Artificial Intelligence
Chapter 1 - Introduction to Machine Learning

Dr. Adnan Ismail Al-Sulaifanie

Department of Electrical and Computer Engineering
University of Duhok, Iraq

Outline

- ➤ Definition of machine learning and list its potential applications.
- ➤ Learning types – supervised, unsupervised, semi-supervised, and reinforcement.
- ➤ Artificial neuron – architecture, mathematical model and activation functions
- ➤ Model training – loss functions and evaluation metrics.

References

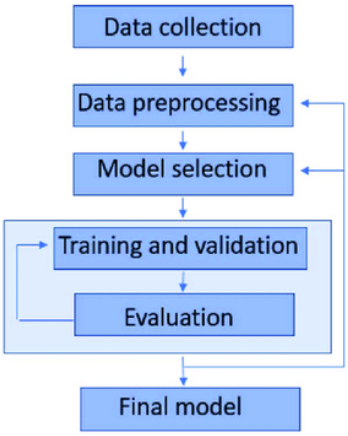
1. Andriy Burkov, he hundred-page machine learning book, 2019
2. Aurélien Géron, **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow**, 2019
3. Ethem Alpaydin **Introduction to Machine Learning**
4. Tom M. Mitchell, **Machine Learning**
5. Christopher M. Bishop, **Pattern Recognition and Machine Learning**
6. Chris Albon, **Machine Learning with Python Cookbook**, 2018
7. Ashwin Pajankar, **Hands-on Machine Learning with Python**, 2022
8. scikit-learn user guide
9. Sebastian Raschka, **Machine Learning with PyTorch and Scikit-Learn**, 2022
10. Max Kuhn and Kjell Johnson, **Feature Engineering and Selection – A Practical Approach for Predictive Models**, 2020

1 Introduction to Machine Learning

1. **Artificial intelligence (AI) –**
   - ➤ AI is a branch of Science which deals with helping machines find solutions to complex problems in human-like fashion. This generally involves borrowing characteristics from human intelligence, and applying them as algorithms in a computer.
   - ➤ It aims to create machines and systems that simulates human intelligence to perform tasks and make decisions through a set of rules/algorithms.
   - ➤ AI systems can learn, reason, make decisions, and perform complex tasks that.
   - ➤ It may or may not require large datasets, i.e., it can use predefined rules (Fuzzy inference).
2. **Machine learning (ML) –**
   - ➤ Machine learning is a field of artificial intelligence that focuses on developing algorithms and models that enable computers to learn from data and make predictions or decisions without being explicitly programmed.
3. **Deep Learning (DL) –**
   - ➤ DL is a branch of ML that uses artificial neural networks to learn complex patterns and representations from large amounts of data.
   - ➤ DL has achieved remarkable results in various domains, such as computer vision, natural language processing, speech recognition, ...

2 Machine Learning Types

- ➤ Machine learning systems are designed to learn patterns and insights from data. This data can come in various forms, such as images, text, numerical values, or a combination of these.
- ➤ ML components:
  1. Input data for learning
  2. Algorithm that learns from data
  3. Model parameters – adjusted by the model during learning
  4. Predictions – output generated by the model
- ➤ **Input data –** email message, image, audio signal, characteristics of an object such as person, house, ...
- ➤ **Model output –** real , integer, label, vector, ...
- ➤ ML workflow



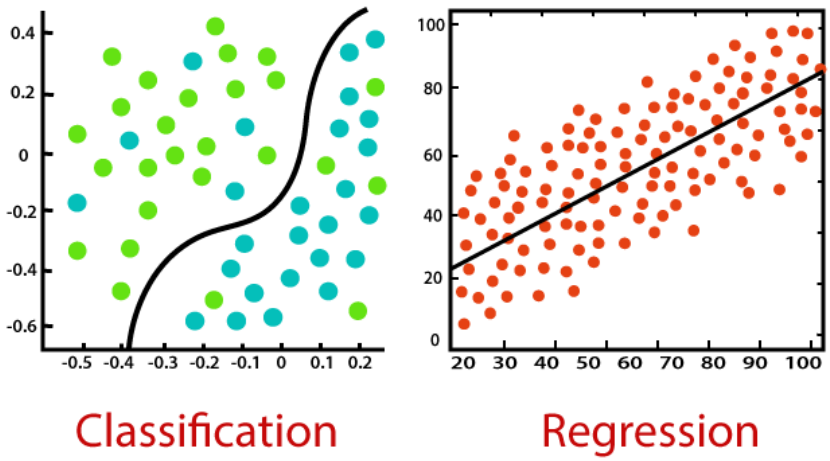
➤ There is four types of ML, depending on how the system work:

1. Supervised learning
2. Unsupervised learning
3. Semi-supervised learning
4. Reinforcement learning

2.1 Supervised ML

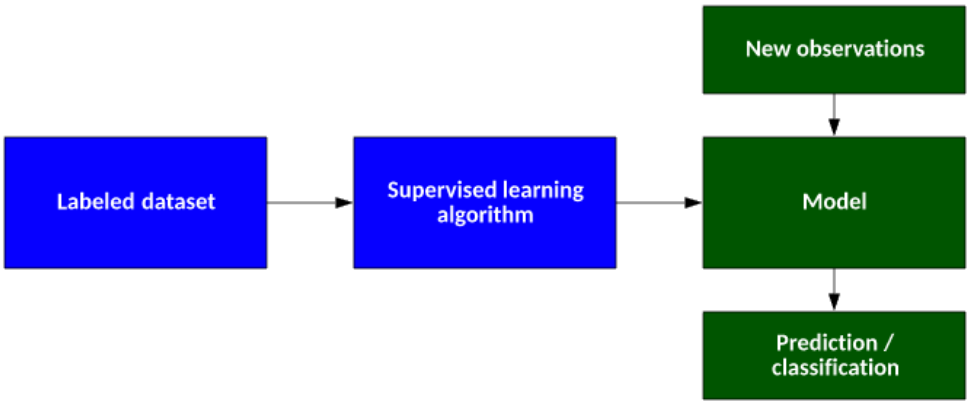
➤ Supervised learning is a type of machine learning where the algorithm is trained on a labeled dataset, where each data point has a corresponding label or output value.

