AI Course

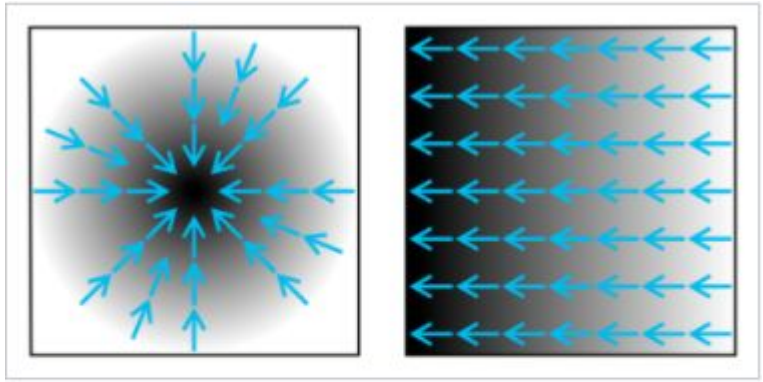
Dr. Mürsel Taşgın

How Gradient Descent works?

Gradient Descent

What is a gradient?

Gradient of a function at p shows the direction in which the function *increases most quickly from* p



The gradient, represented by the blue arrows, denotes the direction of greatest change of a scalar function. The values of the function are represented in greyscale and increase in value from white (low) to dark (high).

Source: wikipedia

$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_n}(p) \end{bmatrix}$$

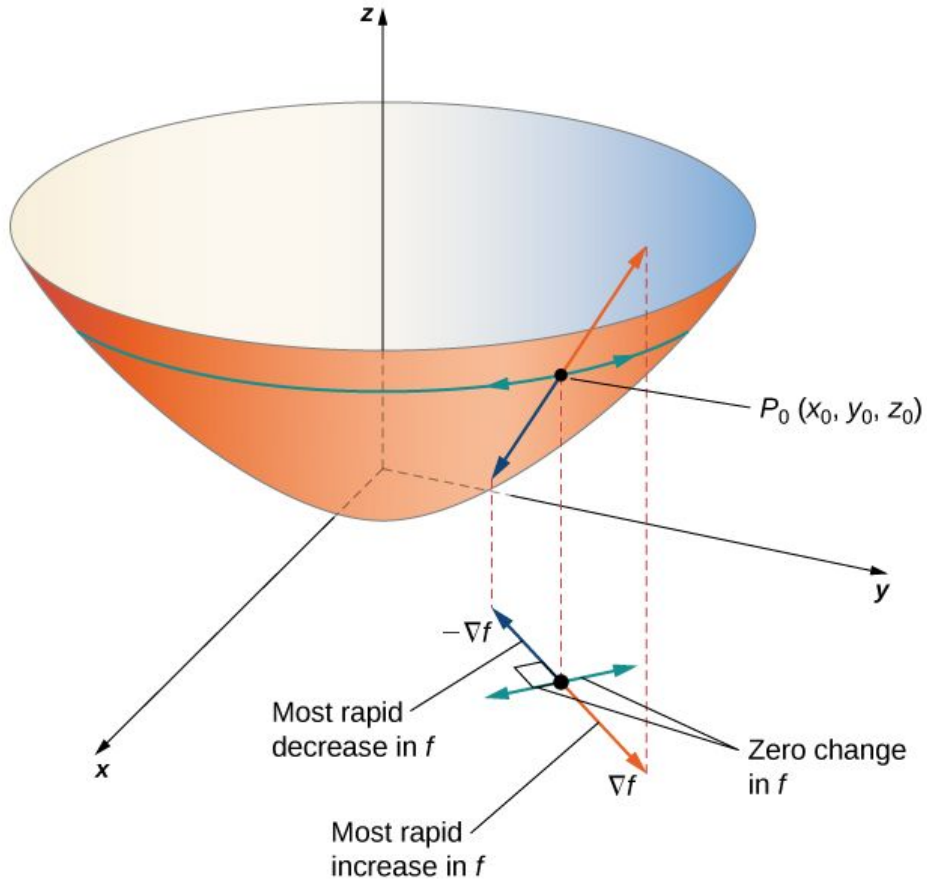
→ Gradient vector
(partial derivatives)

Gradient Descent

What is a gradient?

