

# GRADIENT DESCENT

**Arshia Anjum and Sibabrata Biswal**

National Institute of Science Education and Research ,  
Bhubaneswar,Odisha

January 31, 2023

# PART I: WHAT IS GRADIENT DESCENT?

- 1 Introduction . . . . . 8
- 2 The Formula . . . . . 9

## PART II: GRADIENT DESCENT - DETAILED WORKING

<b>1</b>	<b>Algorithm</b> . . . . .	<b>11</b>
<b>2</b>	<b>Gradient Descent of Simple Linear Regression Model (Example)</b> . . . . .	<b>12</b>
<b>3</b>	<b>Requirements of Gradient Descent</b> . . . . .	<b>15</b>
<b>4</b>	<b>Function Requirements</b> . . . . .	<b>16</b>
<b>5</b>	<b>All the Requirements of Gradient Descent: Listed</b> . . . . .	<b>21</b>

# PART III: VARIOUS TYPES OF GRADIENT DESCENT

- 1 **Types of Gradient Descent** . . . . . 23
- 2 **Stochastic Gradient Descent (SGD)** . . . . . 24
- 3 **Batch Gradient Descent (BGD)** . . . . . 25
- 4 **Mini-Batch Gradient Descent (MBGD)** . . . . . 26

## PART IV: PSEUDO-CODE FOR GRADIENT DESCENT

<b>1</b>	<b>Pseudo-Code for Gradient Descent . . . . .</b>	<b>28</b>
<b>2</b>	<b>Pseudo-Code for Stochastic Gradient Descent . . . . .</b>	<b>29</b>
<b>3</b>	<b>Pseudo-Code for Batch Gradient Descent . . . . .</b>	<b>30</b>
<b>4</b>	<b>Pseudo-Code for Mini-Batch Gradient Descent . . . . .</b>	<b>31</b>
<b>5</b>	<b>Python Code for Gradient Descent . . . . .</b>	<b>32</b>

# PART V: APPLICATION OF GRADIENT DESCENT

- 1 Gradient Descent Example 1 . . . . . 34
- 2 Gradient Descent Example 2 . . . . . 35

# PART VI: IS GRADIENT DESCENT A GOOD ALGORITHM?

- 1 Advantages . . . . . 37
- 2 Disadvantages . . . . . 38

# Part I

## WHAT IS GRADIENT DESCENT?



# INTRODUCTION

Gradient Descent is

- ▶ An optimisation technique/algorithm.
- ▶ Mostly used in supervised machine learning models and deep learning.
- ▶ Also called as first order optimisation algorithm.
- ▶ One of the most used algorithms for optimisation of parameters in ML models.

The meaning of Gradient Descent:

- ▶ The meaning of Gradient - first order derivative/ slope of a curve.
- ▶ The meaning of descent - movement to a lower point.
- ▶ The algorithm thus makes use of the gradient/slope to reach the minimum/ lowest point of a Mean Squared Error (MSE) function.

## THE FORMULA

While performing the algorithm of gradient descent, the machine iteratively calculates the next point it has to reach by using the gradient at the current position, and subtracting it from the parameter value by scaling it with a learning rate. The formula for the same looks like:

$$p_{n+1} = p_n - l\nabla f(p_n) \quad (1)$$

Where,  $l$  is the learning rate,  $p_n$  is the parameter to be optimised, and  $\nabla f(p_n)$  depicts the gradient of the expected loss function.

## Part II

# GRADIENT DESCENT - DETAILED WORKING

## ALGORITHM

- ▶ Suppose we have two unknown parameters,  $p_1$  and  $p_2$ .
- ▶ Assume the parameters (initial values of  $p_1$  and  $p_2$ ).
- ▶ Plot the expected loss function for various values of the other parameter,  $p_2$ .
- ▶ The point on the curve, where the value of the function is minimum, is the required value of  $p_2$  given  $p_1$ . (Note that the value of  $p_2$  might change with a change in  $p_1$ ).
- ▶ Now we change the values of  $p_1$  and  $p_2$  such that the expected loss reduces after each iteration.
- ▶ To define the iterations, we have the formula:

$$p_{n+1} = p_n - l\nabla f(p_n) \quad (2)$$

where,  $l$  defines the learning parameter and  $n$  defines the iteration number.

