Multi Layer Perceptron & Gradient Descent

AI Applied to Medical Images

M. Alexander Perez

2021

Galileo University – BiomedLab

Outline

MLP & Backpropagation

Activation Functions

One main disadvantage of the use of Heaviside functions is that we can't really calculate its derivative. One of the strategies in optimization for a simple model was to find critical points i.e. where $L'(w) = 0$. In the case of the MLP using the Heaviside function we simply can't use derivatives since the involved function doesn't have one. Sometimes we use what is called a subgradient. An improvement we have found to be useful is the use of activation functions different to Heaviside.

Some Activation Functions.

- Linear: $h(x) = x$. Not very useful with the methods we try to implement.
- Sigmoid: $h(x) = \frac{1}{1+e^{-x}}$.
- Hyperbolic Tangent: $h(x) = \tanh x = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$.
- ReLU: $h(x) = \max\{0, x\}$.
- Leaky ReLU: $h(x) = \max\{\alpha\, x, x\}$.

Feedforward

This is the name we use to the process of computing the output of a network. For example:

$$f(x) = 3 * \exp(x^2 + 1).$$

When we compute $f(3)$ we follow an evaluation graph, we apply a series of operations as we do that we move forward on the graph.

BackPropagation

This is the process of computing the derivative of a function. We follow the graph from output to input to get the derivative, all this is a consequence of the chain rule of derivatives. For example we can see:

$$y = f(g(x)) \Rightarrow \frac{dy}{dx} = g'(x)\, f'(g(x)).$$

So a good strategy is to save the value at each node of the graph so we use them when we compute the gradient.

Steepest Descent

Why use it?

Unfortunately for us the models we are going to work are way to complex to compute, analytically or an exact answer, the critical points. So our option next option is to use a numerical method. The one method that has work very well, or maybe the best we have, is steepest descent. We use this in combination with a Loss Function that measures how bod is our model with a set of weight W.

Why gradient?

To understand the steepest descent we must recall a property of the gradient of a function. The gradient point towards the direction where the function grows the most, and the negative gradient point in the direction where the function decreases the most.

Example:

Lets look at a simple example with $f(x, y) = 3x^3 + 10y^2$ at the point $(3, 3)$.

- The gradient at $(3, 3)$ is $(81, 60)$.
- So if we want to decrease the value of f we move in the direction of $(-81, -60)$ against the gradient.

If we study the gradient at $(1, 4)$:

- The gradient of f at $(1, 4)$ is $(9, 30)$.
- We can see now that from this point the direction to move to make f decrease is $(-9, -30)$.

Limits and Learning Rate

One problem with that is that gradients and derivatives are related to limits, specifically to limits with very small numbers, so the gradient property is only valid if we take a very small steep in the direction of $-\nabla f$.

Basically to find a minimum for the Loss Function $L(W_0)$, we change our weights in the direction of $-\nabla L(W_0)$ so the loss function, that measures how bad are we doing, this will make (hopefully) the function L decrease. So we are doing less bad now, i.e. better. Mathematically it is:

$$W_1 = W_0 - \alpha \nabla L(W_0).$$

The α parameter manages how big is the steep that we take.

Intro

We are going to see a couple of situations that we have to work with in the practice, one of them is the existence of saddle points, and other is that some loss functions are not convex. We also are going to see a couple of variations of the Steepest Descent method.

- $f(x, y) = 2x^3 - 6xy + 3y^2$.
- $f(x, y) = xy \exp\left(-\frac{x^2 + y^2}{2}\right)$.

Using the Autograd

- Torch
- Tensorflow

Implementation of MLP

MLP

To apply all that we see about gradients, backpropagation and steepest descent to our problem with perceptrons we are going to change the Heaviside function for one that we can derivate.

Quickly we notice that the perceptron has a lot of limitations we can see that through the xor function.

Deep in Deep Learning

Four component that make possible the advances in AI that we see today are:

- The existence of more data.
- We find that we can stack layers to form a new, deeper, neural network. And the deeper the better.
- The use of convolutions.
- Production of hardware that allow us to tackle massive problems, specially the advances on GPUs.

Of course there are a lot more factor, like the interest of companies and researchers. And this things also bring new challenges to mathematician, engineers, and mathematicians.

Implementation

We are going to build a simple two layers neural network, this kind of neural networks are called multi-layer perceptrons.

- Torch
- Keras