# Andrew Ng Course Note 3

Shu Wang

May 16 2023

## 1 Gradient Descent

### 1.1 Implementation of Gradient Descent

Gradient Descent is also used in deep learning models.

Have some function $J(w, b)$

Want $\underset{w,b}{\text{minimize}}\, J(w, b)$

If you have : $J(w_1, w_2, ....w_n, b)$

Your objective is to minimize $J$ over $w_1, w_2, ..w_n, b$.

In other words, your goal is to find $(w_1, w_2, ...w_n, b)$ s.t $J$ is minimized.

Outline: Start with some $w, b$, set $w = b = 0$.

Keep changing $w, b$ to reduce $J(w, b)$

Until we settle at or near a minimum.

It's possible for the surface to have more than one minimum(local).

Direction of steepest descend: a step you take at this direction would take you down faster than in any other direction.

Property of gradient descent:

When you choose a different starting point, you might end up being in a different point. These are called local minimas.

Gradient descent algorithm:

$w = w - \alpha * \frac{\partial}{\partial w} J(w, b)$

Assigning a value to $w$.

Note: the expression in mathematics and code is different.

| Assignment | Truth Assertion |
|---|---|
| a = c | a = c |
| Assigning value of c to a | a equals to c |

Equal signs can be used both in assigning a value to a variable or asserting the truth of the equality of the values.

$\alpha$ is also called the learning rate. It is a small number between 0 and 1. $\alpha$ determines how big of a step you take. If $\alpha$ is very large, that corresponds to a very aggressive gradient descent process and vise versa.

You need to assign a new value of $b$ too.

$b = b - \alpha * \frac{\partial}{\partial b} J(w, b)$

Repeat these two update steps until the algorithm converges.

Converge: reach a point at the local minimum where the parameters $w, b$ no longer change much with each additional step you take.

You want to **simultaneously** update $w, b$

Update $w$ from old one to new one,updating $b$ from old one to new one.

Correct: Simultaneous update

$tmp\_w = w - \alpha * \frac{\partial}{\partial w} J(w, b)$

$tmp\_b = w - \alpha * \frac{\partial}{\partial b} J(w, b)$

$w = tmp\_w$

$b = tmp\_b$

Incorrect way:

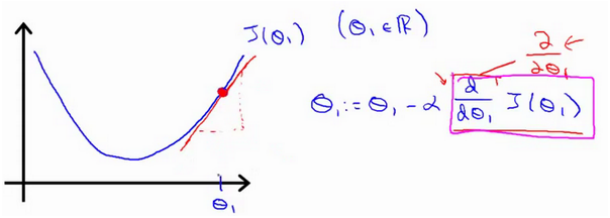
$tmp\_w = w - \alpha * \frac{\partial}{\partial w} J(w, b)$

$w = tmp\_w$

$tmp\_b = w - \alpha * \frac{\partial}{\partial b} J(w, b)$

$b = tmp\_b$

This is the updated $w$ that goes into the calculation of $tmp\_b$




Slope: positive, $\frac{\partial}{\partial b} J(w, b) > 0$

Hence $w = w - \alpha$*positive number

You are now decreasing $w$.

## 1.2 Learning Rate

$w = w - \alpha * \frac{\partial}{\partial w} J(w, b)$

If $\alpha$ is too small, you end up taking a very tiny step. Decrease the cost $J$ at a very slow rate. Gradient Descent will be slow.

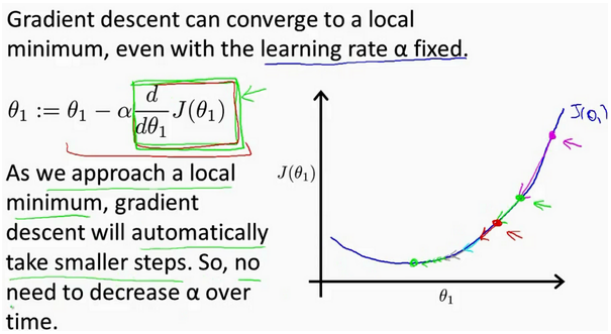
If $\alpha$ is too large, you update $w$ by a giant step directly from left to right, and cost is getting worse. May overshoot and never reach the minimum.It may fail to converge, even diverge.

If your cost $J$ is already at minimum, what will gradient descent do?

$w = w - \alpha * \frac{\partial}{\partial w} J(w,b)$ and now $\frac{\partial}{\partial w} = 0$.

Gradient descent leaves $w$ unchanged as $w = w - \alpha * 0$.

If parameters have already got local minimum, then gradient descent steps change nothing.




The next update step will not be as large as the first step. The derivative will get smaller. As we approach the minimun, the derivative gets closer and closer to 0.

## 1.3 Gradient Descent for Linear Regression

Recall that:

$w = w - \alpha * \frac{\partial}{\partial w} J(w,b)$

$b = b - \alpha * \frac{\partial}{\partial b} J(w,b)$

Where we have that:

$J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})^2$

Taking derivative w.r.t $w$, we have that:

$\frac{\partial}{\partial w} J(w,b) = \frac{\partial}{\partial w} (\frac{1}{2m} \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)})^2)$

$= \frac{\partial}{\partial w} \frac{1}{2m} \sum_{i=1}^{m} (w^2 (x^{(i)})^2 + 2wx^{(i)}b - 2wx^{(i)}y^{(i)})$

Note that in above, we exclude terms that doesn't contain $w$, because taking partial derivative w.r.t to $w$ would change these terms to 0.

$= \frac{1}{2m} \sum_{i=1}^{m} (2w(x^{(i)})^2 + 2x^{(i)}b - 2x^{(i)}y^{(i)})$

$= \frac{1}{m} \sum_{i=1}^{m} (w(x^{(i)})^2 + x^{(i)}b - x^{(i)}y^{(i)})$

Take the common term $x^{(i)}$ out, we have:

$\frac{\partial}{\partial w} J(w,b) = \frac{1}{m} \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)}) * x^{(i)}$

where as we know, $wx^{(i)} + b = f_{w,b}(x^{(i)})$.

So the equation can be written as:

$\frac{\partial}{\partial w} J(w,b) = \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)}) * x^{(i)}$.

Now we work on $\frac{\partial}{\partial b} J(w,b) = \frac{\partial}{\partial b} (\frac{1}{2m} \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)})^2)$.

$\frac{\partial}{\partial b} J(w,b) = \frac{\partial}{\partial b} \frac{1}{2m} \sum_{i=1}^{m} (b^2 + 2wx^{(i)}b - 2b * y^{(i)})$

Note that in above, we exclude terms that doesn't contain $w$, because taking partial derivative w.r.t to $b$ would change these terms to 0.

$\frac{\partial}{\partial b} J(w,b) = \frac{1}{m} \sum_{i=1}^{m} (b + w * x^{(i)} - y^{(i)})$

where as we know, $wx^{(i)} + b = f_{w,b}(x^{(i)})$.

So the equation can be written as:

$\frac{\partial}{\partial b} J(w,b) = \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})$

So repeat until convergence:

$w = w - \alpha * \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)}) * x^{(i)}$

$b = b - \alpha * \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})$

Reminder: Update $w, b$ simultaneously

Initial value of $w, b$ will result in different final points.

If you implement gradient descent on a convex function, it will always converge to the global minimum.

Batch gradient descent:

BATCH: each step of gradient descent uses all the training examples.