Linear Regression*

February 15, 2023

This course note has two parts. The first part is on linear regression, and the second part is on the gradient descent optimization technique.

1 Introduction

1.1 Definition

In machine learning, a regression problem is the problem of determining a relation between one or more independent variables and an output variable which is a real continuous variable, given a set of observed values of the set of independent variables and the corresponding values of the output variable.

1.2 Examples

- Consider the data on car prices given in Table 1. Suppose we are required to estimate the price of a car aged 25 years with distance 53240 km, and weight 1200 pounds. This is an example of a regression problem because we have to predict the value of the numeric variable "Price".

- Consider the navigation of a mobile robot, say an autonomous car. The output is the angle by which the steering wheel should be turned at each time, to advance without hitting obstacles and deviating from the route. Inputs are provided by sensors on the car like a video camera, GPS, and so forth.

- In finance, the capital asset pricing model uses regression for analyzing and quantifying the systematic risk of an investment.

- In economics, regression is the predominant empirical tool. For example, it is used to predict consumption spending, inventory investment, purchases of a country's exports, spending on imports, labor demand, and labor supply.

*References

- Ethem Alpayidin. Introduction to Machine Learning, MIT Press, 2004
- Christopher Bishop. Pattern Recognition and Machine Learning. 2006
- V. N. Krishnachandran. Lecture Notes in Machine Learning. 2018
- Robert Kwiatkowski. Gradient Descent Algorithm - a deep dive. May 22, 2021

Table 1: Prices of used cars: example data for regression.

Price (US$)	Age (years)	Distance (km)	Weight (pounds)
13500	23	46986	1165
13750	23	72937	1165
13950	24	41711	1165
14950	26	48000	1165
13750	30	38500	1170
12950	32	61000	1170
16900	27	94612	1245
18600	30	75889	1245
21500	27	19700	1185
12950	23	71138	1105

1.3 General approach

Let x denote the set of input variables and y the output variable. In machine learning, the general approach to regression is to assume a model, that is, some mathematical relation between x and y, involving some parameters θ, in the following form:

$$y = f(x, \theta). \tag{1}$$

The function $y = f(x, \theta)$ is called the *regression function*. The machine learning algorithm optimizes the parameters in the set θ such that the approximation error is minimized; that is, the estimates of the values of the dependent variable y are as close as possible to the correct values given in the training set.

Example. If the input variables are "Age", "Distance" and "Weight" and the output variable is "Price", the model $y = f(x, \theta)$ may be

$$\text{Price} = a_0 + a_1 \cdot \text{Age} + a_2 \cdot \text{Distance} + a_3 \cdot \text{Weight},$$

where $x = \{\text{Age}, \text{Distance}, \text{Weight}\}$ denotes the the set of input variables and $\theta = \{a_0, a_1, a_2, a_3\}$ denotes the set of parameters of the model. The goal of regression is to determine the unknown parameters of the model and to use the model for prediction.

1.4 Different regression models[1]

There are various types of regression techniques available to make predictions. These techniques mostly differ in three aspects, namely, the number and type of independent variables, the type of dependent variables and the shape of regression line. Some of these are listed below.

- **Simple linear regression**. There is only one continuous independent variable x, and the assumed relation between the independent variable x and the dependent variable y is

$$y = a + bx. \tag{2}$$

[1]Note: *logistic regression* has a binary dependent variable (i.e., a variable which takes only the values 0 and 1), which is referred to as a classification model.

- **Multivariate linear regression**. There are more than one independent variable, say $x1, ..., x_n$, and the assumed relation between the independent variables and the dependent variable is

$$y = a_0 + a_1x_1 + ... + a_nx_n. \tag{3}$$

- **Polynomial regression**. There is only one continuous independent variable x and the assumed model is

$$y = a_0 + a_1x + a_2x^2 + ... + a_nx^n. \tag{4}$$

1.5 Criterion for minimization of error

In regression, we would like to write the numeric output y, called the dependent variable, as a function of the input x, called the independent variable. We assume that the output is the sum of a function of the input, i.e., $f(x)$, and some random error denoted by ϵ, i.e.,

$$y = f(x) + \epsilon. \tag{5}$$

Here the function $f(x)$ is unknown and we would like to approximate it by some estimator $g(x, \theta)$ containing a set of parameters θ. We assume that the random error ϵ follows normal distribution with mean 0.

Let $x_1, ..., x_n$ be a random sample of observations of the input variable x and $y_1, ..., y_n$ the corresponding observed values of the output variable y. The value of desired θ minimizes the following sum of squares:

$$E(\theta) = (y_1 - g(x_1, \theta))^2 + ... + (y_n - g(x_n, \theta))^2. \tag{6}$$

The method of finding the value of θ that minimizes $E(\theta)$ is known as the ordinary least squares method.