➤ The algorithm learns to map the input data to the desired output, allowing it to make predictions for new, unseen data.



➤ The training process involves presenting the algorithm with a set of examples where both the input and the correct output are known.

➤ The algorithm adjusts its internal parameters based on the provided examples, attempting to minimize the difference between its predictions and the actual correct outputs.



➤ Supervised learning is classified into two categories of algorithms:

1. Regression – is the task of predicting a continuous (real) value, such as the price of a house.
2. Classification – is the task of assigning data points to predefined categories, such as ”red” or ”blue” , ”disease” or ”no disease’.

➤ Common supervised learning algorithms:

1. Linear Regression
2. Polynomial Regression
3. Support Vector Machine
4. Decision Tree
5. Random Forest
6. Logistic Regression
7. Naive Baye
8. ...

➤ Applications:

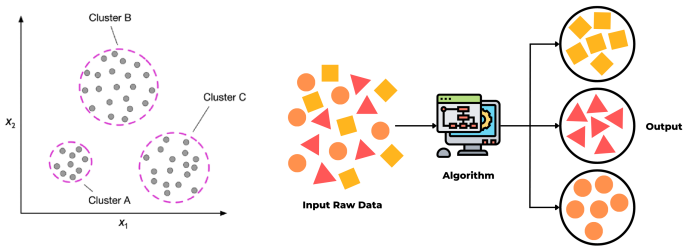
1. Image recognition and classification.
2. Recognize facial expressions (angry,joyful,...)
3. Speech recognition – identify speakers
4. Recognize sentiment in voices.
5. Automated medical diagnosis
6. Autonomous Vehicles
7. Email filtering and spam detection
8. Weather prediction
9. Anomaly detection
10. .....

➤ Limitations

- ➢ Supervised learning is slow as it requires human experts to manually label training examples one by one.
- ➢ It is costly as a model should be trained on large volumes of hand-labeled data to provide accurate predictions.

2.2 Unsupervised ML

- ➤ Unsupervised learning deals with unlabeled data. Dataset consists of input vectors without any target values.
- ➤ The primary goal of unsupervised learning is to find patterns, relationships, or structures within the data without any predefined categories or target values.
- ➤ Two common types of unsupervised learning are clustering and dimensionality reduction:
- ➤ Clustering is an algorithm applied to unlabeled dataset to categorize them into groups according to a similarity criteria.



- ➤ Advantages of unsupervised machine learning include:
  - ➢ Requires less data preparation (i.e., no hand labeling) than supervised machine learning.
  - ➢ Capable of finding unknown patterns in data, which is impossible with supervised machine learning models.
- ➤ Disadvantages of Unsupervised learning
  - ➢ Results may be unpredictable or difficult to understand. The user needs to spend time interpreting and label the classes which follow that classification.
  - ➢ Difficult to measure accuracy or effectiveness due to lack of predefined answers during training.
  - ➢ The results often have lesser accuracy.

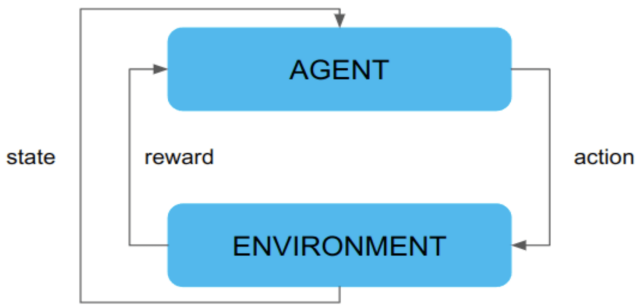
2.3 Semi-supervised ML

- ➤ Semi-supervised learning combines the characteristics of supervised learning and unsupervised learning.

- It is a method that uses a small amount of labeled data and a large amount of unlabeled data to train a model.
- Algorithm procedure:
    1. Labeled examples are used to build the model.
    2. Use clustering algorithm to assign unlabeled data to different groups .
    3. Use the labeled data to label groups (clusters)
    4. After labeling all examples, significant amount of labeled data is used to train machine learning model.
- Semi-supervised provides more accurate than unsupervised learning.

2.4 Reinforcement ML

- Reinforcement Learning (RL) is the science of decision making. It is about learning the optimal behavior in an environment to obtain maximum reward. It is a way of teaching a computer to do something by rewarding or punishing it based on its actions.
- RL enables software agents and machines to automatically evaluate the optimal behavior in an environment to improve its efficiency.
- The system has the following components:
    1. Environment
    2. Agent (AI system called function)
    3. Input (state)
    4. Output (action)
    5. Reward (performance metric)



- The computer (agent) has to learn by itself how to achieve a goal in a certain situation, without being told what to do. The goal is to learn policy function which is an optimal action to execute in each state.

- The agent receives state of the environment as a vector of features and execute actions in every state It tries different actions and gets feedback from the environment, which tells it how good or bad its actions are. The feedback is used to improve the future actions and get more rewards.

  Example – Robot learns how to walk.

