Chapter 1

**Introduction to**

**MACHINE LEARNING**

*What is machine learning*

*Applications of machine learning*

*Supervised Learning*

*Unsupervised Learning*

**1.1 MACHINE LEARNING**

* **AI is the new electricity**

Just as electricity transformed countless industries—transportation, manufacturing, healthcare, communications, and more—AI (including Machine Learning, Deep Learning, and Reinforcement Learning) will bring about an equally significant transformation.

* ***Demand for AI Skills:*** The demand for AI skills is vast, offering numerous opportunities to apply learning algorithms both in industry and academia.
* For instance, investment departments in industries utilize learning algorithms to analyze investment history data
* Lawyers apply machine learning to process legal documents.

From technology companies to manufacturing, healthcare, and logistics firms, there is a widespread effort to implement ***Machine Learning***. As a result, development in machine learning is rapidly increasing.

* Our Goal in this course is to:
* Develop meaningful Machine Learning applications.
* Solve engineering problems (mechanical, electrical), language processing challenges, legal issues, and more using ML techniques.
* Equip students with the ability to conduct research in ML, including reading and understanding research papers, and empowering them to push forward the state of the art.
* In this specialization, you will learn the following courses in sequence:

1. ***Neural Networks*** and ***Deep Learning***
2. Improving Deep Neural Networks: ***Hyperparameter tuning***, ***Regularization*** and ***Optimization***
3. ***Structuring*** your ***Machine Learning project***
4. ***Convolutional Neural Networks***
5. ***Natural Language Processing***: Building ***sequence*** models

* Prerequisites:

|  |  |  |
| --- | --- | --- |
| * Basic ***Computer Skills and Principles***: * Big O notations * Queue * Stacks * Binary trees | * Basic knowledge of ***Probability***: * Random variable * Expected value of Random variable * Variance of Random variable | * Basic knowledge of ***Linear Algebra***: * Matrices * Vectors * Matrix and Vector multiplications * Eigen vector |

* ***Homework & Projects:*** Practice the concepts learned with homework and projects. Use ***MATLAB*** and ***Octave*** for the assignments. However, we'll use ***Python*** and ***NumPy***, and for production-level work, we'll use C++.

|  |  |
| --- | --- |
| * Projects: Its goal is to make you well-qualified to do *meaningful Machine Learning projects*. * Start brainstorming to accomplish the project: pick an area/application (e.g., biology, medicine, engineering, math) that excites you and try to apply machine learning to it. See if you can build a good ***Machine Learning System*** for some application in that area. * Go to the website: <https://cs229.stanford.edu/> * Get an overview of the previous years' projects for inspiration (e.g. <https://cs229.stanford.edu/proj2018/> ; <https://cs229.stanford.edu/proj2019spr/> ). For example: * Diagnosing cancer * Generating art * Some areas of engineering * Understanding literature | * Some of other Prerequisites: * Linear algebra, * Basic probability and statistics, * Python, NumPy * Other advanced materials like: For example: * CONVEX optimization algorithm (lots of learning algorithms relay on it); * Hidden Markov models, * Time Series |

* What is machine learning?

Machine learning is the science of getting computers to *learn without explicit programming*. It's used in many daily applications, often without us realizing it. Common Examples of Machine Learning:

* *Web Search:* Machine learning helps search engines like Google, Bing, or Baidu *rank web pages*, improving the accuracy of search results.
* *Tagging Friends on Social Media:* Apps like Instagram or Snapchat use machine learning to *recognize* and *tag friends* in *photos* (so that they can see their pictures).
* *Movie Recommendations:* Streaming services use machine learning to suggest movies or shows based on your viewing history.
* *Text-to-Speech:* Features like *Siri*, *Google Assistant*, or *voice-to-text* use machine learning to understand and process speech.
* *Spam Detection:* Email services flag potential spam emails using machine learning algorithms.
* *Industrial and Business Applications:* Machine learning is rapidly being adopted by industries and businesses.
* *Climate Change:* It helps optimize wind turbine power generation.
* *Healthcare:* Machine learning is being used to assist doctors in making accurate diagnoses.
* *Manufacturing:* Computer vision is used to inspect products for defects in factories (for example Landing.AI).

That's machine learning, it's the science of getting computers to learn without being explicitly programmed. In this book, you learn about machine learning and get to implement machine learning and code yourself.

* Why is machine learning so widely used today?

*Machine learning* has evolved as a *sub-field of artificial intelligence (AI)*. We wanted to build intelligent machines. It turns out that there are a *few basic* *things* that we could *program a machine to do*, such as: calculating the *shortest route* from point A to B (like in GPS systems).

* But for the most part, we just *did not know* how to write an *explicit program* to do many of the more interesting things, such as perform *web search*, *recognize human speech*, *diagnose diseases from X-rays* or build a *self-driving car*.
* The only way we knew how to do these things was to have a *machine learn to do it by itself*. It enables machines to perform complex tasks that are *difficult or impossible to program explicitly*.
* In these cases, the only viable approach has been to create algorithms that allow machines to learn to perform these tasks independently. This has led to significant advancements across multiple industries, as machine learning enables new capabilities and automation.

Machine learning professionals have applied this technology to a wide range of applications, from *speech recognition* and *computer vision* (Google Maps) to *fraud* *detection*, *augmented reality*, and *autonomous vehicles*. For instance, AI applications are now common in manufacturing, large-scale agriculture, healthcare, and e-commerce (Landing.AI, Stanford University).

Today, thousands, perhaps millions, of people are developing machine learning applications across various industries, and the demand for these skills continues to grow. Machine learning is expected to impact nearly every industry significantly, creating exciting opportunities for those who acquire these skills.

|  |
| --- |
| This book will teach you about the *latest advancements* in *machine learning*, along with hands-on experience in *implementing* these *algorithms*.   * You'll explore some of the most significant *machine learning algorithms* used in leading AI and tech companies today, gaining insight into the current *state of the art in AI*. * In addition to learning the algorithms, this book covers essential practical tips and techniques to optimize their performance. You'll implement these algorithms yourself and see firsthand how they work. |

**Artificial General Intelligence (AGI) :**

Looking further into the future, many people are excited about the idea of creating *machines as intelligent as humans*, often referred to as Artificial General Intelligence (AGI). However, the development of AGI is still considered to be a *distant goal*, with some experts estimating it could take *anywhere from 10 to 50 years* or longer to achieve. Although AGI is often discussed, most AI researchers believe that progress toward AGI will come through *advancements* in *learning algorithms*, potentially drawing inspiration from the human brain.

* The pursuit of AGI has sparked considerable interest, and you may hear more about this vision as you continue learning about **AI**. Beyond AGI, *machine* *learning* and *AI* have already created immense economic value. According to a study by McKinsey, *AI* and *machine learning* could add an estimated *$13* *trillion* of value annually by *2030*. While these technologies have transformed the *software industry*, there is even greater potential for machine learning to drive innovation and efficiency in sectors like *retail*, *travel*, *transportation*, *automotive*, and *materials manufacturing*.
* Given the vast, untapped opportunities in these industries, the demand for machine learning expertise remains high, making this an excellent time to develop these skills. If you find machine learning applications exciting, mastering these concepts could open numerous rewarding opportunities.

In the following sections, we'll dive into the *formal definitions of machine learning*, explore the main *types* of *machine learning problems* and *algorithms*, and introduce essential *terminology*. By the end, you'll have a solid foundation in machine learning concepts and an understanding of when to apply different algorithms.

