Module – I
Introduction

- Data objects and Attribute types
- Overview of Machine Learning Algorithms
- Basics of Supervised and Unsupervised Algorithms.
- Well posed learning problems
- Perspectives and issues in Machine Learning
- Concept Learning: Concept learning task
- Concept learning as search
- Find-S algorithm
- Version Space
- Candidate Elimination Algorithm

Data Objects

The Type of Data

Data sets differ in a number of ways. For example, the attributes used to describe data objects can be of different types—quantitative or qualitative—and data sets may have special characteristics; e.g., some data sets contain time series or objects with explicit relationships to one another. Not surprisingly, the type of data determines which tools and techniques can be used to analyze the data. Furthermore, new research in data mining is often driven by the need to accommodate new application areas and their new types of data.

The Quality of the Data Data is often far from perfect. While most data mining techniques can tolerate some level of imperfection in the data, a focus on understanding and improving data quality typically improves the quality of the resulting analysis. Data quality issues that often need to be addressed include the presence of noise and outliers; missing, inconsistent, or duplicate data; and data that is biased or, in some other way, unrepresentative of the phenomenon or population that the data is supposed to describe.

Preprocessing Steps to Make the Data More Suitable for Data Mining

Often, the raw data must be processed in order to make it suitable for analysis. While one objective may be to improve data quality, other goals focus on modifying the data so that it better fits a specified data mining technique or tool. For example, a continuous attribute, e.g., length, may need to be transformed into an attribute with discrete categories, e.g., short, medium, or long, in order to apply a particular technique. As another example, the number of attributes in a data set is often reduced because many techniques are more effective when the data has a relatively small number of attributes.

Attribute Types

Attribute: An **attribute** is a property or characteristic of an object that may vary, either from one object to another or from one time to another.

For example, eye color varies from person to person, while the temperature of an object varies over time. Note that eye color is a symbolic attribute with a small number of possible values {brown, black, blue, green, hazel, etc.}, while temperature is a numerical attribute with a potentially unlimited number of values.