## GRADIENT DESCENT OF SIMPLE LINEAR REGRESSION MODEL (EXAMPLE)

- ▶ The Simple Linear Regression Model predicts the outcome through the equation [“Normal Equations” 2008]

$$y^{predicted} = y^p = w \times x + b \quad (3)$$

where,  $w$  is the slope of the linear curve, and  $b$  is the intercept value.  $x$  being the independent variable (the data given to us) and  $y$  being the dependent variable (the label we need to find).

- ▶ The Loss function for the same becomes

$$L(w, b) = \frac{1}{n} \sum_{i=0}^n (y_i(predicted) - y_i(actual))^2 = \frac{1}{n} \sum_{i=0}^n (y_i^p - y_i^a)^2 \quad (4)$$

- ▶ Now, Gradient Descent has to minimize this Loss function by choosing a set of  $w$  and  $b$  appropriately.
- ▶ We begin by choosing a set  $w_0$  and  $b_0$ . Then the equation becomes:

$$y_i^p = w_0 x_i + b_0 \quad (5)$$

- ▶ Now we have to update  $w$  and  $b$  re-iteratively to minimize the Loss function  $L(w, b)$ . The equations for the same become:

$$w = w - l\Delta w \quad (6)$$

$$b = b - l\Delta b \quad (7)$$

where  $l$  is the learning parameter.

## GRADIENT DESCENT OF SIMPLE LINEAR REGRESSION MODEL (EXAMPLE)

- ▶ To calculate  $\Delta w$  and  $\Delta b$  we use the relation:

$$\Delta w = \frac{\partial L(w, b)}{\partial w} \quad (8)$$

$$\Delta b = \frac{\partial L(w, b)}{\partial b} \quad (9)$$

- ▶ The definition of the Loss function was given in Eq. (4). Using the Simple Linear Regression Model Line (3),

$$L(w, b) = \frac{1}{n} \sum_{i=0}^n (wx_i + b - y_i^a)^2 \quad (10)$$

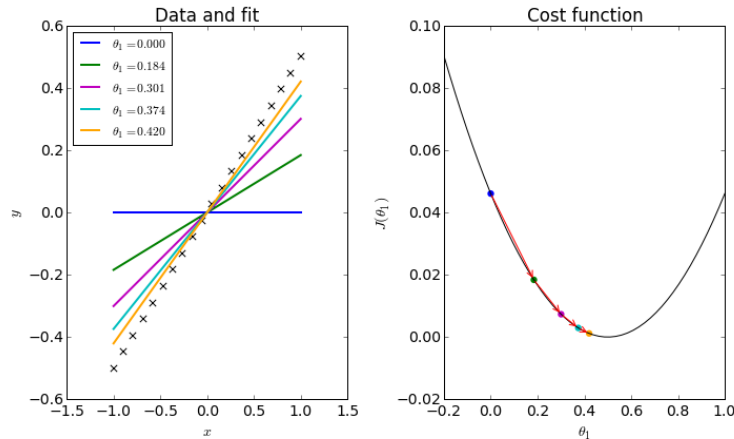
- ▶ This makes the Delta functions[*Gradient Descent* n.d.]/ Gradient functions to be:

$$\frac{\partial L(w, b)}{\partial w} = \frac{1}{n} 2 \sum_{i=0}^n (wx_i + b - y_i^a) w \quad (11)$$

$$\frac{\partial L(w, b)}{\partial c} = \frac{1}{n} 2 \sum_{i=0}^n (wx_i + b - y_i^a) b \quad (12)$$

## GRADIENT DESCENT OF SIMPLE LINEAR REGRESSION MODEL (EXAMPLE)

- ▶ After getting the Delta function values/ gradients through (11 and 12), we substitute them in Eq. (6 and 7).
- ▶ This counts for one iteration, however, it possible to get more accurate values, hence more such iterations are done until we reach the best fit.



**Figure.** Linear Regression Example for Gradient Descent.[*Visualizing the gradient descent method* n.d.]

## LEARNING PARAMETER

- ▶ The learning rate is mentioned in Eq (6 and 7) is an important parameter.
- ▶ It decides the rate at which the algorithm learns/ improves its unknown parameters [Jordan 2018].
- ▶ The gradients were calculated in Eq. (8 and 9), however, those used alone are not enough to make sure we reach the best fit at a low computational time.
- ▶ The learning parameter  $l$  decides the step size of the learning/iteration.
- ▶ There can be three cases when we don't use the learning parameter:
  1. The gradient is too small, and we reach the best fit point in too many iterations.
  2. The gradient is too high, and we jump the best fit point to go farther away from it.
  3. The gradient is just right, and we reach the best fit just fine.
- ▶ To make sure that the step size is just right, we use the learning parameter, which scales the change in the previous value of the parameter to get the new value.