* Machine Learning : The definition

Now, lets explore the definition of machine learning, along with insights on when to apply it.

**Machine Learning definition :**

* A widely accepted definition of machine learning comes from Arthur Samuel, who described it as *the field of study that enables computers to learn without being explicitly programmed*. Samuel gained recognition in the 1950s for developing a *checkers-playing program*. Interestingly, he was not a particularly skilled checkers player himself.
* He programmed the computer to play tens of thousands of games against itself.
* By observing which board positions led to wins and which led to losses, the checkers program learned to identify *favorable* and *unfavorable positions*. Over time, this iterative process helped the program enhance its gameplay.
* Because the computer had the patience to play tens of thousands of games against itself, it gained so much checkers-playing experience that it eventually became a better player than Arthur Samuel himself.

Arthur Samuel’s definition was informal, but in the next sections, we’ll explore the major types of machine learning algorithms in greater depth.

|  |  |
| --- | --- |
| * **Arthur Samuel (1959). Machine Learning:** Field of study that gives computers the ability to *learn without being explicitly programmed*. * The Samuel Checkers-playing Program was among the world's first successful self-learning programs, and as such a very early demonstration of the fundamental concept of artificial intelligence (AI). * He wrote the program in such a way that it could learn patterns that lead to winning through self-play. The program eventually became better than the programmer himself. * It used techniques such as the minimax algorithm and alpha-beta pruning to play the game, and it improved by playing thousands of games against itself. * It used a technique called "rote learning" for self-improvement. It also incorporated a *heuristic evaluation function* to estimate the ***likelihood*** of ***winning*** from a particular position, allowing the program to make better decisions over time. |  |

Arthur Samuel's work laid the groundwork for the *development of algorithms that could learn from data*, a fundamental concept in modern *Machine Learning*.

His approach to self-learning systems demonstrated the potential for computers to *adapt and improve based on experience*, influencing future research and development in *AI* and *Machine Learning*.

His work continues to influence the field, underscoring the importance of algorithms that can learn and adapt without explicit programming.

Nowadays, machine learning programs are outperforming humans in various tasks. However, we still have to choose narrow tasks, such as speech recognition or playing a specific game like: the game of Go.

**Well-posed Learning Problem :**

A "well-posed learning problem" in the context of machine learning refers to a problem that meets certain criteria ensuring it can be meaningfully and effectively addressed by a *Machine Learning Algorithm*. These criteria are generally inspired by ***Hadamard's definition*** of well-posed problems in ***mathematical modeling*** and are adapted to the machine learning context.

* **Tom Mitchell (1998). Well-posed Learning Problem:** A computer program is said to learn from *experience* E with respect to some *task* T and some *performance measure* P, if its *performance* on T, as *measured* by P, ***improves*** with *experience* E.
* A well-posed learning problem in machine learning involves having a *clear objective*, *sufficient and relevant data*, a *definable evaluation metric*, and ensuring that the *model can* ***generalize*** *to* ***new data***. This ensures that the learning problem can be effectively addressed using machine learning techniques.
* In the case of playing the checkers,
* *Experience* E would be computer playing checkers against itself
* *Task* T Would be playing checkers.
* *Performance measure* P would be Computer's chance of winning the game against the next opponent.

Throughout these lessons, questions will occasionally be asked to help reinforce understanding. Here’s a sample question related to the checkers program:

***Quiz :-***

*If the checkers program had been allowed to play only ten games against itself, instead of tens of thousands, how would this have affected its performance?*

*a. It would have improved performance.*

*b. It would have worsened performance.*

***The correct answer is:***

It would have worsened performance.

Playing only ten games would give the checkers program less experience, making it less skilled.

More games allow it to learn better strategies, so fewer games would result in poorer performance.

In general, the more opportunities you give a learning algorithm to learn, the better it will perform.

Overview of major areas of Machine Learning | Overview of topics we'll cover

Supervised Learning | Unsupervised Learning

ML Strategy

Deep Learning

Reinforcement Learning

**1.2 SUPERVISED** learning

There are many different ***tools*** we use in ***Machine Learning***. In our course, we'll try to learn this variety of tools. Let's discuss some of the ***major categories*** of Machine Learning tools:

* Machine Learning Algorithms

This course covers various machine learning algorithms, focusing on two main types: *supervised and unsupervised learning*. These terms will be defined in more detail in upcoming sections.

* *Supervised learning*, the most widely used type of machine learning in real-world applications, has seen rapid advancements and is central to many innovations. Consequently, about *two-thirds* of the course will focus on *supervised learning*, while the remaining one-third will cover *unsupervised* *learning*, *recommender systems*, and *reinforcement learning*.
* The primary types of learning algorithms used today are supervised learning, unsupervised learning, and recommender systems.
* In addition to learning about algorithms, this course emphasizes *practical advice* for applying them effectively. Understanding *how to use* these tools is as essential as having the tools themselves. Just as simply owning a high-quality hammer and hand-drill doesn’t mean you can build a house, *knowing* *machine learning algorithms without understanding their application limits your effectiveness*.
* Many teams at top tech companies, including those with experienced machine learning professionals, sometimes *spend months on approaches that* *ultimately don’t succeed*. In many cases, a different method could have offered a much better chance of success from the start.

In this course, you'll not only gain the tools but also learn how to use them effectively to achieve results. This course covers best practices for developing *practical and valuable machine learning systems*, reducing the likelihood of spending months pursuing ineffective directions. By understanding how skilled engineers build successful systems, learners will gain the rare ability to design and build robust machine learning solutions.

Next, we’ll dive deeper into supervised and unsupervised learning, exploring when to use each.

* **Supervised Learning**

Machine learning is driving *significant economic value*, and the majority of this value around 99%— comes from *supervised learning*.