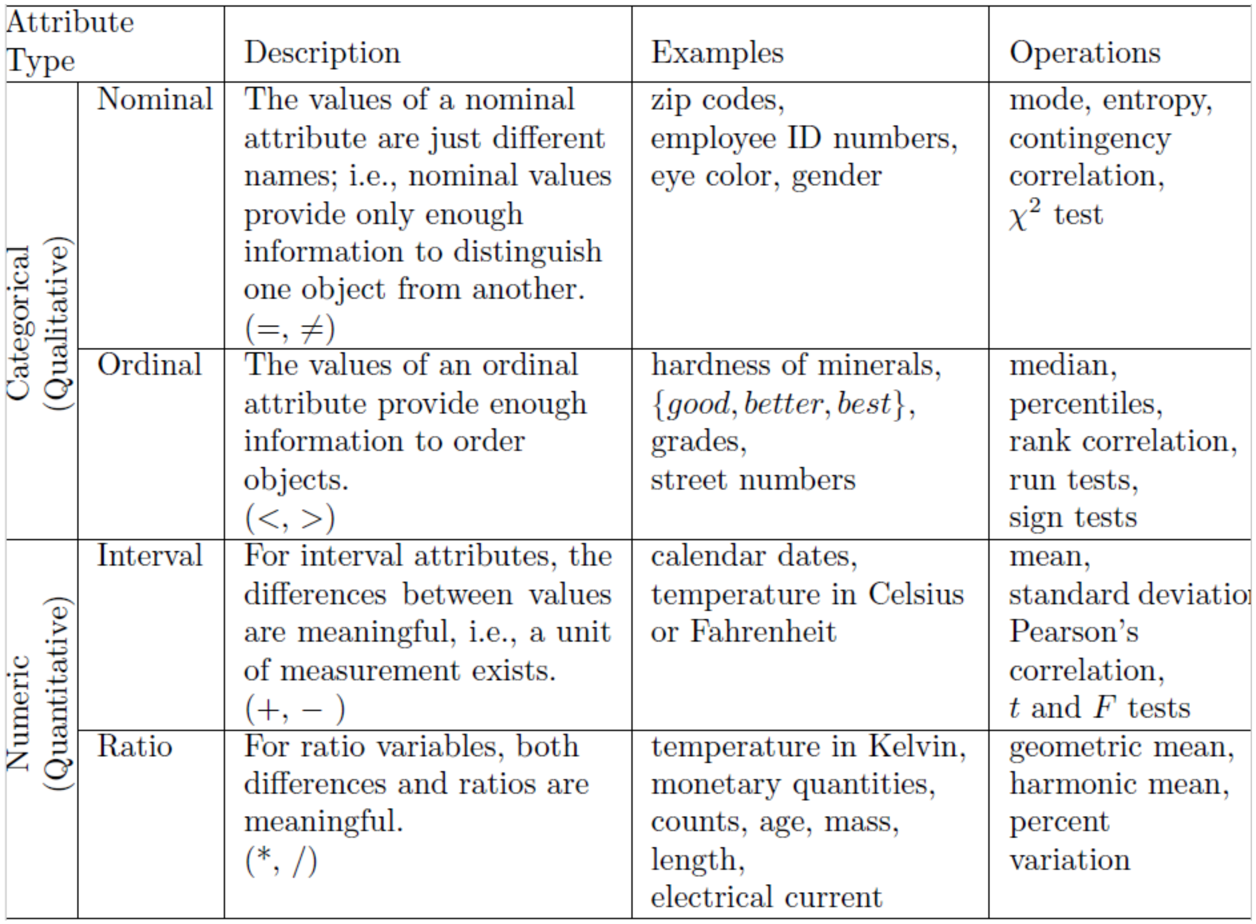
The Different Types of Attributes

A useful (and simple) way to specify the type of an attribute is to identify the properties of numbers that correspond to underlying properties of the attribute. For example, an attribute such as length has many of the properties of numbers. It makes sense to compare and order objects by length, as well as to talk about the differences and ratios of length. The following properties (operations) of numbers are typically used to describe attributes.

1. Distinctness = and not equal to
2. Order $<$, $\leq$, $>$, and $\geq$
3. Addition $+$ and $-$
4. Multiplication $*$ and $/$

Given these properties, we can define four types of attributes: **nominal, ordinal, interval, and ratio.** Table below gives the definitions of these types, along with information about the statistical operations that are valid for each type. Each attribute type possesses all of the properties and operations of the attribute types above it. Consequently, any property or operation that is valid for nominal, ordinal, and interval attributes is also valid for ratio attributes. In other words, the definition of the attribute types is cumulative. However, this does not mean that the operations appropriate for one attribute type are appropriate for the attribute types above it.

Table 1.1. Different Attribute Types



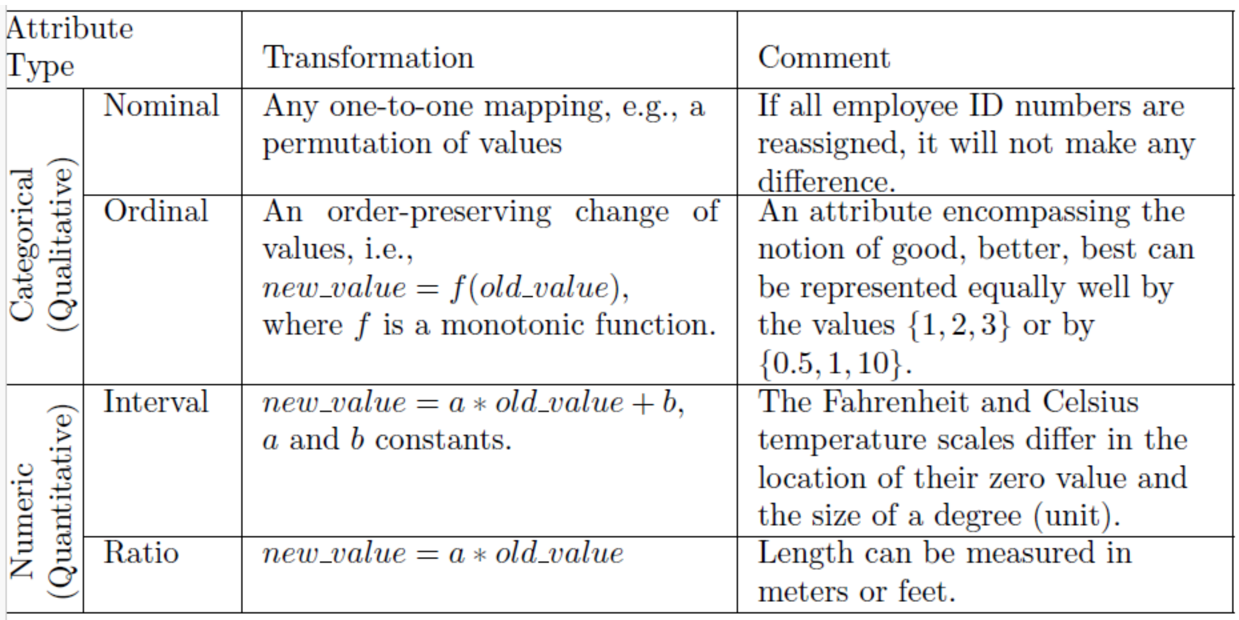
Attribute Type		Description	Examples	Operations
Categorical (Qualitative)	Nominal	The values of a nominal attribute are just different names; i.e., nominal values provide only enough information to distinguish one object from another. $(=, \neq)$	zip codes, employee ID numbers, eye color, gender	mode, entropy, contingency correlation, χ^2 test
	Ordinal	The values of an ordinal attribute provide enough information to order objects. $(<, >)$	hardness of minerals, $\{good, better, best\}$, grades, street numbers	median, percentiles, rank correlation, run tests, sign tests
Numeric (Quantitative)	Interval	For interval attributes, the differences between values are meaningful, i.e., a unit of measurement exists. $(+, -)$	calendar dates, temperature in Celsius or Fahrenheit	mean, standard deviatio Pearson's correlation, t and F tests
	Ratio	For ratio variables, both differences and ratios are meaningful. (*, /)	temperature in Kelvin, monetary quantities, counts, age, mass, length, electrical current	geometric mean, harmonic mean, percent variation

