\frac{dC}{da_0} = -\frac{1}{n}\sum_{i=1}^{n} [y_i - \sigma(a_0 + a_1 x_i)]$$

$$\frac{dC}{da_1} = -\frac{1}{n}\sum_{i=1}^{n} [y_i - \sigma(a_0 + a_1 x_i)]x$$

- If the prediction involves several features, we need to calculate the derivatives with respect to the different parameters that we have.

- Here is an example of 2 features x_1 and x_2

$$\frac{dC}{da_0} = -\frac{1}{n}\sum[y_i - \sigma(a_0 + a_1x_{1i} + a_2x_{2i})]$$

$$\frac{dC}{da_1} = -\frac{1}{n}\sum[y_i - \sigma(a_0 + a_1x_{1i} + a_2x_{2i})]x_1$$

$$\frac{dC}{da_2} = -\frac{1}{n}\sum[y_i - \sigma(a_0 + a_1x_{1i} + a_2x_{2i})]x_2$$

Note that: x_1 and x_2 refer to the features, and "i" refers to the index of the data example.

- Afterwards, we just perform the updates as usual using the famous GD update formula

$$a_i := a_i - \text{learning\_rate} \times \frac{dCost}{da_i}$$

- We keep performing the updates until we reach the stopping criteria that we set (convergence, maximum number of iterations...)