UNIT I

Ashok Kumar Yadav

Jan 2018

1 Concept of learning system

Machine learning is a subfield of computer science, but is often also referred to as predictive analytics, or predictive modeling. Its goal and usage is to build new and/or leverage existing algorithms to learn from data, in order to build generalizable models that give accurate predictions, or to find patterns, particularly with new and unseen similar data.
A computer program is said to learn from experience E with respect to some class of task T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.
A checkers learning problem:

- Task T: playing checkers
- Performance measure P: percentage 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 highway using vision sensors
- Performance measure P: average distance travelled before an error
- Training experience E: a sequence of image and steering commands recorded while observing a human driver.

2 Goals of Machine Learning

The primary goal of machine learning research is to develop general purpose algorithms of practical value. Such algorithms should be efficient. As usual, as computer scientists, we care about time and space efficiency. But in the context of learning, we also care a great deal about another precious resource, namely, the amount of data that is required by the learning algorithm. Learning algorithms should also be as general purpose as possible. We are looking for algorithms that can be easily applied to a broad class of learning problems, such as those listed above.

Of primary importance, we want the result of learning to be a prediction rule that is as accurate as possible in the predictions that it makes. Occasionally, we may also be interested in the interpretability of the prediction rules produced by learning. In other words, in some contexts (such as medical diagnosis), we want the computer to find prediction rules that are easily understandable by human experts.

Machine learning can be thought of as "programming by example." What is the advantage of machine learning over direct programming? First, the results of using machine learning are often more accurate than what can be created through direct programming. The reason is that machine learning algorithms are data driven, and are able to examine large amounts of data. On the other hand, a human expert is likely to be guided by imprecise impressions or perhaps an examination of only a relatively small number of examples. Imagine a dataset as a table, where the rows are each observation (aka measurement, data point, etc), and the columns for each observation represent the features of that observation and their values. At the outset of a machine learning project, a dataset is usually split into two or three subsets. The minimum subsets are the training and test datasets, and often an optional third validation dataset is created as well. Once these data subsets are created from the primary dataset, a predictive model or classifier is trained using the training data, and then the model's predictive accuracy is determined using the test data. As mentioned, machine learning leverages algorithms to automatically model and find patterns in data, usually with the goal of predicting some target output or response. These algorithms are heavily based on statistics and mathematical optimization. Optimization is the process of finding the smallest or largest value (minima or maxima) of a function, often referred to as a loss, or cost function in the minimization case. One of the most popular optimization algorithms used in machine learning is called gradient descent, and another is known as the the normal equation. In a nutshell, machine learning is all about automatically learning a highly accurate predictive or classifier model, or finding unknown patterns in data, by leveraging learning algorithms and optimization techniques.

3 Applications of Machine Learning

Machine learning algorithms are used primarily for the following types of output:

- Clustering (Unsupervised)
- Two-class and multi-class classification (Supervised)
- Regression: Univariate, Multivariate, etc. (Supervised)
- Anomaly detection (Unsupervised and Supervised)
- Recommendation systems (aka recommendation engine)

Specific algorithms that are used for each output type are discussed in the next section, but first, let's give a general overview of each of the above output, or problem types. As discussed, clustering is an unsupervised technique for discovering the composition and structure of a given set of data. It is a process of clumping data into clusters to see what groupings emerge, if any. Each cluster is characterized by a contained set of data points, and a cluster centroid. The cluster centroid is basically the mean (average) of all of the data points that the cluster contains, across all features. Classification problems involve placing a data point (aka observation) into a pre-defined class or category. Sometimes classification problems simply assign a class to an observation, and in other cases the goal is to estimate the probabilities that an observation belongs to each of the given classes. A great example of a two-class classification is assigning the class of Spam or Ham to an incoming email, where ham just means 'not spam'. Multi-class classification just means more than two possible classes. So in the spam example, perhaps a third class would be 'Unknown'. Regression is just a fancy word for saying that a model will assign a continuous value (response) to a data observation, as opposed to a discrete class. A great example of this would be predicting the closing price of the Dow Jones Industrial Average on any given day. This value could be any number, and would therefore be a perfect candidate for regression. Note that sometimes the word regression is used in the name of an algorithm that is actually used for classification problems, or to predict a discrete categorical response (e.g., spam or ham). A good example is logistic regression, which predicts probabilities of a given discrete value. Another problem type is anomaly detection. While we'd love to think that data is well behaved and sensible, unfortunately this is often not the case. Sometimes there are erroneous data points due to malfunctions or errors in measurement, or sometimes due to fraud. Other times it could be that anomalous measurements are indicative of a failing piece of hardware or electronics. Sometimes anomalies are indicative of a real problem and are not easily explained, such as a manufacturing defect, and in this case, detecting anomalies provides a measure of quality control, as well as insight into whether steps taken to reduce defects have worked or not. In either case, there are times where it is beneficial to find these anomalous values, and certain machine learning algorithms can be used to do just that. The final type of problem is addressed with a recommendation system, or also called recommendation engine. Recommendation systems are a type of information filtering system, and are intended to make recommendations in many applications, including movies, music, books, restaurants, articles, products, and so on. The two most common approaches are

content-based and collaborative filtering. Two great examples of popular recommendation engines are those offered by Netflix and Amazon. Netflix makes recommendations in order to keep viewers engaged and supplied with plenty of content to watch. In other words, to keep people using Netflix. They do this with their "Because you watched ... ", "Top Picks for Alex", and "Suggestions for you" recommendations. Amazon does a similar thing in order to increase sales through up-selling, maintain sales through user engagement, and so on. They do this through their "Customers Who Bought This Item Also Bought", "Recommendations for You, Alex", "Related to Items You Viewed", and "More Items to Consider"recommendations.

The value of machine learning technology has been recognized by companies across several industries that deal with huge volumes of data. By leveraging insights obtained from this data, companies are able work in an efficient manner to control costs as well as get an edge over their competitors. This is how some sectors / domains are implementing machine learning :-

- **Financial Services:** Companies in the financial sector are able to identify key insights in financial data as well as prevent any occurrences of financial fraud, with the help of machine learning technology. The technology is also used to identify opportunities for investments and trade. Usage of cyber surveillance helps in identifying those individuals or institutions which are prone to financial risk, and take necessary actions in time to prevent fraud.

- **Marketing and Sales:** Companies are using machine learning technology to analyze the purchase history of their customers and make personalized product recommendations for their next purchase. This ability to capture, analyze, and use customer data to provide a personalized shopping experience is the future of sales and marketing.

- **Government:** Government agencies like utilities and public safety have a specific need FOR Ml, as they have multiple data sources, which can be mined for identifying useful patterns and insights. For example sensor data can be analyzed to identify ways to minimize costs and increase efficiency. Furthermore, ML can also be used to minimize identity thefts and detect fraud.

- **Healthcare:** With the advent of wearable sensors and devices that use data to access health of a patient in real time, ML is becoming a fast-growing trend in healthcare. Sensors in wearable provide real-time patient information, such as overall health condition, heartbeat, blood pressure and other vital parameters. Doctors and medical experts can use this information to analyze the health condition of an individual, draw a pattern from the patient history, and predict the occurrence of any ailments in the future. The technology also empowers medical experts to analyze data to identify trends that facilitate better diagnoses and treatment.

- **Transportation:** Based on the travel history and pattern of traveling across various routes, machine learning can help transportation compa-

nies predict potential problems that could arise on certain routes, and accordingly advise their customers to opt for a different route. Transportation firms and delivery organizations are increasingly using machine learning technology to carry out data analysis and data modeling to make informed decisions and help their customers make smart decisions when they travel.

