OPTIMIZATION IN MACHINE LEARNING

Dr. Dinesh K Vishwakarma

Professor, Department of Information Technology

Delhi Technological University, Delhi-110042, India

Email: dinesh@dtu.ac.in

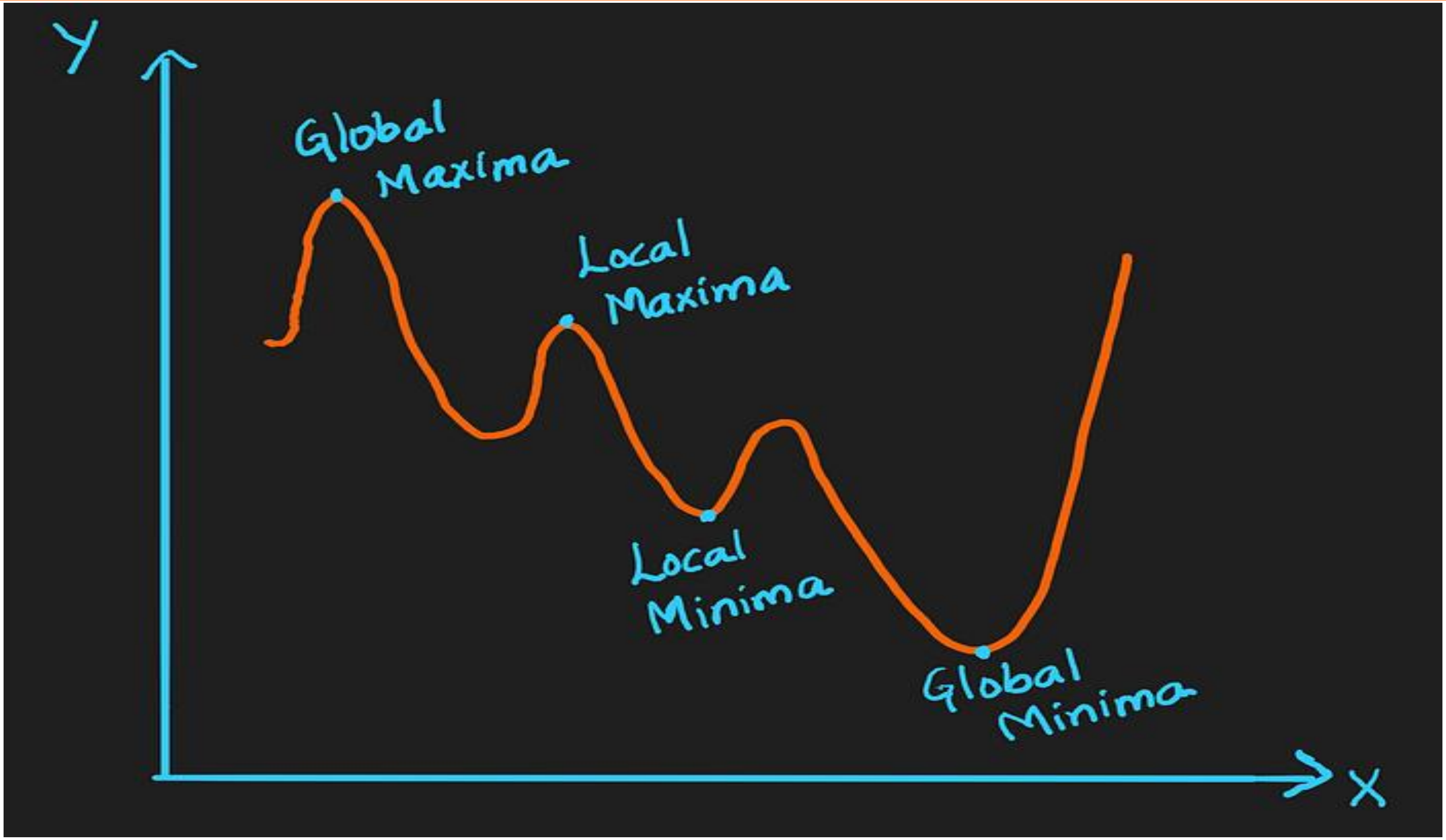
Mobile:9971339840

Webpage: http://www.dtu.ac.in/web/departments/informationtechnology/faculty/dkvishwakarma.php

Introduction

- **Optimization** is the process where we train the model iteratively that results in a **maximum** and **minimum** function evaluation. One of the Important phenomena in ML is to get **better results**.
- An **optimization problem** consists of maximizing or minimizing a real function by systematically choosing Input values.
- Optimization is the **most essential ingredient** in the recipe of machine learning algorithms. It starts with defining some kind of **loss function/cost function**.
- The choice of optimization algorithm can make a difference between getting a good accuracy in hours or days.

Maxima & Minima



Types of Optimization

- Most popular types of optimizations are:
  - Maximum likelihood
  - Expectation maximization
  - Gradient descent

Maximum Likelihood

- Many methods are used for estimating unknown parameters from data.
- The maximum likelihood estimate (MLE), works on the principle of parameter value that has biggest probability?
- The MLE is an example of a point estimate because it gives a single value for the unknown parameter.
- It is often easy to compute and that it agrees with our intuition in simple examples.

Maximum Likelihood…

– **Problem Statement**

- Consider a Random Samples (RS) $X_1, X_2 \dots\dots X_n$ and their probability distribution depends on some unknown parameter θ.
- Goal of ML is to find a point estimator $\mathrm{u}(X_1, X_2 \dots\dots X_n)$, such that $\mathrm{u}(x_1, x_2 \dots\dots x_n)$ is a "good" point estimate of θ, where $x_1, x_2 \dots x_n$ are the observed values of the random sample.
- For e.g., consider a RS $X_1, X_2 \dots\dots X_n$ for which the X_i are assumed to be normally distributed with mean μ and variance σ^2, then our goal will be to find a good estimate of μ, say, using the data $x_1, x_2 \dots\dots x_n$ that we obtained from our specific random sample.

Maximum Likelihood…

– **Basic Idea**

- A good estimate of the unknown parameter θ would be the value of that maximizes the probability
- Suppose we have RS $X_1, X_2 \ldots\ldots X_n$, for probability distribution or mass function of each X_i is $f(x_i, \theta)$.
- The joint probability distribution or mass function of $X_1, X_2 \ldots\ldots X_n$ is $L(\theta)$, *also called Likelihood Function*.
- $L(\theta) = P(X_1 = x_1, X_2 = x_2 \ldots\ldots X_n = x_n) = f(x_1, \theta).f(x_2, \theta).\ldots f(x_n, \theta) = \prod_{i=1}^{n} f(x_i, \theta)$.
- The first equality is just the definition of the joint probability mass function. The second equality comes from that fact that we have a RS, which implies by definition that the X_i are independent.

