Gradient Descent and Siblings

Ananda Biswas

Contents

1 The What ?

Gradient Descent is a **first-order iterative** algorithm for **minimizing** a **differentiable** multivariate function.

The idea is to take repeated steps in the opposite direction of the gradient (or approximate gradient) of the function at the current point, because this is the direction of steepest descent.

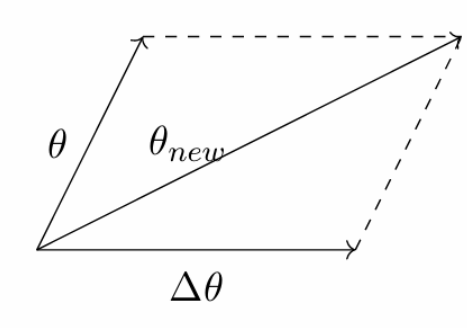
In a machine learning / deep learning problem, we have a number of parameters that we want to estimate. Our goal is to find a better way of traversing the error surface so that we can reach the minimum value quickly without resorting to guess work or brute force search which is any how infeasible. Here gradient descent provides an efficient and principled way of doing this (traversing the error surface).

2 The Math

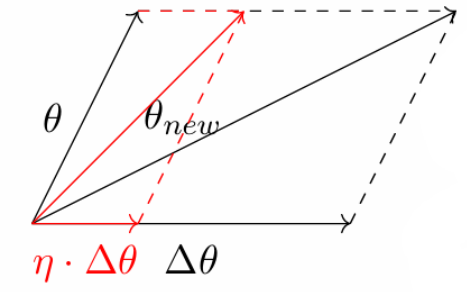
2.1 Set-up

Suppose we have a randomly initialized vector of parameters $\boldsymbol{\theta} = [w, b]$.

Let $\Delta\boldsymbol{\theta} = [\Delta w, \Delta b]$ denote change in the values of w and b respectively. So we move in the direction of $\Delta\boldsymbol{\theta}$.



Let us be a bit conservative: we move only by a small amount $\eta > 0$.



$$i.e.\ \boldsymbol{\theta}_{new} = \boldsymbol{\theta} + \eta \cdot \Delta\boldsymbol{\theta}$$

Now the question is what is the right $\boldsymbol{\theta}$ to use. The answer comes from Taylor Series.

2.2 Multivariate Taylor Series

For a function $\mathcal{F}(\mathbf{x})$ expanded around $\mathbf{a} \in \mathbb{R}^n$, we have,

$$\mathcal{F}(\mathbf{x}) = \mathcal{F}(\mathbf{a}) + (\mathbf{x}-\mathbf{a})'\nabla\mathcal{F}(\mathbf{a}) + \frac{1}{2!}(\mathbf{x}-\mathbf{a})'\nabla^2\mathcal{F}(\mathbf{a})(\mathbf{x}-\mathbf{a}) + \cdots$$

2.3 Derivation

For ease of notation, take $\Delta\boldsymbol{\theta} = \boldsymbol{u}$.

With $\mathcal{F} = \mathscr{L}$(our Loss Function), $\mathbf{x} = \boldsymbol{\theta} + \eta\boldsymbol{u}$, $\mathbf{a} = \boldsymbol{\theta}$; from the Taylor Series we have,

$$\begin{aligned}
\mathscr{L}(\boldsymbol{\theta}+\eta\boldsymbol{u}) &= \mathscr{L}(\boldsymbol{\theta}) + (\eta\boldsymbol{u})'\nabla\mathscr{L}(\boldsymbol{\theta}) + \frac{1}{2!}(\eta\boldsymbol{u})'\nabla^2\mathscr{L}(\boldsymbol{\theta})(\eta\boldsymbol{u}) + \cdots \\
&= \mathscr{L}(\boldsymbol{\theta}) + \eta\,\boldsymbol{u}'\nabla\mathscr{L}(\boldsymbol{\theta}) + \frac{\eta^2}{2!}\,\boldsymbol{u}'\nabla^2\mathscr{L}(\boldsymbol{\theta})\boldsymbol{u} + \cdots \\
&= \mathscr{L}(\boldsymbol{\theta}) + \eta\,\boldsymbol{u}'\nabla\mathscr{L}(\boldsymbol{\theta}) \left[\eta \text{ typically being small } \eta^2, \eta^3, \ldots \to 0\right]
\end{aligned}$$

Note that the move $(\eta\boldsymbol{u})$ would be favourable only if,

$$\mathscr{L}(\boldsymbol{\theta}+\eta\boldsymbol{u}) - \mathscr{L}(\boldsymbol{\theta}) < 0$$

i.e. if the new loss is less than the previous loss.

This implies $\boldsymbol{u}'\nabla\mathscr{L}(\boldsymbol{\theta}) < 0$, as η is a positive quantity and more the negative $\boldsymbol{u}'\nabla\mathscr{L}(\boldsymbol{\theta})$ is, the more is favourable the move $\eta\boldsymbol{u}$. Now let us find out the range of $\boldsymbol{u}'\nabla\mathscr{L}(\boldsymbol{\theta})$.

