

Gradient Descent: Takeaways

by Dataquest Labs, Inc. - All rights reserved © 2019

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:

—

- 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 θ .
- Repeat until convergence (usually implemented with a max number of iterations):
 - Calculate the error (MSE) of the model that uses current parameter value:
—
 - Calculate the derivative of the error (MSE) at the current parameter value:
—
 - Update the parameter value by subtracting the derivative times a constant (α , called the learning rate):
—

- 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:

$$J(\theta_0, \theta_1) = \frac{1}{2} \sum_{i=0}^{n-1} (y_i - \theta_0 - \theta_1 x_i)^2$$
 - 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:

$$\frac{d}{d\theta_0} \left(\frac{1}{2} \sum_{i=0}^{n-1} (y_i - \theta_0 - \theta_1 x_i)^2 \right) = \frac{1}{2} \sum_{i=0}^{n-1} \frac{d}{d\theta_0} (y_i - \theta_0 - \theta_1 x_i)^2$$
 - Using the power rule and the chain rule to simplify:

$$\frac{d}{d\theta_0} (y_i - \theta_0 - \theta_1 x_i)^2 = 2(y_i - \theta_0 - \theta_1 x_i) \cdot \frac{d}{d\theta_0} (y_i - \theta_0 - \theta_1 x_i)$$
 - Because we're differentiating with respect to θ_0 , we treat θ_1 and x_i as constants.

$$\frac{d}{d\theta_0} (y_i - \theta_0 - \theta_1 x_i) = -1$$
- For every iteration of gradient descent:
 - The derivative is computed using the current θ_0 value.
 - The derivative is multiplied by the learning rate (α): $-\alpha \sum_{i=0}^{n-1} (y_i - \theta_0 - \theta_1 x_i)$. The result is subtracted from the current parameter value and assigned as the new parameter value: $\theta_0 = \theta_0 - \alpha \sum_{i=0}^{n-1} (y_i - \theta_0 - \theta_1 x_i)$
- Multivariate case of gradient descent:
 - When we have two parameter values (θ_0 and θ_1), the cost function is now a function of two variables instead of one. Our new cost function is:

$$J(\theta_0, \theta_1) = \frac{1}{2} \sum_{i=0}^{n-1} (y_i - \theta_0 - \theta_1 x_i)^2$$
 - We also need two update rules:
 - $\theta_0 = \theta_0 - \alpha \sum_{i=0}^{n-1} (y_i - \theta_0 - \theta_1 x_i)$
 - $\theta_1 = \theta_1 - \alpha \sum_{i=0}^{n-1} x_i (y_i - \theta_0 - \theta_1 x_i)$
 - Computed derivative for the multivariate case:
 - $\frac{d}{d\theta_0} J(\theta_0, \theta_1) = -\sum_{i=0}^{n-1} (y_i - \theta_0 - \theta_1 x_i)$

- 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 θ_0 . 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](#)



Takeaways by Dataquest Labs, Inc. - All rights reserved © 2019