"**likelihood function**" L(θ) as a function of θ, and find the value of θ that maximizes it.

Maximum Likelihood…

– **Example**

- Consider RS $X_1, X_2 \ldots\ldots X_n$, where
- $X_i = 0$ if a randomly selected student does not own a sports car, and
- $X_i = 1$ if a randomly selected student does own a sports car.
- Assuming X_i are independent Bernoulli random variables with unknown parameter p, find the maximum likelihood estimator of p, the proportion of students who own a sports car.

Maximum Likelihood…

DTU Delhi Technological UNIVERSITY

– **Solution**

- X_i are independent Bernoulli random variables with unknown parameter p then the PMF is $f(x_i, p) = p^{x_i}.(1-p)^{1-x_i}$ for $x_i = 0 \; or \; 1$ and $0 < p < 1$.
- The likelihood function
- $L(p) = \prod_{i=1}^{n} f(x_i, p) = p^{x_1}(1-p)^{1-x_1} \times p^{x_2}(1-p)^{1-x_2} .. \times p^{x_n}.(1-p)^{1-x_n} = p^{\sum x_i}(1-p)^{\,n-\sum x_i}$
- Now, in order to implement the method of maximum likelihood, we need to find the p that maximizes the likelihood $L(p)$.
- Here, we can recall our calculus.

Maximum Likelihood...

– **Solution**

- To have easy calculation we can take logarithmic of both the side of $L(p) = p^{\sum x_i}(1-p)^{\,n-\sum x_i}$.
- $\log\big(L(p)\big) = \log\big(p^{\sum x_i}\big) + (n - \sum x_i)\, log\ \ (1-p)$ and taking derivative of the both the side and setting to '0'.
- $\frac{\partial \log(L(p))}{\partial p} = \frac{\sum x_i}{p} + \frac{n-\sum x_i)}{1-p} = 0$
- Multiply both side with $\mathrm{p}(1-p)$ and get
- $\sum x_i - \mathrm{p} \times \sum x_i - \mathrm{np} - \mathrm{p} \times \sum x_i = 0$ or $\sum x_i - np = 0$
- Hence the estimate $\boldsymbol{\hat{p}} = \frac{\boldsymbol{\sum_{i=1}^{n} x_i}}{\mathbf{n}}$, Alternatively, an estimator $\boldsymbol{\hat{p}} = \frac{\sum_{i=1}^{n} X_i}{n}$

Example 1

- *A coin is flipped 100 times, given that there were 55 heads, find the maximum likelihood estimate for the probability p of head on single toss.*
- **Solution**
- For a given value of p, the probability of getting 55 heads in this experiment is the binomial probability.
- The probability of getting 55 heads depends on the value of p, the conditional probability:
  - $P(heads/p) = \binom{100}{55} p^{55}(1-p)^{45}$: the probability of 55 heads given p
  - The **Maximum Likelihood Estimate** can be used to find the value of p.
  - $\frac{dP(heads/p)}{dp} = \binom{100}{55}(55p^{54}(1-p)^{45} - 45p^{54}(1-p)^{45}) = 0$
  - **The MLE is $\hat{p} = .55$**

Example 1

- **Alternate Solution: Log Likelihood**
- The log likelihood is :
  - $\ln\left(P\left(\frac{heads}{p}\right)\right) = \ln\left(\binom{100}{55}\right) + 55\ln(p) + 45\ln(1-p)$:
  - The Maximum Likelihood Estimate same as maximizing Log likelihood.
  - $\frac{d(\log liklihood)}{dp} = \frac{d\left[\ln\left(P\left(\frac{heads}{p}\right)\right) = \ln\left(\binom{100}{55}\right) + 55\ln(p) + 45\ln(1-p)\right]}{dp}$
  - $\frac{55}{p} - \frac{45}{1-p} = 0$
  - $55(1-p) = 45p$
  - **The log LE is $\hat{p} = .55$**

Expectation Maximization

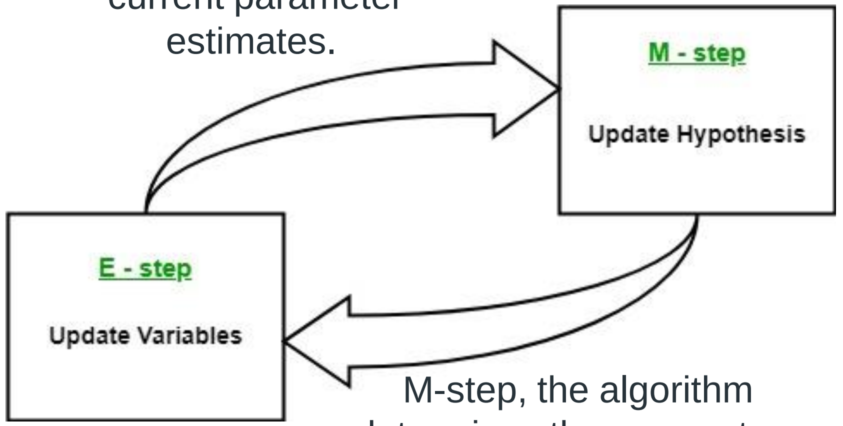
- Expectation Maximization (EM) Algorithm
  - EM algorithm is an approach for maximum likelihood estimation in the presence of latent variables.
  - The EM algorithm is an iterative approach that cycles between two modes.
  - The **first** mode attempts to estimate the missing or latent variables, called the estimation-step or E-step.
  - The **second** mode attempts to optimize the parameters of the model to best explain the data, called the maximization-step or M-step.
  - EM Algorithm can be used in unsupervised machine learning such as clustering and density estimation.

Latent variables are variables that are not directly observed but are inferred or estimated from other observed variables in a statistical or mathematical model.