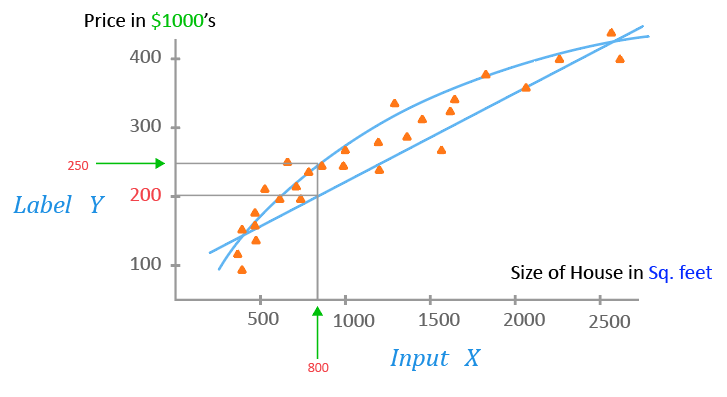
* Supervised learning involves *algorithms* that learn to *map* *inputs (x)* to *outputs (y)*. In other words, it uses *labeled data* to *predict outcomes* or make *decisions*.
* The key feature of supervised learning is that the algorithm *learns* from *examples with correct answers (correct labels y for a given input x)*. By observing *pairs* of *input (x)* and *correct output (y)*, the algorithm *learns to predict the output* for *new inputs*, even *without* being given the *label*.
* In supervised learning, the goal is to learn a function that maps *inputs to outputs* *based on example* ***input-output pairs***.
* **Dataset (X, Y):** You start with a dataset composed of pairs *(X, Y)* where X represents the *input data (features)* and Y represents the *output data (labels)*.
* **Inputs (X):** *X* could be anything that you want to ***make predictions about***. For example, in a housing price prediction model, *X* could include *features* like the *size of the house*, the *number of bedrooms*, *location*, etc.
* **Labels (Y):** is what you want to predict. In the housing price example, would be the price of the house.
* **Mapping Function** : The *objective* is to *learn a function* that takes as input and produces as output.
* This **Mapping Function** is usually *derived through training* on the dataset.
* Here are some examples of *supervised learning applications*, where a learning algorithm is trained on input-output (x, y) pairs and can later predict outputs for new inputs:
* ***Online Advertising:*** Most large online ad platforms use algorithms that analyze ad information and user data to predict whether you'll click on an ad. By showing ads you're more likely to click, these platforms increase clicks, which directly boosts their revenue.
* *Input (x):* Information about an ad and user data. *Output (y):* Will the user click on the ad? (Yes = 1, No = 0)
* This application generates significant revenue for online ad platforms by predicting and showing ads that users are more likely to click.
* ***Self-Driving Cars:*** If you want to build a self-driving car, the learning algorithm would take as input an *image* and some *information from other sensors* such as a *radar* or *other things* and then try to output the position of, say, other cars so that your self-driving car can *safely drive around* the other cars.
* *Input (x):* Images and sensor data (e.g., radar). *Output (y):* Position of other cars or obstacles, enabling safe navigation
* ***Visual Inspection in Manufacturing:*** You can have a learning algorithm takes as input a picture of a manufactured product, say a *cell phone* that just rolled off the production line and have the learning algorithm output whether or not there is a *scratch*, *dent*, or other *defect* in the product. This is called *visual inspection* and it's helping manufacturers reduce or prevent defects in their products (eg: (Landing.AI)).
* *Input (x):* An image of a manufactured product (e.g., a phone). *Output (y):* Presence of defects like scratches or dents (Yes = 1, No = 0)
* This helps manufacturers detect and reduce product defects.
* ***Speech recognition:*** If the input is an audio clip and the algorithm's job is output the text transcript.
* ***Machine translation:*** If you want to input English and have it output to corresponding Spanish, Arabic, Hindi, Chinese, Japanese, or something else translation.

In all these applications, the model is trained using labeled data (x and y pairs). Once trained, it can predict the output (y) for unseen inputs (x).

|  |  |  |
| --- | --- | --- |
| Input (x) | Output (y) | Application |
| Email | Spam? (Yes/No) | Spam Filtering |
| Product image | Defect? (Yes/No) | Visual Inspection |
| Advertise, user info | Click on ad? (Yes/No) (0/1) | Online Advertising |
| Image | Object (1, . . . , 1000) | Photo tagging |
| Audio (An audio clip) | Text transcript | Speech recognition |
| English text | Chinese (Translated text) | Machine translation |
| Image, Radar info | Position of other cars | Autonomous driving |

|  |  |
| --- | --- |
| **Regression: Housing price prediction :**  The most widely used Machine Learning tool is *Supervised Learning*, for example— *Regression Analysis*. Consider this regression problem: Let's say we have a database (dataset) of housing prices:   * We want to *predict housing prices* based on the *size of the house*. We've collected some data and say we plot the data and it looks like below. * ***Horizontal Axis (x):*** Size of the house (e.g., square feet). * ***Vertical Axis (y):*** Price of the house (e.g., in thousands of dollars). |  |

* If your friend's house is *800 square feet*, how can a learning algorithm help *predict* its *price*? We can have two types of line fitting:
* ***Fitting a Straight Line:*** A learning algorithm might fit a *straight line* to the data. Based on the line, it could *predict* the price of the *800-square-foot* house to be about *$200,000*.
* ***Fitting a Curve:*** Sometimes, a straight line might not be the best fit for the data. Instead, using a *curve* (a more complex function) might work better. In this case, the *predicted price* might be closer to *$250,000*.



**Regression:** Housing price prediction

* ***Supervised Learning:*** Above is an example of supervised learning because the algorithm is trained on a dataset where the "*correct answer*" (i.e. the *label* or the correct price y is given for every house) is already provided. The algorithm learns from this data and uses it to *predict* the price of other houses (which are *not given in the training data*), such as your friend's.

Let's consider a dataset where:

= size of houses in square feet

= price of houses in dollars

* You have multiple pairs of in your dataset. For instance:

(500 sq. ft, $150,000)

(1000 sq. ft, $215,000)

(1200 sq. ft, $275,000)

Using *supervised learning*, you aim to learn a function that can predict the price given the size of a house.

* After training, if you input 800 sq. ft into your learned function , it should output an estimate of the price, say $250,000, based on the *learned relationship* from the data (let's consider the straight line).
* So, in supervised learning, you're given a dataset with inputs and labels , and the goal is to learn a mapping from to ,
* For example, if you want to ***sell*** a house of 750 sq. feet, you can *estimate the price* from this kind of *dataset* based on the relationship between inputs and labels .
* ***Different mappings:*** In machine learning, there are various ways to create a mapping between input data and output data . For instance:
* ***Linear Model:*** The simplest approach is to fit a straight line through the data points.
* ***Non-Linear Model:*** For better accuracy, a more complex model like a quadratic curve can be used, which may fit the data more closely.
* The choice of model depends on the method of selection, which can be done *automatically using algorithms* or with *manual intervention* based on *expertise* and data analysis.
* Later we'll learn is how we can decide whether to fit a straight line, a curve, or another function that is even more complex to the data.
* We'll see is how to get an *algorithm* to *systematically choose* the most *appropriate line or curve* or other thing to fit to this data.
* We'll use algorithm to find the most appropriate function (straight line, curve, or something else) to make accurate predictions. The choice of function depends on the data, not personal preferences.

**Supervised learning :**

***Supervised learning*** is a type of *machine learning* where the model is trained on labeled data, meaning that each training example (input X) is paired with an output label (Y). There are *several types* *of supervised learning*, each suited for different types of problems. The two main types are:

* ***Regression:*** Above example is a specific *type* of *supervised learning* is called regression, which *predicts* a *continuous value* from an *infinite range of possibilities* (e.g., $150,000, $70,000, $183,000, etc.).
* By regression, we mean we're trying to predict a number from infinitely many possible numbers such as the house prices in our example, which could be 150,000 or 70,000 or 183,000 or any other number in between.
* In regression, the output is a number, like the house price in this example.
* ***Similar Examples:*** Predicting *house prices*, forecasting *stock prices*, estimating *temperatures*.
* ***Common Algorithms are:***
* Linear Regression
* Polynomial Regression
* Ridge Regression
* Lasso Regression
* Support Vector Regression (SVR)
* Decision Trees
* Random Forest

We also can use Neural Networks (for regression tasks)

* ***Classification:*** These kind of algorithms are used to predict a ***discrete*** *class label (such as Yes, No)*.Examples: Spam detection, image classification, medical diagnosis (benign vs malignant). Common Algorithms:
* Logistic Regression
* K-Nearest Neighbors (KNN)
* Support Vector Machines (SVM)
* Decision Trees
* Random Forest
* Gradient Boosting Machines (GBM)
* Naive Bayes

And we can use Neural Networks (for classification tasks)

Additional Types of Supervised Learning

|  |  |
| --- | --- |
| * ***Time Series Forecasting:*** To predict *future values* based on previously *observed* *values*. Examples: Weather forecasting, sales forecasting. Common Algorithms: * ARIMA (AutoRegressive Integrated Moving Average) * Exponential Smoothing * Long Short-Term Memory Networks (LSTM) | * ***Anomaly Detection:*** To identify *rare items, events, or observations* which raise suspicions by differing significantly from the majority of the data. Examples: Fraud detection, network security. Common Algorithms: * Isolation Forest * One-Class SVM * Autoencoders (for anomaly detection tasks) |

These types of supervised learning algorithms are chosen based on the nature of the problem, the type of data available, and the specific requirements of the task at hand.