Nominal and **ordinal attributes are collectively referred to as categorical** or **qualitative attributes**. As the name suggests, qualitative attributes, such as employee ID, lack most of the properties of numbers. Even if they are represented by numbers, i.e., integers, they should be treated more like symbols. The remaining two types of attributes, **interval and ratio, are**

collectively referred to as quantitative or numeric attributes. Quantitative attributes are represented by numbers and have most of the properties of numbers. Note that quantitative attributes can be integer-valued or continuous.

The types of attributes can also be described in terms of transformations that do not change the meaning of an attribute. For example, the meaning of a length attribute is unchanged if it is measured in meters instead of feet. The statistical operations that make sense for a particular type of attribute are those that will yield the same results when the attribute is transformed using a transformation that preserves the attribute's meaning. To illustrate, the average length of a set of objects is different when measured in meters rather than in feet, but both averages represent the same length. Table 1.2 shows the permissible (meaning-preserving) transformations for the four attribute types of Table 1.1.

Table 1.2. Transformations that define attribute levels



Attribute Type		Transformation	Comment
Categorical (Qualitative)	Nominal	Any one-to-one mapping, e.g., a permutation of values	If all employee ID numbers are reassigned, it will not make any difference.
	Ordinal	An order-preserving change of values, i.e., $new\_value = f(old\_value)$, where f is a monotonic function.	An attribute encompassing the notion of good, better, best can be represented equally well by the values $\{1, 2, 3\}$ or by $\{0.5, 1, 10\}$.
Numeric (Quantitative)	Interval	$new\_value = a * old\_value + b$, a and b constants.	The Fahrenheit and Celsius temperature scales differ in the location of their zero value and the size of a degree (unit).
	Ratio	$new\_value = a * old\_value$	Length can be measured in meters or feet.

Describing Attributes by the Number of Values

An independent way of distinguishing between attributes is by the number of values they can take.

Discrete A discrete attribute has a finite or countably infinite set of values. Such attributes can be categorical, such as zip codes or ID numbers, or numeric, such as counts. Discrete attributes are often represented using integer variables.

Binary attributes are a special case of discrete attributes and assume only two values, e.g., true/false, yes/no, male/female, or 0/1. Binary attributes are often represented as Boolean variables, or as integer variables that only take the values 0 or 1.

Continuous A continuous attribute is one whose values are real numbers. Examples include attributes such as temperature, height, or weight. Continuous attributes are typically represented as floating-point variables. Practically, real values can only be measured and represented with limited precision.

What is Machine Learning

In the real world, we are surrounded by humans who can learn everything from their experiences with their learning capability, and we have computers or machines which work on our instructions. But can a machine also learn from experiences or past data like a human does? So here comes the role of **Machine Learning**.

Introduction to Machine Learning

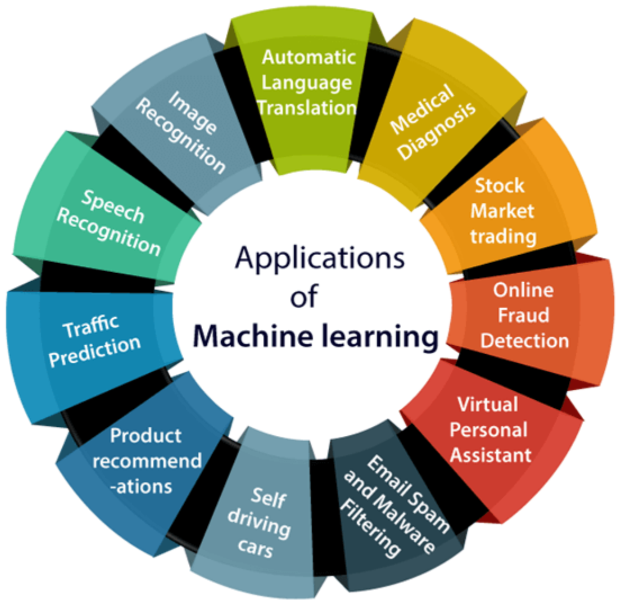
A subset of artificial intelligence known as machine learning focuses primarily on the creation of algorithms that enable a computer to independently learn from data and previous experiences.

Without being explicitly programmed, machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things.

Machine learning algorithms create a mathematical model that, without being explicitly programmed, aids in making predictions or decisions with the assistance of sample historical data, or training data. For the purpose of developing predictive models, machine learning brings together statistics and computer science. Algorithms that learn from historical data are either constructed or utilized in machine learning. The performance will rise in proportion to the quantity of information we provide.