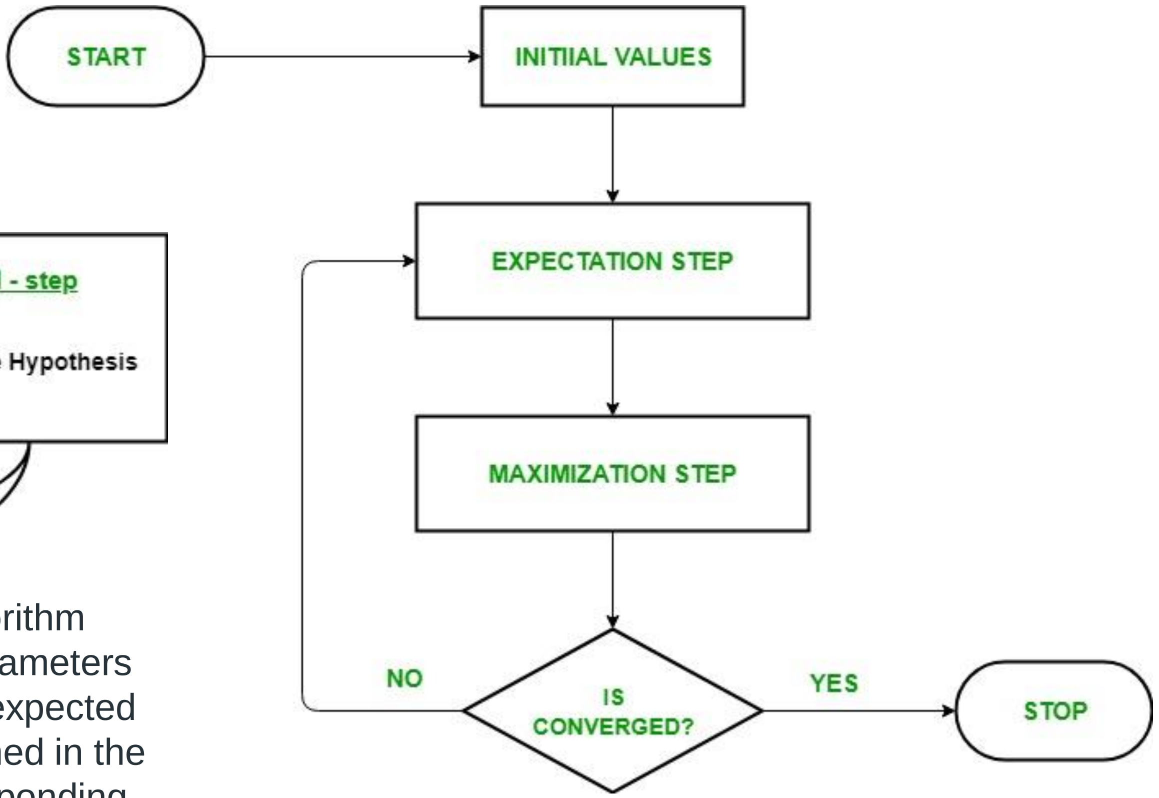
Expectation Maximization

– Expectation Maximization (EM) Algorithm

E-step, the algorithm computes the latent variables i.e. expectation of the log-likelihood using the current parameter estimates.



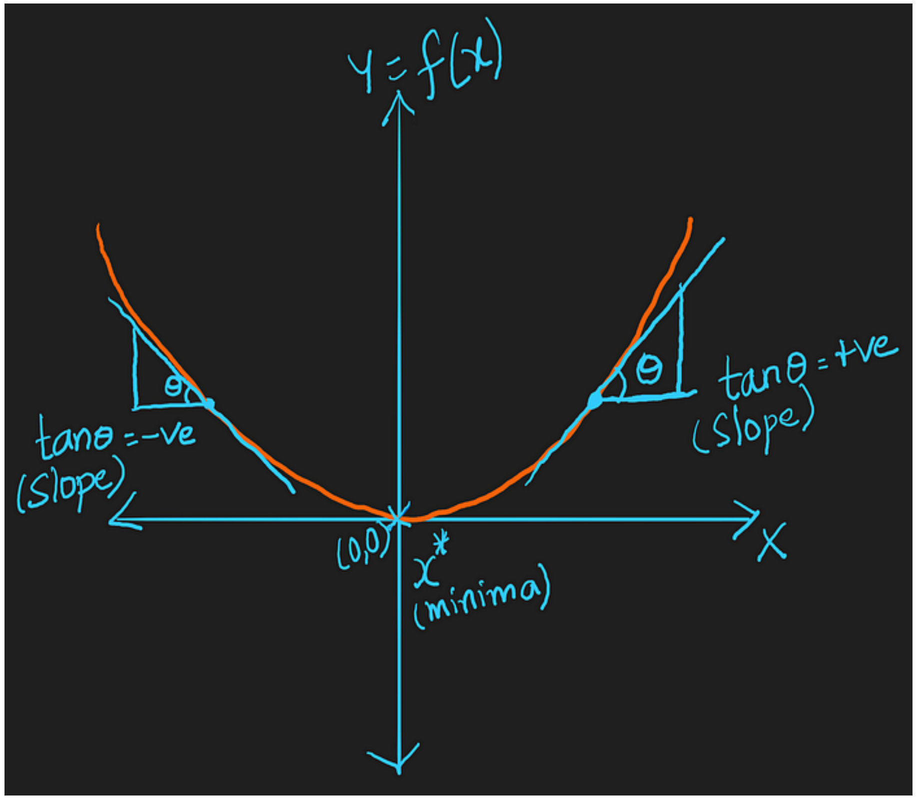
M-step, the algorithm determines the parameters that maximize the expected log-likelihood obtained in the E step, and corresponding model parameters are updated based on the estimated latent variables.