That's supervised learning— learning input, output, or x to y mappings. And above example is a specific *type* of *supervised learning­*—***regression***, where the task is to predict a number. But there's also a second major type of supervised learning problem called ***classification***, where the goal is to categorize data into discrete groups. The next section will cover that concept.

* **Regression and Classification**

**Regression problem :**

In the "Housing Price Prediction" example, the problem is known as a *Regression Problem*. The term "regression" refers to the fact that the value we're trying to ***predict*** is *continuous*.

**Classification problem :**

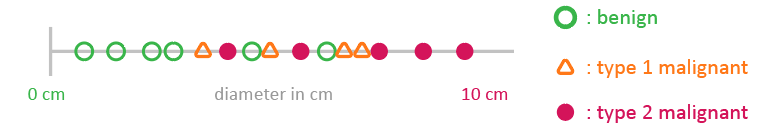
Now, let's consider a different type of problem: a *healthcare problem* where we're looking at ***Breast Tumors*** that can be either "Benign" or "Malignant". The goal is to predict whether a *Tumor* will turn into *Breast Cancer*.

* Imagine creating a *machine learning system* to help doctors detect breast cancer, which can save lives through *early diagnosis*. Using a patient's medical *records*, the system determines whether a tumor (lump) is *malignant (cancerous and dangerous)* or *benign (non-cancerous and harmless)*.
* The *dataset* might include *tumors of various sizes*, labeled as either *benign* (represented as 0 in this example) or *malignant* (represented as 1). This data can be plotted on a graph, where the *horizontal* axis represents the *size of the tumor*, and the *vertical* axis takes on only two values: 0 for benign tumors and 1 for malignant tumors.

|  |  |
| --- | --- |
| * In this case, on the *horizontal axis* (X-axis), we place the "tumor size." On the vertical axis (Y-axis), we indicate whether the *tumor is malignant or no*t, with possible values of *1 (malignant)* or *0 (benign)*. I.e. in the vertical line (Y-axis), it takes two values 1 or 0. * Given such a dataset, the task is to *learn a mapping* from X to Y, (). Then, when a new patient walks into a doctor's office, the *learning* *algorithm* can predict whether the *tumor* is *benign* or *malignant* based on the patient's *tumor size*. * From our demo dataset, we can predict that there's a high chance the patient's tumor (light-green point) is malignant. |  |
| * The problem described above is a classification problem. The term "classification" refers to situations where the output variable Y takes on a *discrete* number of *values*. * ***Different way to visualize the dataset:*** We can represent our data from the above *2D plot* in a *1D format*. Positive data points are represented as 1 (denoted by 'o'), and negative data points are represented as 0 (denoted by 'x'). This is simply an alternative way to visualize the same data, *using a line* and *two symbols* to denote the two discrete values, 0 and 1. * If a new patient arrives for diagnosis with a *lump* of a *specific size*, the question becomes whether the system will *classify* the *tumor* as *benign* or *malignant*. * This differs from *regression* because the goal is to *predict* a *limited number* of possible *outputs* or *categories*. In this case, there are only two possible outputs: 0 (benign) or 1 (malignant). * Regression, on the other hand, *predicts* from *infinitely many possible numerical values*. * The limited number of possible outputs (in this example two possible outputs) defines this task as classification. |  |
| Since there are only two possible outputs or categories in this example, the dataset can also be plotted on a line. Different symbols can be used to represent the categories: a **circle (O)** for **benign** examples and a **red dot** for **malignant** examples. |

* **CLASSIFICATION: more than two possible output categories**

Classification problems can have more than two output categories. For example, a learning algorithm could predict *different cancer types* if the tumor is **malignant**, such as **type 1** or **type 2**.



* In this case, the algorithm would have three possible output categories: **benign**, **type 1**, or **type 2**.
* The terms "output classes" and "output categories" are used interchangeably in classification, meaning the same thing.
* ***Classification problem with more than 2 classes:*** If we have K - discrete outputs (for example, detecting 5 different types of cancers), we then have 5 different possible outputs. This is also a classification problem because the output is discrete.
* To summarize, *classification* algorithms *predict categories*.
* Categories can be ***non-numeric***,
* such as determining if a *picture* shows a *cat* or a *dog*, or
* predicting if a *tumor* is *benign* or *malignant*.
* Categories can also be ***numeric***, like 0, 1, or 2.

The key difference between classification and regression is that *classification* predicts a *limited set* of *discrete categories* (e.g., 0, 1, 2) and *not* *continuous values* like 0.5 or 1.7.

*In contrast, for a* ***regression*** *problem, Y is a* ***continuous*** *real number. For example, in our "Housing Price Prediction" problem, the price of houses is a continuous variable. Therefore, the "Housing Price Prediction" problem is a regression problem.*

*In a* ***classification*** *problem, there are two or more possible outcomes, such as {0, 1}.*

* **Classification** with **many features**

Notice that in both examples above, the input x is one-dimensional, but in most Machine Learning problems, the ***input*** will be multi-dimensional. You won't be given just one number (feature) to predict another number (output); instead, you will often be given *multiple features (numbers)* to predict a *target* *value* (i.e. multiple input values can be used to predict an output).

|  |  |
| --- | --- |
| * For example, instead of using only **'tumor size'** to predict whether a tumor is *malignant* or *benign*, we may use two features, such as **'tumor size'** and **'age of the patient'**, and create a dataset as shown below: * The new dataset now has two inputs: **age** and tumor **size**. *Red circles* represent patients with *malignant (positive)* tumors, while *blue crosses* represent patients with *benign (negative)* tumors. * Now, our goal is to predict whether a tumor is malignant or benign based on these two input features (like a 2D vector). According to the ***point*** shown in this figure, the tumor is *benign*, i.e. *not cancerous* (where 'o' represents a positive, i.e. malignant data example). * So when a new patient visits, the doctor measures the tumor *size* and records the patient's *age*. Using this information, how can it be determined if the tumor is benign or malignant? |  |

|  |  |
| --- | --- |
| * **How to classify:** As part of the classification problem, one task is to draw a *boundary line* through the data points to separate the **negative ('x')** and **positive ('o')** examples using a logistic regression algorithm. * With a dataset like this, a learning algorithm can *identify a boundary* that separates *malignant* tumors from *benign* ones. The algorithm's task is to determine the most *appropriate boundary line* based on the data. * This boundary line, once established, can assist the doctor in making a diagnosis. For instance, in this scenario, the tumor is more likely to be *benign* based on its *position* relative to the boundary.   This example demonstrates how two inputs (features), the patient's age and tumor size, can be utilized for prediction. However, in many machine learning problems, a *larger number of input features* are often required for *accurate predictions*. For instance, in breast cancer detection, additional inputs such as : |  |

* The thickness of the tumor clump,
* Uniformity of cell size,
* Uniformity of cell shape,

and other features are commonly used to improve the model's accuracy.