Applications of Machine learning

Machine learning is a buzzword for today's technology, and it is growing very rapidly day by day. We are using machine learning in our daily life even without knowing it such as Google Maps, Google assistant, Alexa, etc. Below are some most trending real-world applications of Machine Learning:



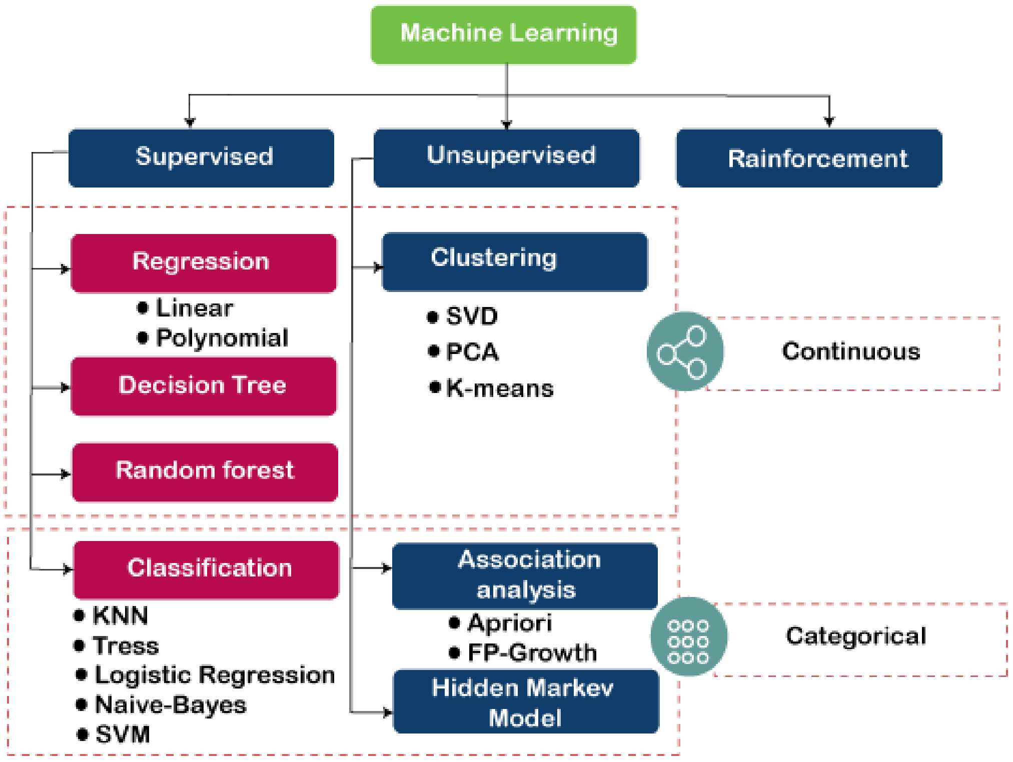
Machine Learning Algorithms

Machine Learning algorithms are the programs that can learn the hidden patterns from the data, predict the output, and improve the performance from experiences on their own. Different algorithms can be used in machine learning for different tasks.

Types of Machine Learning Algorithms

Machine Learning Algorithm can be broadly classified into three types:

- Supervised Learning Algorithms
- Unsupervised Learning Algorithms
- Reinforcement Learning algorithm



1) Supervised Learning Algorithm

Supervised learning is a type of Machine learning in which the machine needs external supervision to learn. The supervised learning models are trained using the labeled dataset. Once the training and processing are done, the model is tested by providing a sample test data to check whether it predicts the correct output. The goal of supervised learning is to map input data with the output data. Supervised learning is based on supervision, and it is the same as when a student learns things in the teacher's supervision.

Supervised learning can be divided further into two categories of problem:

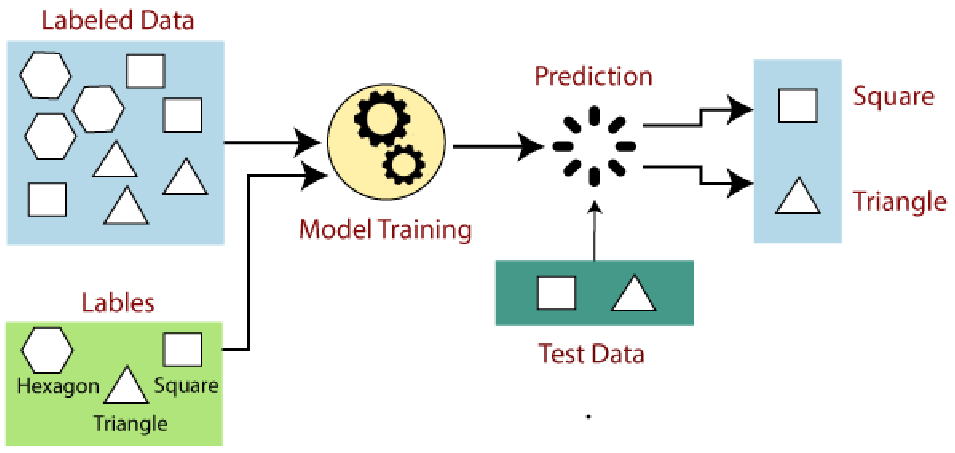
- **Classification**
- **Regression**

Examples of some popular supervised learning algorithms are Simple Linear regression, Decision Tree, Logistic Regression, KNN algorithm, etc.

Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output. In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher. Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to find a mapping function to map the input variable(x) with the output variable(y). In the real-world, supervised learning can be used for Risk Assessment, Image classification, Fraud Detection, spam filtering, etc.

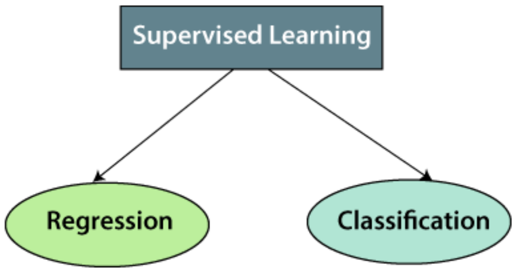
How Supervised Learning Works?

In supervised learning, models are trained using labelled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output. The working of Supervised learning can be easily understood by the below example and diagram:



Types of supervised Machine learning Algorithms:

Supervised learning can be further divided into two types of problems:



1. Regression

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

- Linear Regression
- Regression Trees
- Non-Linear Regression
- Bayesian Linear Regression
- Polynomial Regression

2. Classification

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

- Random Forest
- Decision Trees
- Logistic Regression
- Support vector Machines

Unsupervised Machine Learning

In the previous topic, we learned supervised machine learning in which models are trained using labeled data under the supervision of training data. But there may be many cases in which we do not have labeled data and need to find the hidden patterns from the given dataset. So, to solve such types of cases in machine learning, we need unsupervised learning techniques.

What is Unsupervised Learning?

As the name suggests, unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things.

Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to **find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format**.

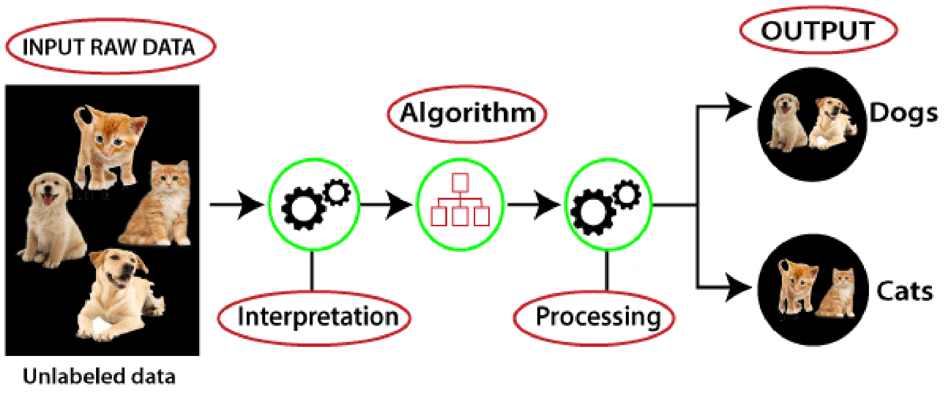
Example: Suppose the unsupervised learning algorithm is given an input dataset containing images of different types of cats and dogs. The algorithm is never trained upon the given dataset, which means it does not have any idea about the features of the dataset. The task of the unsupervised learning algorithm is to identify the image features on their own. Unsupervised learning algorithm will perform this task by clustering the image dataset into the groups according to similarities between images.

Working of Unsupervised Learning

Working of unsupervised learning can be understood by the below diagram:

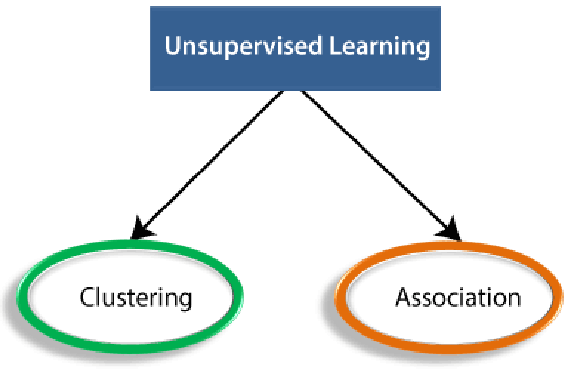
Here, we have taken an unlabeled input data, which means it is not categorized and corresponding outputs are also not given. Now, this unlabeled input data is fed to the machine learning model in order to train it. Firstly, it will interpret the raw data to find the hidden patterns from the data and then will apply suitable algorithms such as k-means clustering, etc.