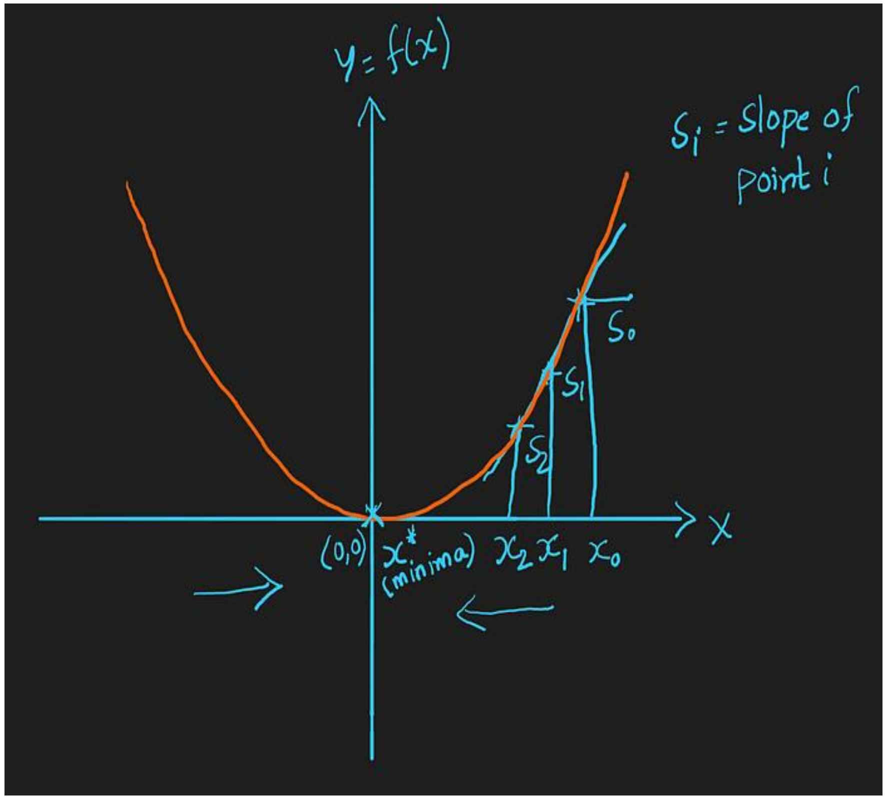
Gradient Descent (GD)

- **Gradient descent** is an iterative first-order optimization algorithm, which finds a local **minimum/maximum** of a given **function**.
- GD is commonly used in *machine learning* (ML) and *deep learning*(DL) to minimize a cost/loss function (e.g. in a linear regression).
- Also used in **Control Engineering** (robotics, chemical, etc.), **Computer games** & mechanical engineering
- Augustin-Louis Cauchy, who first suggested it in 1847.

Gradient Descent (GD)…



Slope of Y=X²



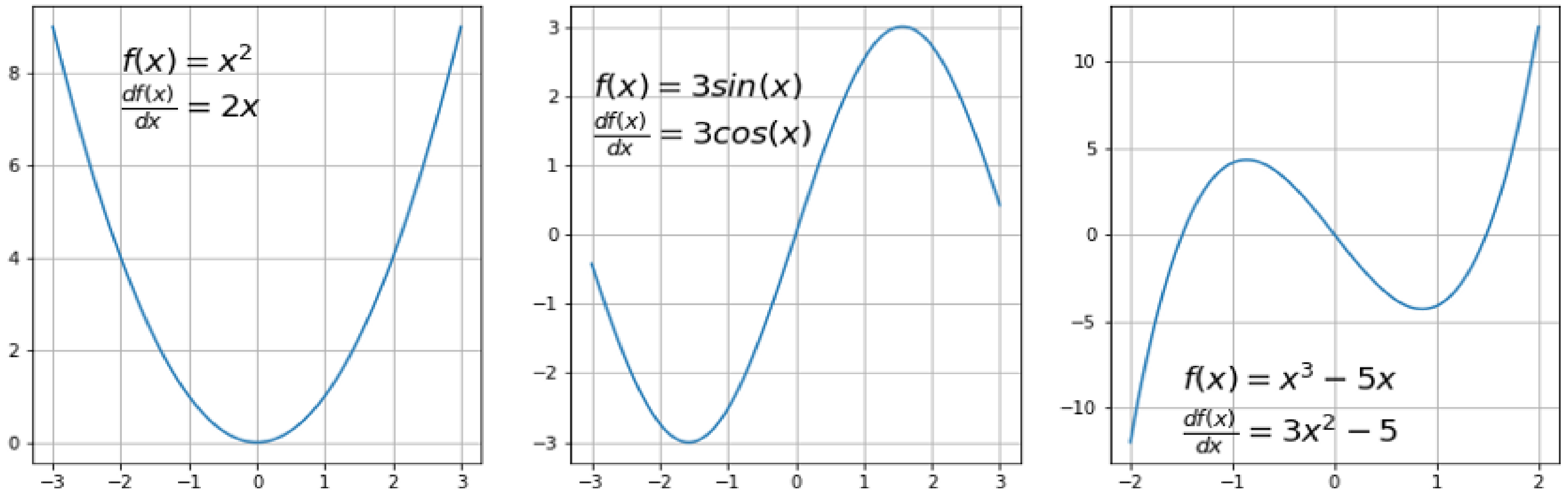
Slope of points as moved towards minima

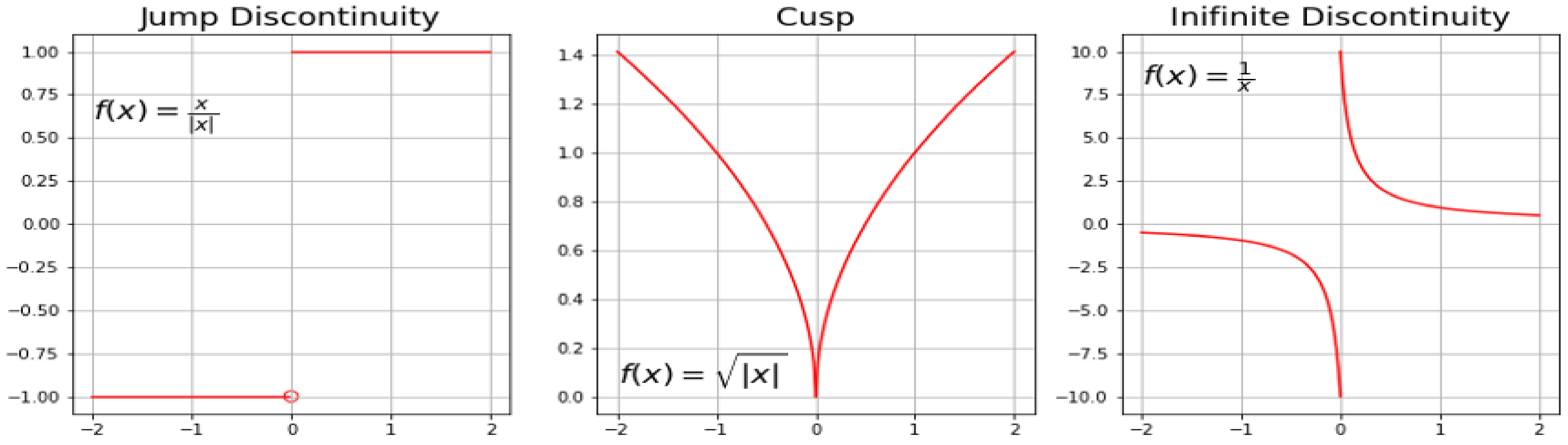
Fundamentals of GD

- **Function requirements**
- GD algorithm doesn't work for all functions. Hence, it has two specific requirements that a ***function*** has to be:
  - **Differentiable**
  - **Convex**
- A differentiable function has, its derivative for each point in its domain and not all functions meet this criteria, such as … next slide…