We can think of a gradient as the *slope of a function* at a point.

- First derivative of a function shows the slope of the function (tangent)
- The higher the gradient, *steeper* the slope
- Zero (0) slope shows no increase (decrease) at that point *If slope is flat, original function hits a minimum/maximum*

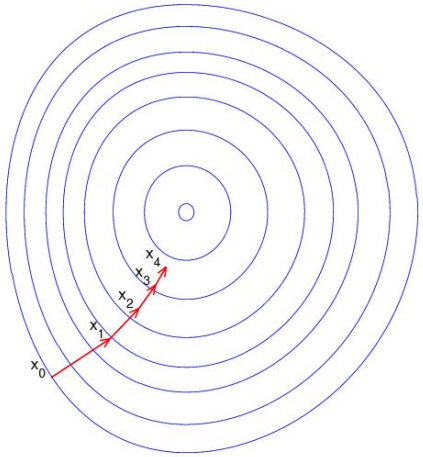


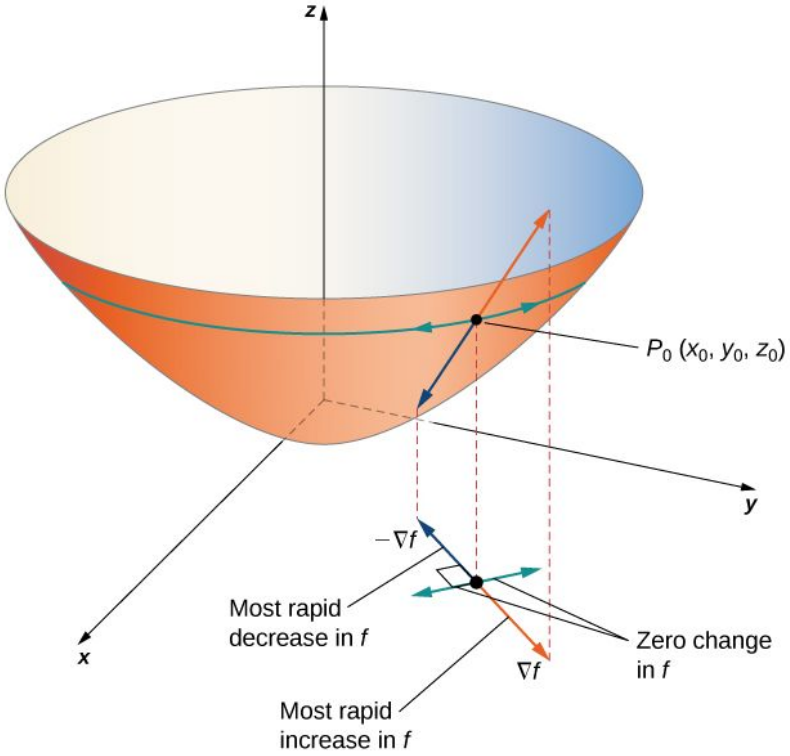
Gradient Descent

What is a gradient?

We can think of a gradient as the *slope of a function* at a point.

- First derivative of a function shows the slope of the function (tangent)
- The higher the gradient, *steeper* the slope
- Zero (0) slope shows no increase (decrease) at that point *If slope is flat, original function hits a minimum/maximum*





Climbing a hill: Red arrows shows the steps of our climber. Think of a gradient in this context as a vector that contains the *direction of the steepest step* the blindfolded man can take and *also how long that step should be*.

Gradient Descent

How Gradient Descent works?

Instead of climbing up a hill, gradient descent is similar to *hiking down* to the bottom.