2 Simple linear regression

Let x be the independent predictor variable and y the dependent variable. Assume that we have a set of observed values of x and y as shown in Table 2.

Table 2: Data set for simple linear regression.

x	x_1	x_2	...	x_n
y	y_1	y_2	...	y_n

A simple linear regression model defines the relationship between x and y using a line defined by an equation in the following form:

$$y = \alpha + \beta x. \tag{7}$$

To determine the optimal estimates of α and β, an estimation method known as Ordinary Least Squares (OLS) is used.

The OLS method. In the OLS method, the values of y-intercept and slope are chosen such that they minimize the sum of the squared errors; that is, the sum of the

squares of the vertical distance between the predicted y-value and the actual y-value (see Figure 1). Let $\hat{y}_i$ be the predicted value of y_i, the sum of squares of errors is given by

$$\begin{aligned} E &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^{n} [y_i - (\alpha + \beta x_i)]^2 \end{aligned} \tag{8}$$

So we are required to find the values of α and β, such that E is minimum. Using methods of calculus, we can show that the values of α and β can be obtained by solving the following equations:

$$\begin{aligned} \sum_{i=1}^{n} y_i &= n\alpha + \beta \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i y_i &= \alpha \sum_{i=1}^{n} x_i + \beta \sum_{i=1}^{n} x_i^2 \end{aligned} \tag{9}$$

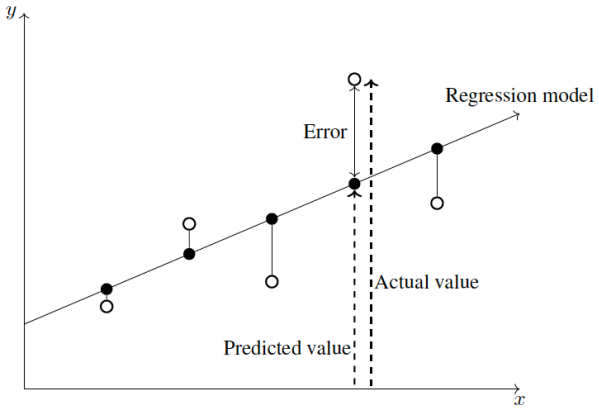


Figure 1: Errors in observed values.

Formulas to find α and β. Recall that the means of x and y are given by

$$\begin{aligned} \bar{x} &= \frac{1}{n} \sum x_i \\ \bar{y} &= \frac{1}{n} \sum y_i \end{aligned}, \tag{10}$$

and also that the variance of x is given by

$$\mathrm{Var}(x) = \frac{1}{n-1} \sum (x_i - \bar{x}_i)^2 . \tag{11}$$

The *covariance of x and y*, denoted by $Cov(x, y)$ is defined as

$$\mathrm{Cov}(x, y) = \frac{1}{n-1} \sum (x_i - \bar{x}) (y_i - \bar{y}) \tag{12}$$

It can be shown that the values of α and β can be computed using the following formulas:

$$\begin{aligned} \beta &= \frac{\mathrm{Cov}(x, y)}{\mathrm{Var}(x)} \\ \alpha &= \bar{y} - \beta \bar{x} \end{aligned} \tag{13}$$

Remarks. It is interesting to note why the least squares method discussed above is christened as "ordinary" least squares method. Several different variants of the least squares method have been developed over the years. For example, in the weighted least squares method, the coefficients α and β are estimated such that the weighted sum of squares of errors

$$E = \sum_{i=1}^{n} w_i \left[y_i - (\alpha + \beta x_i) \right]^2 \tag{14}$$

for some positive constants $w_1, ..., w_n$, is minimum. There are also methods known by the names generalized least squares method, partial least squares method, total least squares method, etc. The reader may refer to Wikipedia to obtain further information about these methods. The OLS method has a long history. The method is usually credited to Carl Friedrich Gauss (1795), but it was first published by Adrien-Marie Legendre (1805).

Example. Obtain a linear regression for the data in Table 3 assuming that y is the independent variable.

Table 3: Example data for simple linear regression.

x	1.0	2.0	3.0	4.0	5.0
y	1.00	2.00	1.30	3.75	2.25

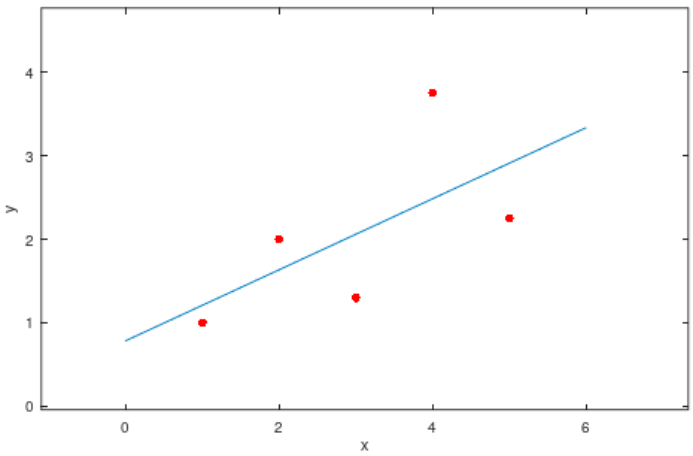


Figure 2: Regression model for Table 3.