* ***How to Deal with a Large Number of Features:*** In many cases, there can be a large number of features to consider. For example, in *Breast Cancer Detection*, features such as clump thickness, uniformity of cell size, uniformity of cell shape, adhesion, and more could be important.
* We can visualize these high-dimensional datasets (we'll discuss this further in learning theory). As we develop *Learning Algorithms* to build *Regression* or *Classification* models, we can manage these large numbers of features.
* ***Use SVM:*** One of the most fascinating concepts we'll explore is the Support Vector Machine (SVM) algorithm, which can work with an infinite number of input features.
* In our earlier examples, we first had one feature (tumor size), then two features ("tumor size" and "age of the patient"), and in the breast cancer example, we have even more features. *SVM* allows us to represent a cancer patient using an *infinite-dimensional vector*.
* So, how do we handle that? How can a computer store an infinite-dimensional vector? Obviously, we can't store an infinite amount of real numbers in computer memory. We'll cover the technical details of ***Support Vector Machines***, including the kernel technique, which allows us to build learning algorithms that work with an infinite number of features.
* Since having *more features* can provide *valuable information* about a cancer patient, for example, kernel SVM is one of the most effective learning algorithms for such scenarios.

**Recap :**

* *Supervised Learning* involves mapping input x to output y, where the algorithm learns from provided "*correct answers*" or *labeled data*.
* The two main types of supervised learning are:
* ***Regression:*** Predicts *continuous numbers* from an *infinite range* of possible values, such as predicting house prices.
* ***Classification:*** Predicts *categories* from a *limited set* of possible outputs, such as classifying whether a tumor is benign or malignant.

This concludes the overview of *supervised learning*, including *regression* and *classification*. Up next is the second major type of machine learning: ***unsupervised learning***. Let’s move on to explore it.

Real-World application of Supervised Learning

* **Autonomous Car-Driving Using Supervised Learning**

In supervised learning, our task is: to provide "input X" along with "label Y" simultaneously, and the learning algorithm’s job is to find a ***mapping*** so that given a *new input X*, it will produce the most *appropriate output Y*.

* ***Autonomous Driving Using Supervised Learning:*** An example of supervised learning can be seen in autonomous car driving. While this approach *isn't considered state-of-the-art for autonomous driving* anymore, it's still an illustrative example. You may also hear about the technique called "backpropagation", we will learn this technique in the later chapters.

|  |  |  |
| --- | --- | --- |
|  | *This application of Artificial Neural Networks (ANN) for autonomous driving using supervised learning was developed at* **Carnegie Mellon University***.*   * ***Here's how it works:*** during training, the system observes a human driving the vehicle (10 times per second) and captures images in front of the vehicle using a front-facing camera. * To collect *labeled data*, the system records *both the images* (like below) the *front image* and the *steering directions chosen by the human driver*: | |
| Front camera image | | steering directions chosen by the human driver  (using steering-wheel sensors) |

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| * Notice at the bottom, the image is converted to *grayscale* and *lower-resolution* (***input X***). * On top (upper-left), the ribbon indicates the *driving direction*, which we can consider as the ***Y-label*** chosen by the human driver. * The position of the *white blob* shows how the human steered the car. * The *second line (second ribbon)* shows the output of the Neural Network. Initially, it doesn’t know how to drive, so you see a *white smear* or *gray blur* across the bar with *no distinct white blob*. * However, as the Neural Network gets trained using the *backpropagation learning algorithm* (or *gradient descent*), a *white blob* appears in the *second ribbon or bar*, and the algorithm starts to mimic the human driver's actions. |  |



As Neural Network gets trained, white blob appears in the second ribbon

|  |  |
| --- | --- |
|  |  |
| *Untrained network* | *Trained Network* |

* This is an example of supervised learning because the human driver demonstrates the ***input X*** *(the lower-resolution image of the road)* and the ***Y-Label*** *(the human's steering of the car, represented by the upper-ribbon)*. After the learning algorithm has been trained, the human is no longer needed to drive—the vehicle can drive itself, using a small motor to turn the steering wheel.

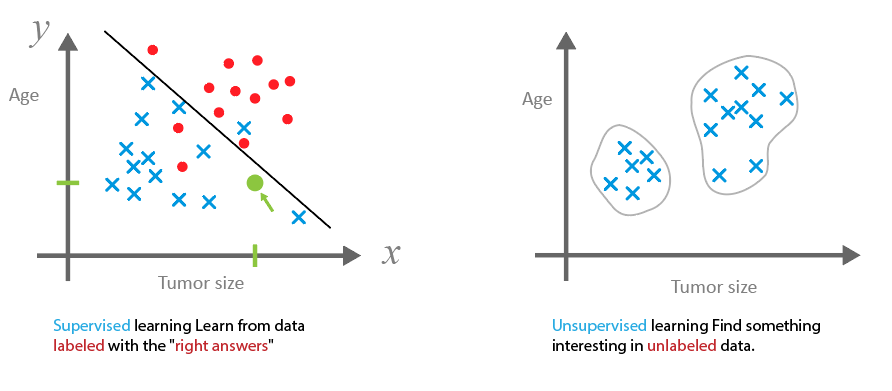
|  |  |
| --- | --- |
| *Upper ribbon represents* **Y-label** *(driving direction by human) and*  *Lower ribbon represents* **Output** *(driving direction by network) from the trained Network*   * ***In a more advanced version of this approach, two separate models are trained:*** one for 2-lane roads and another for 1-lane roads. The *third bar* *(ribbon)* represents the *2-lane road*. An ***arbitrator algorithm*** decides whether the *2-lane road model* or the *1-lane road model* is appropriate, and then it takes the necessary action. | *A more advanced version* |

**1.3 UNSUPERVISED** learning

|  |  |
| --- | --- |
| After supervised learning, the second most widely used type of machine learning is *unsupervised learning*. In the previous section, we discussed supervised learning, such as the *Tumor: Malignant/Benign classification problem*. That was a supervised learning task because we had to learn a function f mapping from X to Y, where both X (the input) and Y (the label) were provided.  Now, let's explore unsupervised learning. Unsupervised learning is just as powerful and important as supervised learning. Let’s explore what this means.   * **What is Unsupervised Learning?**   In ***unsupervised learning***, however, we’re given a dataset like the one below, there are **no labels**, so we are just given an **input X** *with no corresponding* **Y**.   * The goal in unsupervised learning is to discover interesting patterns or structures within the data (e.g., clusters, anomalies). | Classification (supervised learning) |

|  |  |
| --- | --- |
| Unsupervised learning | * For example, in the dataset on the left-figure, you might notice that there are *two clusters* of *data points*. * We'll learn about ***clustering algorithms***, which are a type of *unsupervised learning algorithm*, such as ***k-means clustering***. These algorithms can help discover structures like clusters in a dataset. |

* **Supervised vs Unsupervised**
* In *supervised learning*, each example (data-point x) in the dataset has a corresponding output label y (like "benign" or "malignant" in classification problems). These labels *guide* the *learning algorithm* during *training*. For example, *benign* cases may be represented by *crosses* and *malignant* cases by *circles*, helping the algorithm distinguish between the two.



* However, in *unsupervised learning*, there are *no output labels y* or *predefined categories* to *guide the algorithm*. Say you're given *data on patients* and their tumor size and the patient's age. But *not whether* the tumor was *benign* or *malignant*, so the dataset looks like this on the right.
* In this case, we are not diagnosing whether a tumor is *benign* or *malignant*, as the dataset does not include any *labels (y)*. Instead, the objective is to identify *patterns*, *structures*, or other *notable insights* within the data.

So the algorithm works with *input data only* and aims to uncover *patterns*, *structures*, or *groupings* within the dataset. Despite the *absence of labels*, unsupervised learning is highly valuable and versatile, just like supervised learning.

