CHAPTER 4: MACHINE LEARNING

Subject: Introduction to data science

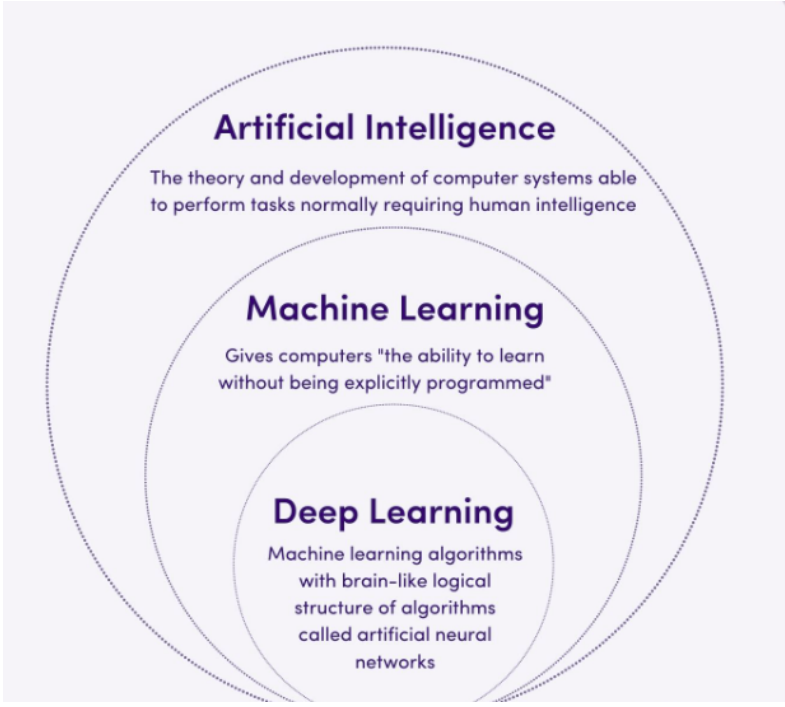
Instructor: Đinh Xuân Trường

Posts and Telecommunications Institute of Technology
Faculty of Information Technology 1

Hanoi, 2024

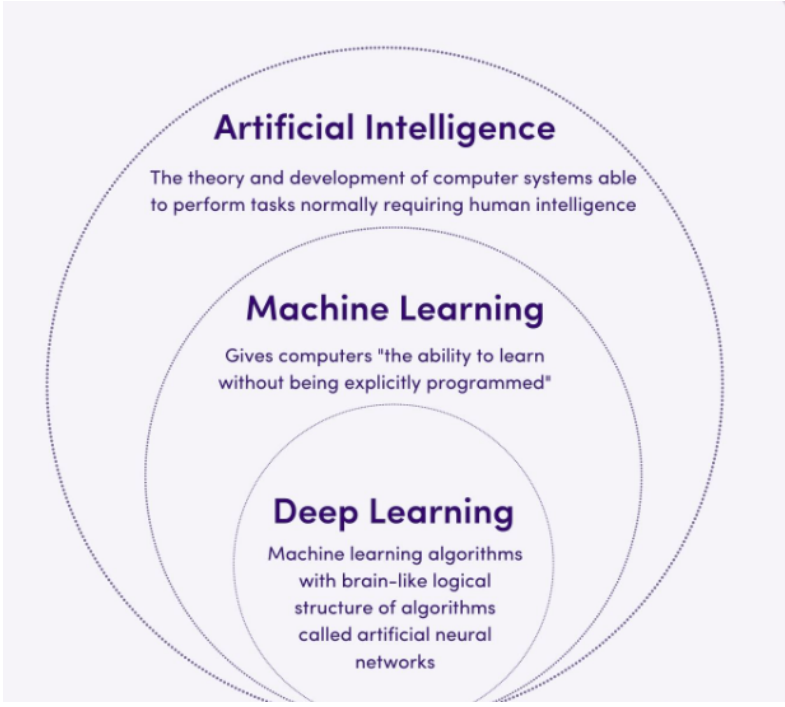
http://www.ptit.edu.vn

What is Machine Learning?



Machine Learning, often abbreviated as ML, is a subset of artificial intelligence (AI) that focuses on the development of computer algorithms that improve automatically through experience and by the use of data.

What is Machine Learning?



Machine learning enables computers to learn from data and make decisions or predictions without being explicitly programmed to do so.