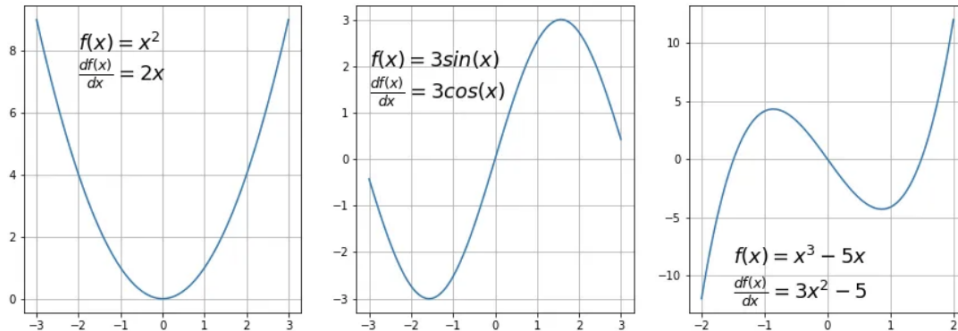
## FUNCTION REQUIREMENTS

- ▶ The gradient descent algorithm is not written for all types of functions.
- ▶ The function has to satisfy two conditions for Gradient Descent to be applicable on it:
  1. It has to be differentiable
  2. It has to be a convex function

# FUNCTION REQUIREMENTS

## DIFFERENTIABILITY

- Being differentiable means the function should have a derivative at each point in its domain. (cf. 2)

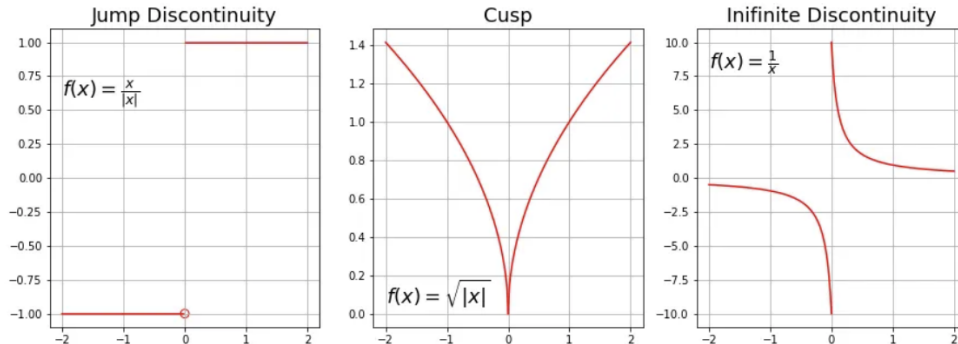


**Figure.** Examples of differentiable functions.[Kwiatkowski 2022]

# FUNCTION REQUIREMENTS

## DIFFERENTIABILITY

- ▶ Not all functions satisfy this condition. Some examples of functions not satisfying this condition is given below in Fig. 3
- ▶ If the function is not differentiable at all points, there may arise a circumstance where the gradient algorithm is not able to iterate the values as there is no gradient to work with.

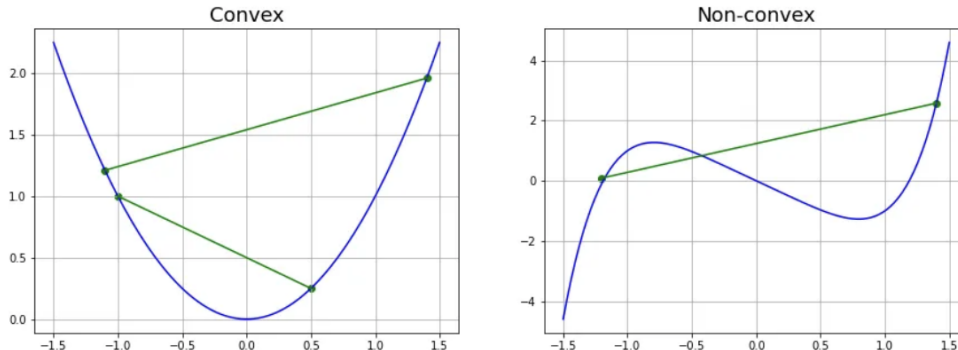


**Figure.** Examples of non-differentiable functions.[Kwiatkowski 2022]

# FUNCTION REQUIREMENTS

## CONVEX FUNCTION

- ▶ A convex function is one in which if any two points are connected on the curve, the line segment lies on or above the curve.
- ▶ If a function is non-convex, we do not have a surity that the gradient descent algorithm would give us the local minima or the global minima as the result.