- Applications:

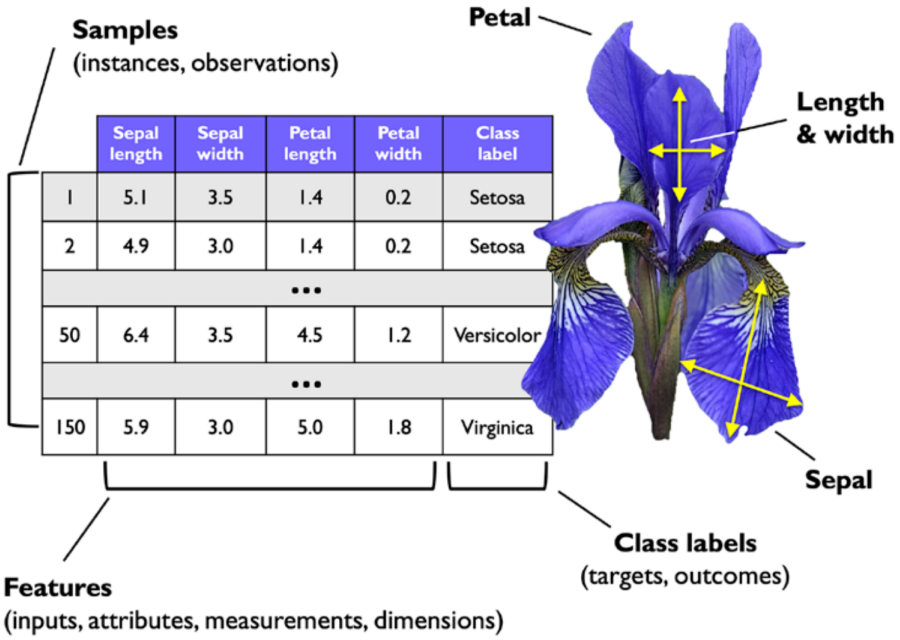
  1. Robotics
  2. Gaming like chess
  3. Self-driving vehicle, or a drone.
  4. .....

3 Dataset

- A dataset is a collection of data that can be used to train and test machine learning models (algorithms).
- The data in a dataset should be cleaned, organized and homogeneous so that they can be easily understood by a machine.
- Dataset is a collection of labeled or unlabeled examples.
- Each row is called feature vector (collection of features).
- The number of examples (vectors) in dataset should be at least 2 or 3 times larger than feature length (model parameters).
- Larger dataset results in better generalization.
- sklearn provides several datasets for different applications.

```
from sklearn.datasets import load_boston
                             load_breast_cancer
                             load_diabetes
                             load_digits
                             load_iris
                             fetch_california_housing
```

- Iris dataset is an example of supervised dataset



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
			...		
50	6.4	3.5	4.5	1.2	Versicolor
			...		
150	5.9	3.0	5.0	1.8	Virginica

➤ SKLEARN provides some functions to create pseudo dataset to evaluate performance of different ML algorithms:

```
from sklearn.datasets import   make_regression
                               make_classification
                               make_multilabel_classification
                               make_biclusters
                               make_blobs
                               make_circles
                               make_moons
                               make_sparse_spd_matrix
```

Example 1 - the following code show how to read data set.

```
from sklearn import datasets

iris = datasets.load_iris()  # Load Iris dataset
X = iris.data   # X = features
y = iris.target  # y = target
print('x-size = ',X.shape)  #  x-size =  (150, 4)
print('y-size = ',y.shape)  # y-size =  (150,)
print(X[0:5,:])
```

```
x-size =  (150, 4)
y-size =  (150,)
[[5.1  3.5  1.4  0.2]
 [4.9  3.   1.4  0.2]
 [4.7  3.2  1.3  0.2]
 [4.6  3.1  1.5  0.2]
 [5.   3.6  1.4  0.2]]
```

Homework 1 - modify the previous code to load other datasets and display their sizes.

Example 2 - write a python code to create regression dataset have 500 samples and 2 features.

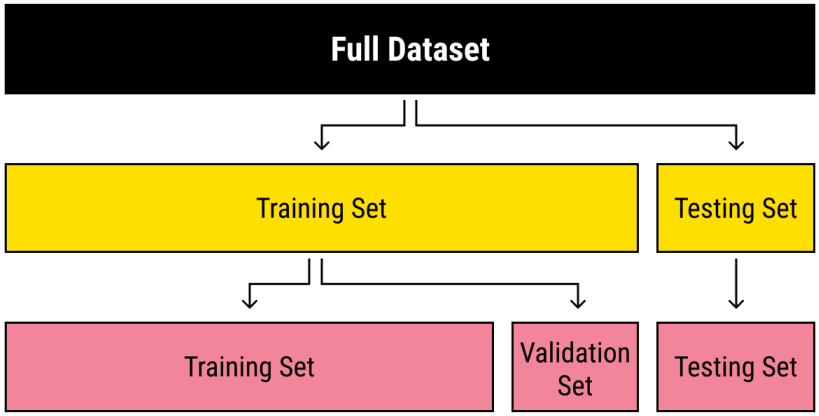
```
import pandas as pd
from sklearn import datasets, linear_model

X, y, _ = datasets.make_regression(n_samples=500, n_features=2, n_informative=1,
                                   noise=10, coef=True, random_state=0)

df = pd.DataFrame()
df['x1'] = pd.Series(X[:,0])
df['x2'] = pd.Series(X[:,1])
df['y'] = pd.Series(y)
print(df.head(7))
print(df.describe())
```

x1	x2	y
-2.255564	-1.022507	-199.604827
-0.739563	1.543015	-71.257642
0.324870	0.997118	18.324064
-0.451303	0.265688	-46.493412
-1.951804	-0.659892	-161.570066
0.498052	-0.026192	45.907832
-0.056133	-0.001385	-0.346197

	x1	x2	y
count	500.000000	500.000000	500.000000
mean	-0.065366	-0.025148	-5.287591
std	0.974521	1.000935	80.526480
min	-2.659172	-3.046143	-225.227850
25%	-0.726459	-0.680994	-61.599110
50%	-0.094417	-0.034669	-10.310688
75%	0.593690	0.609228	51.202090
max	2.696224	2.759355	221.234806

3.1 Training Set and Testing Set

Dataset is divided into two or three set.