- **Oil and Gas:** This is perhaps the industry that needs the application of machine learning the most. Right from analyzing underground minerals and finding new energy sources to streaming oil distribution, ML applications for this industry are vast and are still expanding.

4 Aspects of Training Data

Data pre-processing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we preprocess our data before feeding it into our model. The process for getting data ready for a machine learning algorithm can be summarized in these steps:

1 Select Data

2 Pre-process Data

3 Transform Data

4.1 Select Data

Selecting the right dataset for Machine learning is very important to make the AI model functional with right approach. Though selecting the right quality and amount of data is challenging task but there are few rules needs to be followed for machine learning on big data. There is always a strong desire for including all data that is available, that the maxim "more is better" will hold. This may or may not be true.
We need to consider what data we actually need to address the question or problem we are working on. Make some assumptions about the data we require and be careful to record those assumptions so that we can test them later if needed.
Below are some questions to help we think through this process:

a What is the extent of the data we have available? For example, through time, database tables, connected systems. Ensure we have a clear picture of everything that we can use.

b What data is not available that we wish we had available? For example, data that is not recorded or cannot be recorded. We may be able to derive or simulate this data.

c What data don't we need to address the problem? Excluding data is almost always easier than including data. Note down which data we excluded and why.

It is only in small problems, like competition or toy datasets where the data has already been selected for us. Unstructured data such as images, text or video will eventually need to be converted into a data frame before applying predictive methods, so these metrics listed below apply to all types of data during the machine learning process.
n: Usually the first characteristic of interest in a dataset is its size, N measured as the number of rows or examples.
d: the next descriptor of the data is its dimension, d measured by the number of columns or attributes
k: this descriptor applies only to classification problems. k represents the number of classes. Today most classification problems are binary or k=2. But it is not difficult to envision situations where $k >> 2$.
m: is the ratio of number of samples of the minority class to number of samples of the majority class in a 2-class problem. In a multiclass problem ($k > 2$), this could be the ratio of the number of samples of a given class to the rest of the samples.

4.2 Pre-Process Data

The collected data cannot be used directly for performing analysis process. Whenever the data is gathered from different sources, it is collected in raw format which is not feasible for the analysis. Therefore, certain steps are executed to convert the data into a small clean data set. These are the common data pre-processing methods:

1 **Formatting:** The data you have selected may not be in a format that is suitable for you to work with. The data may be in a relational database and you would like it in a flat file, or the data may be in a proprietary file format and you would like it in a relational database or a text file.

2 **Cleaning:** Cleaning data is the removal or fixing of missing data. There may be data instances that are incomplete and do not carry the data you believe you need to address the problem. These instances may need to be removed. Additionally, there may be sensitive information in some of the attributes and these attributes may need to be anonymized or removed from the data entirely.

3 **Sampling:** There may be far more selected data available than you need to work with. More data can result in much longer running times for algorithms and larger computational and memory requirements. You can take a smaller representative sample of the selected data that may be much faster for exploring and prototyping solutions before considering the whole dataset.

4 **Binarize Data (Make Binary):** We can transform our data using a binary threshold. All values above the threshold are marked 1 and all equal to or below are marked as 0. This is called binarizing your data or threshold your data. It can be useful when you have probabilities that you want to make crisp values. It is also useful when feature engineering and you want to add new features that indicate something meaningful.

5 **Standardize Data:** Standardization is a useful technique to transform attributes with a Gaussian distribution and differing means and standard deviations to a standard Gaussian distribution with a mean of 0 and a standard deviation of 1.

6 **Dimensionality Reduction:** When data sets become large in the number of predictor variables or the number of instances, data mining algorithms face the curse of dimensionality problem. It is a serious problem as it will impede the operation of most data mining algorithms as the computational cost rise. This section will underline the most influential dimensionality reduction algorithms according to the division established into Feature Selection (FS) and space transformation-based methods.

4.3 Transform Data

The final step is to transform the process data. This step is used to convert the raw data into a specified format according to the need of the model. The methods used for transformation of data are given below :–

1 **Rescale Data:** When our data is comprised of attributes with varying scales, many machine learning algorithms can benefit from rescaling the attributes to all have the same scale. This is useful for optimization algorithms in used in the core of machine learning algorithms like gradient descent. It is also useful for algorithms that weight inputs like regression and neural networks and algorithms that use distance measures like K-Nearest Neighbours.

2 **Decomposition:** There may be features that represent a complex concept that may be more useful to a machine learning method when split into the constituent parts. An example is a date that may have day and time components that in turn could be split out further. Perhaps only the hour of day is relevant to the problem being solved. consider what feature decompositions you can perform.

3 **Aggregation:** There may be features that can be aggregated into a single feature that would be more meaningful to the problem you are trying to solve. For example, there may be a data instances for each time a customer logged into a system that could be aggregated into a count for the number of logins allowing the additional instances to be discarded. Consider what type of feature aggregations could perform.

Data preparation is a large subject that can involve a lot of iterations, exploration and analysis. Getting good at data preparation will make you a master at machine learning. For now, just consider the questions raised in this post when preparing data and always be looking for clearer ways of representing the problem you are trying to solve.

5 Concept Learning and Concept Representation

Conceptual learning is an educational method that centers on big-picture ideas and learning how to organize and categorize information. Unlike more traditional learning models which concentrate on the ability to recall specific facts (such as the dates of an event or the twenty possible causes of a particular illness), conceptual learning focuses on understanding broader principles or ideas (what we call "concepts") that can later be applied to a variety of specific examples.
To some, conceptual learning can be seen as more of a top-down approach versus the bottom- up model used in more traditional learning (Fig. **??**). To others who view traditional learning as rote memorization of facts and figures, conceptual learning is seen as a means for getting students to think more critically about the new subjects and situations they encounter. The main goal of representation learning or feature learning is to find an appropriate representation of data in order to perform a machine learning task.
In particular, deep learning exploits this concept by its very nature. In a neural network, each hidden layer maps its input data to an inner representation that tends to capture a higher level of abstraction. These learnt features are increasingly more informative through layers towards the machine learning task that we intend to perform (e.g. classification).
Much of human learning involves acquiring general concepts from past experiences. For example, humans identify different vehicles among all the vehicles based on specific sets of features defined over a large set of features. This special set of features differentiates the subset of cars in a set of vehicles. This set of features that differentiate cars can be called a concept.
Similarly, machines can learn from concepts to identify whether an object belongs to a specific category by processing past/training data to find a hypothesis that best fits the training examples.

