Optional: Calculus-Based Gradient Descent

Machine Learning Mini Course

In this handout, we go into the specifics on how to do gradient descent using calculus. We’ll be looking at the case in which we have exactly one input variable and one output variable. In other words, the line will have function y = wx + b. Let’s start with some notation.

**Notation:**

Let X be our dataset of multiple x1, x2, … xm for some integer m. Let y be our corresponding answer dataset containing multiple y1, y2, … ym. In this case, (x1, y1) would be a single example containing one input (x1) and one output (y1). The variables w and b will denote our slope and y-intercept respectively. (In machine learning jargon, this is called the “weights” and the “bias.”) Finally, the predicted y for xi is denoted yi’

Then our loss function can be defined:

loss(w, b) = Σ (yi - yi’)2 = Σ (yi - wxi - b)2

**Algorithm of Gradient Descent:**

Initialize w, b

Initialize α = 0.01 (learning rate)

Repeat until convergence:

wnew = w - α

bnew = b - α

w = wnew

b = bnew

Note that -α is how much we update. The learning rate (α) determines exactly how much to increment, and gives us the slope. Finally, also note that this expression has a negative sign in front of it. If the slope is positive, we should move backwards. If the slope is negative, we should move forwards.

**Derivative of Loss Function:**

loss(w,b) = (Σ (yi - wxi - b)2)

= Σ 2(yi - wxi - b)(yi - wxi - b)

= Σ (yi - wxi - b)xi

Likewise,

loss(w,b) = (Σ (yi - wxi - b)2)

= Σ 2(yi - wxi - b)(yi - wxi - b)

= Σ (yi - wxi - b)

Remember, if we take the partial wrt w, we assume xi, yi, and b are all constants. Similarly with b.

And thus, we have an algorithm which can do gradient descent computationally.

**Open challenge problem**:

If you have time after implementation time, you could also try implementing gradient descent yourself.