Types of Machine Learning

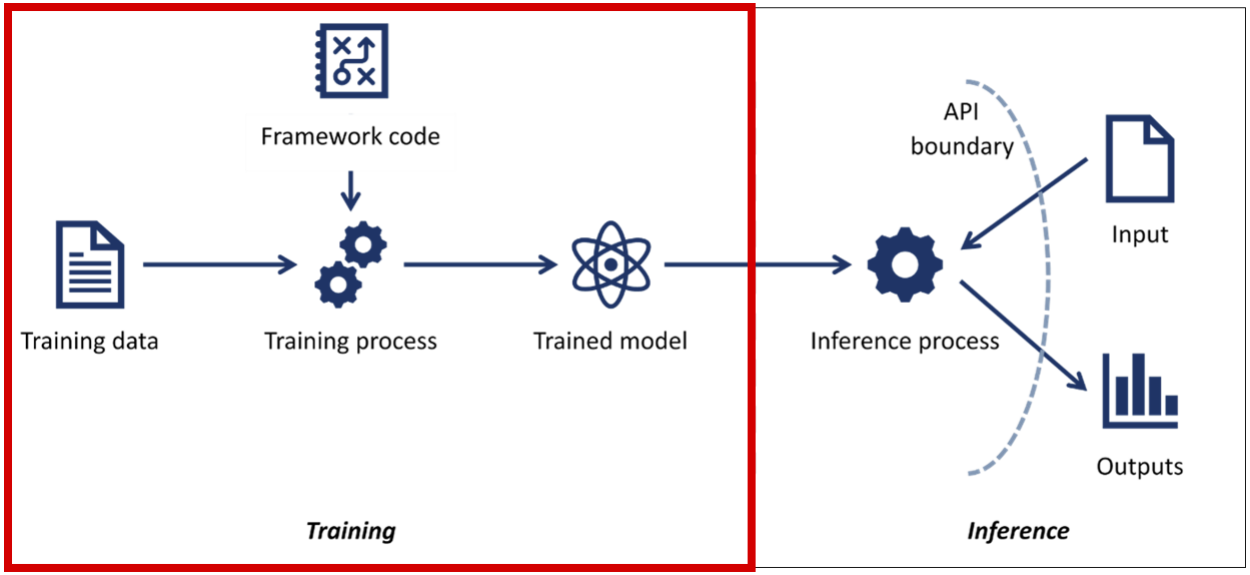
Machine learning can be broadly classified into four types based on the nature of the learning system and the data available:

- Supervised learning,
- Unsupervised learning,
- Semi-Supervised Learning,
- Reinforcement Learning.

Content

Content

Training and inference



Machine learning works in two main phases: training and inference. In the training phase, a developer feeds their model a curated dataset to "learn" everything it needs about the data type it will analyze. Then, in the inference phase, the model can make predictions based on live data to produce actionable results.

Machine Learning Inference

Machine learning inference is the ability of a system to make predictions from novel data. This can be helpful if you need to process large amounts of newly accumulated information.

Example

Suppose that you're teaching a child to sort different fruits by color. You might first show them a tomato, an apple, and a cherry to learn that those fruits are red. Later, when you show them a strawberry for the first time, they'll be able to infer that that fruit is red, too since it's similar in color to the apple, tomato, and cherry.

Content

Model Evaluation

Model evaluation is the process of using different evaluation metrics to understand a machine learning model's performance, as well as its strengths and weaknesses. Model evaluation is important to assess the efficacy of a model during initial research phases, and it also plays a role in model monitoring.

How to Evaluate Classification Models

The most popular metrics for measuring classification performance include accuracy, precision, recall, F1-Score, ...
When performing a classification problem, there are 4 cases of possible predictions:

- True Positive (TP): predicted positive, the real value was positive
- True Negative (TN): predicted negative, the real value was negative
- False Positive (FP): predicted positive, the real value was negative – Type I Error
- False Negative (FN): predicted negative, the real value was positive – Type II Error

How to Evaluate Classification Models

Example

Classify 1100 images as cats or not, in the prediction data there are 100 images that are cats, 1000 images that are not cats (non-cat). Here, the prediction result is as follows

- Among 100 cat images (cat is "positive" and non-cat is "negative"):
  - 90 images predicted to be cat, called True Positive,
  - 10 images predicted to be non-cat are called False Negative
- Among 1000 non-cat images:
  - 940 images are predicted to be non-cat, called True Negative,
  - 60 images predicted to be cat are called False Positive

Confusion matrix

		Predicted Class: Positive	Predicted Class: Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

Accuracy

Accuracy measures how often the classifier makes the correct predictions, as it is the ratio between the number of correct predictions and the total number of predictions.

$$\text{Accuracy} = \frac{\text{correct predictions}}{\text{all predictions}}$$

That is,

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}.$$

Accuracy

Example

Applying to the Cat/Non-cat problem above, calculate Accuracy?

Accuracy

Example

Applying to the Cat/Non-cat problem above, calculate Accuracy?

$$\text{Accuracy} = \frac{90 + 940}{1000 + 100} = 93.6\%$$

Accuracy

Example

Applying to the Cat/Non-cat problem above, calculate Accuracy?

$$\text{Accuracy} = \frac{90 + 940}{1000 + 100} = 93.6\%$$

The disadvantage of this evaluation method is that it only tells us what percentage of data is correctly classified without specifying how each type is classified. There will be many cases where the Accuracy measure does not accurately reflect the performance of the model.
Assuming the model predicts all 1100 images as Non-cat, Accuracy still reaches $1000/1100 = 90.9\%$.

Precision

Precision measures the proportion of predicted Positives that are truly Positive. Precision is a good choice of evaluation metrics when you want to be very sure of your prediction.

$$\text{Precision} = \frac{TP}{TP + FP}.$$

Example

Applying to the Cat/Non-cat problem above, calculate Precision(cat) and Precision(non-cat)?

Precision

Precision measures the proportion of predicted Positives that are truly Positive. Precision is a good choice of evaluation metrics when you want to be very sure of your prediction.

$$\text{Precision} = \frac{TP}{TP + FP}.$$

Example

Applying to the Cat/Non-cat problem above, calculate Precision(cat) and Precision(non-cat)?

$$\text{Precision(cat)} = \frac{90}{90 + 60} = 60\%,$$

$$\text{Precision(non-cat)} = \frac{940}{940 + 10} = 98.9\%.$$

Recall

Recall is also an important metric, it measures the rate of accurate prediction of positive cases across all samples in the positive group. The formula for Recall is as follows:

$$\text{Recall} = \frac{TP}{TP + FN}.$$

Example

Applying to the Cat/Non-cat problem above, calculate Recall(cat) and Recall(non-cat)?