5.1 Concepts and Exemplars

Concepts are mental categories for facts, objects, events, people, ideas — even skills and competencies — that have a common set of features across multiple situations and contexts. Concepts can range from simple to complex according to how easily they can be defined.
Examples of concepts :-
Concrete Concepts have aspects or dimensions that are easily seen, heard, or

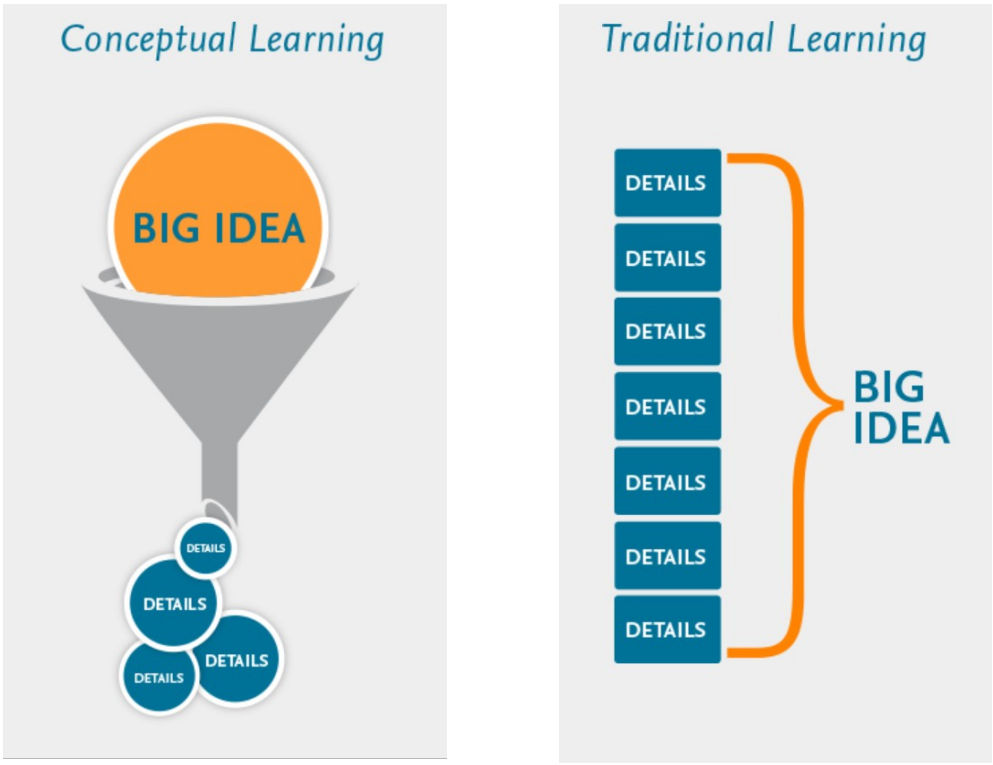

Fig. 1 Conceptual Learning and Traditional Learning

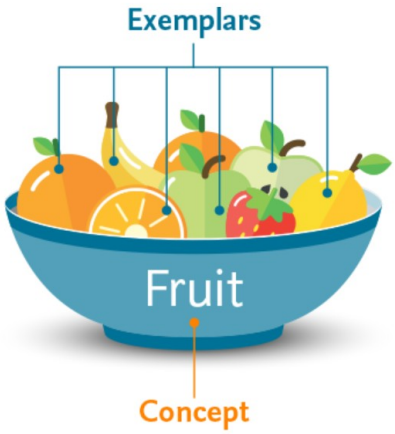

Fig. 2 Concept of Fruit

touched. Fruit would be an example of a concrete concept due to its tangible characteristics of being seed-associated, fleshy, and plant-derived.

Semi-concrete Concepts have some combination of concrete and non-concrete characteristics. Take the semi-concrete concept of a politician, for instance. Some characteristics of a politician could be concrete, such as a holder or candidate for an elected office. However, other characteristics may not be as concrete, such as one who serves the public.

Abstract Concepts do not have many (if any) absolute characteristics that are easy to comprehend with the senses. Unlike concrete and semi-concrete concepts, abstract concepts are not explained by a list of well-defined rules or characteristics. More often, they are understood by mental images or beliefs about its characteristics. Love would be a good example of an abstract concept, as the characteristics of love might differ from one person to the next.

So if concepts are the broad principles or classifications, **Exemplars** then, are the "typical examples" or "excellent models" of that principle. For example, if you are teaching about the concept of fruit, then some good exemplars would be apples, oranges, and bananas (Fig. **??**). If love is the concept at hand, depending on the type of course you are teaching, some exemplars to use could be the relationship of a mother and daughter, or a group of friends.

Target Concept

The set of items/objects over which the concept is defined is called the set of instances and denoted by X. The concept or function to be learned is called the target concept and denoted by c. It can be seen as a Boolean valued function defined over X and can be represented as $c : X -> \{0, 1\}$.

If we have a set of training examples with specific features of target concept C, the problem faced by the learner is to estimate C that can be defined on training data. H is used to denote the set of all possible hypotheses that the

learner may consider regarding the identity of the target concept. The goal of a learner is to find a hypothesis H that can identify all the objects in X so that $h(x) = c(x)$ for all x in X.
An algorithm that supports concept learning requires:
1. Training data (past experiences to train our models)
2. Target concept (hypothesis to identify data objects)
3. Actual data objects (for testing the models)

Inductive Learning Hypothesis

As we discussed earlier, the ultimate goal of concept learning is to identify a hypothesis H identical to target concept C over data set X with the only available information about C being its value over X. Our algorithm can guarantee that it best fits the training data. In other words:
"Any hypothesis found approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples."
For example, whether a person goes to a movie is based on four binary features with two values (true or false):
1. Has money
2. Has free time
3. It's a holiday
4. Has pending work

With the training data, we have with two data objects as positive samples and one as negative:
1. $x1 :< true, true, false, false >: +ve$
2. $x2 :< true, false, false, true >: +ve$
3. $x3 :< true, false, false, true >: -ve$

Hypothesis Notations

Each of the data objects represents a concept and hypotheses. Considering a hypothesis $< true, true, false, false >$ is more specific because it can cover only one sample. Generally, we can add some notations into this hypothesis. We have the following notations:
1. Ø(represents a hypothesis that rejects all)
2. $<?, ?, ?, ? >$ (accepts all) 3. $< true, false, ?, ? >$ (accepts some) The hypothesis Ø will reject all the data samples. The hypothesis $<?, ?, ?, ? >$ will accept all the data samples. The ? notation indicates that the values of this specific feature do not affect the result.
The total number of the possible hypothesis is $(3 * 3 * 3 * 3) + 1|3$ because one feature can have either true, false, or ? and one hypothesis for rejects all (Ø).

General to Specific

Many machine learning algorithms rely on the concept of general-to-specific ordering of hypothesis.
1. $h1 =< true, true, ?, ? >$
2. $h2 =< true, ?, ?, ? >$

Any instance classified by h1 will also be classified by h2. We can say that h2 is more general than h1. Using this concept, we can find a general hypothesis that can be defined over the entire dataset X. To find a single hypothesis defined on

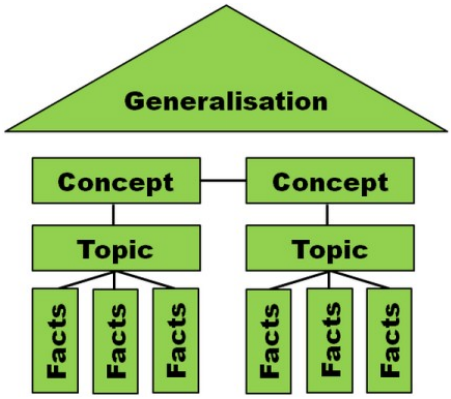

Fig. 3 Generalisation and Concept

X, we can use the concept of being more general than partial ordering. One way to do this is start with the most specific hypothesis from H and generalize this hypothesis each time it fails to classify and observe positive training data object as positive.

1. The first step in the Find-S algorithm is to start with the most specific hypothesis, which can be denoted by $h < - < \emptyset, \emptyset, \emptyset, \emptyset >$.

2. This step involves picking up next training sample and applying Step 3 on the sample.

3. The next step involves observing the data sample. If the sample is negative, the hypothesis remains unchanged and we pick the next training sample by processing Step 2 again. Otherwise, we process Step 4.