What should be the next position of our hiker? (going down)

We can use gradient descent to select the best move to decrease the height

$$p' = p - \gamma \nabla f(p)$$

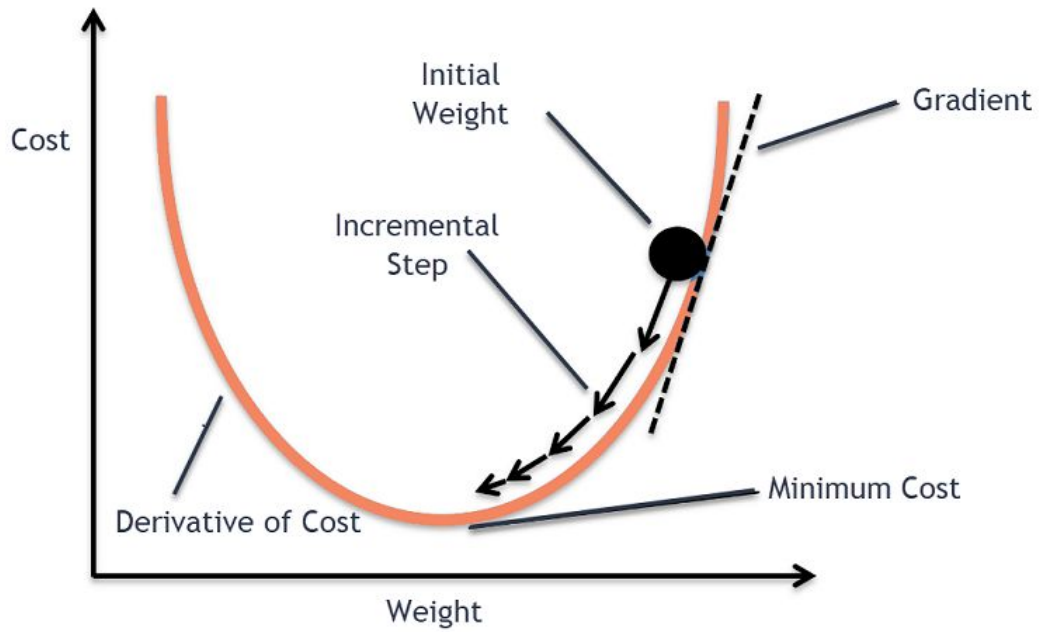
- γ: Learning rate
- $\nabla f(p)$: Gradient vector
- p: Current location
- p': New location

Gradient Descent

How Gradient Descent works?

Gradient descent is an *iterative optimization* algorithm for finding the local minimum of a function.

To find the local minimum of a function using gradient descent, we must take steps proportional to the negative of the gradient (*move away from the gradient*) of the function at the current point.



1. Select a point
2. Calculate gradient of f at that point
3. Using the following, find the new point p'

$$p' = p - \gamma \nabla f(p)$$

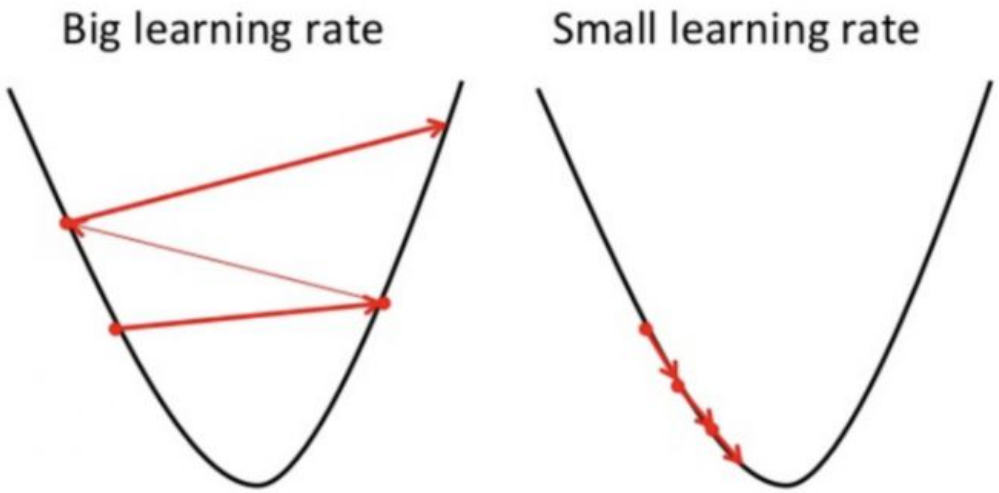
4- If there is no update (i.e., p==p') Stop
Else go to 2 and iterate