Solution. In the usual notations of simple linear regression, we have

$$
\begin{aligned}
n &= 5 \\
\bar{x} &= \frac{1}{5}(1.0 + 2.0 + 3.0 + 4.0 + 5.0) = 3.0 \\
\bar{y} &= \frac{1}{5}(1.00 + 2.00 + 1.30 + 3.75 + 2.25) = 2.06 \\
\mathrm{Cov}(x, y) &= \frac{1}{4}[(1.0 - 3.0)(1.00 - 2.06) + \cdots + (5.0 - 3.0)(2.25 - 2.06)] = 1.0625 \\
\mathrm{Var}(x) &= \frac{1}{4}\left[(1.0 - 3.0)^2 + \cdots + (5.0 - 3.0)^2\right] = 2.5 \\
b &= \frac{1.0625}{2.5} = 0.425 \\
a &= 2.06 - 0.425 \times 3.0 = 0.785
\end{aligned}
$$

Therefore, the linear regression model for the data is

$$y = 0.785 + 0.425x.$$

3 Polynomial regression

Let x be the independent predictor variable and y the dependent variable. Assume that we have a set of observed values of x and y as in Table 2. A polynomial regression model defines the relationship between x and y by an equation in the following form:

$$y = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \cdots + \alpha_k x^k \tag{15}$$

To determine the optimal values of the parameters $\alpha_0, \alpha_1, ..., \alpha_k$, the method of ordinary least squares can be used. The desired values of the parameters are those values that minimize the sum of squared errors, i.e.,

$$E = \sum_{i=1}^{n} \left[y_i - \left(\alpha_0 + \alpha_1 x_i + \alpha_2 x_i^2 + \cdots + \alpha_k x_i^k\right)\right]^2 \tag{16}$$

The optimal values of the parameters are obtained by solving the following system of equations

$$\frac{\partial E}{\partial \alpha_i} = 0, \quad \forall i = 0, 1, \ldots, k. \tag{17}$$

Simplifying and transforming Equation 17, we get a system of $k + 1$ linear equations

$$
\begin{aligned}
\sum y_i &= \alpha_0 n + \alpha_1 \left(\sum x_i\right) + \cdots + \alpha_k \left(\sum x_i^k\right) \\
\sum y_i x_i &= \alpha_0 \left(\sum x_i\right) + \alpha_1 \left(\sum x_i^2\right) + \cdots + \alpha_k \left(\sum x_i^{k+1}\right) \\
\sum y_i x_i^2 &= \alpha_0 \left(\sum x_i^2\right) + \alpha_1 \left(\sum x_i^3\right) + \cdots + \alpha_k \left(\sum x_i^{k+2}\right) \\
&\vdots \\
\sum y_i x_i^k &= \alpha_0 \left(\sum x_i^k\right) + \alpha_1 \left(\sum x_i^{k+1}\right) + \cdots + \alpha_k \left(\sum x_i^{2k}\right)
\end{aligned} \tag{18}
$$

We can see that the values of α_i can be obtained by solving the above linear system.

Remarks. The linear system of equations to find α_i has a compact matrix representation, $\vec{y} = D\vec{\alpha}$, where

$$\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad D = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^k \\ 1 & x_2 & x_2^2 & \cdots & x_2^k \\ \vdots & & & & \\ 1 & x_n & x_n^2 & \cdots & x_n^k \end{bmatrix}, \text{ and } \vec{\alpha} = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_k \end{bmatrix}. \tag{19}$$

Then we have

$$\vec{\alpha} = \left(D^T D\right)^{-1} D^T \vec{y}, \tag{20}$$

where the superscript T denotes the transpose of a matrix.

Example. Find a quadratic regression model for the data show in Table 4.