4. If the sample is positive and we find that our initial hypothesis is too specific because it does not cover the current training sample, then we need to update our current hypothesis. This can be done by the pairwise conjunction (logical and operation) of the current hypothesis and training sample.

If the next training sample is ¡true, true, false, false¿ and the current hypothesis is $< \emptyset, \emptyset, \emptyset, \emptyset >$, then we can directly replace our existing hypothesis with the new one.

If the next positive training sample is $< true, true, false, true >$ and current hypothesis is $< true, true, false, false >$, then we can perform a pairwise conjunctive. With the current hypothesis and next training sample, we can find a new hypothesis by putting ? in the place where the result of conjunction is false: $< true, true, false, true > \emptyset < true, true, false, false >=< true, true, false, ? >$ Now, we can replace our existing hypothesis with the new one: $h < - < true, true, false, ? >$

5. This step involves repetition of Step 2 until we have more training samples.

6. Once there are no training samples, the current hypothesis is the one we wanted to find. We can use the final hypothesis to classify the real objects.

Example of working with conceptual learning:-

Fig. 4 Example of conceptual learning

Consider you are tasked to classify a set of shapes. The way you do that is to find unique characteristics of each of your input shape. An example is number of corners (or vertices). Circle has 0, Triangle has 3 and a square has 4 Fig. **??**. Your system may work something like this:
Input - An image
Representation - No of corners in the image (you might use tools like openCV)
Model - Gets an input representation or feature (e.g. no of corners) and applies rules to detect the shape. (Like if feature input is 0 then circle).
Output - We have a working system.
But what happens when you start getting inputs as cuboid, trapezium or all sort of shapes. You would realize that designing features gets not just difficult, time consuming and requires a deep domain expertise as you start working with real world use-cases. E.g. consider you need to recognize equilateral versus obtuse triangle (both have same no of corners). It is observed that designing features is a complex process and the way to solve that is how our brain is able to design these features. In this example, you looked at shapes and decided that no of corners seems like a good way to uniquely classify images. It is this task of brain that is performed by feature or representation learning algorithms. Deep learning is just one of such methods. Deep Learning learns tries to learn features on its own. All you would like to do is pass an image and let the system learn features like we do.

6 Function Approximation

Statistical and connectionist approaches to machine learning are related to function approximation methods in mathematics. For the purposes of illustration let us assume that the learning task is one of classification. That is, we wish to find ways of grouping objects in a universe. In Fig. **??** we have a universe of objects that belong to either of two classes '+' or '-'. By function approximation, we describe a surface that separates the objects into different regions. The simplest function is that of a line and linear regression methods and perceptrons are used to find linear discriminant functions. A perceptron is a simple pattern classifier. Given a binary input vector, x, a weight vector, w, and a threshold

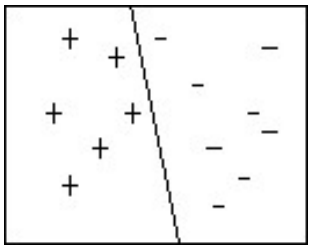

Fig. 5 A linear discrimination between two classes

value, T, if,

$$\sum_i w_i \times x_i > T$$

Then, the output is 1, indicating membership of a class, otherwise it is 0, indicating exclusion from the class. Clearly, $w.x - T$ describes a hyper plane and the goal of perceptron learning is to find a weight vector, w, that results in correct classification for all training examples. The perceptron learning algorithm is quite straight forward. All the elements of the weight vector are initially set to 0. For each training example, if the perceptron outputs 0 when it should output 1 then add the input vector to the weight vector; if the perceptron outputs 1 when it should output 0 then subtract the input vector to the weight vector; otherwise, do nothing. This is repeated until the perceptron yields the correct result for each training example. The algorithm has the effect of reducing the error between the actual and desired output.
The perceptron is an example of a linear threshold unit (LTU). A single LTU can only recognize one kind of pattern, provided that the input space is linearly separable. If we wish to recognize more than one pattern, several LTU's can be combined. In this case, instead of having a vector of weights, we have an array. The output will now be a vector:

$$u = Wx$$

Where, each element of u indicates membership of a class and each row in W is the set of weights for one LTU. This architecture is called a pattern associators. LTU's can only produce linear discriminant functions and consequently, they are limited in the kinds of classes that can be learned. However, it was found that by cascading pattern associators, it is possible to approximate decision surfaces that are of a higher order than simple hyper planes. In cascaded system, the outputs of one pattern associators are fed into the inputs of another, thus:

$$u = W(Vx)$$

To facilitate learning, a further modification must be made. Rather than using a simple threshold, as in the perceptron, multi-layer networks Fig. **??**

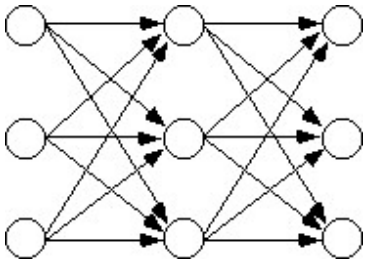

Fig. 6 A multi-layer network

usually use a non-linear threshold such a sigmoid function, such as

$$\frac{1}{1+e^{-x}}$$

Like perceptron learning, back-propagation attempts to reduce the errors between the output of the network and the desired result. However, assigning blame for errors to hidden units (ie. nodes in the intermediate layers), is not so straightforward. The error of the output units must be propagated back through the hidden units. The contribution that a hidden unit makes to an output unit is related strength of the weight on the link between the two units and the level of activation of the hidden unit when the output unit was given the wrong level of activation. This can be used to estimate the error value for a hidden unit in the penultimate layer, and that can, in turn, be used in make error estimates for earlier layers. Despite the non-linear threshold, multi-layer networks can still be thought of as describing a complex collection of hyper planes that approximate the required decision surface.

Version Space

A version space is a hierarchical representation of knowledge that enables you to keep track of all the useful information supplied by a sequence of learning examples without remembering any of the examples. The version space method is a concept learning process accomplished by managing multiple models within a version space.

Version Space Characteristics

Tentative heuristics are represented using version spaces. A version space represents all the alternative plausible descriptions of a heuristic. A plausible description is one that is applicable to all known positive examples and no known negative example. A version space description consists of two complementary trees:

1. One that contains nodes connected to overly general models, and
2. One that contains nodes connected to overly specific models.

Node values/attributes are discrete.

Fundamental Assumptions

1. The data is correct; there are no erroneous instances.
2. A correct description is a conjunction of some of the attributes with values.

Diagrammatical Guidelines

There is a generalization tree and a specialization tree. Each node is connected to a model. Nodes in the generalization tree are connected to a model that matches everything in its subtree. Nodes in the specialization tree are connected to a model that matches only one thing in its subtree. Links between nodes and their models denote:
- generalization relations in a generalization tree, and
- specialization relations in a specialization tree.