Recall

Recall is also an important metric, it measures the rate of accurate prediction of positive cases across all samples in the positive group. The formula for Recall is as follows:

$$\text{Recall} = \frac{TP}{TP + FN}.$$

Example

Applying to the Cat/Non-cat problem above, calculate Recall(cat) and Recall(non-cat)?

$$\text{Recall(cat)} = \frac{90}{90 + 10} = 90\%,$$

$$\text{Recall(non-cat)} = \frac{940}{940 + 60} = 94\%.$$

High recall means high True Positive Rate, meaning the rate of missing truly positive points is low

F-score

Precision and Recall are useful in cases where the classes are not evenly distributed. Therefore, both Precision and Recall of a model must be evaluated to draw accurate conclusions. To combine Precision and Recall, we can calculate the F-score.

$$F_\beta = (1 + \beta^2) \frac{\text{Precision} * \text{Recall}}{(\beta^2 * \text{Precision}) + \text{Recall}}$$

The β parameter allows us to control the balance between Precision and Recall.

- $\beta < 1$ focuses more on Precision.
- $\beta > 1$ focuses more on Recall.
- $\beta = 1$ focuses on both Precision and Recall.

F1-score

When $\beta = 1$, we use F1-score, which is the harmonic expectation of Precision and Recall. F1-score is large when both Precision and Recall values are large. On the contrary, just a small value will make the F1-Score small.

$$F1 = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

The larger the F1-Score, the better. Ideally, F1-score = 1 (when Recall = Precision=1).

Example

Apply to Cat/Non-cat problem, calculate F1-Score and give comments?

How to Evaluate a Regression Model

- The metrics for a regression model are quite different from the metrics for a classification model because it must predict over a continuous range instead of a number of discrete classes.
- For example, building a model that predicts the price of a house is 2 billion VND but it sells for 2.1 billion VND, then that is considered a good model, while the classification problem only cares about See if that house can be sold for 2 billion or not.

Mean squared error (MSE)

Mean squared error is defined MSE is defined as the average sum of squared errors between the predicted output and the actual result.

$$MSE = \frac{1}{n}\sum_{1}^{n}(y_i - \hat{y}_i)^2.$$

MSE has a range of values from $[0, +\infty]$. On the same data set, the smaller the MSE, the higher the accuracy. However, because the error is squared, the unit of MSE is different from the unit of the predicted result.

Mean squared error (MSE)

```
1
2  from sklearn.metrics import mean_squared_error
3  # sklearn có thư viện giúp tính RMSE một cách dễ dàng
4  # tương tự như trên, y_true là vector lưu kết quả chính xác
5  #                    y_pred là vector lưu dự đoán
6  y_true = [3, -0.5, 2, 7]
7  y_pred = [2.5, 0.0, 2, 8]
8  mean_squared_error(y_true, y_pred)
```

Mean Absolute Error (MAE)

Mean Absolute Error (MAE) is a measure to evaluate regression models. MAE is defined as the average of the total absolute error between the predicted output and the actual result:

$$MAE = \frac{1}{n}\sum_{1}^{n}|y_i - \hat{y}_i|.$$

MAE has a range of values from $[0, +\infty]$. On the same data set, the smaller the MAE, the higher the accuracy.

Mean Absolute Error (MAE)

```
1
2  from sklearn.metrics import mean_absolute_error #gọi thư viện để tính MAE
3  #giả sử y_true là vector lưu kết quả chính xác
4  #       y_pred là vector lưu dự đoán
5  y_true = [3, -0.5, 2, 7]
6  y_pred = [2.5, 0.0, 2, 8]
7  mean_absolute_error(y_true, y_pred)
```

Content

Bias vs Variance

* The capacity of classification and prediction models in the class of supervised learning models of machine learning is often expressed through two aspects of **bias and variance**.
* Understanding the exact meaning of these two concepts helps us create models that are less biased and have uniform accuracy on both the training set and test set and at the same time have the ability to the ability to apply the model into practice without worrying about errors that may arise.

Bias

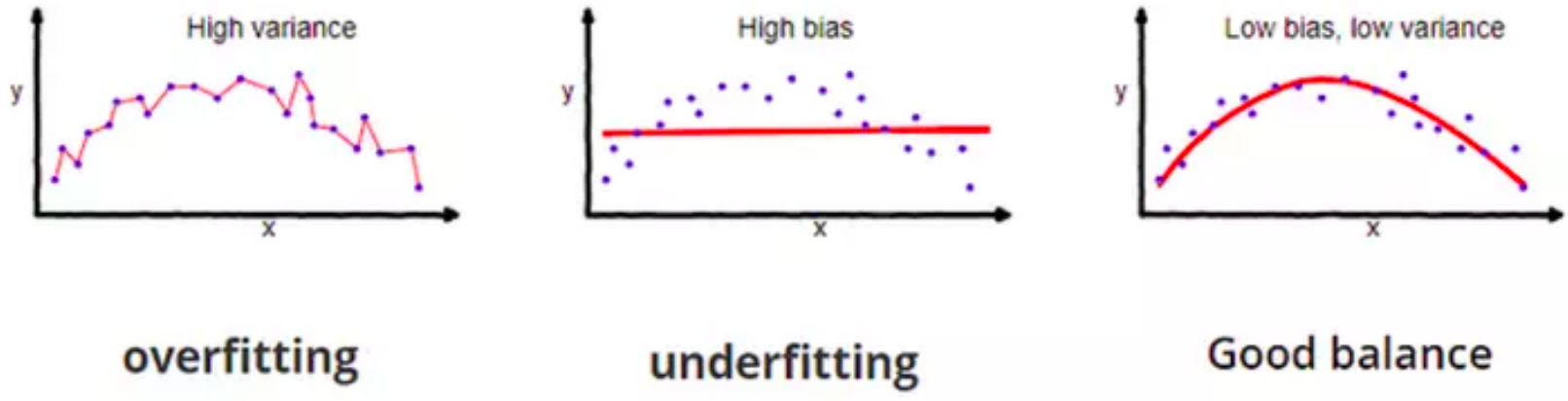
- **Bias**: is the error between the average predicted value of the model and the actual value. When building the model we want to create low deviation. That is, the predicted value will be closer to the ground truth.
- **High bias**: large error, simple model, but the prediction accuracy is not high
- **Low bias**: small error, complex model, good prediction results.