**Figure.** Example of convex and non-convex functions.[Kwiatkowski 2022]

# FUNCTION REQUIREMENTS

## CONVEX FUNCTION

- ▶ A special case is where the function has a saddle point. At the saddle point, the function has gradient as zero, but the double derivative is also zero at that point. This defines that the point is a saddle and hence not a global minimum.

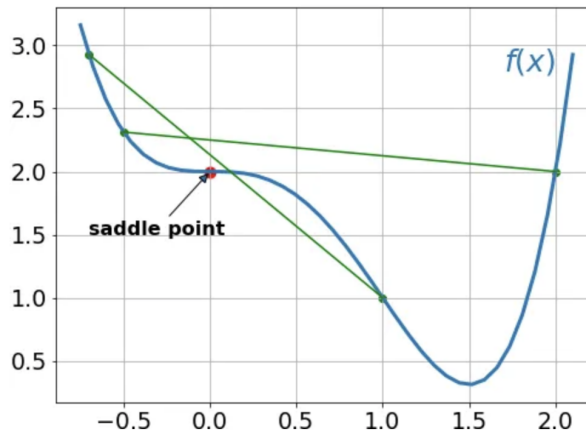


Figure. Example of saddle point in a function.[Kwiatkowski 2022]

## ALL THE REQUIREMENTS OF GRADIENT DESCENT: LISTED

Therefore, the gradient descent algorithm takes 5 parameters as its basic requirement:

- ▶ Initial Point ( $w_0, b_0$ )
- ▶ Gradient Function
- ▶ Learning Rate ( $l$ )
- ▶ Number of iterations ( $n$ )
- ▶ Tolerance - helps to give an end point/ stop to an algorithm.

## Part III

### TYPES OF GRADIENT DESCENT

# TYPES OF GRADIENT DESCENT

There are three types of Gradient Descent Algorithms:

1. Stochastic Gradient Descent (SGD)
2. Batch Gradient Descent (BGD)
3. Mini-Batch Gradient Descent (MBGD)



## STOCHASTIC GRADIENT DESCENT (SGD)

- ▶ SGD computes the gradient for only one random sample at each iteration.
- ▶ This property of SGD helps in it being faster and efficient as it does not have to process all the data in each of its iterations.
- ▶ However, the randomness of SGD contributes to the fact that it can in some cases give the suboptimal solutions/local minima as the result rather than the global minimum.
- ▶ One of the techniques to overcome this fault is to decrease the learning rate of the model over time, which helps in reducing the updates in the parameter with each iteration.
- ▶ SGD also has its variants, like Mini-Batch SGD, where the Gradient descent is done for a random subset of data, and Momentum SGD, where a term is added to the gradient update to help with the optimisation and avoiding getting stuck at a local minima.
- ▶ SGD is majorly used in Deep Learning and has found applications in classification, regression, and neural machine translation.

## BATCH GRADIENT DESCENT (BGD)

- ▶ BGD computes the gradient based on the average of the gradients of all data samples in the training set.
- ▶ Therefore, for each iteration, the gradient has to be calculated for the entire dataset, making BGD a computationally expensive process for huge datasets.
- ▶ However, it also gives BGD the advantage of being more stable and avoiding overfitting compared to SGD.
- ▶ The learning parameter has to be chosen carefully, so that with each iteration the step size doesn't get bigger or diverge from the minima.
- ▶ BGD is usually used in simpler models with lesser data.
- ▶ It is widely used in linear regression and logistic regression, not so much in deep learning where the training set is typically large and the models have many parameters.

## MINI-BATCH GRADIENT DESCENT (MBGD)

- ▶ MBGD is a combination of SGD and BGD.
- ▶ MBGD, in comparison to BGD, computed the gradient over a subset of the data, called the mini-batch.
- ▶ This helps in computing faster than BGD and avoiding overfitting as compared to SGD.
- ▶ The mini-batch size is a trade-off between speed and stability, with smaller sizes leading to faster convergence but increased variability in the optimization, and larger sizes leading to more stable convergence but slower processing times.
- ▶ The learning rate, which determines the size of the parameter update at each iteration, must be carefully tuned to ensure that the optimization converges to the minimum and does not oscillate or diverge.
- ▶ Mini-Batch Gradient Descent is widely used in deep learning and has been applied to a variety of tasks, including classification, regression, and neural machine translation.