Once it applies the suitable algorithm, the algorithm divides the data objects into groups according to the similarities and difference between the objects.



Types of Unsupervised Learning Algorithm:

The unsupervised learning algorithm can be further categorized into two types of problems:



Clustering: Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.

Association: An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

Below is the list of some popular unsupervised learning algorithms:

- K-means clustering
- Hierarchal clustering
- Anomaly detection
- Neural Networks
- Principle Component Analysis
- Independent Component Analysis
- Apriori algorithm
- Singular value decomposition

Well posed Learning Problems

Definition: A computer program is said to learn from **experience E** with respect to some **class of tasks T** and **performance measure P,** if its performance at tasks in T, as measured by P, improves with experience E.

For example, a computer program that learns to play checkers might improve its performance as measured *by its ability to win* at the class of tasks involving *playing checkers games*, through experience *obtained by playing games against itself*.

In general, to have a well-defined learning problem, we must identity these **three features**: the class of tasks, the measure of performance to be improved, and the source of experience.

A checkers learning problem:

Task T: playing checkers

Performance measure P: percent of games won against opponents

Training experience E: playing practice games against itself

A handwriting recognition learning problem:

Task T: recognizing and classifying handwritten words within images

Performance measure P: percent of words correctly classified

Training experience E: a database of handwritten words with given classifications

A robot driving learning problem:

Task T: driving on public four-lane highways using vision sensors

Performance measure P: average distance traveled before an error (as judged by human)

Training experience E: a sequence of images and steering commands recorded while observing a human driver

Perspectives and Issues in Machine Learning

One useful perspective on machine learning is that it involves searching a very large space of possible hypotheses to determine one that best fits the observed data.

Issues in Machine Learning

- What algorithms exist for learning general target functions from specific training examples?
- Which algorithms perform best for which types of problems and representations?
- How much training data is sufficient?
- When and how can prior knowledge held by the learner guide the process of generalizing from examples?
- What is the best strategy for choosing a useful next training experience, and how does the choice of this strategy alter the complexity of the learning problem?
- What is the best way to reduce the learning task to one or more function approximation problems? What specific functions should the system attempt to learn? Can this process itself be automated?
- How can the learner automatically alter its representation to improve its ability to represent and learn the target function?

Concept Learning

Concept learning can be formulated as a problem of searching through a predefined space of potential hypotheses for the hypothesis that best fits the training examples.

Definition: Concept learning. Inferring a Boolean-valued function from training examples of its input and output.

A Concept Learning Task

Consider the example task of learning the target concept "days on which Aldo enjoys his favorite water sport." Table 2.1 describes a set of example days, each represented by a set of ***attributes***. The attribute ***EnjoySport*** indicates whether or not Aldo enjoys his favorite water sport on this day. The task is to learn to predict the value of ***EnjoySport*** for an arbitrary day, based on the values of its other attributes.

Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

TABLE 2.1
Positive and negative training examples for the target concept *EnjoySport*.

Consider a simple representation in which each hypothesis consists of a conjunction of constraints on the instance attributes. In particular, let each hypothesis be a vector of six constraints, specifying the values of the six attributes Sky, AirTemp, Humidity, Wind, Water, and Forecast. For each attribute, the hypothesis will either

- indicate by a "?' that any value is acceptable for this attribute,
- specify a single required value (e.g., Warm) for the attribute, or
- indicate by a "0" that no value is acceptable.

If some instance x satisfies all the constraints of hypothesis h, then h classifies x as a positive example (h(x) = 1). To illustrate, the hypothesis that Aldo enjoys his favorite sport only on cold days with high humidity (independent of the values of the other attributes) is represented by the expression (?, Cold, High, ?, ?, ?).

The most general hypothesis-that every day is a positive example-is represented by
(?, ?, ?, ?, ?, ?)
and the most specific possible hypothesis-that no day is a positive example-is represented by
(Ø, Ø, Ø, Ø, Ø, Ø).

To summarize, the EnjoySport concept learning task requires learning the set of days for which EnjoySport = yes, describing this set by a conjunction of constraints over the instance attributes. In general, any concept learning task can be described by the set of instances over which the target function is defined, the target function, the set of candidate hypotheses considered by the learner, and the set of available training examples. The definition of the EnjoySport concept learning task in this general form is given in Table 2.2.

- **Given:**
  - Instances X: Possible days, each described by the attributes
    - *Sky* (with possible values *Sunny*, *Cloudy*, and *Rainy*),
    - *AirTemp* (with values *Warm* and *Cold*),
    - *Humidity* (with values *Normal* and *High*),
    - *Wind* (with values *Strong* and *Weak*),
    - *Water* (with values *Warm* and *Cool*), and
    - *Forecast* (with values *Same* and *Change*).
  - Hypotheses H: Each hypothesis is described by a conjunction of constraints on the attributes *Sky*, *AirTemp*, *Humidity*, *Wind*, *Water*, and *Forecast*. The constraints may be "?" (any value is acceptable), "Ø" (no value is acceptable), or a specific value.
  - Target concept c: $EnjoySport : X \rightarrow \{0, 1\}$
  - Training examples D: Positive and negative examples of the target function (see Table 2.1).
- **Determine:**
  - A hypothesis h in H such that $h(x) = c(x)$ for all x in X.