1. **Training set –** is used to fit the models and learn from the data. The model adjusts its learnable parameters (weights) based on the patterns and information present in the training set.
2. **Validation set –** used to choose optimal value for hyperparameters. This set is used with cross validation process.
3. **Test set –** is reserved for evaluating the performance of the trained model. The testing set helps assess how well the model can make predictions on new, unseen data that it has not encountered during training. The testing set allows the model designer to estimate the model's performance on real-world data.

Example 3 -

```
import numpy as np
from sklearn.model_selection import train_test_split
X = np.arange(10).reshape((5, 2))
y = np.arange(5)

xtrain, xtest, ytrain, ytest = train_test_split(X, y,  test_size = 0.33, random_state = 42)
```

```
  X       y       xtrain     xtest
0 1       0        4 5        2 3
2 3       1        0 1        8 9
4 5       2        6 7
6 7       3
8 9       4
```

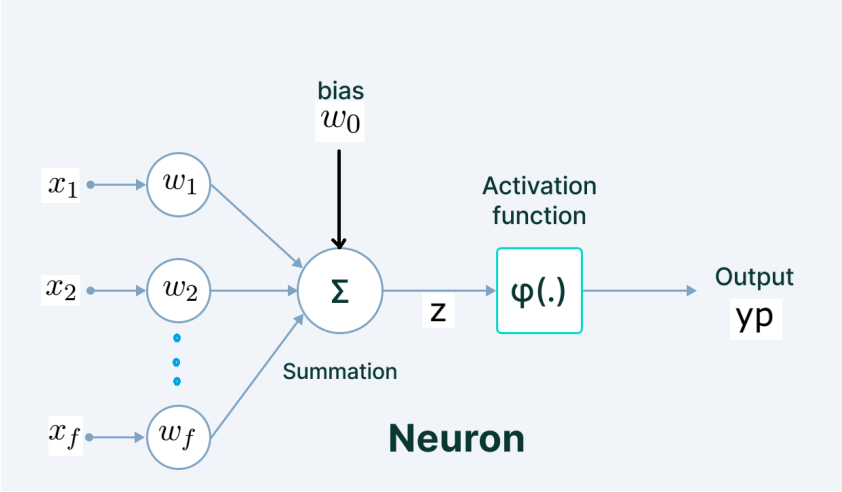
Example 4 -

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

X, y = load_iris(return_X_y = True)
xtrain, xtest, ytrain, ytest = train_test_split(X, y, test_size = 30, random_state = 42)
print('X-size = ' , X.shape)
print('xtrain-size = ' , xtrain.shape)
print('xtest-size = ' , xtest.shape)
```