Table 4: Example data for quadratic polynomial regression.

x	3.0	4.0	5.0	6.0	7.0
y	2.5	3.2	3.8	6.5	11.5

Solution. Let the quadratic regression model be

$$y = \alpha_0 + \alpha_1 x + \alpha_2 x^2.$$

The values of α_0, α_1 and α_2 that minimize the sum of squares of errors satisfy the following system of equations

$$\begin{aligned} \sum y_i &= n\alpha_0 + \alpha_1 \left(\sum x_i\right) + \alpha_2 \left(\sum x_i^2\right) \\ \sum y_i x_i &= \alpha_0 \left(\sum x_i\right) + \alpha_1 \left(\sum x_i^2\right) + \alpha_2 \left(\sum x_i^3\right) \\ \sum y_i x_i^2 &= \alpha_0 \left(\sum x_i^2\right) + \alpha_1 \left(\sum x_i^3\right) + \alpha_2 \left(\sum x_i^4\right) \end{aligned}$$

Using the given data (shown in Table 4), we have

$$\begin{aligned} 27.5 &= 5\alpha_0 + 25\alpha_1 + 135\alpha_2 \\ 158.8 &= 25\alpha_0 + 135\alpha_1 + 775\alpha_2 \\ 966.2 &= 135\alpha_0 + 775\alpha_1 + 4659\alpha_2 \end{aligned}$$

Solving this system of equations we get

$$\begin{aligned} \alpha_0 &= 12.4285714 \\ \alpha_1 &= -5.5128571 \\ \alpha_2 &= 0.7642857 \end{aligned}$$

Thus, the required quadratic polynomial model is

$$y = 12.4285714 - 5.5128571x + 0.7642857x^2.$$

Figure 3 shows plots of the data and the quadratic polynomial model.

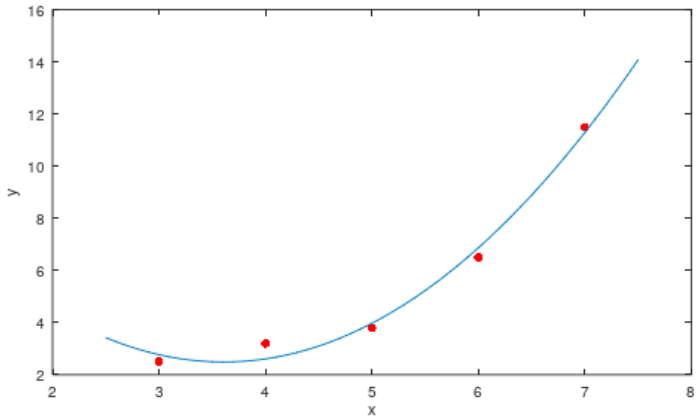


Figure 3: Plot of quadratic polynomial model.

4 Multiple linear regression

We assume that there are N independent variables $x_1, x_2, ..., x_N$. Let the dependent variable be y. Let there also be n observed values of these variables, as shown in Table 5.

Table 5: Datasets overview

Variables	Values (examples)			
	Example 1	Example 2	...	Example n
x_1	x_{11}	x_{12}	...	x_{1n}
x_1	x_{21}	x_{22}	...	x_{2n}
...				
x_N	x_{N1}	x_{N2}	...	x_{Nn}
y (outcomes)	y_1	y_2	...	y_n

The multiple linear regression model defines the relationship between the N independent variables and the dependent variable by an equation of the following form:

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_N x_N \tag{21}$$

As in simple linear regression, here also we use the ordinary least squares method to obtain the optimal estimates of $\beta_0, \beta_1, ..., \beta_N$. The method yields the following procedure for the computation of these optimal estimates. Let

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad X = \begin{bmatrix} 1 & x_{11} & x_{21} & \cdots & x_{N1} \\ 1 & x_{12} & x_{22} & \cdots & x_{N2} \\ \vdots & & & & \\ 1 & x_{1n} & x_{2n} & \cdots & x_{Nn} \end{bmatrix}, \text{ and } B = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_N \end{bmatrix}. \tag{22}$$

It can be shown that the regression coefficients are given by

$$B = \left(X^T X\right)^{-1} X^T Y. \tag{23}$$

Example. Fit a multiple linear regression model to the data show in Table 6.

Table 6: Example data for multi-linear regression.

x_1	1	1	2	0
x_2	1	2	2	1
y	3.25	6.5	3.5	5.0

Solution. In this problem, there are two independent variables and four sets of values of the variables. Thus, in the notations used above, we have $n = 2$ and $N = 4$. The multiple linear regression model for this problem has the form

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2.$$

The computations are shown below.

$$Y = \begin{bmatrix} 3.25 \\ 6.5 \\ 3.5 \\ 5.0 \end{bmatrix}, \quad X = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 2 \\ 1 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 4 & 4 & 6 \\ 4 & 6 & 7 \\ 6 & 7 & 10 \end{bmatrix}$$

$$\left(X^T X\right)^{-1} = \begin{bmatrix} \frac{11}{4} & \frac{1}{2} & -2 \\ \frac{1}{2} & 1 & -1 \\ -2 & -1 & 2 \end{bmatrix}$$

$$B = \left(X^T X\right)^{-1} X^T Y = \begin{bmatrix} 2.0625 \\ -2.3750 \\ 3.2500 \end{bmatrix}$$

Thus the required model is

$$y = 2.0625 - 2.3750 x_1 + 3.2500 x_2.$$

Figure 4 shows plots of the data and the multiple linear regression model (a plane in this example).