7 Types of Learning

7.1 Supervised Learning

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). Supervised learning, in the context of artificial intelligence (AI) and machine learning, is a type of system in which both input and desired output data are provided. Input and output data are labelled for classification to provide a learning basis for future data processing. Supervised machine learning systems provide the learning algorithms with known quantities to support future judgements, Chatbots, self-driving cars, facial recognition programs, systems and robots are among the systems that may use either supervised or unsupervised learning. Supervised learning systems are mostly associated with retrieval-based AI, but they may also be capable of using a generative learning model. In supervised learning for image processing, for example, an AI system might be provided with labelled pictures of vehicles in categories such as cars and trucks. After an enough observation, the system should be able to distinguish between and categorize unlabelled images, at which time training can be said to be complete. Supervised learning models have some advantages over the unsupervised approach, but they also have limitations. The systems are more likely to make judgements that humans can relate to, for example, because humans have provided the basis for decisions. However, in the case of a retrieval-based method, supervised learning systems have trouble dealing with new information. If a system with categories for cars and trucks is presented with a bicycle, for example, it would have to be incorrectly lumped in one category or the other. If the AI system was generative, however, it may not know what the bicycle is but would be able to recognize it as belonging to a separate category. Supervised learning is the Data mining task of inferring a function from labelled training data. The training data consist of a set of training example. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the

algorithm to correctly determine the class labels for unseen instance . This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way. There are some of the points:-

- You already learn from your previous work about the physical characters of fruits.
- So, arranging the same type of fruits at one place is easy now.
- Your previous work is called as training data in data mining.
- so, you already learn the things from your train data, this is because of response variable.
- Response variable mean just a decision variable.
- You can observe response variable below (FRUIT NAME) .
- Suppose you have taken a new fruit from the basket then you will see the size , colour and shape of that fruit.
- If size is Big , colour is Red , shape is rounded shape with a depression at the top, you will conform the fruit name as apple, and you will put in apple group.
- Likewise, for other fruits also.
- Job of groping fruits was done and happy ending.
- You can observe in the table that a column was labelled as "FRUIT NAME" this is called as response variable.
- If you learn the thing before from training data and then applying that knowledge to the test data(for new fruit), This type of learning is called as Supervised Learning.
- Classification comes under Supervised learning.

7.1.1 Supervised Learning Algorithms

- Support Vector Machines
- Linear regression
- Logistic regression
- Naive Bayes
- Linear discriminant analysis
- Decision trees
- K-Nearest neighbor algorithm
- Neural Networks (Multilayer perceptron)
- Similarity learning

7.1.2 Steps taken to implement supervised algorithm

In order to solve a given problem of supervised learning, one has to perform the following steps:

- Determine the type of training examples. Before doing anything else, the user should decide what kind of data is to be used as a training set. In case of Handwriting Analysis, for example, this might be a single handwritten character, an entire handwritten word, or an entire line of handwriting.
- Gather a training set. The training set needs to be representative of the real-world use of the function. Thus, a set of input objects is gathered and corresponding outputs are also gathered, either from human experts or from measurements.
- Determine the input feature representation of the learned function. The accuracy of the learned function depends strongly on how the input object is represented. Typically, the input object is transformed into a feature vector, which contains a number of features that are descriptive of the object. The number of features should not be too large, because of the curse of dimensionality; but should contain enough information to accurately predict the output.
- Determine the structure of the learned function and corresponding learning algorithm. For example, the engineer may choose to use support vector machines or decision trees.
- Complete the design. Run the learning algorithm on the gathered training set. Some supervised learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing performance on a subset (called a validation set) of the training set, or via cross-validation.
- Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set

7.1.3 Major issues in supervised learning

Bias-variance tradeoff
A first issue is the tradeoff between bias and variance. Imagine that we have available several different, but equally good, training data sets. A learning algorithm is biased for a particular input . The prediction error of a learned classifier is related to the sum of the bias and the variance of the learning algorithm. But if the learning algorithm is too flexible, it will fit each training data set differently, and hence have high variance.
Function complexity and amount of training data
The second issue is the amount of training data available relative to the complexity of the "true" function (classifier or regression function). If the true

function is simple, then an "inflexible" learning algorithm with high bias and low variance will be able to learn it from a small amount of data. But if the true function is highly complex (e.g., because it involves complex interactions among many different input features and behaves differently in different parts of the input space), then the function will only be learn able from a very large amount of training data and using a "flexible" learning algorithm with low bias and high variance.

Dimensionality of the input space

A third issue is the dimensionality of the input space. If the input feature vectors have very high dimension, the learning problem can be difficult even if the true function only depends on a small number of those features. This is because the many "extra" dimensions can confuse the learning algorithm and cause it to have high variance. Hence, high input dimensional typically requires tuning the classifier to have low variance and high bias This is an instance of the more general strategy of dimensionality reduction, which seeks to map the input data into a lower-dimensional space prior to running the supervised learning algorithm.

Noise in the output values

A fourth issue is the degree of noise in the desired output values (the supervisory target variables). If the desired output values are often incorrect (because of human error or sensor errors), then the learning algorithm should not attempt to find a function that exactly matches the training examples.In such a situation, the part of the target function that cannot be modeled "corrupts" your training data - this phenomenon has been called deterministic noise. When either type of noise is present, it is better to go with a higher bias, lower variance estimator.

7.2 Unsupervised Learning

Unsupervised Learning is a class of Machine Learning techniques to find the patterns in data. The data given to unsupervised algorithm are not labeled, which means only the input variables(X) are given with no corresponding output variables. In unsupervised learning, the algorithms are left to themselves to discover interesting structures in the data. Yan Lecun, director of AI research, explains that unsupervised learning-teaching machines to learn for themselves without having to be explicitly told if everything they do is right or wrong-is the key to "true" AI. The image to the left in Fig. **??** is an example of supervised learning; we use regression techniques to find the best fit line between the features. While in unsupervised learning the inputs are segregated based on features and the prediction is based on which cluster it belonged. Unsupervised Learning mainly divided into two categories:- Clustering and Classification.

7.2.1 Clustering

The cluster analysis as a branch of machine learning that groups the data that has not been labelled, classified or categorized. Instead of responding to feedback, cluster analysis identifies commonalities in the data and reacts based on

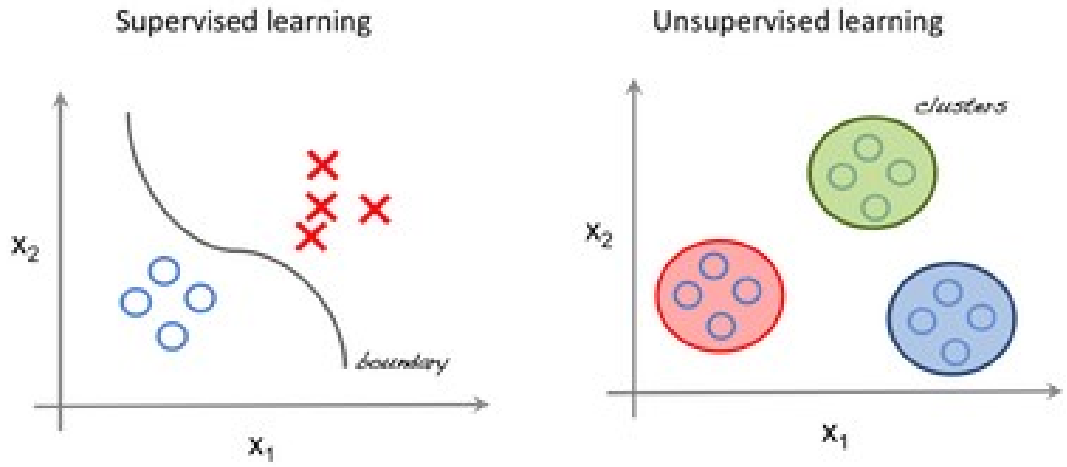

Fig. 7 Supervised and Unsupervised Learning

the presence or absence of such commonalities in each new piece of data.
Common clustering algorithms:
k-Means clustering: partitions data into k distinct clusters based on distance to the centroid of a cluster
Gaussian mixture models: models clusters as a mixture of multivariate normal density components
Self-organizing maps: uses neural network that learn the topology and distribution of the data
Hidden Markov models: uses observed data to recover the sequence of states.

