Difference between Batch Gradient Descent and Stochastic Gradient Descent

[WARNING: TOO EASY!]

Aerin Kim Follow
Sep 21, 2017 · 3 min read ★

Let's take the simplest example, which is Linear Regression.

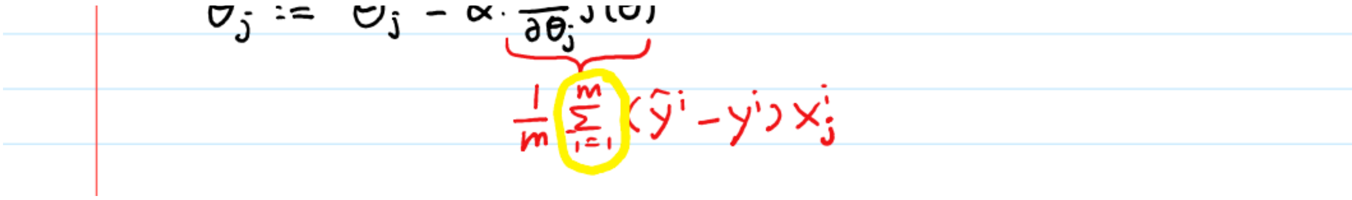
As always, we start with the cost function.

① Linear Regression Cost function $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}^i - y^i)^2$

m training data

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}^i - y^i) \cdot x_j^i$$

Linear Regression Recap done.



In the above algorithm says, **to perform the GD**, we need to calculate the gradient of the cost function J. And **to calculate the gradient of the cost function**, we need to sum (**yellow circle!**) the cost of each sample. If we have 3 million samples, we have to loop through 3 million times or use the dot product.

Here is the Python code:

```
def gradientDescent(X, y, theta, alpha, num_iters):
    """
       Performs gradient descent to learn theta
    """
    m = y.size  # number of training examples
    for i in range(num_iters):
        y_hat = np.dot(X, theta)
        theta = theta - alpha * (1.0/m) * np.dot(X.T, y_hat-y)
    return theta
```

Do you see `np.dot(X.T, y_hat-y)` above? That's the **vectorized version of "looping through (summing) 3 million samples"**.

> Wait... just to move a single step towards the minimum, do we really have to calculate each cost 3 million times?

$$\theta_j := \theta_j - \alpha \cdot \boxed{\text{only one example}} \; (\hat{y}^i - y^i) x_j^i$$

Basically, in SGD, we are using the cost gradient of **1 example** at each iteration, instead of using the sum of the cost gradient of **ALL** examples.

```
def SGD(f, theta0, alpha, num_iters):
    """
       Arguments:
       f -- the function to optimize, it takes a single argument
            and yield two outputs, a cost and the gradient
            with respect to the arguments
       theta0 -- the initial point to start SGD from
       num_iters -- total iterations to run SGD for

       Return:
       theta -- the parameter value after SGD finishes
    """
    start_iter = 0
    theta= theta0

    for iter in xrange(start_iter + 1, num_iters + 1):
        _, grad = f(theta)
        theta = theta - (alpha * grad) # there is NO dot product!
    return theta
```

Well, Stochastic Gradient Descent has a fancy name, but I guess it's a pretty simple algorithm!

. . .

c) Mini-batch gradient descent uses **n** data points (instead of **1** sample in SGD) at each iteration.

If you like my post, could you please clap? It gives me motivation to write more. :)

Machine Learning Gradient Descent Deep Learning Optimization