5 Gradient descent

In the previous sections, we have provided a closed form solution to the linear regression problems (see Equations 13, 20, and 23). In this section, we show that how linear regression can be solved as an optimization problem. We will introduce an optimization algorithm called *gradient descent*[2]. Gradient descent is an iterative first-order optimization algorithm used to find a local minimum/maximum of a given function. This method is commonly used in machine learning and deep learning to minimize a cost/loss function. It also has wide applications in areas such as control engineering (robotics, chemical, etc.), computer games, and mechanical engineering.

[2]The gradient descent method was proposed before the era of modern computers and there was an intensive development meantime that led to numerous improved versions of it. In this course note, we focus on the basic version/idea of the algorithm.

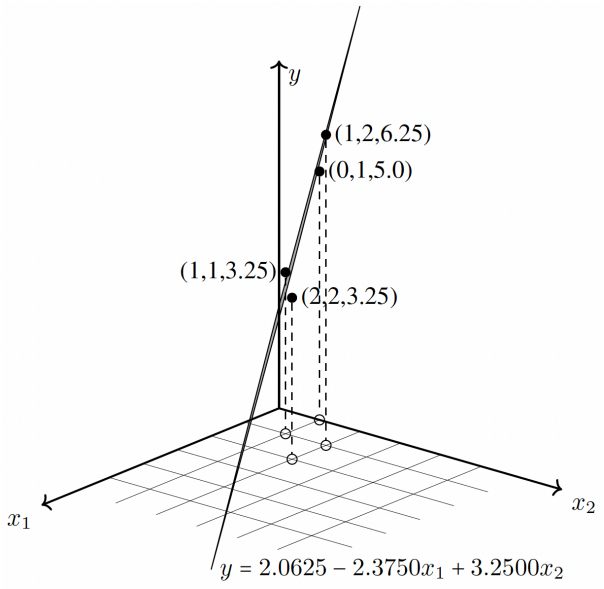


Figure 4: The regression plane for the data in Table 6.

5.1 Function requirements

Gradient descent algorithm does not work for all functions. There are two specific requirements. A function has to be:

- *differentiable*. This requires that a function has a derivative for each point in its domain. See Figure 5 for a few examples of differentiable functions and Figure 6 for a few examples of non-differentiable functions.

- *convex*. For a univariate function, this means that the line segment connecting two function's points lays on or above its curve (i.e., it does not cross it). A line crossing the curve means that the function has a local minimum (which may not be the global one). Mathematically, for two points $(x_1, f(x_1))$, $(x_2, f(x_2))$ laying on or above the function's curve, this condition can be expressed as

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2),$$

  where λ determines a point's location on a section line and its value is between 0 (left point) and 1 (right point). For example, $\lambda = 0.5$ means a location in the middle. See Figure 7 for two functions with exemplary section lines. Another way to check mathematically if a univariate function is convex is to calculate the second derivative and check if its value is always bigger than 0, i.e., $d^2 f(x)/dx^2 > 0$.

Example. Let's investigate a simple quadratic function given by $f(x) = x^2 - x + 3$. Its first and second derivative are

$$\frac{df(x)}{dx} = 2x - 1, \quad \frac{d^2 f(x)}{dx^2} = 2.$$

Because the second derivative is always bigger than 0, $f(x)$ is strictly convex. It is also possible to find extreme value of quasi-convex functions using a gradient descent

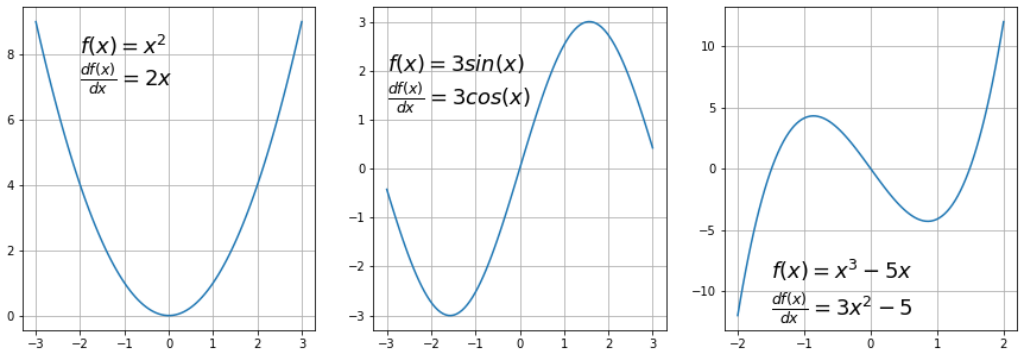
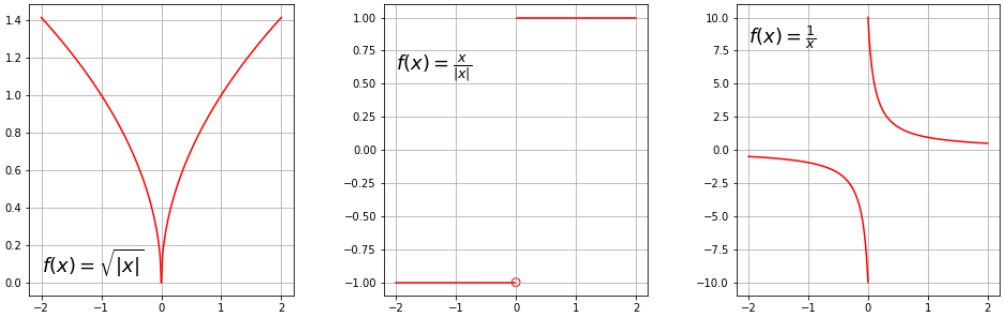