7.2.2 Classification

Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y). For example, spam detection in email service providers can be identified as a classification problem. This is s binary classification since there are only 2 classes as spam and not spam. A classifier utilizes some training data to understand how given input variables relate to the class. In this case, known spam and non-spam emails have to be used as the training data. When the classifier is trained accurately, it can be used to detect an unknown email. Classification belongs to the category of supervised learning where the targets also provided with the input data. There are many applications in classification in many domains such as in credit approval, medical diagnosis, target marketing etc.

Classification algorithms:

Decision Tree

Decision tree builds classification or regression models in the form of a tree structure. It utilizes an if-then rule set which is mutually exclusive and exhaustive for classification. The rules are learned sequentially using the training data one at a time. Each time a rule is learned, the tuples covered by the rules are removed. This process is continued on the training set until meeting a termination condition. The tree is constructed in a top-down recursive divide-and-conquer manner. All the attributes should be categorical. Otherwise, they should be discretized in advance. Attributes in the top of the tree have more impact towards in the classification and they are identified using the information gain concept. A decision tree can be easily over-fitted generating too many branches and may reflect anomalies due to noise or outliers. An over-fitted model has a very poor performance on the unseen data even though it gives an impressive performance on training data. This can be avoided by pre-pruning which halts tree construction early or post-pruning which removes branches from the fully grown tree.

Naive Bayes

Naive Bayes is a probabilistic classifier inspired by the Bayes theorem under a simple assumption which is the attributes are conditionally independent. The classification is conducted by deriving the maximum posterior which is the maximal with the above assumption applying to Bayes theorem. This assumption greatly reduces the computational cost by only counting the class distribution. Even though the assumption is not valid in most cases since the attributes are dependent, surprisingly Naive Bayes has able to perform impressively. Naive Bayes is a very simple algorithm to implement and good results have obtained in most cases. It can be easily scalable to larger datasets since it takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

Artificial Neural Networks

Artificial Neural Network is a set of connected input/output units where each connection has a weight associated with it started by psychologists and neurobiologists to develop and test computational analogs of neurons. During the learning phase, the network learns by adjusting the weights so as to be able to predict the correct class label of the input tuples. There are many network architectures available now like Feed-forward, Convolutional, Recurrent etc. The appropriate architecture depends on the application of the model. For most cases feed-forward models give reasonably accurate results and especially for image processing applications, convolutional networks perform better.

k-Nearest Neighbor (KNN)

k-Nearest Neighbor is a lazy learning algorithm which stores all instances correspond to training data points in n-dimensional space. When an unknown discrete data is received, it analyzes the closest k number of instances saved (nearest neighbors)and returns the most common class as the prediction and for real-valued data it returns the mean of k nearest neighbors. In the distance-weighted nearest neighbor algorithm, it weights the contribution of each of the

k neighbors according to their distance using the following query giving greater weight to the closest neighbors.

7.2.3 Challenges in Implementing Unsupervised Learning

a) In addition to the regular issues of finding the right algorithms and hardware, unsupervised learning presents a unique challenge: it's difficult to figure out if you're getting the job done or not.
b) In supervised learning, we define metrics that drive decision making around model tuning. Measures like precision and recall give a sense of how accurate your model is, and parameters of that model are tweaked to increase those accuracy scores. Low accuracy scores mean you need to improve, and so on.
c) Since there are no labels in unsupervised learning, it's near impossible to get a reasonably objective measure of how accurate your algorithm is. In clustering for example, how can you know if K-Means found the right clusters? Are you using the right number of clusters in the first place? In supervised learning we can look to an accuracy score; here you need to get a bit more creative.
d) A big part of the "will unsupervised learning work for me?" question is totally dependent on your business context. In our example of customer segmentation, clustering will only work well if your customers actually do fit into natural groups. One of the best (but most risky) ways to test your unsupervised learning model is by implementing it in the real world and seeing what happens! Designing an A/B test–with and without the clusters your algorithm outputted–can be an effective way to see if it's useful information or totally incorrect.
e) Researchers have also been working on algorithms that might give a more objective measure of performance in unsupervised learning. Check out the below section for some examples.

8 Training Dataset

The actual dataset that we use to train the model (weights and biases in the case of Neural Network). The model sees and learns from this data. Machine learning is a subfield of computer science that gives the computer the ability to learn without being explicitly programmed. For this to happen, a machine needs to be "trained" by explicitly feeding it data that has the correct answers attached. This training data will help the machine to connect the patterns in the data to the right answer. Once trained in this way, a machine can now be given test data that has no answers. The machine will then predict the answers based on the training it received. Most data scientists divide their data (with answers, that is historical data) into three portions: training data, cross-validation data and testing data. The training data is used to make sure the machine recognizes patterns in the data, the cross-validation data is used to ensure better accuracy and efficiency of the algorithm used to train the machine, and the test data is used to see how well the machine can predict new answers based on its training.

8.1 How to create training data?

Machine Learning models are trained using data with specific features. The way in which the data is structured helps the models to learn and develop relationship between these features. A well-processed training set is required to build a robust model which in turn generates accurate results. In this article we shall look at some of the ways in which one can build a structured dataset for training. To build a robust model, one has to keep in mind the flow of operations involved in building a quality dataset. The data should be accurate with respect to the problem statement. For example, while trying to determine the height of a person, feature such as age, sex, weight, or the size of the clothes, among others, are to be considered. Here, the person's clothes will account for his/her height, whereas the colour of the clothes and the material will not add any value in this case. Hence these features have very low weightage for predicting the height of a person. A golden rule of machine learning is: Larger the data better the results. There are several steps included in this process:

1 **Data Selection** In this step, one should be concerned about opting the right number of features for the particular dataset. The data should be consistent and should have least number of missing values. If a feature has more than 25 to 30 percent missing values then it is usually considered not fit to be a part of the training set. But there are instances where the relationship between this feature and the Y feature is high. In that case, one has to impute and handle the missing values for better results. For example, let us say an institution has borrowed a loan from a bank. A feature containing the GDP value of the particular country is available with 30 percent missing values. If one infers that the particular feature has a very high importance to predict whether the institution is able to repay the loan or not, then this feature has to be considered. If the feature does not hold high importance for developing the AI model, one should exclude the data. At the end of this particular step, one should have an idea about how to deal with the preprocessing data.

2 **Data Preprocessing** Once the right data is selected, preprocessing includes selection of the right data from the complete dataset and building a training set. Here, some of the common steps are:
Organise and Format: The data might be scattered in different files, for example, classroom datasets of various grades in a school which needs to be clubbed together to form a dataset. One has to find the relation between these datasets and preprocess to form a dataset of required dimensions. Also if the datasets are in different language they have to be transformed into a universal language before proceeding.
Data Cleaning: This is one of the major steps in data preprocessing. Cleaning refers to mainly dealing with the missing values and removal of unwanted characters from the data. For example, if a feature consists of age of a person, with 4 percent missing values, it can either deleted or

replaced. Here, is an in-depth article of how to handle missing in machine learning.
Feature Extraction: This step involves analysis and optimisation of the number of features. One has to find out which features are important for prediction and select them for faster computations and low memory consumption. For example, while dealing with an image classification problem, images with noise (irrelevant images with respect to the dataset) should be removed.