```
X-size =  (150, 4)
xtrain-size =  (120, 4)
xtest-size =  (30, 4)
```

4 Artificial Neuron

- A neuron (also called perceptron) is the basic element in any artificial neural networks. It is the simplest cell in a neural network architecture.
- The neuron is used in supervised learning.
- The architecture of an artificial neuron is shown below.



$$z = W.X = w_1\, x_1 + w_2\, x_2 + \ldots + w_f\, x_f + b$$

$$yp = \varphi(z)$$

➤ How it work?

1. The neuron receives multiple input values (x_i). Each of these inputs is multiplied by its corresponding weight (w_i).
2. All the weighted inputs are added up, and the bias is added to this sum.
3. The result (z) is passed through the activation function (φ) to produce the neuron's output (yp).

➤ The bias allows the neuron to produce outputs other than zero, even if all the inputs are zero.

➤ The activation function introduces non-linearity to the model, enabling it to learn non linear relationship between input and output.

➤ An optimization algorithm is needed to adjust the value of w and b to minimize the error between the target value (y) and the predicted one (yp).

4.1 Gradient Descent (GD)

➤ In artificial neuron, the output is expressed as a weighted sum of the input plus the bias term.

$$yp = W.X = w_1\ x_1 + w_2\ x_2 + ... + w_f\ x_f + b$$

➤ It si required to minimize the error between y and yp. The error function (also called cost function) should be minimized.

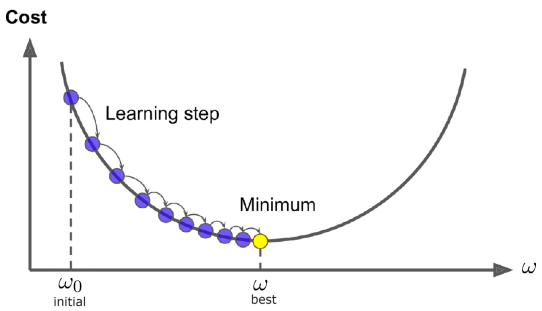
$$e = \frac{1}{N} \sum (y_i - (w.x_i + b))^2$$

➤ The primary objective is to optimize the weight values in order to achieve maximum performance, specifically by minimizing the previously defined error function.

➤ Gradient descent (GD) is an optimization algorithm that is used to minimize the error function.

➤ A gradient is a derivative of a function that has more than one input variable. It measures how much the output of a function changes if the input is changed a little bit. It is known as the slope of a function in mathematical terms

➤ GD works by iteratively updating the parameters of the model based on the gradient. The goal is to find the minimum value of the cost function, where the error is the smallest.



➤ Procedure:

1. Define a cost function to be minimized. It measures how well the model performs on the training data.

   $e = \frac{1}{N}\sum(y_i - (w.x_i + b))^2$

2. Initialize $w_0 = 0$ and $b_0 = 0$ and then iterate through the training examples.

3. Compute the gradient of the cost function with respect to each parameter. The gradient is a vector of partial derivatives that indicates the rate of change of the cost function with respect to each parameter.

   $\frac{\partial e}{\partial w} = \frac{1}{N}\sum -2x_i(y_i - (w.x_i + b))$

   $\frac{\partial e}{\partial b} = \frac{1}{N}\sum -2(y_i - (w.x_i + b))$

4. Parameters are updated by adjusting them in the opposite direction of the gradient to reduce the cost. This step involves multiplying the gradient by a small positive number called the learning rate (α) and subtracting the result from the current parameter values.

   $w_{n+1} = w_n - \alpha \frac{\partial e}{\partial w} = w_n - \alpha \frac{1}{N}\sum -2x_i(y_i - (w_n.x_i + b_n))$

   $b_{n+1} = b_n - \alpha \; ; \frac{\partial e}{\partial b} = b_n - \alpha \; ; \frac{1}{N}\sum -2(y_i - (w_n.x_i + b_n))$

   α is the learning rate determines the step size of each iteration and is crucial for the algorithm's convergence.

➤ One pass through all training examples is called an **epoch**.

➤ After each epoch, the parameters are updated.

➤ The process is repeated multiple epochs until we start seeing that the values for w and b don't change much; then we stop.

➤ The GD can be applied in three different ways:

1. **Batch Gradient Descent –** utilizes the complete dataset to compute the gradient in each iteration. However, it can become computationally expensive, particularly when dealing with large datasets.
2. **Stochastic Gradient Descent –** randomly selects a single data point to compute the gradient and update the parameters. While it is computationally less expensive, it can introduce noise into the optimization process.
3. **Mini-batch Gradient Descent –** a compromise between batch and stochastic gradient descent. It randomly selects a small batch of data points to compute the gradient and update the parameters.

➤ Each model has two type of parameters:

1. Learnable parameters (w and b) adjusted by the model during training process.

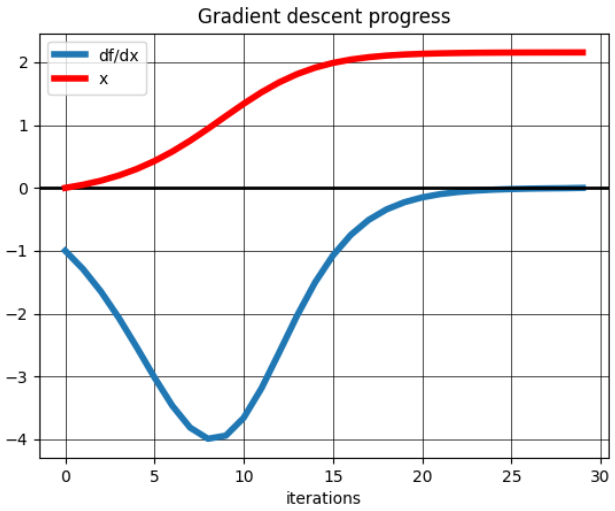
2. Hyper parameters (such as α) set by user. The optimal value can be chosen with cross validation process.

Example 5 - Assume the loss function $f(x) = x^3 - 3 * x^2 - x$, show the gradient descent can be applied to minimize $f(x)$ toward 0. Use α = 0.05

$$\acute{f}(x) = \frac{df}{dx} = 3 * x^2 - 6 * x - 1$$

$$x_{n+1} = x_n - \alpha * \acute{f}(x)$$

```
import numpy as np
import matplotlib.pyplot as plt
N = 30  # number of iterations
alpha = 0.05  # learning rate
x = np.zeros(N)
df = np.zeros(N)
for n in range(N-1):
    df[n] = 3 *x[n]**2 - 6 * x[n] - 1
    x[n+1] = x[n] - alpha * df[n]