Figure 5: Examples of differentiable functions.



(a) Cusp (b) Jump discontinuity (c) Infinite discontinuity

Figure 6: Examples of non-differentiable functions. Typical non-differentiable functions have a step a cusp or a discontinuity.

algorithm. However, often they have so-called saddle points (also called minimax points) where the algorithm can get stuck. An example of a quasi-convex function is

$$f(x) = x^4 - 2x^3 + 2,$$

whose first order derivative is

$$\frac{df(x)}{dx} = 4x^3 - 6x^2 = x^2(4x - 6),$$

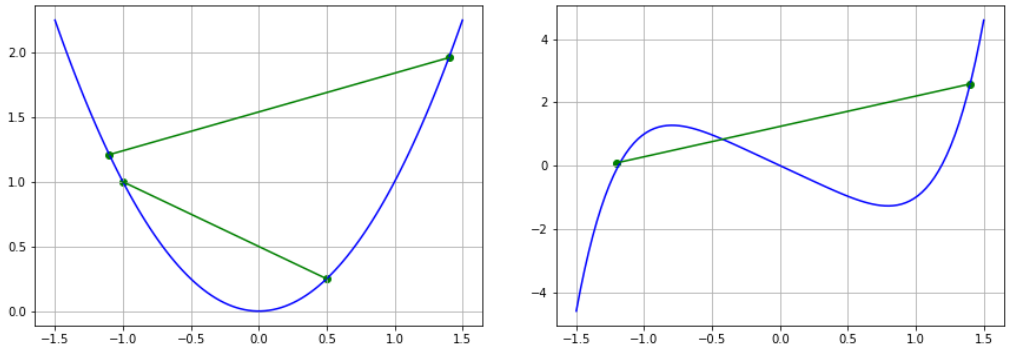
and its second order derivative is

$$\frac{d^2f(x)}{dx^2} = 12x^2 - 12x = 12x(x - 1).$$

The value of the second order derivative is zero for $x = 0$ and $x = 1$. These locations are called an inflection point, i.e., a place where the curvature changes sign. In other words, the function changes from convex to concave or vice-versa. By analyzing this equation we can conclude that

- for $x < 0$: function is convex
- for $0 < x < 1$: function is concave
- for $x > 1$: function is convex again

Now we see that point $x = 0$ has both first and second derivatives equal to zero meaning this is a saddle point, and point $x = 1.5$ is a global minimum. The graph of this function is shown in Figure 8. As calculated before, a saddle point is at $x = 0$ and its minimum at $x = 1.5$.



(a) Convex (b) Non-convex

Figure 7: Exemplary convex and non-convex functions.

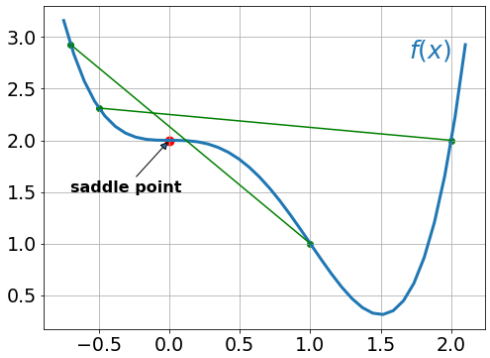


Figure 8: Example of a semi-convex function with a saddle point.