3 **Data Conversion**
Scaling: This is necessary when the dataset is placed. Considering a linear dataset - bank data. If the feature containing the transaction amount is important, then the data has to be scaled in order to build a robust model. By default in correlation matrix, the Pearson method is used to find the relationship. This might lead to a misunderstanding of the data if it is not scaled by a definite value.
Disintegration and Composition: This step is considered when one needs to split a particular feature to build a better training data for the model to understand. One of the best examples of the data disintegration is splitting up the time-series feature. Where one can extract the days, months, year, hour, minutes, seconds, etc. from a particular sample. And also let us say, the Project ID is IND0002. Here the first three characters refer to the country code and 0002 refer to a categorical value. Separating and processing may result in better accuracy.
Composition: This process involves combining different features to a single feature for more meaningful data. For example, in the Titanic dataset, the prefix of the passengers with Dr, Mr, Miss. etc can be clubbed into a particular age groups of categorical data which adds more weight in predicting the passengers' survival.

one can understand how processed training set helps a machine learning to develop the relationship between the features. This process involves a lot of time, analysis and examination of the data. With a well - structured data, machine learning model can train faster and give robust results.

9 Test Dataset

The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset. The Test dataset provides the gold standard used to evaluate the model. It is only used once a model is completely trained (using the train and validation sets). The test set is generally what is used to evaluate competing models (For example on many Kaggle competitions, the validation set is released initially along with the training set and the actual test set is only released when the competition is about to close, and it is the result of the the model on the Test set that decides the winner). Many a times the validation set

is used as the test set, but it is not good practice. The test set is generally well curated. It contains carefully sampled data that spans the various classes that the model would face, when used in the real world. This corresponds to the final evaluation that the model goes through after the training phase (utilizing training and validation sets) has been completed. This step is critical to test the generalizability of the model (Step 3). By using this set, we can get the working accuracy of our model. It is worth mentioning that we need to be subjective and honest by not exposing the model to the test set until the training phase is over. This way, we can consider the final accuracy measure to be reliable. Training a model involves looking at training examples and learning from how off the model is by frequently evaluating it on the validation set. However, the last and most valuable pointer on the accuracy of a model is a result of running the model on the testing set when the training is complete.

10 Validation Dataset

Validation Dataset: The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. We as machine learning engineers use this data to fine-tune the model hyperparameters. Hence the model occasionally sees this data, but never does it "Learn" from this. We use the validation set results and update higher level hyperparameters. So the validation set in a way affects a model, but indirectly.

11 Dataset split ratio

Now that you know what these datasets do, you might be looking for recommendations on how to split your dataset into Train, Validation and Test sets. This mainly depends on 2 things. First, the total number of samples in your data and second, on the actual model you are training. Some models need substantial data to train upon, so in this case you would optimize for the larger training sets. Models with very few hyperparameters will be easy to validate and tune, so you can probably reduce the size of your validation set, but if your model has many hyperparameters, you would want to have a large validation set as well(although you should also consider cross validation). Also, if you happen to have a model with no hyperparameters or ones that cannot be easily tuned, you probably don't need a validation set too! All in all, like many other things in machine learning, the train-test-validation split ratio is also quite specific to your use case and it gets easier to make judgement as you train and build more and more models. Many a times, people first split their dataset into two - Train and Test. After this, they keep aside the Test set, and randomly choose $X\%$ of their Train dataset to be the actual Train set and the remaining $(100 - X)\%$ to

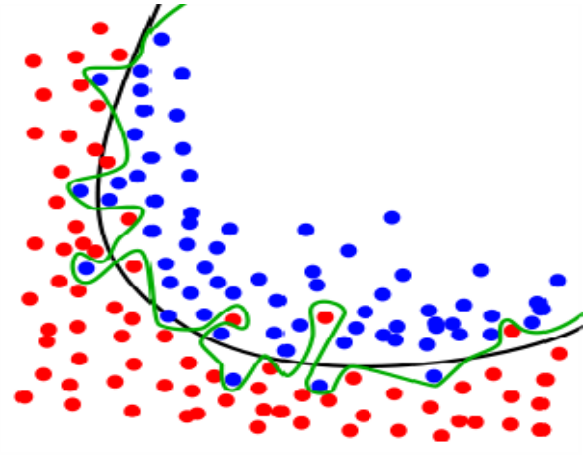

Fig. 8 Overfitting

be the Validation set, where X is a fixed number(say 80%), the model is then iteratively trained and validated on these different sets. There are multiple ways to do this, and is commonly known as Cross Validation. Basically you use your training set to generate multiple splits of the Train and Validation sets. Cross validation avoids over fitting and is getting more and more popular, with K-fold Cross Validation being the most popular method of cross validation.

12 Over fitting

Overfitting refers to a model that models the training data too well. Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data Fig. **??**. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the models ability to generalize. Overfitting is more likely with nonparametric and nonlinear models that have more flexibility when learning a target function. As such, many nonparametric machine learning algorithms also include parameters or techniques to limit and constrain how much detail the model learns. For example, decision trees are a nonparametric machine learning algorithm that is very flexible and is subject to overfitting training data. This problem can be addressed by pruning a tree after it has learned in order to remove some of the detail it has picked up. The cause of poor performance in machine learning is either overfitting or underfitting the data. Supervised machine learning is best understood as approximating a target function (f) that maps input variables (X) to an output variable (Y). Y = f(X) This characterization describes the range of classifica-

tion and prediction problems and the machine algorithms that can be used to address them. An important consideration in learning the target function from the training data is how well the model generalizes to new data. Generalization is important because the data we collect is only a sample, it is incomplete and noisy.

12.1 Generalization

In machine learning we describe the learning of the target function from training data as inductive learning. Induction refers to learning general concepts from specific examples which is exactly the problem that supervised machine learning problems aim to solve. This is different from deduction that is the other way around and seeks to learn specific concepts from general rules. Generalization refers to how well the concepts learned by a machine learning model apply to specific examples not seen by the model when it was learning. The goal of a good machine learning model is to generalize well from the training data to any data from the problem domain. This allows us to make predictions in the future on data the model has never seen. There is a terminology used in machine learning when we talk about how well a machine learning model learns and generalizes to new data, namely overfitting and underfitting. Overfitting and underfitting are the two biggest causes for poor performance of machine learning algorithms.

12.2 Statistical Fit

In statistics, a fit refers to how well you approximate a target function. This is good terminology to use in machine learning, because supervised machine learning algorithms seek to approximate the unknown underlying mapping function for the output variables given the input variables. Statistics often describe the goodness of fit which refers to measures used to estimate how well the approximation of the function matches the target function. Some of these methods are useful in machine learning (e.g. calculating the residual errors), but some of these techniques assume we know the form of the target function we are approximating, which is not the case in machine learning. If we knew the form of the target function, we would use it directly to make predictions, rather than trying to learn an approximation from samples of noisy training data. Overfitting refers to a model that models the training data too well. Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the models ability to generalize. Overfitting is more likely with nonparametric and nonlinear models that have more flexibility when learning a target function. As such, many nonparametric machine learning algorithms also include parameters or techniques to limit and constrain how much detail the model learns. For example, decision trees are a nonparametric machine learning algorithm that is very flexible and is subject

to overfitting training data. This problem can be addressed by pruning a tree after it has learned in order to remove some of the detail it has picked up.