Fundamentals of GD…

E.g. differentiable functions

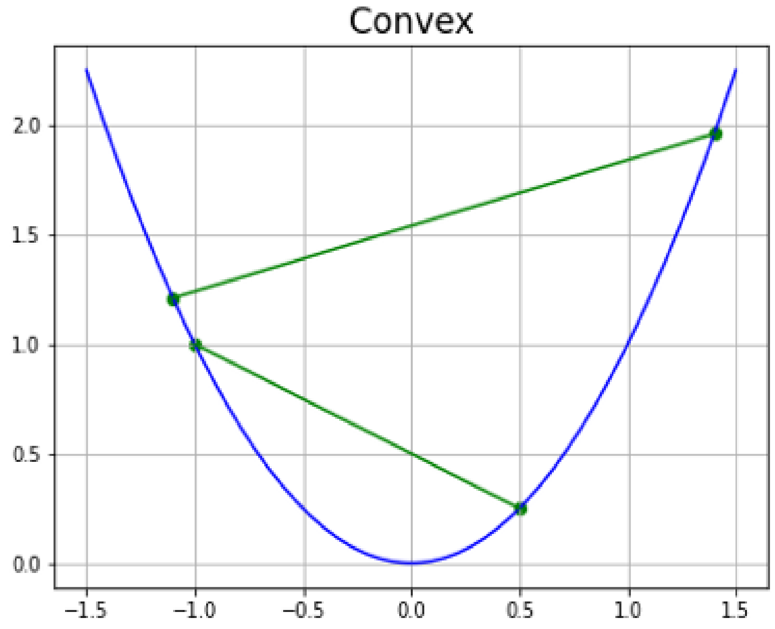


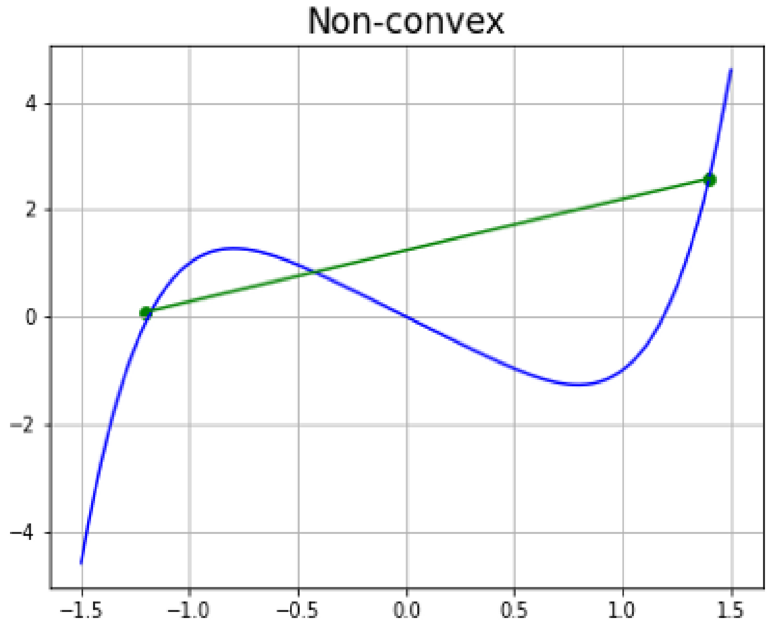


Non-differentiable functions have a step a cusp or a discontinuity

Fundamentals of GD…

- **Convexity in GD optimization**
  - Our goal is **to minimize the cost function in order to improve the accuracy of the model**. MSE is a convex function (it is differentiable twice). This **means** there is **no local minimum**, but only the ***global minimum***. Thus gradient descent would converge to the global minimum.