5.2 Gradient

Before jumping into the code of the method, one more thing has to be explained - what is a gradient. Intuitively it is a slope of a curve at a given point in a specified direction. In the case of a univariate function, it is simply the first derivative at a selected point. In the case of a multivariate function, it is a vector of derivatives in each main direction (along variable axes). Because for each variable we are interested only in the slope along its axis and we don't care about other variables, these derivatives are called partial derivatives. A gradient for an $n-$dimensional function $f(\vec{x})$ at a given point $\vec{p}$ is defined as

$$\nabla f(\vec{p}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\vec{p}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\vec{p}) \end{bmatrix}. \tag{24}$$

The upside-down triangle ∇ is the so-called nabla symbol and you read it "del". To better understand how to calculate it, let's do a hand calculation for an exemplary 2-dimensional function below (the function is plotted in Figure 9),

$$f(x, y) = 0.5x^2 + y^2.$$

Its gradient can be derived as

$$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f}{\partial x}(x, y) \\ \\ \frac{\partial f}{\partial y}(x, y) \end{bmatrix} = \begin{bmatrix} x \\ \\ 2y \end{bmatrix}.$$

Let's assume we are interested in a gradient at point $p(10, 10)$, which is

$$\nabla f(10, 10) = \begin{bmatrix} 10 \\ 20 \end{bmatrix}.$$

By looking at these values we can see that the slope is twice steeper along the y axis.

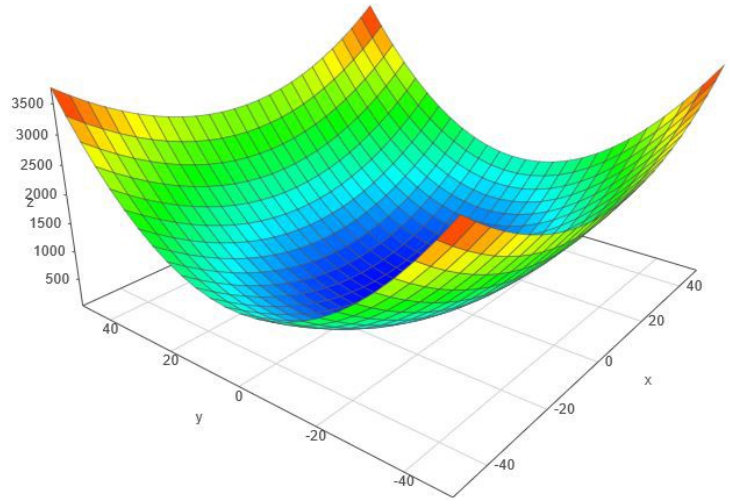


Figure 9: The 3D plot of the 2-dimensional function $f(x, y) = 0.5x^2 + y^2$.

5.3 Gradient descent algorithm

The gradient descent (GD) algorithm iteratively calculates the next point using gradient at the current position, then 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 a function (to maximize it would be adding). This process can be written as:

$$\vec{p}_{n+1} = \vec{p}_n - \eta \nabla f\left(\vec{p}_n\right). \tag{25}$$

There's an important parameter η that scales the gradient and thus controls the step size. In machine learning, it is called learning rate and has a strong influence on performance.

- The smaller learning rate the longer GD converges, or may reach maximum iteration before reaching the optimum point.
- If learning rate is too big the algorithm may not converge to the optimal point (jump around) or even to diverge completely.

In summary, the main steps of the gradient descent method are:

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 steps 2) and 3) until one of the criteria is met:

- maximum number of iterations reached.
- step size is smaller than a given tolerance.

Below is an exemplary implementation of the Gradient Descent algorithm (with steps tracking):

```python
import numpy as np

def gradient_descent(start, gradient, learn_rate, max_iter, tol=0.01):
  steps = [start] # history tracking
  x = start

  for _ in range(max_iter):
    diff = learn_rate*gradient(x)
    if np.abs(diff)<tol:
      break
    x = x - diff
    steps.append(x) # history tracing

  return steps, x
```

This function takes 5 parameters:

1) starting point - in our case, we define it manually but in practice, it is often an initial guess or even random initialization.

2) gradient function - a function that computes the gradient of the original function, which has to be specified before-hand.

3) learning rate - scaling factor for step sizes.

4) maximum number of iterations.

5) tolerance, to conditionally stop the algorithm (in this case a default value is 0.01).

5.4 Examples

Example 1: a quadratic function. Let's take a simple univariate quadratic function

$$f(x) = x^2 - 4x + 1.$$

Its gradient function is

$$\frac{df(x)}{dx} = 2x - 4.$$

Let's write these functions in Python:

```python
def func1(x):
  return x**2-4*x+1

def gradient_func1(x):
  return 2*x - 4
```

For this function, by taking a learning rate of 0.1 and starting point at $x = 9$ we can easily calculate each step by hand using Equation 25. Let's do it for the first 3 steps:

$$\begin{aligned} x_0 &= 9, & f(9) &= 46 \\ x_1 &= 9 - 0.1 \times (2 \times 9 - 4) = 7.6, & f(7.6) &= 28.36 \\ x_2 &= 7.6 - 0.1 \times (2 \times 7.6 - 4) = 6.48, & f(6.48) &= 17.07 \\ x_3 &= 6.48 - 0.1 \times (2 \times 6.48 - 4) = 5.584, & f(5.584) &= 9.845 \end{aligned}$$