It is called *unsupervised learning* because the algorithm is *not provided* with *"correct answers"* or *labels* for each *input*. Instead, the algorithm is tasked with independently discovering *patterns*, *structures*, or anything interesting within the data.

**Clustering Algorithm :**

With the given dataset, an unsupervised learning algorithm might identify two distinct *groups* or *clusters* within the data. For instance, it could determine that one cluster is located in one area and another cluster is in a different area.

* This is a particular type of *unsupervised learning*, called a *clustering algorithm*. It works by organizing unlabeled data into meaningful clusters and since it places the unlabeled data, into different clusters it is widely used in various applications.
* **Examples of Unsupervised Learning**

One of the main goals of clustering is to *getting data* and figuring out *what group it belongs together* with similar data-points.

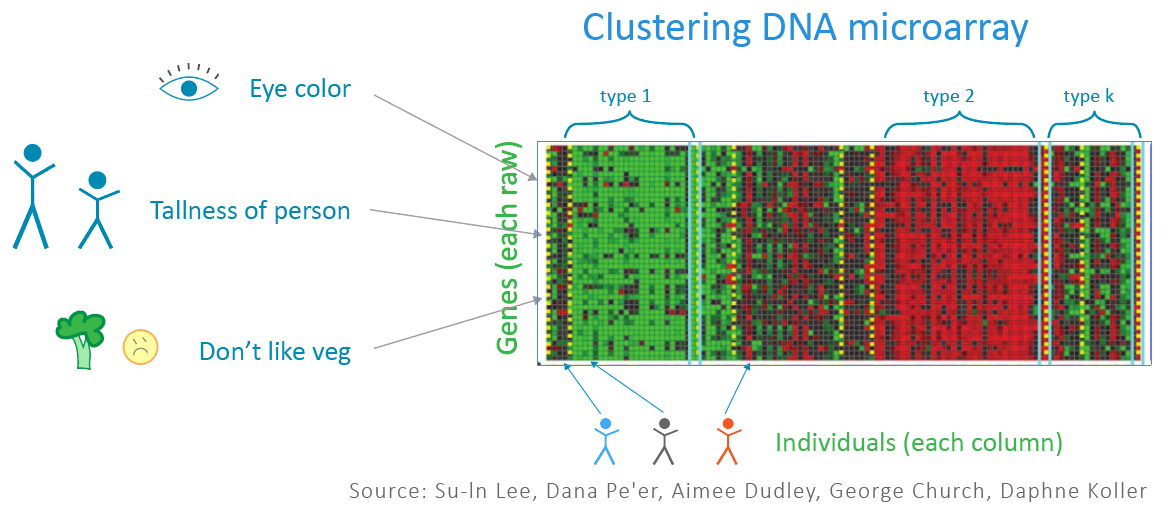
|  |  |
| --- | --- |
| * ***Example 1:*** Clustering in Google News   *Clustering* is used by Google News to *group related news* stories together. Every day, Google News scans hundreds of thousands of news articles online and organizes *similar stories* into *clusters*. For instance, it may gather articles on the topic of *Artificial Heart Transplants* from *various sources* (different articles from different reporters) and group them into *one cluster*. This is done using a ***clustering algorithm*** that recognizes *which stories are related to the same topic* (like artificial heart transplants).   * *How it works:* Let's consider a sample where the top headline is: "Giant panda gives birth to rare twin cubs at Japan's oldest zoo". Beneath this headline, you might see other *related articles*. Upon closer inspection, words like "panda," "twins," and "zoo" appear in all the articles. * The *clustering* algorithm identifies *patterns* in the *text*—such as common words or topics—and *groups articles* with similar *content* into *clusters*. This helps users easily *find* and read *related stories* from various sources. * The impressive part about clustering algorithms is that they independently figure out which *words* indicate that *certain articles* belong in the *same* *group*. What we mean that, there isn't a Google News employee *manually instructing* the algorithm to group articles containing words like "panda," "twins," and "zoo" into one cluster. * Since news topics constantly change and there are thousands of new stories every day, it would be *impossible for people* to *manually organize* these articles for every topic. Instead, the *algorithm autonomously identifies patterns* in the data and determines how to *group related articles* into *clusters* without human guidance. |  |

This is why clustering is considered a type of *unsupervised learning*—the algorithm works *without supervision* to uncover patterns or groups in the data.

* ***Example 2:*** *Clustering Genetic or DNA data.*

In *genetics*, clustering algorithms are often used to *group* data with *similar characteristics*. Let’s consider another example of *unsupervised learning* applied to *clustering genetic or DNA data*. *Genetic Microarray Data* [Source: Su-ln Lee, Dana Pe'er, Aimee Dudley, George Church, Daphne Koller] shows how individuals are grouped based on *different genetic traits*.

* The image below shows *DNA microarray data*, which resembles a tiny grid, much like a spreadsheet. *Each column* represents the genetic or *DNA* *activity* of a *single individual*. For example, one entire column corresponds to one person’s DNA, while another column represents a different person’s DNA.
* *Each row* in the data corresponds to a *specific gene*. For instance, one row could represent a gene that influences *eye color*, while another might affect a *person’s height*.
* Researchers have even found a *genetic links* to whether someone *dislikes* *certain vegetables*, such as broccoli, Brussels sprouts, or asparagus. So, if someone asks why you didn’t finish your salad, you can say (jokingly), “Maybe it’s genetic!”



* For DNA microarrays, the *goal* is to measure how much *specific genes* are *expressed* in each *individual*. The colors, such as *red*, *green*, and *gray*, (colors in the chart) represent the degree to which particular genes are *active* or *inactive* for different individuals.
* A *clustering algorithm* can then be applied to *group individuals* into different *categories* or *types* based on *genetic similarities*. For example, the algorithm might group some individuals together and label them as Type 1, while others are grouped into Type 2, and yet another group as Type 3.
* This process is an example of unsupervised learning because the *algorithm* is not given *prior knowledge* of what defines a Type 1 or Type 2 person. Instead, the task is to *analyze the data*, *find* *patterns*, and automatically *determine* the *underlying structure of the data*, identifying major *types* or *clusters* of individuals.
* Since the algorithm is not provided with predefined "right answers" or labels for the examples, it operates without supervision, making this a classic example of unsupervised learning.

|  |  |
| --- | --- |
| * Also Clustering algorithms can be applied to: * ***Organizing Computer Clusters:*** Clustering algorithms can also be used to *organize computer clusters* by determining *which machines work closely together* or are more related to each other. * ***Social Networking:*** On platforms like *LinkedIn* or *Facebook*, clustering algorithms are used to *identify groups of friends* or *cohesive* *communities* within a social network. * ***Astronomical Data Analysis:*** Clustering is also used in astronomy to group different galaxies based on certain characteristics. |  |

* ***Example 3:*** *Grouping customers. Clustering Example for Market Segmentation*

Many companies have massive *databases of customer information*. Given this data, can you *automatically group customers* into different *market* *segments* to serve them more efficiently?

* Companies use Clustering Algorithms to *segment* their *customer* base into different types, such as by **age**, **profession**, or **gender** (cluster the users in different types according to their age, profession, gender and other factors). This helps the company market its products more effectively by *targeting specific groups* (reach out to the appropriate client) with tailored offerings.