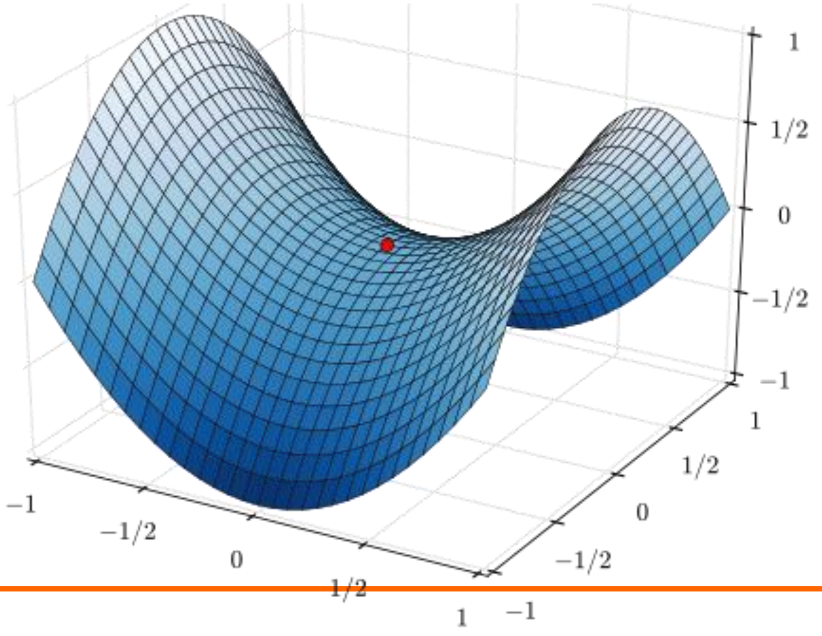


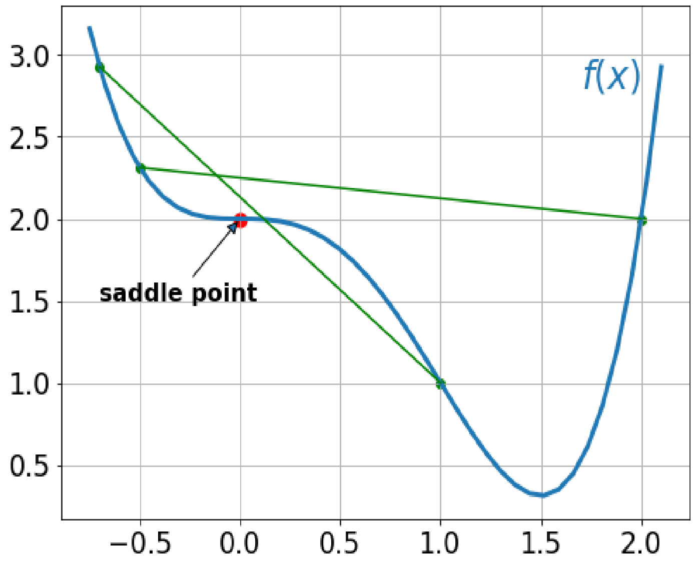
Fundamentals of GD…

- **Convexity in GD optimization**
  - Another way to check mathematically if a univariate function is ***convex*** then the second derivative is always **greater than 0**.
  - $\frac{d^2f(x)}{dx^2} > 0$; E.g. $f(x) = x^2 - x + 3$; $\frac{df(x)}{dx} = 2x - 1 \; and \; \frac{d^2f(x)}{dx^2} = 2$
  - Hence, $f(x)$ is **convex**.
  - There may be a case of **Quasi-Convex function** such as $f(x) = x^4 - 2x^3 + 2$; $\frac{df(x)}{dx} = 4x^3 - 6x^2 = x^2(4x - 6)$; here $x = 0 \; and \; x = 1.5$ where this function has extrema (maximum & Minimum).
  - Lets check the $\frac{d^2f(x)}{dx^2} = 12x^2 - 12x = 12x(x-1)$. The value of $\frac{d^2f(x)}{dx^2}$ is zero for *x=0* and *x=1*. These locations are called an **inflexion point**; a place where the curvature changes the sign from convex to concave or vice-versa

Fundamentals of GD…

- **Convexity in GD optimization: Quasi-Convex function…**
  - Now we see that point $x = 0$ has both first and second derivative equal to zero, meaning this is a **saddle point** and point x=1.5 is a global minimum.
    - For multivariate functions the most appropriate check if a point is a saddle point is to calculate a Hessian matrix which involves a bit more complex calculations





Gradient Descent…

- There are **three variants** of gradient descent, which differ in **how much data we use to compute the gradient** of the **objective function.**
- **Batch gradient descent**
- Vanilla gradient descent, aka batch gradient descent, computes the gradient of the cost function w.r.t. to the parameters **θ** for the entire training dataset:
- $\boldsymbol{\theta_{t+1} = \theta_t - \eta \nabla J(\theta)}$ **:** $\boldsymbol{\eta \rightarrow \textit{Learning Rate}}$
- GD algorithm iteratively calculates the next point using gradient at the current position, scales it (by a **learning rate**) and subtracts obtained value from the current position (makes a step).
- It **subtracts** the value because we want to minimize the function (to maximize it would be adding).

Batch gradient descent

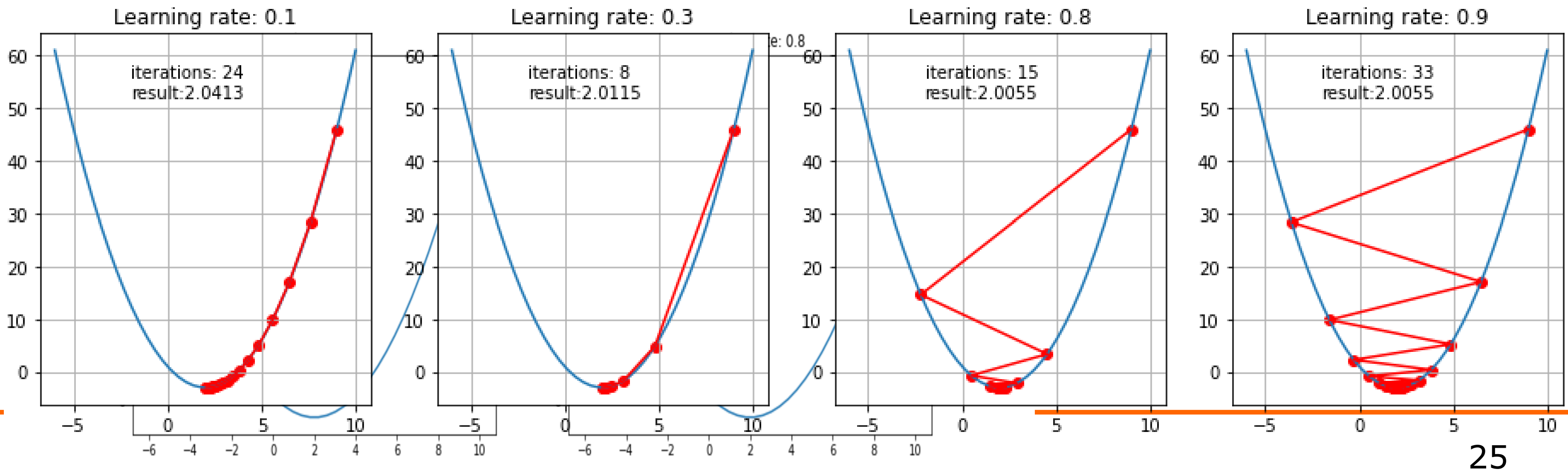
- η which scales the gradient and thus controls the step size. In machine learning, it is called **learning rate** and have a strong influence on performance.
- Smaller value η GD may converges in longer, or may reach maximum iteration before reaching the optimum point.
- Higher Value η, algorithm may not converge to the optimal point (jump around) or even to diverge completely.

