Chapter 2

Math for Machine Learning

We are going to look at few basic concepts of linear algebra, optimization that are core for any machine learning enthusiast

Linear Algebra

Solving two linear Equations

$$3x + 4y = 9$$

$$9x + 22y = 15$$

Matrix notation

$$AX = B$$

$$\begin{bmatrix} 3 & 4 \\ 9 & 22 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 9 \\ 15 \end{bmatrix}$$

One of the methods to find a solution to this kind of system of equations by finding inverese to A

Simply put,

$$X = A^{-1}B, where A^{-1} = 1/ad - bc \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Optimization

Optimization is a process of finding the best possible values to solve a system of equation(s) or inequation(s). There is usually an objective, forexample to minimize cost also equivalent to maximize profit subject to constraints

Linear Optimization

Linear optimization, also called linear programming (LP), is a powerful technique for making the best possible decisions in situations where relationships between variables and constraints are all linear. Think of it as finding the perfect balance in a system where everything is represented by straight lines.

NonLinear Optimization

Non-linear optimization, as the name suggests, is the optimization counterpart to linear optimization where the relationships between variables and constraints are not straight lines, but rather involve curves, exponents, or

other non-linear functions. This makes the problem more complex to solve compared to linear programming, but it also allows for modeling a wider range of real-world scenarios

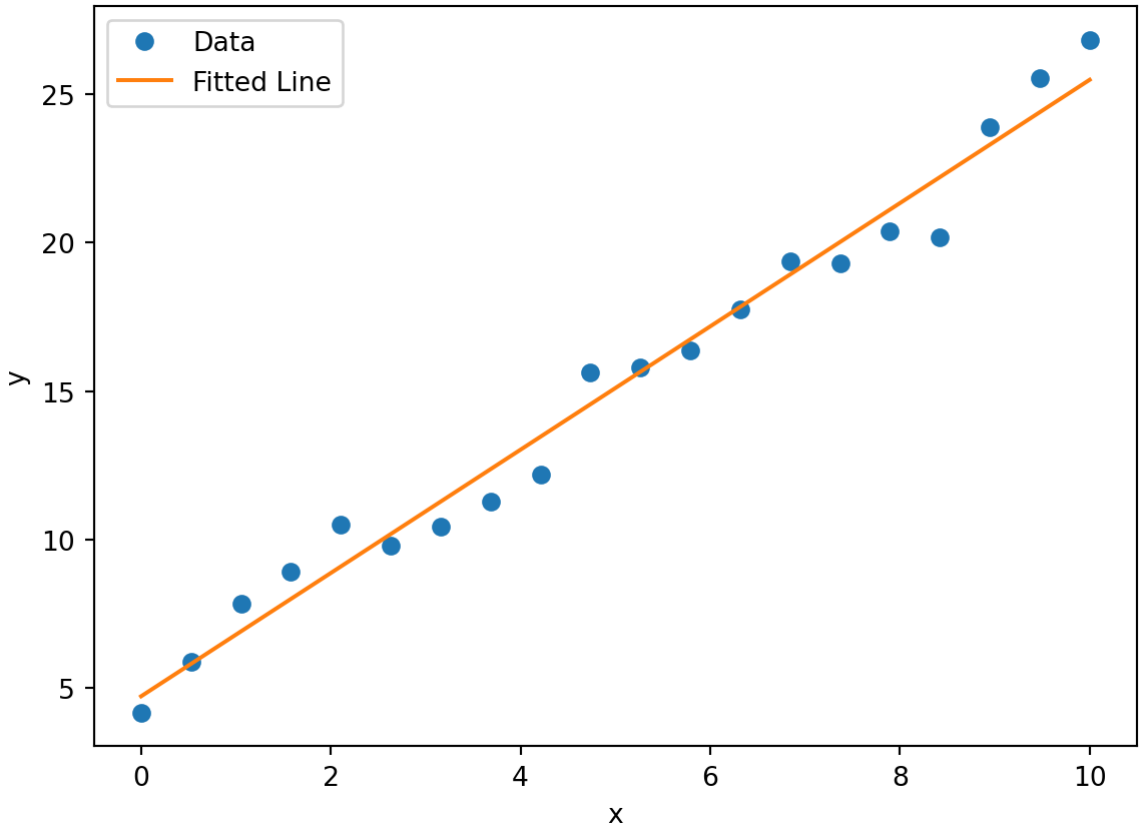
Method of Least Squares

The method of least squares is a powerful statistical technique used to find the best-fitting line or curve to a set of data points. In simpler terms, it helps you draw the line that most closely “hugs” the data points, minimizing the overall distance between the line and the actual data

This method works by minimizing the sum of the squared errors and finding best possible values for the variables.

Please refer to the example in excel(google sheets) associated with this section

```
import numpy as np
import matplotlib.pyplot as plt

# Generate sample data with some noise
x = np.linspace(0, 10, 20)
y = 2 * x + 5 + np.random.randn(20)

# Create design matrix with a constant term
X = np.vstack([np.ones(len(x)), x]).T

# Calculate least squares solution (beta = (X^T X)^(-1) X^T y)
beta = np.linalg.inv(X.T @ X) @ X.T @ y

# Extract slope and intercept
slope = beta[1]
intercept = beta[0]

# Plot data and fitted line
plt.plot(x, y, 'o', label='Data')
plt.plot(x, slope * x + intercept, label='Fitted Line')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()

# Print the slope and intercept
print("Slope:", slope)
print("Intercept:", intercept)
```




```
Slope: 2.075510769337758
Intercept: 4.725644675780904
```

Gradient Descent

Imagine a hiker descending a mountain:

*They want to reach the lowest point (the valley).

*They can't see the entire path, but they can sense the slope under their feet.

*They take steps in the direction of steepest descent, gradually getting closer to the valley.

Gradient descent works similarly:

1. Start with an initial guess:
   Like the hiker starting at a random spot on the mountain, you begin with an initial set of parameters for your function.

2. Calculate the gradient:
   The gradient represents the direction of steepest ascent (like uphill). To descend, you take the negative of the gradient.

   It tells you how each parameter affects the overall function output.

3. Update the parameters:
   Adjust the parameters in the opposite direction of the gradient (downhill).

The amount of adjustment is controlled by the learning rate (like the hiker's step size).

4. Repeat until convergence:
   Keep calculating the gradient and updating parameters until you reach a point where the gradient is nearly zero (the valley, where the slope is flat).

Key points:

Loss function: The function you want to minimize, representing the error or difference between your model's predictions and the actual values.

Learning rate: Controls how much you adjust the parameters in each step. Too large might overshoot the minimum, too small might take too long to converge.

Iterations: The number of times you repeat the process.

Local minima: Gradient descent might get stuck in local minima (small valleys) instead of the global minimum (lowest point). Techniques like momentum and adaptive learning rates can help.

Common applications:

Linear regression: Finding the best-fitting line to a set of data points.
Logistic regression: Classifying data into two categories.
Neural networks: Training complex models with multiple layers.