Variance

- **Variance**: is the error that represents the "sensitivity"of the model to fluctuations in the training data. Models with high variance often perform very well on the training data set, but do not give positive results on the test data set.
- **Low-variance**: model with little variation according to changes in training data
- **High-variance**: strong variation model, closely following the changes of training data.

Content

Overfitting vs Underfitting



Underfitting can be seen as a model that is "learning poorly", while Overfitting indicates that the model is "rote learning".

Overfitting và Underfitting

- **Underfitting:** is the phenomenon where the model has high bias and low varriance, giving poor prediction results on both the training set and the testing set. Underfitting is often easily detected because it gives bad results on the training set.
- **Overfitting**: is the phenomenon where the model has low bias and high variance, at which point the model becomes complex, closely following the training data. The model gives very good results on data that has been learned, but gives bad results on data that has never been seen before. This problem occurs when the model tries to fit all training data points, including noise.

Content

Feature transformation and feature extraction

* Feature transformations are techniques that help transform input data into data suitable for the research model (See Section 4, Chapter 2).
* Not all the information provided by a predictor variable is of complete value in classification. Therefore we need to extract the main information from that variable.

Content

Text Characteristics

* Text data can come from many sources and many different formats (lowercase characters, uppercase characters, special characters,...). There are many data processing methods suitable for each specific topic. The lecture will introduce the most popular method.
* Since text is characters, how can we quantify characters? Tokenization techniques will help us do this. Encoding simply means we divide paragraphs into sentences, sentences into words.
* In encryption, the word is the basic unit. We need a tokenizer whose size is equal to all the words that appear in the text or to all the words in the dictionary.

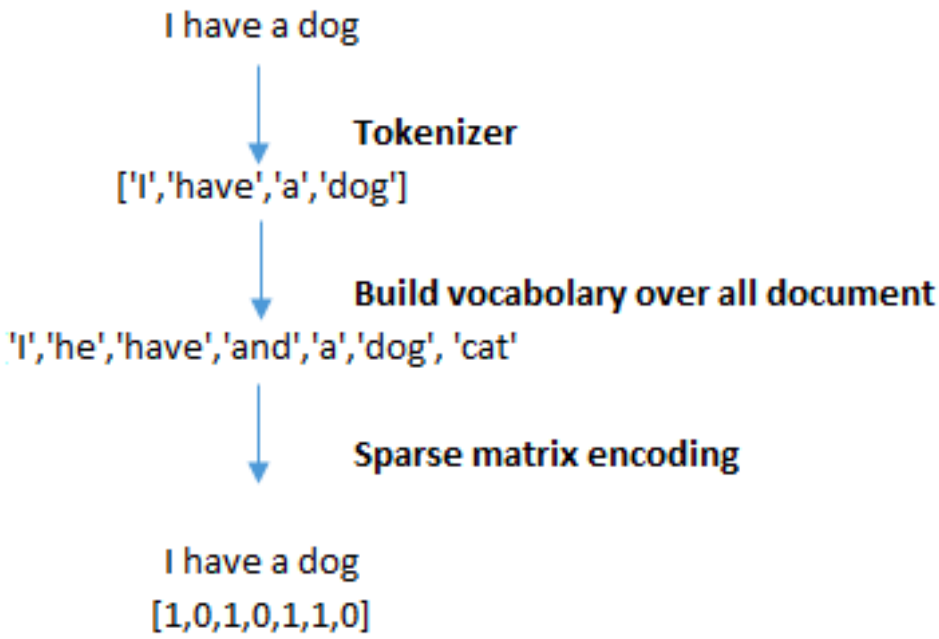
Text Characteristics

* A sentence will be represented by a sparse vector where each element represents a word, its value is 0 or 1 corresponding to the word not appearing or appearing.
* Tokernizers will be different for each different language.
* A sentence will be represented by a sparse vector where each element represents a word, its value is 0 or 1 corresponding to the word not appearing or appearing.
* Use the "bags of words"method to create a vector whose length is equal to the length of the tokenizer and each element of the bag of words will count the number of occurrences of a word in the sentence and sort Arrange them in a suitable position in the vector.

```python
from functools import reduce
import numpy as np

# Đầu vào là một texts bao gồm 3 câu văn:
texts = [['i', 'have', 'a', 'cat'],
        ['he', 'has', 'a', 'dog'],
        ['he', 'has', 'a', 'dog', 'and', 'i', 'have', 'a', 'cat']]

# B1: Xây dựng từ điển
dictionary = list(enumerate(set(reduce(lambda x, y: x + y, texts))))

# B2: Mã hoá câu sang véc tơ tần suất
def bag_of_word(sentence):
    # Khởi tạo một vector có độ dài bằng với từ điển.
    vector = np.zeros(len(dictionary))
    # Đếm các từ trong một câu xuất hiện trong từ điển.
    for i, word in dictionary:
        count = 0
        # Đếm số từ xuất hiện trong một câu.
        for w in sentence:
            if w == word:
                count += 1
        vector[i] = count
    return vector

for i in texts:
    print(bag_of_word(i))
```

