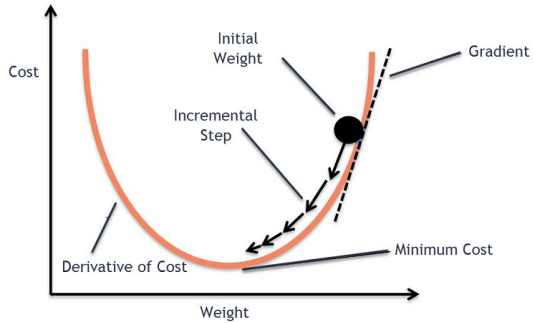
Gradient of a Single-Point Regression

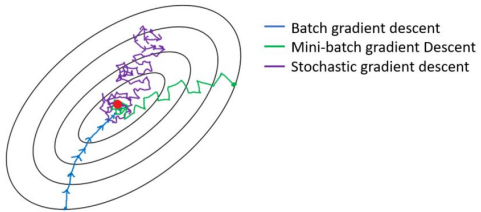
Putra Bahy Helmi Hartoyo
bahy@sju.ac.kr

Gradient descent



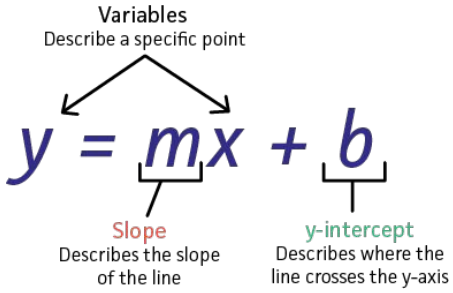
- **Gradient descent** is an iterative method of updating model parameters (weight) to find a **minimum point** of a cost function.
- It is done by iteratively **calculating the gradient** & taking the steps in the descending direction.
- Calculating gradient using just **a single point** at a time is commonly called **Stochastic Gradient Descent (SGD)**.
- There are **three types** of gradient descent, batch gradient descent, mini-batch gradient descent, and stochastic gradient descent.

Types of gradient descent



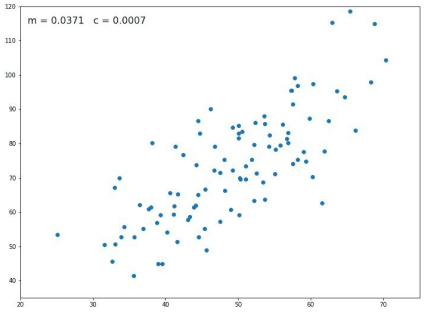
- **Batch gradient descent.**
  - Updates the parameters after all training examples are being pass.
  - Computationally efficient, produces stable error gradient and convergence.
- **Stochastic gradient descent.**
  - Updates the parameters every one forward pass of training example.
  - Detailed improvement rate, converges faster than batch gradient descent in certain situation.
  - Computationally expensive, can result in noisy gradients.
- **Mini-batch gradient descent.**
  - Splits the training examples into several batches and perform updates for each of those batches.
  - Create a balance between the robustness of batch gradient descent and the efficiency of stochastic gradient descent.

Regression



- **Regression** is a statistical processes for estimating the relationships between a **dependent variable (y)** and one or more **independent variables (x)**.
- The most common regression form is **linear regression**.
- Linear regression can gives us the **most optimal value** for the parameters (intercept and the slope), one of the way by using the **gradient descent algorithm**.

Steps (SGD)



1. Shuffle the training data, select **single sample**.
2. Do a **forward pass** with that single sample **(x)** to the model and obtain the **predicted value ($\hat{y}$)**.
3. **Compare** the **actual value (y)** and the **predicted value ($\hat{y}$)** to calculate the error.
4. Compute the **derivative of the cost function** w.r.t. parameters.
5. **Update** the weight parameters.
6. Repeat step 1-5 for all training data.
7. Repeat step 1-6 until the cost function reaches near 0 (minimized).

Code

Let’s look at the notebook!

https://github.com/bahyhelmihp/sgd-from-scratch/blob/main/midterm.ipynb