TABLE 2.2
The *EnjoySport* concept learning task.

Notation

We employ the following terminology when discussing concept learning problems. The set of items over which the concept is defined is called the **set of instances**, which we denote by **X**. In the current example, X is the set of all possible days, each represented by the attributes Sky, AirTemp, Humidity, Wind, Water, and Forecast. The concept or function to be learned is called the **target concept**, which we denote by c. In general, c can be any boolean valued function defined over the instances X; that is, c : X → {O, 1). In the current example, the target concept corresponds to the value of the attribute EnjoySport (i.e., c(x) = 1 if EnjoySport = Yes, and c(x) = 0 if EnjoySport = No).

When learning the target concept, the learner is presented a set of ***training examples,*** each consisting of an instance ***x*** from X, along with its target concept value ***c(x)*** (e.g., the training examples in Table 2.1). Instances for which ***c(x)*** = 1 are called ***positive examples,*** or members of the target concept. Instances for which ***C(X)*** = 0 are called ***negative examples,*** or nonmembers of the target concept. We will often write the ordered pair ***(x, c(x))*** to describe the training example consisting of the instance **x** and its target concept value ***c(x).*** We use the symbol ***D*** to denote the set of available training examples.

Given a set of training examples of the target concept ***c,*** the problem faced by the learner is to hypothesize, or estimate, ***c.*** We use the symbol H to denote the set of ***all possible hypotheses*** that the learner may consider regarding the identity of the target concept. Usually H is determined by the human designer's choice of hypothesis representation. In general, each hypothesis ***h*** in H represents a boolean-valued function defined over X; that is, ***h*** : X → {O, 1). The goal of the learner is to find a hypothesis ***h*** such that ***h(x)*** = ***c(x)*** for all ***x*** in X.

Concept Learning as Search

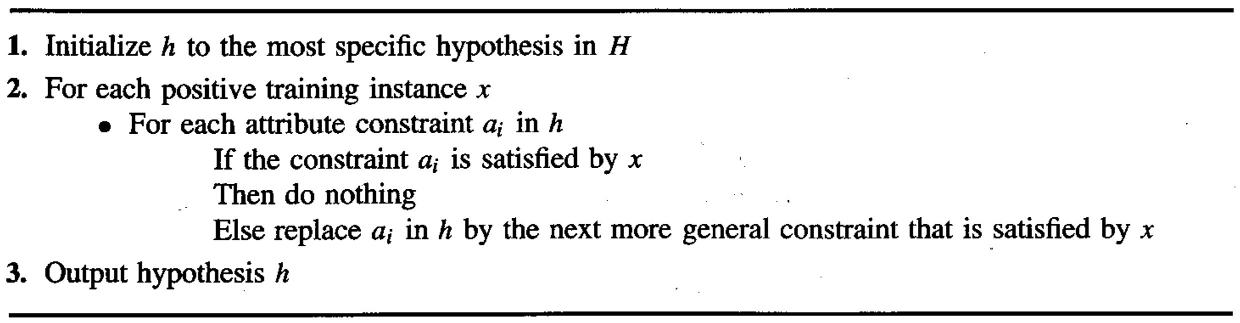
Concept learning can be viewed as the task of searching through a large space of hypotheses implicitly defined by the hypothesis representation. The goal of this search is to find the hypothesis that best fits the training examples. It is important to note that by selecting a hypothesis representation, the designer of the learning algorithm implicitly defines the space of all hypotheses that the program can ever represent and therefore can ever learn.

Consider, for example, the instances X and hypotheses H in the ***EnjoySport*** learning task. Given that the attribute ***Sky*** has three possible values, and that ***AirTemp, Humidity, Wind, Water,*** and ***Forecast*** each have two possible values, the instance space X contains exactly 3 .2 .2 .2.2 .2 = **96** distinct instances. **A** similar calculation shows that there are **5.4.4.4.4.4** = **5120 syntactically distinct hypotheses** within H.

Every hypothesis contains one or more "Ø" symbols that represents the empty set of instances; that is, it classifies every instance as negative. Therefore, the number of **semantically distinct hypotheses** is only 1 + **(4.3.3.3.3.3)** = **973.**

FIND-S Algorithm (Finding a Maximally Specific Hypothesis)

In this algorithm, begin with the most specific possible hypothesis in H, then generalize this hypothesis each time it fails to cover an observed positive training example. (We say that a hypothesis "covers" a positive example if it correctly classifies the example as positive.)



```
1. Initialize $h$ to the most specific hypothesis in $H$
2. For each positive training instance $x$
       • For each attribute constraint $a_i$ in $h$
               If the constraint $a_i$ is satisfied by $x$
               Then do nothing
               Else replace $a_i$ in $h$ by the next more general constraint that is satisfied by $x$
3. Output hypothesis $h$
```

TABLE 2.3
FIND-S Algorithm.