Bags of words

The above process can be described by the diagram below:



Bags of words

* For practical applications, the dictionary can be very large, so the resulting feature vector will be very long. There are many words in the dictionary that do not appear in a text. Thus, the obtained feature vectors often have many zero elements (sparse vector).
* The Bags of words representation has the limitation that it cannot distinguish between two sentences with the same words because Bags of words does not distinguish the before or after order of words in a sentence.
* For example: "you have no dogs"and "no, you have dogs"are two sentences that have the same performance even though they have opposite meanings.

$\Longrightarrow$ **The bag-of-n-gram method will be used to fix it.**

Bag-of-n-gram

* The bag-of-n-grams method is an extension of bag-of-words. An n-grams is a string consisting of n tokens.
* In case $n = 1$ words are called unigrams, for $n = 2$ words are bigrams and $n = 3$ words are trigrams.

Bag-of-n-gram

Example

* We have the sentence "I love machine learning",
* the unigram of this sentence will be ["I", "love","machine", "learning"],
* and the bigram of this sentence will be ["I love", "love machine", "machine learning"].

Phương pháp bag-of-n-gram

* In sklearn, to use bigram, in CountVectorizer we change ngram-range = (2, 2).
* The first value is the minimum length and the latter value is the maximum allowed length of ngrams. Here we declare the smallest and largest lengths to be 2, so we get ngrams as bigrams.

```
from sklearn.feature_extraction.text import CountVectorizer

# bigram
bigram = CountVectorizer(ngram_range = (2, 2))
n1, n2, n3 = bigram.fit_transform(['you have no dog', 'no, you have dog', 'you have a dog']).toarray()

# trigram
trigram = CountVectorizer(ngram_range = (3, 3))
n1, n2, n3 = trigram.fit_transform(['you have no dog', 'no, you have dog', 'you have a dog']).toarray()
```

Phương pháp TF-IDF

* Suppose we have a corpus consisting of many sub-texts. Words that are rarely found in the corpus but are present in certain topics may play a more important role.
* There are also words that appear a lot in the text, but they appear in almost every topic, every text such as "the, a, an". Such words are called **stopwords** because they do not have much significance for text classification.
* When encoding language, we will find a way to remove stopwords by using a dictionary with important stopwords.

Phương pháp TF-IDF

The TF-IDF method is a method in which we will give greater weight to words that appear in some specific texts through the formula:

$$\text{idf}(t, D) = \log \frac{|D|}{|\{d \in D; t \in d\}| + 1} = \log \frac{|D|}{\text{df}(d, t) + 1}$$

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D)$$

where:
- $|D|$ is the number of documents in the text set.
- $\text{df}(d, t) = |\{d \in D; t \in d\}|$ is the frequency in texts $d \in D$ in which the word t appears.
- $\text{tf}(t, d)$ is the frequency of appearance of word t in text d.

Thus, a word is more popular when idf is smaller and tfidf is larger.

Content

Image characteristics

* In the past, when computing resources were limited and neural networks were not yet really developed, feature mining for image data was a complex field. Then it is necessary to design manual filters to extract features such as corners, edges, horizontal, vertical, diagonal lines, borders, colors, ...
* Before deep learning exploded, the algorithm commonly used in image processing was HOG (histogram of oriented gradient).
https://phamdinhkhanh.github.io/2019/11/22/HOG.html