plt.plot(df, linewidth = 4, label = 'df/dx')
plt.grid(color='k',  linewidth=0.5)
plt.axhline(linewidth=2, color='k')
plt.plot(x,'r',linewidth=4,label = 'x')
plt.title("Gradient descent progress")
plt.xlabel('iterations')
plt.legend(loc="upper left")
plt.show()
```



4.2 Cost and Loss Functions

- The loss function (also known as the objective function) measures the difference between the target value and predcited one. It serves as a gauge for how well the model can predict the desired result.
- Minimizing loss function aims to optimize the model's performance.
- **Difference between loss and cost functions –** cost function is calculated by taking the average of all values whereas we calculate the loss function for each sample output compared to its actual value.
- The choice of a loss function depends on the type of task (classification or regression) and the characteristics of the data.
- The common loss functions used for regression algorithms:

1. **Mean Squared Error (MSE) –** it is also referred to as L2 Loss. It measures the squared difference between predicted and actual values.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - yp_i)^2$$

Larger errors contribute more significantly than smaller errors, thus it is sensitive to outliers.

2. **Root Mean Squared Error (RMSE) –** takes the square root of MSE.

   RMSE = $\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - yp_i)^2}$

   The resulting values have the same units as the target variable, thus provides a meaningful and interpretable measure of the average prediction error.

3. **Mean Absolute Error (MAE) –** it is also referred to as L1 Loss. It measures the average of absolute differences between y and yp.

   MAE = $\frac{1}{N}\sum_{i=1}^{N}|y_i - yp_i|$

   The positive and negative errors don't cancel out each other. MAE is robust to outlier and final error is in the same units as the original target variable. Small errors are as important as large errors.

4. **Huber Loss –** is a hybrid of MSE and MAE. It incorporates a threshold parameter (δ) to strike a balance between the characteristics of both MSE and MAE.

   $$L_\delta(y, yp) = \frac{1}{N}\sum_{i=1}^{N} d_i$$

   $$d_i = \begin{cases} \frac{1}{2}(y_i - yp_i)^2 & |y - yp| \le \delta \\ \delta.(|y_i - yp_i| - \frac{\delta}{2}) & |y - yp| > \delta \end{cases}$$

   Huber loss combines the best properties of the mean squared error (L2 loss) and the mean absolute error (L1 loss). It is designed to be less sensitive to outliers than MSE loss.

   If |d| < δ, its behavior is similar to MSE, while when |d| > δ, its behavior is similar to MAE.

   A larger threshold value makes the loss function more tolerant to outliers, resembling MAE behavior for a wider range of errors.

   A smaller threshold value makes the loss function more sensitive to outliers, resembling MSE behavior over a larger range of errors.

➤ The common loss functions for classification algorithms:

1. **Binary Cross-Entropy –** commonly referred to as log loss, is widely used in binary classification problems.

   The output y is either **1** (positive class) or **0** (negative class), while the predicted output represents the probability of the example belonging to the positive class.

   BCE measures the difference between the predicted probabilities and the true binary labels.

   BCE = $\frac{-1}{N}\sum_{i=1}^{N}\left[(y_i * log(p_i)) + (1 - y_i) * log(1 - p_i)\right]$

Log value is used because it offers less penalty for small differences between predicted probability and corrected probability. When the difference is large the penalty will be higher.

Example 6 - explain how to use BCE in binary classification.

y = [1, 0, 1]

p = [0.9, 0.1, 0.8]

BCE = $\frac{-1}{3}$ [1 log(0.9) + (1−0) log(1−0.1) + 1 log(0.8)] = 0.324

2. **Categorical Cross-Entropy Loss (CCE) –** is an extension of Binary Cross Entropy loss to multi-class classification tasks. It measures the difference between the predicted probabilities and the true labels for each class.

   It penalizes the model for assigning low probabilities to the correct class and high probabilities to the incorrect classes.

   CCE = $\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{i,j} * ln(p_{i,j})$

   The CCE loss is widely used in multi-label classification in conjunction with a softmax activation function in the output layer.

3. **Hinge Loss –** is also known as max-margin loss or large-margin loss. It is often used in binary classification problems, where y = ∓1.

   L = $max(0, 1 - y * yp)$

   or

   $$L = \begin{cases} 1 - z & z \leq 1 \\ 0 & z > 1 \end{cases}$$

   When yp and y have the same sign meaning yp predicts the right class and the hing loss L(y,yp) = 0. When they have opposite signs, the loss increases linearly with the difference.

   Hinge loss penalizes wrong predictions and does not do so for the right predictions.

4.3 Activation Functions

➤ An activation function is a mathematical operation applied to the weighted sum (z) in order to determine the output of a neuron.

➤ It is usually has a squashing effect. It bounds the range of z between a given range, e.g., 0 to 1 or −1 to 1.

➤ Activation functions play a crucial role in introducing non-linearity to the network. It converts these linear inputs to non-linear outputs, which enables the model to learn more complex relationships between inputs and outputs.

➤ There are several activation functions available, each has its own characteristics and.

➤ A typical activation should

➢ Have bounded range

➢ Be differentiable and continuous everywhere.

➢ Should produce one-to-one mapping

➤ Common activation functions:

1. **Linear –** is one of the simplest activation functions.

   $f(z) = z$

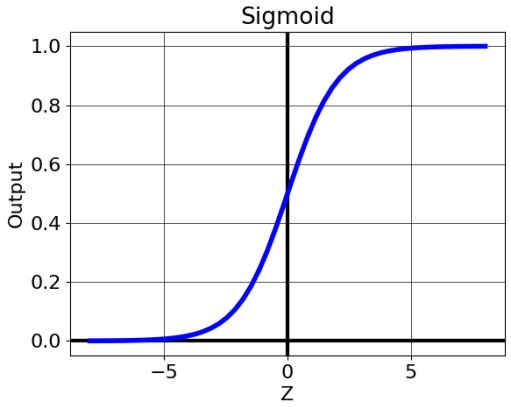
   This function doesn't help in adding non-linearity to the neural networks. It is used in simple regression algorithms.

2. **Threshold (sign, step) –** produce output based on the sign of z

   $f(z) = \begin{cases} +1 & z \leq 0 \\ -1 & z < 0 \end{cases}$ It is not differentiable, therefore, the sign activation is rarely used in neural networks.

3. **Sigmoid –** tt maps input value into a range between 0 and 1.

   $f(z) = \frac{1}{1 + e^{-z}}$



   This function helps in adding non-linearity, and it is often used as the activation function for the output layer of a binary classification neural network. It is one of the popular non-linear functions.

   Example 7 - explain how to use sigmoid activation function in binary classification.

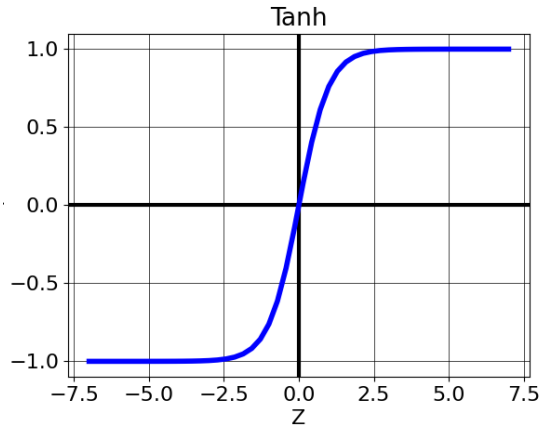
   z = 0.5

   f(z) = 0.622

   Since f(z) $\geq$ 0.5, yp = 1

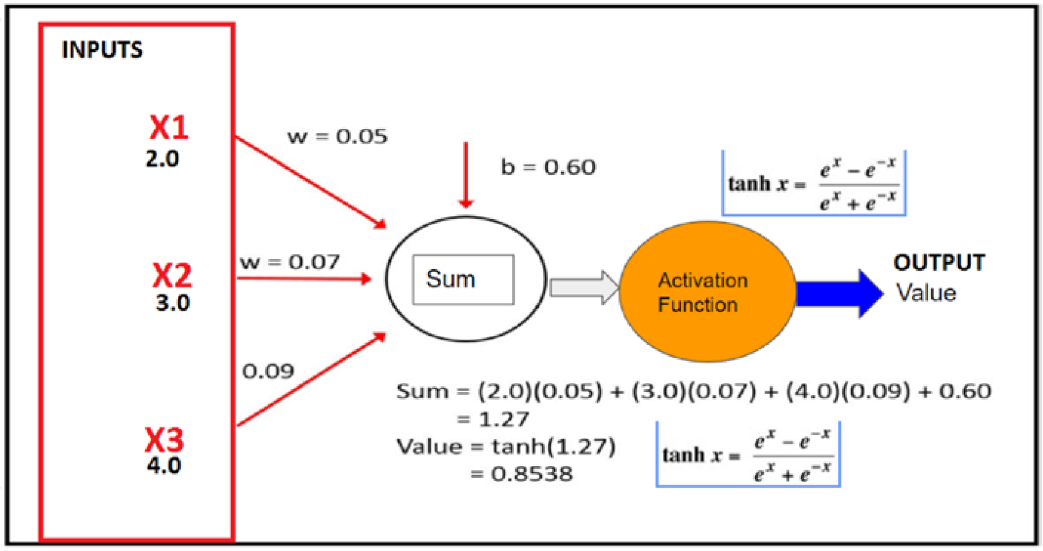
4. **Tanh (hyperbolic tangent) –** maps input values into a range between -1 and 1.

   $f(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$



The advantage of this is that negative values of the weighted sum are not forgotten by setting them to zero, but are given a negative weight. In practice tanh makes the network learn much faster than sigmoid.

Example 8 -



5. **Softmax –** is an extension sigmoid function to multidimensional data. It is used for multi-label classification.

$$f(z) = \frac{e^{z_i}}{\sum_{i=1}^{N_c} e^{z_i}}$$

N_c number of classes

The softmax function ensures that the predicted probabilities sum up to 1, representing a valid probability distribution over the classes.

Example 9 -

z = { 5, 1, 0, 2}

e^z = {148, 2.72, 1, 7.4}

$\sum_{i=1}^{4} e^{z_i}$ = 148 + 2.72 + 1 + 7.4 = 159.12

$yp = \frac{e^{z_i}}{\sum e^{z_i}}$ = {0.93, 0.02, 0.006, 0.047}

The probability that input belongs to class 1 is 93%

5 Evaluation Metrics

- ML algorithms can be divided into two groups based on their objectives – regression and classification algorithms.
- There are different evaluation metrics for each type of algorithms.
- The common metrics for regression algorithms: MSE, MAE, and maximum error.
- The common metrics for classification algorithms: confusion_matrix, classification_report, acuracy, precision, ...

5.1 Evaluation Metrics - Regression

1. **Mean Square Error (MSE) –** the square of the difference between the predictions and the targets.

   MSE = $\frac{1}{N_{test}} \sum (y - yp)^2$

2. **Mean Absolute Error (MAE) –** the average absolute error between actual and predicted.

   MAE = $\frac{1}{N_{test}} \sum |y - yp|$

3. **Maximum error –** the maximum error between the target and predicted.

   Max-error(y,yp) = max(|y - yp|)

   Example 10 - the Python code below demonstrates how to utilize various metrics to assess the performance of a regression algorithm

