Gradient Descent: Takeaways

Syntax

- Implementing gradient descent for 10 iterations:

```
a1_list = [1000]
alpha = 10
for x in range(0, 10):
    a1 = a1_list[x]
    deriv = derivative(a1, alpha, xi_list, yi_list)
    a1_new = a1 - alpha*deriv
    a1_list.append(a1_new)
```

Concepts

- The process of finding the optimal unique parameter values to form a unique linear regression model is known as model fitting. The goal of model fitting is to minimize the mean squared error between the predicted labels made using a given model and the true labels. The mean squared error is as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

- Gradient descent is an iterative technique for minimizing the squared error. Gradient descent works by trying different parameter values until the model with the lowest mean squared error is found. Gradient descent is a commonly used optimization technique for other models as well. An overview of the gradient descent algorithm is as follows:
    - Select initial values for the parameter a_1.
    - Repeat until convergence (usually implemented with a max number of iterations):
        - Calculate the error (MSE) of the model that uses current parameter value: $MSE(a_1) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}^{(i)} - y^{(i)})^2$
        - Calculate the derivative of the error (MSE) at the current parameter value: $\frac{d}{da_1}MSE(a_1)$
        - Update the parameter value by subtracting the derivative times a constant (α, called the learning rate): $a_1 := a_1 - \alpha\frac{d}{da_1}MSE(a_1)$
- Univariate case of gradient descent:
    - The function that we optimize through minimization is known as a cost function or as the loss function. In our case, the loss function is: $MSE(a_1) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}^{(i)} - y^{(i)})^2$
    - Applying calculus properties to simplify the derivative of the loss function:
        - Applying the linearity of differentiation property, we can bring the constant term outside the summation:
            - $\frac{d}{da_1}MSE(a_1) = \frac{1}{n}\sum_{i=1}^{n}\frac{d}{da_1}(a_1x_1^{(i)} - y^{(i)})^2$
        - Using the power rule and the chain rule to simplify:
            - $\frac{d}{da_1}MSE(a_1) = \frac{1}{n}\sum_{i=1}^{n}2(a_1x_1^{(i)} - y^{(i)})\frac{d}{da_1}(a_1x_1^{(i)} - y^{(i)})$
        - Because we're differentiating with respect to a_1, we treat $y^{(i)}$ and $x_1^{(i)}$ as constants.
            - $\frac{d}{da_1}MSE(a_1) = \frac{2}{n}\sum_{i=1}^{n}x_1^{(i)}(a_1x_1^{(i)} - y^{(i)})$

- For every iteration of gradient descent:
    - The derivative is computed using the current a_1 value.
    - The derivative is multiplied by the learning rate (α): $\alpha \frac{d}{da_1} MSE(a_1)$ The result is subtracted from the current parameter value and assigned as the new parameter value: $a_1 := a_1 - \alpha \frac{d}{da_1} MSE(a_1)$
- Multivariate case of gradient descent:
    - When we have two parameter values (a_0 and a_1), the cost function is now a function of two variables instead of one. Our new cost function is: $MSE(a_0, a_1) = \frac{1}{n} \sum_{i=1}^{n} (a_0 + a_1 x_1^{(i)} - y^{(i)})^2$
    - We also need two update rules:
        - $a_0 := a_0 - \alpha \frac{d}{da_0} MSE(a_0, a_1)$
        - $a_1 := a_1 - \alpha \frac{d}{da_1} MSE(a_0, a_1)$
    - Computed derivative for the multivariate case:
        - $\frac{d}{da_1} MSE(a_0, a_1) = \frac{2}{n} \sum_{i=1}^{n} x_1^{(i)} (a_0 + a_1 x_1^{(i)} - y^{(i)})$
- Gradient descent scales to as many variables as you want. Keep in mind each parameter value will need its own update rule, and it closely matches the update for a_1. The derivative for other parameters are also identical.
- Choosing good initial parameter values, and choosing a good learning rate are the main challenges with gradient descent.

Resources

- Mathematical Optimization
- Loss Function