## Part IV

# PSEUDO-CODE FOR GRADIENT DESCENT

## PSEUDO-CODE FOR GRADIENT DESCENT

Let us look at the pseudo-code of the gradient descent algorithm:

Input: parameters ( $\theta$ ), gradient of the loss function with respect to the parameters ( $d\theta$ ), learning rate ( $\alpha$ )

Update parameters:  $\theta = \theta - \alpha \times d\theta$

Output: updated parameters ( $\theta$ )

## PSEUDO-CODE FOR STOCHASTIC GRADIENT DESCENT

Let us look at the pseudo-code of the gradient descent algorithm:

```
INPUT: cost function  $J(\theta)$ , learning rate  $\alpha$ , number of iterations  $N$ 
INITIALIZE: random  $\theta$ 
FOR i = 1 to  $N$  DO
  FOR j = 1 to number of training examples  $m$  DO
    Compute the gradient of  $J$  with respect to  $\theta$  for a single training example:

      gradient =  $\nabla_{\theta} J(\theta, x^j, y^j)$ 
    Update the parameters  $\theta$ :
       $\theta = \theta - \alpha \times \text{gradient}$ 
  END FOR
END FOR
OUTPUT:  $\theta$ 
```

## PSEUDO-CODE FOR BATCH GRADIENT DESCENT

Let us look at the pseudo-code for Batch Gradient Descent:

```
INPUT: cost function  $J(\theta)$ , learning rate  $\alpha$ , number of iterations  $N$ 
INITIALIZE: random  $\theta$ 
FOR  $i = 1$  to  $N$  DO
    Compute the gradient of  $J$  with respect to  $\theta$  for all training examples:
        gradient =  $1/m * \nabla_{\theta} \sum(J(\theta, x^j, y^j))$ 
    Update the parameters  $\theta$ :
         $\theta = \theta - \alpha \times \text{gradient}$ 
END FOR
OUTPUT:  $\theta$ 
```

## PSEUDO-CODE FOR MINI-BATCH GRADIENT DESCENT

Let us look at the pseudo-code for mini-batch gradient descent:

```
INPUT: cost function  $J(\theta)$ , learning rate  $\alpha$ , number of iterations  $N$ ,  
       batch size  $B$   
INITIALIZE: random  $\theta$   
FOR  $i = 1$  to  $N$  DO  
    Split the training examples into  $B$  mini-batches of size  $b$ :  
    FOR  $j = 1$  to number of mini-batches  $B$  DO  
        Compute the gradient of  $J$  with respect to  $\theta$  for a mini-batch  
        of training examples:  
        gradient =  $1/b \times \nabla_{\theta} \Sigma(J(\theta, x^j, y^j))$   
        Update the parameters  $\theta$ :  
         $\theta = \theta - \alpha \times \text{gradient}$   
    END FOR  
END FOR  
OUTPUT:  $\theta$ 
```



## PYTHON CODE FOR GRADIENT DESCENT

```
def GradientDescent(X, y, w, b, learningRate, MaxIterations):  
    for i in range(MaxIterations):  
        # Compute the error/cost function J(w,b)  
        J = CostFunction(X, y, w, b)  
        # Compute the gradient of the cost function with respect to w and b  
        dw = gradientOfw(X, y, w, b)  
        db = gradientOfb(X, y, w, b)  
        # Update the parameters:  
        w = w - learningRate * dw  
        b = b - learningRate * db  
    return w, b
```

## Part V

# APPLICATIONS OF GRADIENT DESCENT

## GRADIENT DESCENT EXAMPLE 1

One common example of gradient descent is training a linear regression model. The model tries to fit a line to a set of data points by minimizing the mean squared error between the predicted values and the actual target values. The model adjusts the parameters (slope and intercept) in the direction of the negative gradient of the error function, until the error reaches a minimum. See section (2), which clearly explains this example.