```
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import max_error
import numpy as np

y = np.array([3.0, -0.5, 2, 7, 9, 13, 4])
yp = np.array([2.5, 0.0, 2, 8, 9, 13.5, 4.5])
print('y-yp = ', y - yp)
MAE = mean_absolute_error(y, yp)
MSE = mean_squared_error(y,yp)
MAXE = max_error(y, yp)
print('MAE = ', MAE)
print('MSE = ', MSE )
print('MAX-ERROR = ', MAXE)
```

```
y-yp = [0.5  -0.5  0.  -1.  0.  -0.5  -0.5]
MAE =  0.4286
MSE =  0.2857
MAX-ERROR =  1.0
```

Homework 2 - for the previous example, compute the metrics by the hand.

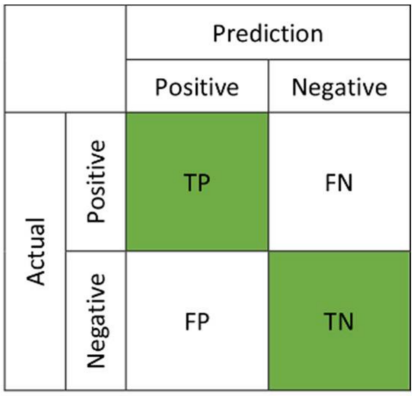
Homework 3 - what is r2-score?

5.2 Evaluation Metrics - Classification

5.2.1 Confusion Matrix

➤ Confusion matrix summarizes the performance of a machine learning model on a set of test data.

➤ The matrix displays number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).



	Original (y_i)	**Predicted (yp_i)**
TP	Positive	Positive
TN	Negative	Negative
FP	Negative	Positive
FN	Positive	Negative

➤ The values of confusion matrix allow you to calculate different performance metrics, as shown below:

1. **Accuracy** measures the number of correct predictions made by a model in relation to the total number of predictions made.

   $$\text{Accuracy} = \frac{\text{correct predictions}}{\text{total predictions}} = \frac{TP + TN}{TP + TN + FP + FN}$$

   When any model gives an accuracy rate of 99%, you might think that model is performing very good but this is not always true and can be misleading in some situations. While accuracy is a common metric, it may not be suitable for imbalanced datasets.

2. **Precision** is a measure of the accuracy of positive predictions made by a model.

   $$\text{Precision} = \frac{\text{correct positive predictions}}{\text{total positive predictions}} = \frac{TP}{TP + FP}$$

   It answers the question: "Of all the instances predicted as positive, how many were actually positive?"

   Precision measures how accurate the model is when it predicts the positive class. A high precision means that the model has a low rate of false positives, and thus minimizes the cost of wrongly labeling a negative instance as positive.

3. **Recall (sensitivity)** is a measure of the ability of a model to capture all the positive instances.

   $$\text{Recall} = \frac{\text{correct positive predictions}}{\text{total actual positives}} = \frac{TP}{TP + FN}$$

   It answers the question: "Of all the actual positive instances, how many did the model correctly predict as positive?"

   Recall measures how well the model can find the positive instances. A high recall means that the model has a low rate of FN, and thus maximizes the benefit of correctly labeling a positive instance as positive.

4. **F1-score** is used to evaluate the overall performance of a classification model. It is the harmonic mean of precision and recall,

   $$\text{F1-score} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

➤ The importance of precision and recall depends on the context and the objective of the classification problem.

➤ In some cases, we might want to prioritize precision over recall, or vice versa.

If we want to filter out spam emails, we might prefer a high precision model that avoids labeling legitimate emails as spam, even if it misses some spam emails.

If we want to detect cancer patients, we might prefer a high recall model that identifies as many positive cases as possible, even if it produces some false alarms.

Example 11 - confusion matrix