For instance, the *DeepLearning.AI (online learning platform)* team conducted research to better understand the *DeepLearning.AI community*. They aimed to *uncover the reason/motivation* for taking the courses, i.e.

|  |  |
| --- | --- |
| * why different individuals take the courses * why subscribe to the Batch Weekly newsletter, or * why attend the Pioneer events.   Let's visualize the *DeepLearning.AI* community as a *collection of individuals* grouped using clustering (market segmentation). This clustering identified a few distinct groups of people based on their primary motivations:   * *Seeking Knowledge:* One group is primarily motivated by the desire to *grow their skills* and *expand* their *knowledge*. * *Career Development:* Another group is focused on *career advancement*. They may want to secure a *promotion*, land a new *job*, or make *progress* in their *professional journey*. * *Staying Updated:* A third group is interested in staying *informed* about how AI is impacting their *specific field of work*. If this is you, that's great as well! * This is a clustering approach used by the *DeepLearning.AI* team to better understand and serve their learner community (by identifying the key learner categories). |  |

So to summarize, a clustering algorithm, which is a type of unsupervised learning algorithm, *takes data without labels* and *groups* it into *meaningful* *clusters*. Besides clustering, there are *other types of unsupervised learning* as well. In the next section, we’ll explore some other unsupervised learning algorithms.

* **Cocktail Party Problem** (Unsupervised Learning)

The Cocktail Party Problem is an interesting example of an unsupervised learning task. The challenge is to *separate one person's voice* from a group of people all talking at the same time.

|  |  |
| --- | --- |
| * ***Problem:*** Imagine two people speaking, each with their own microphone. However, *both microphones pick up the voices of both individuals*. How can we separate their voices from the mixed audio? * ***How it s unsupervised?:*** Now, consider a noisy room with *many people talking simultaneously* and *multiple microphones* capturing sound from *different parts of the room*. * The *recordings* from the microphones will have *overlapping voices*, making it difficult to distinguish who is saying what. * Since there are *no labels (no individual voice tracks)*, this becomes an *Unsupervised Learning Problem*. * Since there are *no labeled voices* for each individual, we need an algorithm to *separate the voices*.   We just have different mics recording various parts of the *crowded room*, capturing *overlapping voices* of multiple people *speaking at the* *same time*. Our goal is to create an algorithm that can *separate the individual voices* from the *mixed audio*. |  |

* ***Future project:*** For instance, in a project scenario where *five people are speaking simultaneously*, and each microphone records the overlapping voices of all five people, how can we implement an algorithm to separate these voices? The result would be a clean recording of just one voice at a time.
* The type of algorithm we would use for this task is called **Independent Component Analysis (ICA)**. We’ll explore and implement ICA in our homework exercises.
* ***Language analysis:*** There are other examples of unsupervised learning as well: The internet contains vast amounts of unlabeled text data. By ***extracting*** this data from the ***internet***—without necessarily having labels—we can learn interesting things about ***language***. For example, learning analogies like:
* "man-woman," "king-queen,"
* "Tokyo-Japan," and "Washington DC-US"

is a form of unsupervised learning.

**UNSUPERVISED LEARNING :**

In this section, we introduced *unsupervised learning*, focusing on one type of it — *clustering*. Now, let's provide a more *formal* *definition of unsupervised learning* and briefly introduce other types beyond *clustering*.

*Formal Definition of Unsupervised Learning:*

In supervised learning, the data comes with both *inputs (x)* and corresponding *output labels (y)*. In unsupervised learning, however, the data includes only the *inputs (x)* *without* any *output labels (y)*. The algorithm must identify *patterns*, *structures*, or *interesting features* within the data *on its own*.

* ***Unlabeled Data:*** While there are only inputs (x) and no corresponding outputs (y), the data is called unlabeled data.
* Unlabeled data is raw information without any predefined categories, tags, or answers. It’s like having a pile of photos without any labels telling you what’s in them— algorithms task is to find patterns or groupings on its own.
* In the following chapters, we will explore *clustering* along with *two other types* of *unsupervised learning*:
* *Anomaly Detection:* Anomaly detection is used to *identify unusual events*. This is particularly important in *fraud detection* within financial systems, where *abnormal transactions* may indicate fraudulent activity. It also has applications in various other fields.
* *Dimensionality Reduction:* Dimensionality reduction allows *large datasets* to be compressed into a *smaller form* while *retaining* as much *essential information* as possible. This technique is useful for *simplifying complex data* and improving *efficiency* in *machine learning models*.
* *Clustering:* Previously, we discussed clustering, a type of unsupervised learning that *groups similar data points together* based on *patterns* in the *data* (e.g., customer segmentation).

If *anomaly detection* and *dimensionality reduction* don't make complete sense yet, there's no need to worry. We will cover them in detail in the following chapters.

*Note: While machine learning (ML) is incredibly useful today, most of the economic value created by ML is driven by supervised learning. However, there are still important applications for unsupervised learning.*

**1.4 REINFORCEMENT** learning

Reinforcement Learning (RL) is a type of machine learning that is used to *train robots*, *AI agents*, and even *autonomous vehicle (eg: cars, helicopters)* to perform complex tasks by using a system of ***rewards*** and ***penalties***.

|  |  |
| --- | --- |
| * **Stanford Autonomous Helicopter** (example of RL)   At Stanford, researchers used *reinforcement learning* to train an autonomous helicopter to perform *difficult maneuvers*, such as *flying upside* *down*.  You have *two controllers to fly*, but no one knows the exact, optimal way to control the helicopter to achieve these maneuvers (like flying a helicopter upside down). |  |

* ***How It Works:*** It's more like training a dog. When a dog does something correctly, you reward it (e.g., "Good Dog!"), and when it does something wrong, you give it a penalty (e.g., "Bad Dog!"). Over time, the dog learns to perform the "good" actions more frequently to earn more rewards. That's how you train a dog to complete a specific task.
* Similarly, in reinforcement learning, the AI ***agent*** or ***robot*** is given the *freedom to explore* and *perform various actions*. When it performs a desirable action (e.g., the helicopter flies smoothly), it receives a **reward** (-ve reward). When it makes a mistake (e.g., the helicopter flies erratically), it receives a **penalty**.
* The goal of the *Reinforcement Learning algorithm* is to *maximize* the *cumulative reward over time*, enabling the AI agent to learn how to *perform a task* more *optimally*.



**Other Examples of Reinforcement Learning :**

* ***Robotic Training:*** Reinforcement learning is used to train robots, such as ***robot dogs***, to *overcome obstacles* or *climb stairs* without needing to explicitly program each action. The robot receives *rewards* for *successful actions* and *penalties* for *failures*, gradually learning to optimize its behavior.
* We simply provide a *reward signal* and use a *learning algorithm* that figures out on its own how to *optimize the rewards* to climb obstacles or complete any task.
* ***Game Playing:*** One of the most famous applications of reinforcement learning is in *playing video games*. For example, AI agents have been trained to play Atari games or the game of Go (like AlphaGo) using reinforcement learning. The AI learns through *trial and error*, gradually improving its performance by *maximizing rewards* (e.g., high scores or wins).
* In video game playing, there are remarkable stunts and an AI can learn from those games and apply to real world.

Reinforcement Learning (RL) has proven to be highly effective in tasks like *game playing*, optimizing *robot movements*, and even controlling *autonomous vehicles* like helicopters. Its ability to learn from rewards and penalties makes it a powerful tool for solving *complex, real-world problems*.

* Is **Reinforcement** Learning an **unsupervised** learning?

Reinforcement learning (RL) is *not considered Unsupervised Learning*, though it shares *some similarities* with both supervised and unsupervised learning.

**What is Reinforcement Learning ?**

Reinforcement learning is a type of machine learning where an agent learns to make decisions by *performing actions* in an *environment* to maximize some notion of *cumulative reward*.