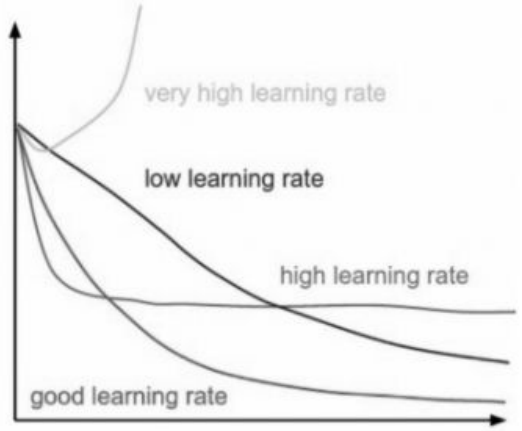
Gradient Descent

Learning rate

Learning rate affects the *size of the next step* in the negative direction of gradient.

Initial steps may be large, but iteratively steps gets smaller to target.



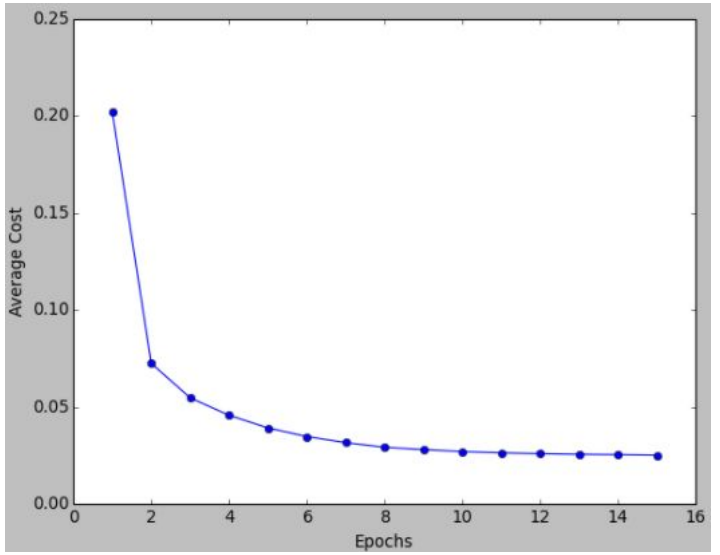


A good way to make sure gradient descent runs properly is by plotting the cost function as the optimization runs. Put the number of iterations on the x-axis and the value of the cost-function on the y-axis. This helps you see the value of your cost function after each iteration of gradient descent, and provides a way to easily spot how appropriate your learning rate is.

Gradient Descent – *Batch Gradient Descent*

- Batch gradient descent, also called vanilla gradient descent, calculates the error for each example within the training dataset, but only after all training examples have been evaluated does the model get updated. This whole process is like a cycle and it's called a training epoch.

- We take the average of the gradients of all the training examples and then use that *mean gradient* to update our parameters. So that's just one step of gradient descent in one epoch.



Batch Gradient Descent is great for convex or relatively smooth error manifolds. In this case, we move somewhat directly towards an optimum solution. The graph of cost vs epochs is also quite smooth because we are averaging over all the gradients of training data for a single step.

Some advantages

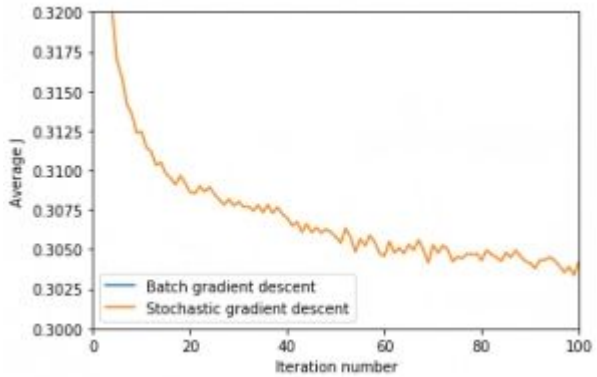
- Computational efficiency *(not updating for each data point)*
- Stable error gradient
- Stable convergence