Steps in BGD

1. Choose a starting point (initialization)
2. Calculate gradient at this point
3. Make a scaled step in the opposite direction to the gradient (objective: minimize)
4. Repeat points 2 and 3 until one of the criteria is met:
   - Maximum number of iterations reached
   - Step size is smaller than the tolerance (due to scaling or a small gradient)

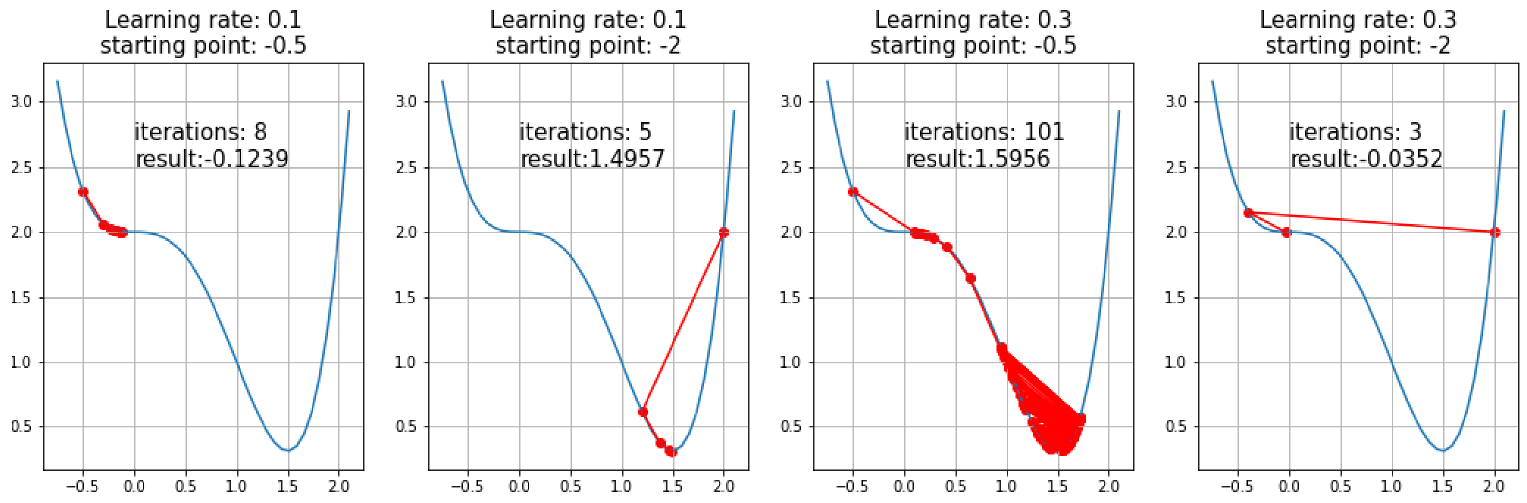
E.g. BGD

- A quadratic function : $f(x) = x^2 - 4x + 1$
- It is a univariate function. $\frac{df(x)}{dx} = 2x - 4$
- let us consider $\eta = 0.1$ and starting point $x = 9.$ Then calculation is as follows
- $\boldsymbol{x_1 = 9 - 0.1 * (2 * 9 - 4) = 7.6}$
- $\boldsymbol{x_2 = 7.6 - 0.1 * (2 * 7.6 - 4) = 6.8}$
- $\boldsymbol{x_3 = 6.8 - 0.1 * (2 * 6.8 - 4) = 5.584}$



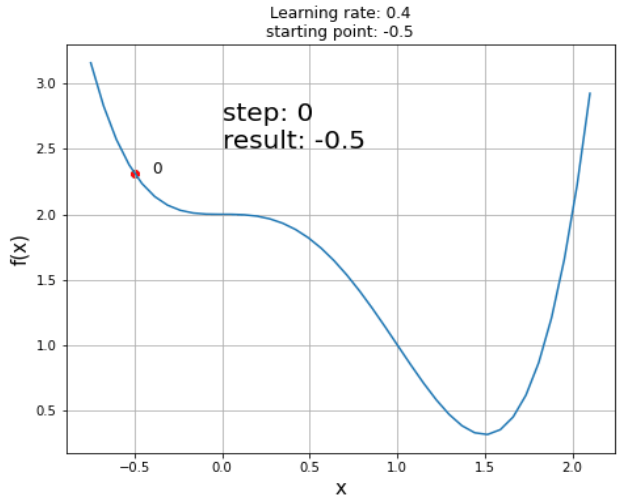
E.g. BGD

– **A function with a saddle point:** $f(x) = x^4 - 2x^3 + 2$.



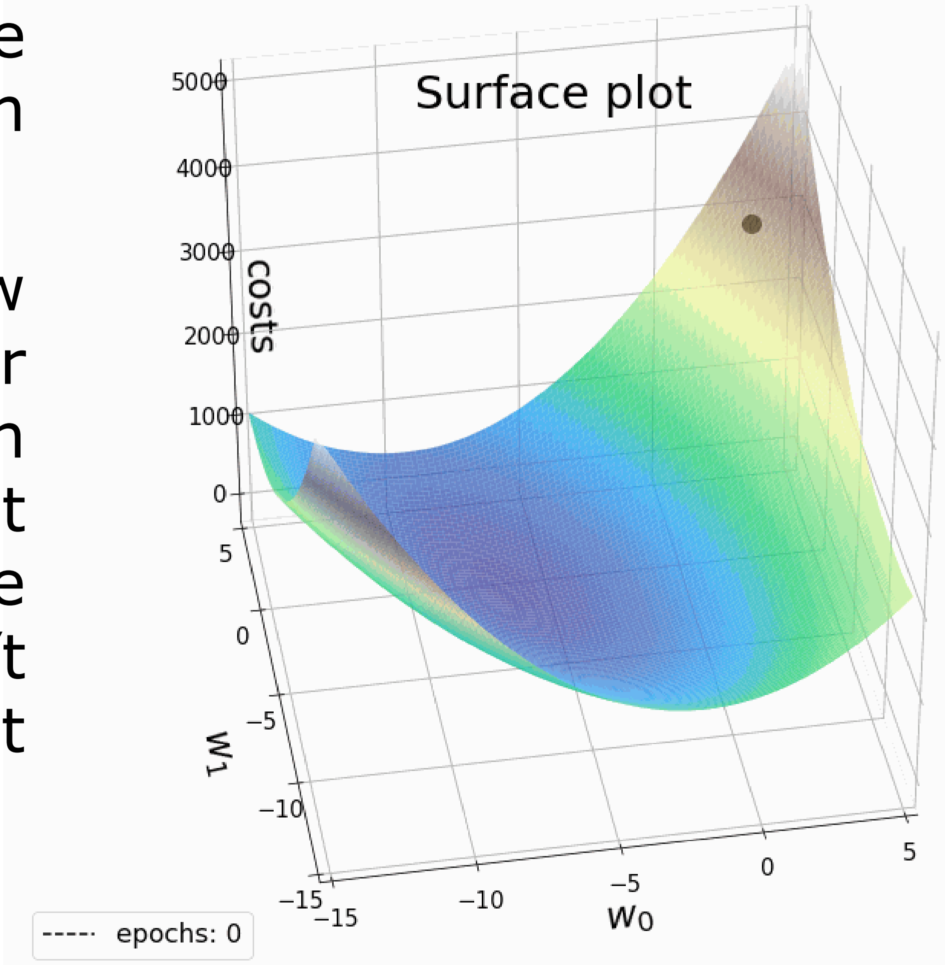
Results for two learning rates and two different staring points.