Image characteristics

* HOG has specific applications such as
  * Human detection

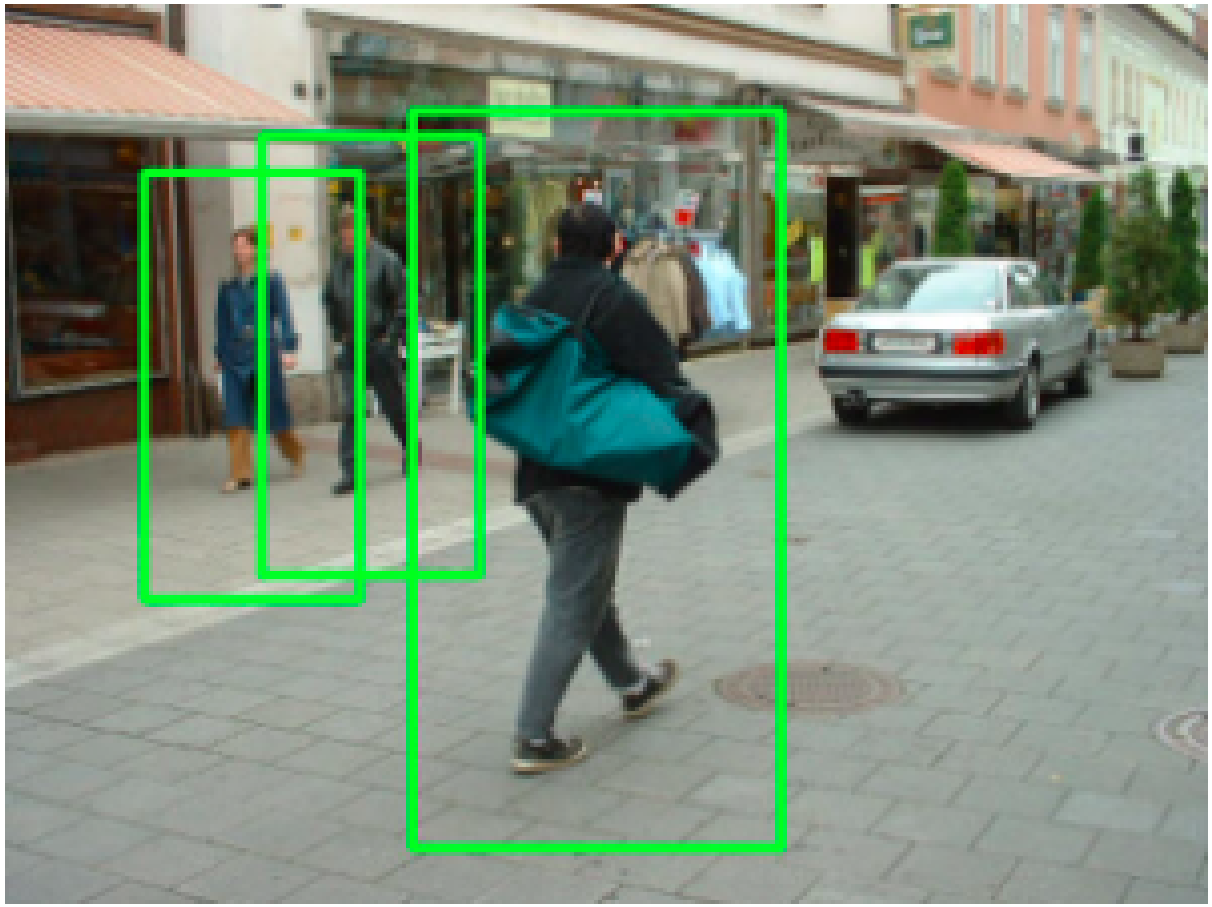


Image characteristics

* Face detection

* Identify other objects
* Create features for image classification problems

Image characteristics

However, currently there are many different methods in computer vision.

* When classifying images, we can apply the CNN family of models (Inception Net, mobile Net, Resnet, Dense Net, Alexnet, Unet,...).
* When detecting objects, YOLO, SSD, Faster RCNN, Fast RCNN, Mask RCNN...

Image characteristics

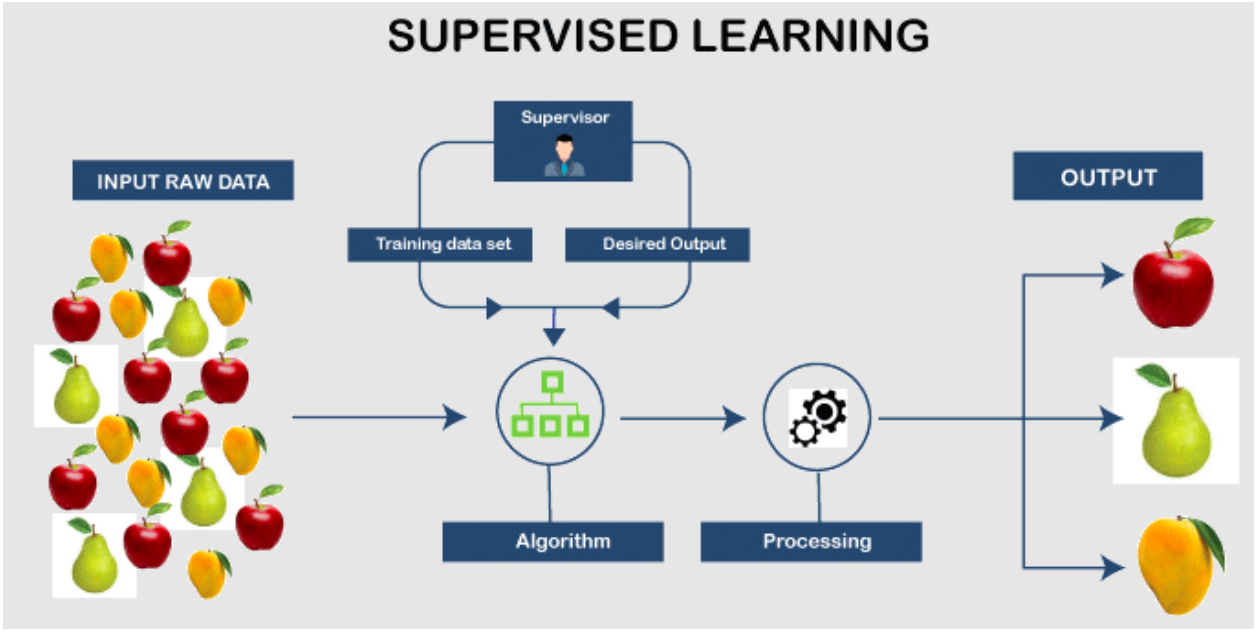
- End-to-end architectures allow feature extractors to be attached to classifiers in a single pipeline.
- Thanks to the available resources of pretrained models, there is no need to figure out the architecture and train the network from scratch, but it is possible to download a modern trained network.
- Adapt these networks to needs by "decoupling" the final fully connected layers of the network, adding new layers designed for a specific task, and then train the network on new data.
- If the task is just to vectorize the image, then simply discard the last layers and use the output from the previous layers.