Continuing this process, you will get

$$\begin{aligned} &\ldots \\ x_{21} &= 2.065, \quad f(2.065) = -2.996 \\ x_{22} &= 2.052, \quad f(2.052) = -2.997 \end{aligned}$$

In the last step, the change in x is 0.007, which is smaller than the specified tolerance 0.01, and thus the process terminated. Please also note that the change in the function value is 0.001.
The python function is called by:

```
history, result = gradient_descent(9, gradient_func1, 0.1, 100)
```

In Figure 10, the trajectories, number of iterations, and the final converged result (within tolerance) for various learning rates are shown.

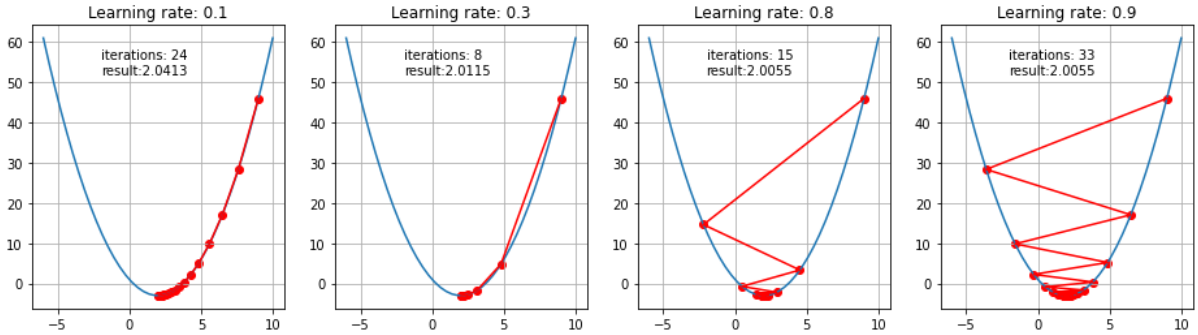


Figure 10: The gradient descent steps for minimizing the function $f(x) = x^2 - 4x + 1$ with different learning rates.

Example 2: a function with a saddle point. Now let's see how the algorithm will cope with a semi-convex function we investigated mathematically before.

$$f(x) = x^4 - 2x^3 + 2.$$

Figure 11 shows the results for two learning rates and two different starting points. We can see that the existence of a saddle point imposes a real challenge for the gradient descent algorithm and obtaining a global minimum is not guaranteed. Second-order algorithms (e.g. Newton's method[3]) deal with these situations better. Investigation of saddle points and how to escape from them is a subject of ongoing studies and various solutions were proposed. For example, Jin et al. proposed a Perturbing Gradient Descent algorithm[4].

[3]Newton's method in optimization. Wikipedia.

[4]Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M. Kakade, Michael I. Jordan. How to Escape Saddle Points Efficiently. PMLR 70:1724-1732, 2017

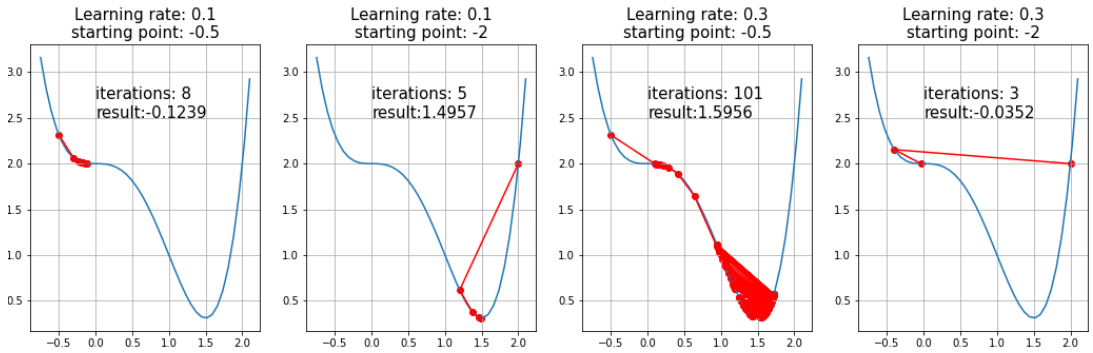


Figure 11: The gradient descent steps for minimizing the function $f(x) = x^4 - 2x^3 + 2$ with two different learning rates and two different starting points.

5.5 Summary

We have learned how a gradient decent algorithm works, when it can be used and the common challenges when using it. This will be a good starting point to explore more advanced gradient-based optimization methods like Momentum or Nesterov (Accelerated) Gradient Descent, ADAM, or second-order ones like the Newton's algorithm.