```
from sklearn import metrics

y  = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
yp = [0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1]
cm = metrics.confusion_matrix(y, yp)
print(metrics.accuracy_score(y, yp))  # 0.7
print(metrics.precision_score(y, yp))  # 0.667
print(metrics.recall_score(y, yp))  # 0.8
print(metrics.f1_score(y, yp))  # 0.727
print(cm)
 [[6 4]
  [2 8]]
```

Homework 4 - repeat previous example manually

5.2.2 Classification Report

➤ A classification report is a tool used in machine learning to evaluate the performance of a classification model.

- The classification report is particularly useful when dealing with problems such as binary or multi-class classification.
- It provides a summary of various metrics that help assess how well the model is performing in terms of classifying instances into different categories or classes.
- Example

```
from sklearn.metrics import classification_report

y  = [0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1]
yp = [0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1]
report = classification_report(y, yp)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.60      0.75      0.67         8
           1       0.80      0.67      0.73        12

    accuracy                           0.70        20
   macro avg       0.70      0.71      0.70        20
weighted avg       0.72      0.70      0.70        20
```

- The report shows the precision, recall, and F1 score for each class, as well as the support (number of samples) for each class.
- Macro average is unweighted average for all classes.
- A weighted average takes into account the number of samples in each class to evaluate the average performance accordingly.
- Confusion matrix versus classification report
  - The confusion matrix shows the actual and predicted counts of each class, while a classification report shows the calculated metrics of each class.
  - A confusion matrix is more visual and intuitive, while a classification report is more numerical and analytical.
  - A confusion matrix can help you identify the types and sources of errors your model makes, while a classification report can help you evaluate the quality and reliability of your model.

6 Overfitting and Underfitting

- Overfitting and underfitting are two common problems that affect the performance and generalization of machine learning models.
- They are related to the concepts of bias and variance, which measure the prediction errors of a model.

- **Bias** refers to the error produced during the training process. If the error is large (high bias), it means that the model is unable to represent the true relationship between input and output accurately. This leads to underfitting, which means the model performs poorly on both the training and the testing data.
- **Variance** is the variability of the model's performance when the model is tested on different dataset. A model with high variance learns too much from the noise and fluctuations in the data. This leads to overfitting, which means the model performs well on the training data but poorly on the testing data.
- To achieve a good balance between bias and variance, we need to choose a model that is neither too simple nor too complex for the data.

6.1 Overfitting

- Overfitting occurs when a ML algorithm learns the training data too well, while it is unable to generalize to new data. This can happen when the algorithm is too complex or when it is trained for too long.

  Imagine a student who memorizes every question and answer on a practice exam before the real test. While they achieve perfect scores on the practice, they might struggle on the actual test with different questions. This is analogous to overfitting in machine learning.
- There is a gap between training error and test error, the gap is too large.
- Causes of overfitting:
  - The algorithm is too complex.
  - Not enough training data.
- Possible solutions:
  - Use a simpler model.
  - Train the algorithm for less time.
  - Increase the size of the training data.
  - Use regularization techniques to penalize the algorithm for learning the noise in the data.
  - Use feature selection techniques to select the most informative features.

6.2 Underfitting

- Underfitting happens when a model is not good enough to understand all the details in the data. This results in a model that performs poorly on both the training and testing data. In other words, the model has not learned the training data well enough and is not able to generalize to new data.

➤ Causes of underfitting:

- ➢ The algorithm is too simple.
- ➢ The algorithm is not trained for long enough.
- ➢ The training data is too small.
- ➢ The features in the training data are not informative enough.

➤ Possible solutions:

- ➢ Use a more complex algorithm.
- ➢ Train the algorithm for longer.
- ➢ Increase the size of the training data.
- ➢ Use feature selection techniques to select the most informative features.

	Overfitting	**Underfitting**
Performance on training data	Very good	Poor
Performance on new data	Poor	Poor
Sensitivity to changes in training data	Sensitive	Insensitive
Causes	Algorithm too complex training too long training data too small correlated features	Algorithm too simple training too short training data too small uninformative features
Prevention	Use simpler algorithm train less increase training data use regularization use feature selection	Use more complex algorithm train longer increase training data use feature engineering