[Problems solved in class]

Limitations of Find-S

- Has the learner converged to the correct target concept? Although FIND-S will find a hypothesis consistent with the training data, it has no way to determine whether it has found the only hypothesis in H consistent with the data (i.e., the correct target concept), or whether there are many other consistent hypotheses as well.
- Why prefer the most specific hypothesis? In case there are multiple hypotheses consistent with the training examples, FIND-S will find the most specific. It is unclear whether we should prefer this hypothesis over, say, the most general, or some other hypothesis of intermediate generality.
- Are the training examples consistent? In most practical learning problems, there is some chance that the training examples will contain at least some errors or noise. Such inconsistent sets of training examples can severely mislead FIND-S, given the fact that it ignores negative examples.
- What if there are several maximally specific consistent hypotheses? In the hypothesis language H for the EnjoySport task, there is always a unique, most specific hypothesis

consistent with any set of positive examples. However, for other hypothesis spaces there can be several maximally specific hypotheses consistent with the data.

Version Spaces and the Candidate Elimination Algorithm

Definition: A hypothesis h is **consistent** with **a** set of training examples ***D*** if **and** only if ***h(x)*** = ***c(x)*** for each example (x, ***c(x))*** in ***D.***

$$Consistent(h, D) \equiv (\forall \langle x, c(x) \rangle \in D)\ h(x) = c(x)$$

The CANDIDATE-ELIMINATION Algorithm represents the set of all hypotheses consistent with the observed training examples. This subset of all hypotheses is called the ***version space*** with respect to the hypothesis space H and the training examples D, because it contains all plausible versions of the target concept.

Definition: The **version space,** denoted ***VS$_{H,D}$*** with respect to hypothesis space ***H*** and training examples D, is the subset of hypotheses from ***H*** consistent with the training examples in D.

$$VS_{H,D} \equiv \{h \in H | Consistent(h, D)\}$$

The LIST-THEN-ELIMINATE Algorithm

The **LIST-THEN-ELIMINATE** algorithm first initializes the version space to contain all hypotheses in H, then eliminates any hypothesis found inconsistent with any training example. The version space of candidate hypotheses thus shrinks as more examples are observed, until ideally just one hypothesis remains that is consistent with all the observed examples.

The LIST-THEN-ELIMINATE Algorithm

```
1. $VersionSpace \leftarrow$ a list containing every hypothesis in $H$
2. For each training example, $\langle x, c(x) \rangle$
        remove from $VersionSpace$ any hypothesis $h$ for which $h(x) \neq c(x)$
3. Output the list of hypotheses in $VersionSpace$
```

TABLE 2.4
The LIST-THEN-ELIMINATE algorithm.

[Examples done in Class]

CANDIDATE-ELIMINATION Learning Algorithm

The CANDIDATE-Elimination Algorithm computes the version space containing all hypotheses from H that are consistent with an observed sequence of training examples. It begins by initializing the version space to the set of all hypotheses in H; that is, by initializing the G boundary set to contain the most general hypothesis in H

$$G_0 \leftarrow \{\langle ?, ?, ?, ?, ?, ? \rangle\}$$

and initializing the S boundary set to contain the most specific (least general) hypothesis

$$S_0 \leftarrow \{\langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle\}$$

These two boundary sets delimit the entire hypothesis space, because every other hypothesis in H is both more general than ***So*** and more specific than ***Go.*** As each training example is considered, the ***S*** and ***G*** boundary sets are generalized and specialized, respectively, to eliminate from the version space any hypotheses found inconsistent with the new training example. After all examples have been processed, the computed version space contains all the hypotheses consistent with these examples and only these hypotheses. This algorithm is summarized in Table 2.5.

```
Initialize G to the set of maximally general hypotheses in H
Initialize S to the set of maximally specific hypotheses in H
For each training example d, do
• If d is a positive example
    • Remove from G any hypothesis inconsistent with d
    • For each hypothesis s in S that is not consistent with d
        • Remove s from S
        • Add to S all minimal generalizations h of s such that
            • h is consistent with d, and some member of G is more general than h
        • Remove from S any hypothesis that is more general than another hypothesis in S
• If d is a negative example
    • Remove from S any hypothesis inconsistent with d
    • For each hypothesis g in G that is not consistent with d
        • Remove g from G
        • Add to G all minimal specializations h of g such that
            • h is consistent with d, and some member of S is more specific than h
        • Remove from G any hypothesis that is less general than another hypothesis in G
```

TABLE 2.5
CANDIDATE-ELIMINATION algorithm using version spaces. Notice the duality in how positive and negative examples influence S and G.

[Example problems done in class]