Learning rate of 0.4 and a starting point *x=-0.5*.



BGD...

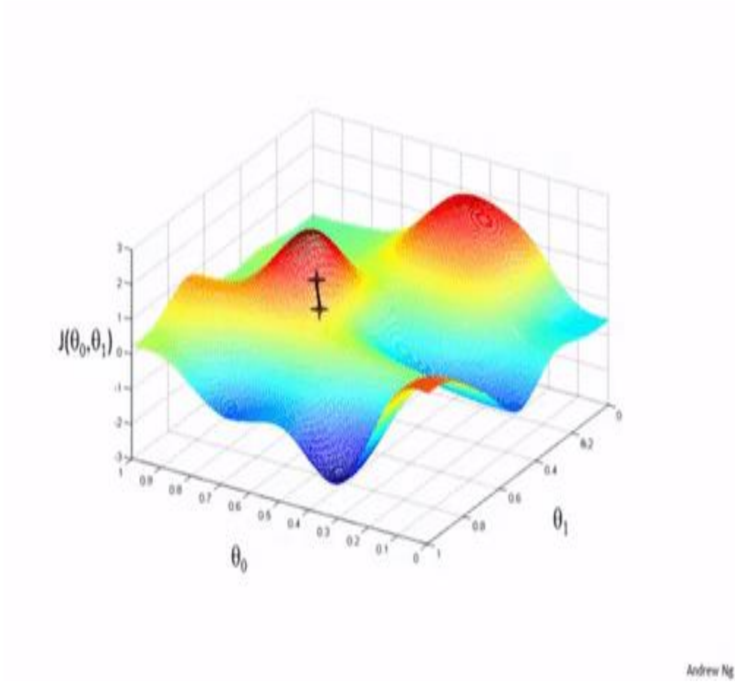
- It requires to calculate the gradients for the whole dataset to perform just one update.
- BGD can be very slow and is intractable for datasets that don't fit in memory, it also doesn't allow us to update the model online i.e BGD isn’t performed on dataset that update continuously.



Stochastic GD

- **Stochastic gradient descent**
  - Stochastic gradient descent (SGD) in contrast performs a parameter update for *each* training example $x(i)$ and label $y(i)$.
  - $\boldsymbol{\theta_{t+1} = \theta_t - \eta \nabla J(\theta; x(i); y(i))}$

- BGD performs redundant computations for large datasets, SGD avoids this redundancy by performing one update at a time.
- It is therefore usually much faster and can also be used to learn online.
- SGD performs frequent updates with a high variance that cause the objective function to fluctuate heavily .



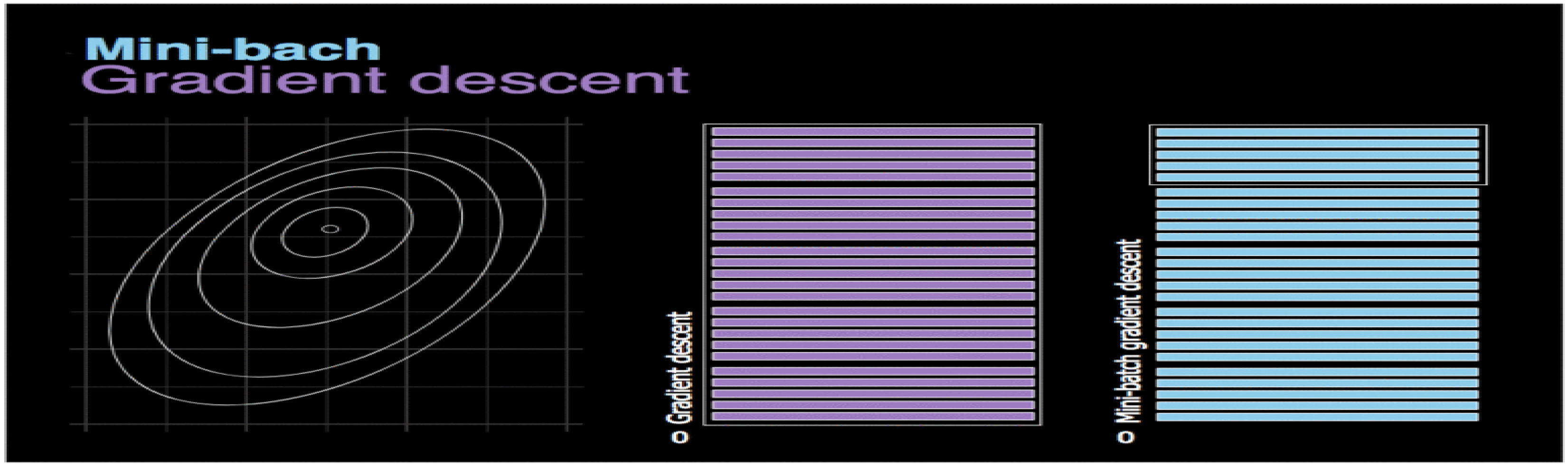
Stochastic GD…

- While SGD’s fluctuation, on the one hand, enables it to jump to new and potentially better local minima. On the other hand, this ultimately complicates convergence to the exact minimum, as SGD will keep overshooting.
- However, it has been shown that when we slowly decrease the learning rate, SGD shows the same convergence behavior as BGD.

Mini Batch GD

- Mini-batch gradient descent finally takes the best of both worlds and performs an update for every mini-batch of $\boldsymbol{n}$ training examples.
- $\boldsymbol{\theta_{t+1} = \theta_t - \eta \nabla J\big(\theta; x(i\!:\!I+n); y(i\!:\!I+n)\big)}$
- a) It reduces the variance of the parameter updates, which can lead to more stable convergence; b) can make use of highly optimized matrix optimizations common to state-of-the-art deep learning libraries that make computing the gradient w.r.t. a mini-batch very efficient.

Common mini-batch sizes range between 50 and 256



References

- https://online.stat.psu.edu/stat415/lesson/1/1.2
- https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21
- https://medium.com/analytics-vidhya/gradient-descent-optimization-techniques-4316419c5b74

THANK YOU

CONTACT: DINESH@DTU.AC.IN
MOBILE: +91-9971339840