Some disadvantages

- Stable error gradient can sometimes miss best solution
- Requires the entire training dataset be in memory and available to the algorithm *impractical for large datasets in limited memory!*
- Convergence is slow

Gradient Descent – *Stochastic Gradient Descent*

- Stochastic gradient descent (SGD) is an iterative method for optimizing an objective function with suitable smoothness properties.
- It can be regarded as a stochastic approximation of gradient descent optimization.
- It replaces the actual gradient *(calculated from the entire data set)* by an estimate thereof *(calculated from a randomly selected subset of the data).*



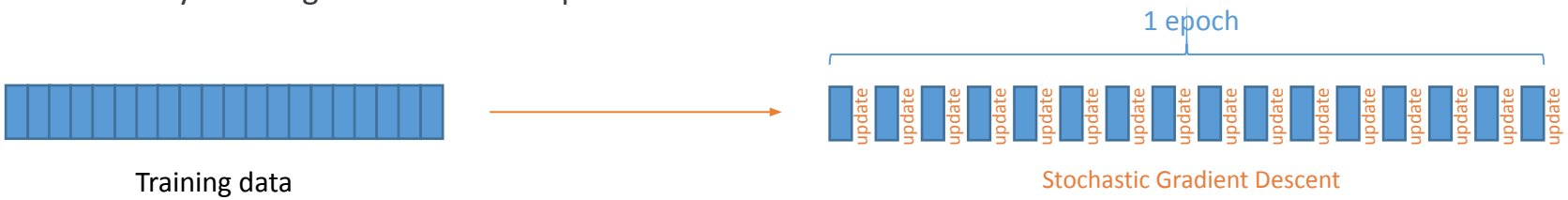
Stochastic Gradient Descent can be noisy; i.e., it is because it responds to the effects of each and every sample. Samples can contain noisiness and can increase errors during gradient descent.

Some advantages

- Sampling efficiency
- Frequent gradient updates → *improved learning*
- Avoids local min/max

Some disadvantages

- Computationally more expensive (more updates)
- May fluctuate → *noisy gradient signal, errors may jump*
- May have high variance over epochs



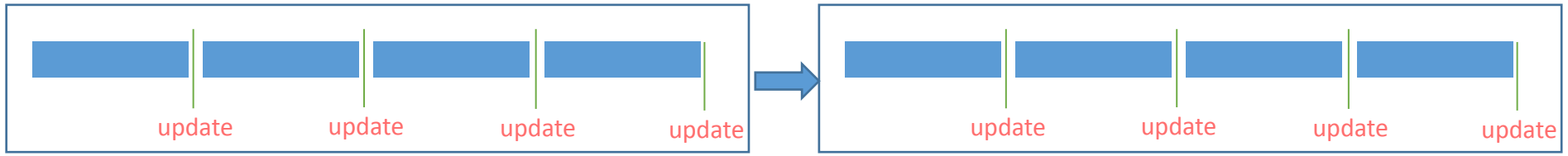
Gradient Descent – *BGD vs SGD*

Batch Gradient Descent	Stochastic Gradient Descent
Computes gradient using the whole Training sample	Computes gradient using a single Training sample
Slow and computationally expensive algorithm	Faster and less computationally expensive than Batch GD
Not suggested for huge training samples.	Can be used for large training samples.
Deterministic in nature.	Stochastic in nature.
Gives optimal solution given sufficient time to converge.	Gives good solution but not optimal.
No random shuffling of points are required.	The data sample should be in a random order, and this is why we want to shuffle the training set for every epoch.
Can't escape shallow local minima easily.	SGD can escape shallow local minima more easily.
Convergence is slow.	Reaches the convergence much faster.

Source:
https://www.geeksforgeeks.org/difference-between-batch-gradient-descent-and-stochastic-gradient-descent/

Gradient Descent – *Mini-batch Gradient Descent*

- Mini-batch gradient descent is a combination of SGD and batch gradient descent
- It splits training dataset into small batches and performs and update for each of those batches



1 epoch *(1 pass over the entire data)*

1 epoch

- It creates a balance between robustness of SGD and efficiency of BGD