- Let β be the angle between $\boldsymbol{u}$ and $\nabla_{\boldsymbol{\theta}}\mathscr{L}(\boldsymbol{\theta})$, then we know that,

$$-1 \leq \cos(\beta) = \frac{\boldsymbol{u}^T\nabla_{\boldsymbol{\theta}}\mathscr{L}(\boldsymbol{\theta})}{\|\boldsymbol{u}\| \cdot \|\nabla_{\boldsymbol{\theta}}\mathscr{L}(\boldsymbol{\theta})\|} \leq 1$$

Multiplying throughout by $k = \|\boldsymbol{u}\| \cdot \|\nabla_{\boldsymbol{\theta}}\mathscr{L}(\boldsymbol{\theta})\|$:

$$-k \leq k \cdot \cos(\beta) = \boldsymbol{u}^T\nabla_{\boldsymbol{\theta}}\mathscr{L}(\boldsymbol{\theta}) \leq k$$

Thus,

$$\mathscr{L}(\boldsymbol{\theta}+\eta\boldsymbol{u}) - \mathscr{L}(\boldsymbol{\theta}) = \boldsymbol{u}^T\nabla_{\boldsymbol{\theta}}\mathscr{L}(\boldsymbol{\theta}) = k \cdot \cos(\beta)$$

will be most negative when $\cos(\beta) = -1$, *i.e.*, when β is 180°.

So our best move is at 180° w.r.t. the gradient $\nabla\mathscr{L}(\boldsymbol{\theta})$. In other words, we should move in a direction opposite to the gradient *i.e.*

$$\eta\Delta\boldsymbol{\theta} = -\eta\nabla\mathscr{L}(\boldsymbol{\theta})$$

The parameter update equations become

$$\begin{aligned}
w_{t+1} &= w_t - \eta\nabla w_t \\
b_{t+1} &= b_t - \eta\nabla b_t
\end{aligned} \tag{1}$$

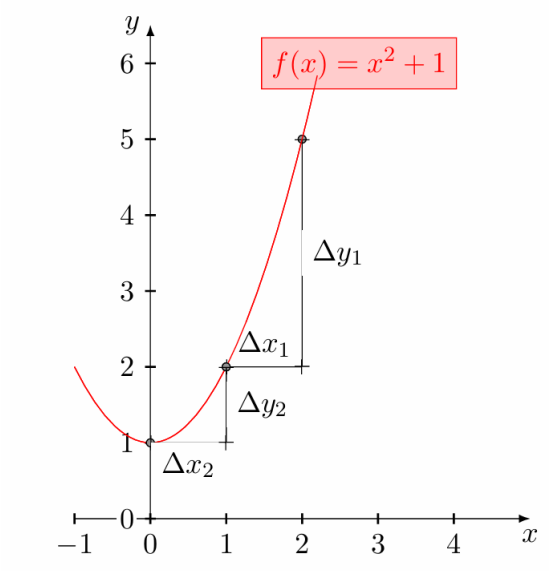
$$\text{where, } \nabla w_t = \left.\frac{\partial\mathscr{L}(w,b)}{\partial w}\right|_{w=w_t,\, b=b_t}, \quad \nabla b = \left.\frac{\partial\mathscr{L}(w,b)}{\partial b}\right|_{w=w_t,\, b=b_t}$$

3 Batch / Vanilla Gradient Descent

This is the simplest form of gradient descent with parameter update rules as in 1.

Algorithm 1: `gradient_descent()`

$t \leftarrow 0$;
$max\_iterations \leftarrow 1000$;
while $t < max\_iterations$ **do**
    $w_{t+1} \leftarrow w_t - \eta \nabla w_t$;
    $b_{t+1} \leftarrow b_t - \eta \nabla b_t$;
    $t \leftarrow t + 1$;
end



- When the curve is steep, the gradient $\left(\frac{\Delta y_1}{\Delta x_1}\right)$ is large.
- When the curve is gentle, the gradient $\left(\frac{\Delta y_2}{\Delta x_2}\right)$ is small.
- Recall that our weight updates are proportional to the gradient: $w_{new} = w - \eta \nabla w$
- Hence, in the areas where the curve is gentle the updates are small, whereas in the areas where the curve is steep the updates are large. That's why in gradient descent whenever the error surface is gentle we move slower and whenever it is steep we move faster.

The fact that it takes a lot of time to navigate regions having a gentle slope, gives 1 demerit point to Vanilla Gradient Descent. In the upcoming section we shall see that we can do better.

1. It's called **batch** gradient descent because it uses the entire batch (full dataset) to compute each gradient.
2. It's called **vanilla** gradient descent because it's the simplest, unmodified form of gradient descent.

4 Momentum Gradient Descent