* The agent interacts with the environment,
* Receives feedback in the form of rewards or penalties, and
* Uses this feedback to learn and improve its actions over time.

**How is it Different from Supervised and Unsupervised Learning ?**

***Supervised Learning:*** In supervised learning, the model learns from *labeled data*, meaning it knows the *correct output for each input* and learns to *map inputs to outputs*. *RL does not require labeled input-output pairs;* instead, it learns from the ***consequences*** of ***actions***.

***Unsupervised Learning:*** In unsupervised learning, the model is provided with data that has *no labels* and tries to find *hidden patterns or structure* within the data. ***RL*** *is different because it involves learning through* ***trial*** *and* ***error****, guided by the* ***reward signal****, rather than just finding patterns*.

Key Differences

* Feedback: In *Supervised Learning*, feedback is *immediate* and *direct* (correct label), whereas in *Reinforcement Learning*, feedback (*rewards*) is often *delayed* and based on the outcome of a sequence of actions.
* Exploration and Exploitation: *Reinforcement learning* involves the challenge of exploration (trying *new actions*) vs. exploitation (using *known* *actions* to maximize reward), which is not *typically a concern* in *Supervised* or *Unsupervised* *learning*.
* Objective: The ***goal*** in *Reinforcement Learning* is to learn a ***policy*** that *maximizes the long-term reward*, whereas in *Supervised Learning*, the goal is to *minimize the error* in predictions, and in *Unsupervised Learning*, the goal is often to *uncover hidden patterns*.

So, while reinforcement learning isn't unsupervised learning, it occupies a unique space within machine learning that involves *learning from* *interaction* and *feedback* in a *dynamic environment*.

* **Is all UNSUPERVISED learning CLUSTERING?**

No, it isn't. *Clustering* is just one type of unsupervised learning. Unsupervised learning is a broader concept that involves working with ***unlabeled*** ***data***—meaning we only have the input data X, (without any corresponding labels Y). The goal is to uncover *interesting patterns*, *structures*, or *characteristics* within this data.

* While *clustering* is a popular method in unsupervised learning (where we group similar data points together), there are other unsupervised learning techniques as well, such as:
* Anomaly Detection,
* Dimensionality Reduction, and
* Association Rule Learning.

Each of these approaches seeks to analyze and understand the data in different ways, without relying on predefined labels.

**1.5** Machine learning **STRATEGY**

***Machine Learning Strategy (Learning Theory):*** This involves making *strategic decisions* during the implementation of ML algorithms to ensure their *effectiveness*. It's not just about applying a machine learning algorithm; it's about *applying it appropriately*.

* How to apply **appropriate** machine learning algorithm?

There can be a *significant difference* in the *effectiveness* of how two different ***ML teams*** implement the same learning algorithms. For instance, one team might have a faulty implementation, causing the algorithm to fail entirely.

* Most skilled ML practitioners are *strategic* in their approach. They are skilled at making decisions such as:
* Should we collect more data? If so, what type of data?
* Should we try a different learning algorithm?
* Should we use faster GPUs to train the algorithm for a longer period?
* How do we choose the appropriate architecture, whether it's *SVM*, *Logistic Regression*, or *Neural Networks*?

Our goal is to approach ML as a ***systematic engineering discipline***, allowing us to make informed decisions in future projects and apply the right steps to solve problems using ML.

* ***Software Engineering Analogy:*** Just as it’s not a good idea to *delete all lines of code* (in C++, for example) that contain *syntax errors*, as it would be *poor debugging practice*, running a *learning algorithm* rarely works perfectly on the *first try*. How you go about *debugging the learning algorithm* greatly impacts your efficiency and how quickly you can build *effective learning systems*.
* Today, creating an effective learning algorithm is often seen as a "**black magic**" process, relying heavily on *years of experience*. For example, an experienced programmer intuitively knows how to implement an effective program. However, our goal is to *move beyond* this experience-based *"black magic"* and towards a *systematic engineering process*.
* We will discuss more detail about this in the *"Machine Learning Strategy (Learning Theory)"* section. We'll explore systematic tools that make the process more strategic, *enabling us to lead a team (or ourselves) to build effective learning systems* and avoid *wasting time* on wrong approaches.
* *Optimizing code to run faster* ***isn’t*** always the *primary concern* in ML development. Less experienced software engineers might focus on *optimizing* *code speed*, but more experienced ones will use a profiler to identify which *part of the code* is truly the *bottleneck* and focus on *optimizing that specific* *part*.

**1.6 JUPYTER**

From the previous sections, you've learned about *supervised* and *unsupervised learning*, along with *examples* of both. To *deepen* your *understanding*, it's important to *engage interactively*—perhaps by *experimenting with code* yourself to *implement* these *concepts*.

* Jupyter Notebooks (**google colab** setting)

One of the most widely used tools among machine learning and data science practitioners today is Jupyter Notebook. This is the default environment many *professionals* use for *coding*, *experimenting*, and *testing* ideas. Throughout this textbook, directly from your web browser, you'll be working within a real *Jupyter Notebook environment* to explore these concepts hands-on.

* This isn’t a simplified or simulated setup—it’s the exact same Jupyter Notebook environment that *developers and data scientists use* in leading tech companies and research institutions worldwide.

**GuideLab :**

*GuideLabs* appear throughout this textbook, allowing you to run code step by step without needing to write any yourself. Simply open the *ipynb* file and execute the provided code. By following along, you'll see *how machine learning code works* in practice. These labs are designed to be completed quickly by running each line from top to bottom.

Working through *GuideLabs* helps you get familiar with machine learning algorithms and code. In the next chapters, you'll also find practice labs where you can write code yourself.

* Let's look at a Jupyter Notebook example. You'll see two types of blocks, called cells.
* ***Markdown cells:*** These contain text explaining the code. You can edit them if needed.
* ***Code cells:*** These contain actual code. To run a code cell, press Shift + Enter.

If you click on a *Markdown cell* and see *raw text* instead of formatted content, pressing Shift + Enter will format it properly again.

**Quiz 1 :**

Of the following examples, which would you address using an unsupervised learning algorithm? 2 are unsupervised and another two are supervised.

* Given **email** labeled as **spam/not spam**, learn a spam filter.
* Given a set of **news articles** found on the web, **group** them into **sets** of articles about the same story.
* Given a **database of customer data**, automatically discover **market segments** and group customers into different market segments.
* Given a dataset of **patients** diagnosed as either having **diabetes or not**, learn to classify new patients as having diabetes or not
* Unsupervised Learning (no labels, find hidden patterns):
* *Group news articles* into sets about the same story.

Goal: Discover *natural groupings* without pre-defined labels.

* *Discover market segments* in customer data and group them.

Goal: Find *hidden structures* (e.g., customer clusters) without prior labels.

We can handle these two tasks using a clustering algorithm— to group news articles together or to perform market segmentation by grouping customers into distinct segments.

* Supervised Learning (labeled data):
* *Given emails labeled as spam/not spam*, learn a spam filter. Uses labeled data (spam/not spam labels).
* *Given patients diagnosed as diabetic/non-diabetic*, classify new patients. Uses labeled data (diabetes labels).

**Quiz 2 :**

Which are the two common types of supervised learning? (Choose two)

* Regression \*
* Classification \*
* Clustering

**Quiz 3 :**

Which of these is a type of unsupervised learning?

* Clustering
* Regression
* Classification \*

In the upcoming chapters, we'll cover five major topics

Supervised Learning

Unsupervised Learning

ML Strategy

Deep Learning

Reinforcement Learning