Content

Content

Supervised learning

* Supervised learning is an algorithm that predicts the output of a new input based on previously known (input, outcome) pairs. This data pair is also called (data, label).



Supervised learning

* Set of input variables $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ and a corresponding set of labels $\mathcal{Y} = \{y_1, y_2, \ldots, y_N\}$, where x_i, y_i are vectors.
* The known data pairs $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ are called the **training data set.** From this set of training data, we need to create a function that maps each element from the set $\mathcal{X}$ to a corresponding (approximate) element of the set $\mathcal{Y}$:

$$y_i \approx f(x_i), \forall i = 1, 2, \ldots, N$$

* The goal is to approximate the function f well enough so that when we have a new data x, we can calculate its corresponding label $y = f(x)$.

Supervised learning

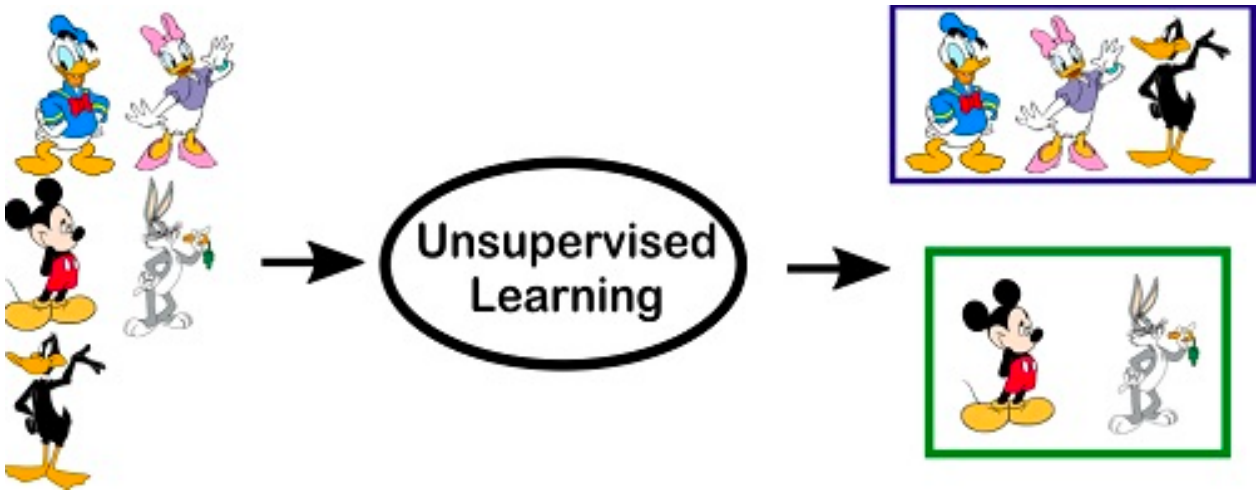
Supervised learning algorithms are further divided into two main types:

* Classification: A problem is called classification if the labels of input data are divided into a finite number of groups. For example, Gmail determines whether an email is spam or not; Credit agencies determine whether a customer has the ability to pay his or her debts.
* Regression: If the label is not divided into groups but is a specific real value. For example, how much would a house x m^2 large, with y bedrooms and z km from the city center cost?

Content

Unsupervised Learning

* Unsupervised Learning is an algorithm that does not know the outcome or label, but only the input data. It will rely on the structure of the data to perform a certain job. For example, clustering or dimension reduction of data to facilitate storage and calculation.
* Unsupervised learning is only having input data $\mathcal{X}$ without knowing the corresponding label $\mathcal{Y}$.



Unsupervised Learning

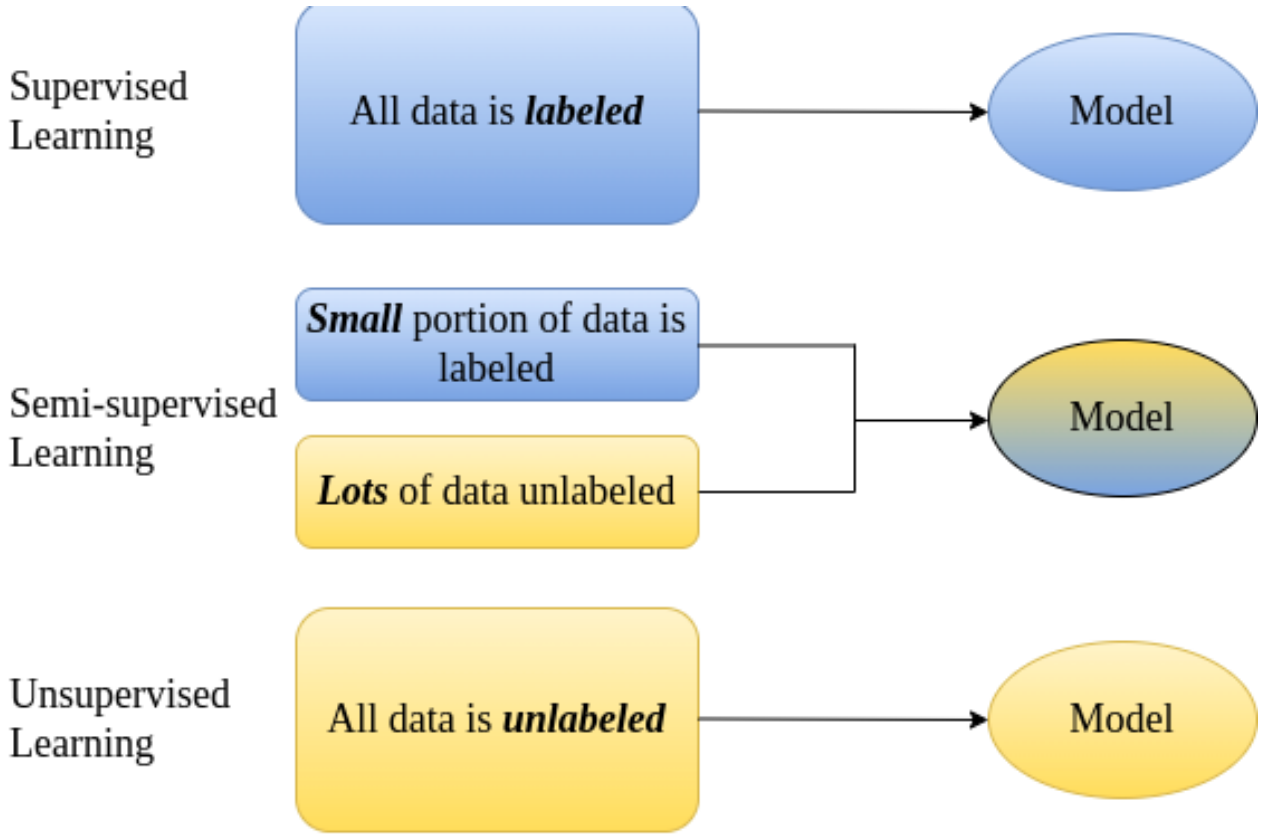
Unsupervised learning algorithms are further broken down into two categories:

* Clustering: A problem of grouping all data into small groups based on the relationship between the data in each group. For example, grouping customers based on purchasing behavior.
* Association: A problem when we want to discover a rule based on a lot of given data. For example, male customers buying clothes are more likely to buy watches or belts.

Content

Semi-Supervised Learning

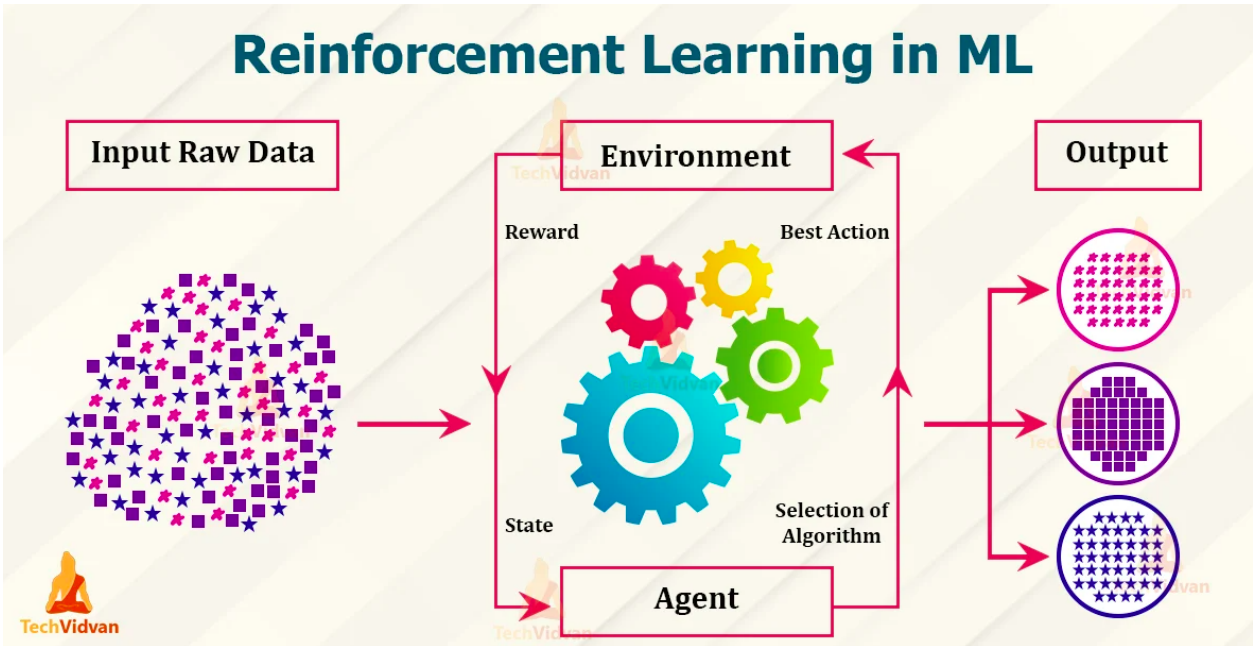
Problems where there is a large amount of data but only a portion is labeled are called Semi-Supervised Learning, located between the two groups of Supervised learning and Unsupervised Learning.



Content

Reinforcement Learning

Reinforcement learning helps a system automatically determine behavior based on circumstances to achieve the highest benefit. Currently, Reinforcement learning is mainly applied to Game Theory, where algorithms need to determine the next move to achieve the highest score.



Group exercise

Each group studies one of the following Machine Learning algorithms:

1. Linear Regresion
2. K-means Clustering
3. K-nearest neighbors
4. Perceptron Learning Algorithm
5. Multi-layer Perceptron
6. Logistic Regression
7. Support Vector Machine
8. Soft Margin Support Vector Machine
9. Kernel Support Vector Machine
10. Multi-class Support Vector Machine
11. Softmax Regression
12. Naive Bayes Classifier