## GRADIENT DESCENT EXAMPLE 2

- ▶ Let's say we want to minimize the following quadratic function:

$$Q(x, y) = (x - 2)^2 + (y - 3)^2 \quad (13)$$

- ▶ The gradient of  $Q(x, y)$  with respect to  $x$  and  $y$  is given by:

$$\nabla Q(x, y) = [2(x - 2), 2(y - 3)] \quad (14)$$

- ▶ The gradient descent algorithm updates  $x$  and  $y$  in each iteration according to the formula:

$$x = x - \alpha \times 2(x - 2) \quad (15)$$

$$y = y - \alpha \times 2(y - 3) \quad (16)$$

where  $\alpha$  is the learning rate.

- ▶ By repeating the above updates, the algorithm converges to the minimum of  $Q(x, y)$ , which is  $(2, 3)$ .
- ▶ The rate of convergence depends on the choice of  $\alpha$ , but as  $\alpha$  approaches 0, the convergence becomes slower, while as  $\alpha$  approaches infinity, the algorithm may not converge at all.

## Part VI

# IS GRADIENT DESCENT A GOOD ALGORITHM?

## ADVANTAGES

Gradient descent is a widely used optimization algorithm in various fields such as machine learning, deep learning, and optimization problems. It has several advantages, including:






1. **Ease of implementation:** Gradient descent is relatively easy to implement, as it only requires the calculation of gradients and updates to the parameters. It does not require complex mathematical techniques like linear algebra, eigenvalue decomposition, or matrix inversion.
2. **Convergence guarantee:** Gradient descent is guaranteed to converge to a minimum, under certain conditions such as the cost function being differentiable and having a unique global minimum. The convergence speed can be controlled by the learning rate, which can be set appropriately to achieve a good balance between convergence speed and accuracy.
3. **Scalability:** Gradient descent can be used for high-dimensional problems and can scale well with large amounts of data. In practice, the convergence of gradient descent can be accelerated through techniques such as batch normalization, early stopping, or momentum.
4. **Versatility:** Gradient descent can be applied to a wide range of optimization problems, including linear regression, logistic regression, and neural networks. It can also be used for non-convex optimization problems, where multiple local minima may exist, although convergence to a good solution may be less certain in such cases.

## DISADVANTAGES

However, gradient descent has some limitations as well, including:

1. Sensitivity to learning rate: The choice of the learning rate can have a significant impact on the convergence of the algorithm. If the learning rate is too high, the algorithm may oscillate or converge slowly, while if it is too low, the convergence may be slow.
2. Sensitivity to initialization: The initial values of the parameters can also have an impact on the convergence of the algorithm. If the parameters are initialized too far from the minimum, the algorithm may converge slowly, or get stuck in a suboptimal solution.
3. Convergence speed: The convergence speed of gradient descent can be slow for some problems, especially for high-dimensional or non-convex optimization problems. This can be mitigated to some extent by using techniques such as mini-batch gradient descent, momentum, or adaptive learning rates.
4. Termination of Algorithm: The termination of this algorithm isn't as easy as it seems. It is either done by giving a tolerance limit or an upper cap on the number of iterations.
5. Correctness: Though the algorithm makes sure that output converges to best fit, it doesn't guarantee the correctness of the solution.

## REFERENCES I

-  [Gradient Descent \(n.d.\)](https://www.niser.ac.in/smishra/teach/cs460/2020/lectures/lec8/). <https://www.niser.ac.in/smishra/teach/cs460/2020/lectures/lec8/>. [Online; accessed 2023-01-31].
-  [Jordan, Jeremy \(Mar. 2018\)](https://www.jeremyjordan.me/nn-learning-rate/). *Setting the learning rate of your neural network*. <https://www.jeremyjordan.me/nn-learning-rate/>. [Online; accessed 2023-01-31].
-  [Kwiatkowski, Robert \(July 2022\)](https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21). *Gradient Descent Algorithm — a deep dive*. <https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21>. [Online; accessed 2023-01-31].
-  [“Normal Equations” \(2008\)](#). In: *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, pp. 380–382. ISBN: 978-0-387-32833-1. DOI: 10.1007/978-0-387-32833-1\_286. URL: [https://doi.org/10.1007/978-0-387-32833-1\\_286](https://doi.org/10.1007/978-0-387-32833-1_286).
-  [Visualizing the gradient descent method \(n.d.\)](https://scipython.com/blog/visualizing-the-gradient-descent-method/). <https://scipython.com/blog/visualizing-the-gradient-descent-method/>. [Online; accessed 2023-01-31].