PadhAI: Variants of Gradient Descent

One Fourth Labs

Highlighting a limitation of Gradient Descent

Let's look at better learning algorithms

1. The gradient descent update rule is as follows
   a. $\omega = \omega - \eta \frac{\partial L(\omega)}{\partial \omega}$ a
   b. The questions we should be asking are: How do we compute the gradients? What data should we use for computing the gradients?
   c. To follow up on those questions: How do we use the gradients? Can we come up with a better update rule?
   d. Here is the Python implementation of Gradient Descent similar to what we have seen earlier

```
X = [0.5, 2.5]
Y = [0.2, 0.9]

def f(w, b, x):
    # sigmoid with parameters w, b
    return 1.0/ (1.0 + np.exp(-(w*x + b)))

def error(w, b):
    err = 0.0
    for x, y in zip(X, Y):
        fx = f(w, b, x)
        err += 0.5 * (fx - y) ** 2
    return err

def grad_b(w, b, x, y):
    fx = f(w, b, x)
    return (fx-y) * fx * (1 - fx)

def grad_w(w, b, x, y):
    fx = f(w, b, x)
    return (fx-y) * fx * (1 - fx) * x

def do_gradient_descent():
    w, b, eta = -2, -2, 1.0
    max_epochs = 1000
    for i in range(max_epochs):
        dw, db = 0, 0
        for x, y in zip(X, Y):
            dw += grad_w(w, b, x, y)
            db += grad_b(w, b, x, y)
        w = w - eta * dw
        b = b - eta * db
```

2. Look at the following plot of w, b and error using Gradient descent