Setting Learning Rate

Selwyn Gomes

June 10, 2021

Approach in 1-Dimension

Every function has different ranges of gradients which can lead to complications in finding a suitable Learning rates for Gradient Descent. Ideally we would prefer not to perform trial and error especially in our setup where there is a heavy cost associated with sampling data. To solve this issue I propose normalizing the gradient values across each dimension to fixed pre-defined ranges given by the formula:

$$\mathbf{G}(\mathbf{x_d})_{std} = \frac{(\mathbf{G}(\mathbf{x_d}) - \mathbf{G}(\mathbf{x_d}).min)}{(\mathbf{G}(\mathbf{x_d}).max) - \mathbf{G}(\mathbf{x_d}).min)}$$

$$\mathbf{G}(\mathbf{x_d})_{scaled} = \mathbf{G}(\mathbf{x_d})_{std} * (max - min) + min$$

Where $\mathbf{G}(\mathbf{x_d})$ is the partial derivative at $\mathbf{x}$ for the $\mathbf{d}$ dimension. To simplify the process and prevent sign flipping due to normalization we find $\mathbf{G}(\mathbf{x_d})_{scaled}$ separately for positive and negative derivatives.

For $\mathbf{G}(\mathbf{x_d}) > 0$, $\mathbf{G}(\mathbf{x}).min = 0$ and $\mathbf{G}(\mathbf{x}).max = Max(\mathbf{G}(\mathbf{x_d}))$

For $\mathbf{G}(\mathbf{x_d}) < 0$, $\mathbf{G}(\mathbf{x}).min = 0$ and $\mathbf{G}(\mathbf{x}).max = |Min(\mathbf{G}(\mathbf{x_d}))|$

The max and min values can be appropriately chosen such that $LR * max$ is the largest stepsize we want in any one dimension and $LR * min$ is the smallest step size. The $LR * max$ value should be chosen depending on the domain size, a larger search space would logically need larger gradient jumps for faster convergence and vice versa. For example if LR is 0.1 and max is set as 2 then the largest jump in any one dimension would be limited to 0.2. The min value is chosen to be 0 as we would like the algorithm to converge at a optimum.

The only flaw with the algorithm is that $Max(\mathbf{G}(\mathbf{x_d}))$ and $Min(\mathbf{G}(\mathbf{x_d}))$ are not known. Hence instead of using $Max(\mathbf{G}(\mathbf{x_d}))$ we use the largest seen gradient value seen upto the current step and similarly for

Algorithm 1 Algorithm for Positive Partial Derivative

```
1: Initially $G(x).max = 0$
2: $GD$:
3: if $G(x) > G(x).max$ then
4:     $G(x).max \leftarrow G(x)$
5: Calculate $G(x)_{scaled}$
6: goto $GD$
```

Algorithm 2 Algorithm for Negative Partial Derivative

```
1: Initially $G(x).max = 0$
2: $GD$:
3: if $G(x) > -G(x).max$ then
4:     $G(x).max \leftarrow |G(x)|$
5: Calculate $G(x)_{scaled}$
6: goto $GD$
```

$Min(\mathbf{G(x_d)})$ we use the smallest seen gradient value. Initially the algorithm starts with the max step size and slowly gets better at adjusting the jump length as we have better estimates on the max/min.