12.3 A Good Fit in Machine Learning

Ideally, we want to select a model at the sweet spot between underfitting and overfitting. This is the goal, but is very difficult to do in practice. To understand this goal, we can look at the performance of a machine learning algorithm over time as it is learning a training data. We can plot both the skill on the training data and the skill on a test dataset we have held back from the training process. Over time, as the algorithm learns, the error for the model on the training data goes down and so does the error on the test dataset. If we train for too long, the performance on the training dataset may continue to decrease because the model is overfitting and learning the irrelevant detail and noise in the training dataset. At the same time the error for the test set starts to rise again as the model's ability to generalize decreases. The sweet spot is the point just before the error on the test dataset starts to increase where the model has good skill on both the training dataset and the unseen test dataset. We can perform this experiment with your favorite machine learning algorithms. This is often not useful technique in practice, because by choosing the stopping point for training using the skill on the test dataset it means that the testset is no longer "unseen" or a standalone objective measure. Some knowledge (a lot of useful knowledge) about that data has leaked into the training procedure. There are two additional techniques you can use to help find the sweet spot in practice: resampling methods and a validation dataset.

12.4 Detection of Overfitting

A key challenge with overfitting, and with machine learning in general, is that we can't know how well our model will perform on new data until we actually test it. To address this, we can split our initial dataset into separate training and test subsets. This method can approximate of how well our model will perform on new data. If our model does much better on the training set than on the test set, then we're likely overfitting.

12.5 Prevention of Overfitting

Detecting overfitting is useful, but it doesn't solve the problem. Fortunately, you have several options to try. Popular solutions for overfitting:

Cross-validation

Cross-validation is a powerful preventative measure against overfitting. The idea is clever: Use your initial training data to generate multiple mini train-test splits. Use these splits to tune your model. Cross-validation allows you to tune hyperparameters with only your original training set. This allows you to keep your test set as a truly unseen dataset for selecting your final model.

Train with more data

It won't work every time, but training with more data can help algorithms detect the signal better. In the earlier example of modeling height vs. age in children, it's clear how sampling more schools will help your model. Of course, that's not always the case. If we just add more noisy data, this technique won't help. That's why you should always ensure your data is clean and relevant.

Remove features

Some algorithms have built-in feature selection. For those that don't, you can manually improve their generalizability by removing irrelevant input features. An interesting way to do so is to tell a story about how each feature fits into the model. This is like the data scientist's spin on software engineer's rubber duck debugging technique, where they debug their code by explaining it, line-by-line, to a rubber duck. If anything doesn't make sense, or if it's hard to justify certain features, this is a good way to identify them. In addition, there are several feature selection heuristics you can use for a good starting point.

Early stopping

When you're training a learning algorithm iteratively, you can measure how well each iteration of the model performs. Up until a certain number of iterations, new iterations improve the model. After that point, however, the model's ability to generalize can weaken as it begins to overfit the training data. Early stopping refers stopping the training process before the learner passes that point. Today, this technique is mostly used in deep learning while other techniques (e.g. regularization) are preferred for classical machine learning.

13 Classification families

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, biometric identification, document classification etc. Here we have the types of classification algorithms in Machine Learning:

1 Linear Classifiers: Logistic Regression, Naive Bayes Classifier

2 Support Vector Machines

3 Decision Trees

4 Boosted Trees

5 Random Forest

6 Neural Networks

7 Nearest Neighbor

13.1 Linear discriminative

Linear Discriminant Analysis (LDA) is a classification method originally developed in 1936 by R. A. Fisher. Linear Discriminant Analysis is a dimensionality reduction technique used as a preprocessing step in Machine Learning and pattern classification applications.

13.2 Non-linear discriminative

13.3 Decision trees

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches and a leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data. Decision Tree Analysis is a general, predictive modelling tool that has applications spanning a number of different areas. In general, decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. The decision rules are generally in form of if-then-else statements. The deeper the tree, the more complex the rules and fitter the model.

- Instances: Refer to the vector of features or attributes that define the input space
- Attribute: A quantity describing an instance
- Concept: The function that maps input to output
- Target Concept: The function that we are trying to find, i.e., the actual answer
- Hypothesis Class: Set of all the possible functions
- Sample: A set of inputs paired with a label, which is the correct output (also known as the Training Set)
- Candidate Concept: A concept which we think is the target concept
- Testing Set: Similar to the training set and is used to test the candidate concept and determine its performance

A decision tree is a tree-like graph with nodes representing the place where we pick an attribute and ask a question; edges represent the answers to the question, and the leaves represent the actual output or class label. They are used in non-linear decision making with simple linear decision surface. Decision trees classify the examples by sorting them down the tree from the root to some leaf node, with the leaf node providing the classification to the example. Each node in the tree acts as a test case for some attribute, and each edge descending from that node corresponds to one of the possible answers to the test case. This process is recursive in nature and is repeated for every sub tree rooted at the new nodes. A general algorithm for a decision tree can be described as follows:

1 Pick the best attribute/feature. The best attribute is one which best splits or separates the data.

2 Ask the relevant question.

3 Follow the answer path.

4 Go to step 1 until you arrive to the answer.

The best split is one which separates two different labels into two sets.
Calculating information gain
As stated earlier, information gain is a statistical property that measures how well a given attribute separates the training examples according to their target classification. In the figure below, we can see that an attribute with low information gain (right) splits the data relatively evenly and as a result doesn't bring us any closer to a decision. Whereas, an attribute with high information gain (left) splits the data into groups with an uneven number of positives and negatives and as a result helps in separating the two from each other. To define information gain precisely, we need to define a measure commonly used in information theory called entropy that measures the level of impurity in a group of examples.

13.3.1 Advantages and Disadvantages

Following are the advantages of decision trees: -

1 Easy to use and understand.

2 Can handle both categorical and numerical data.

3 Resistant to outliers, hence require little data preprocessing.

4 New features can be easily added.

5 Can be used to build larger classifiers by using ensemble methods.

Following are the disadvantages of decision trees: -

1 Prone to over fitting.

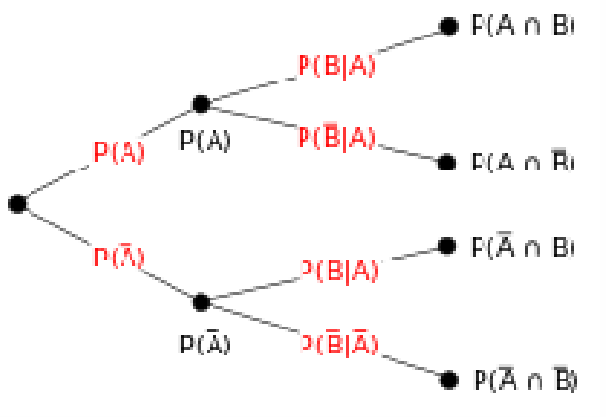

Fig. 9 Conditional Model

2 Require some kind of measurement as to how well they are doing.

3 Need to be careful with parameter tuning.

4 Can create biased learned trees if some classes dominate.

13.4 Conditional Model

Conditional Probabilistic models is a class of statistical models in which sample data are divided into input and output data and the relation between the two kind of data is studied by modelling the conditional probability distribution of the outputs given the inputs Fig. **??**. This is in contrast to unconditional models (sometimes also called generative models) where the data is studied by modelling the joint distribution of inputs and outputs. Before introducing conditional models, let us review the main elements of a statistical model:

1 There is a sample ξ, which can be regarded as a realization of a random vector Ξ. (for example, could be a vector collecting the realizations of some independent random variables);

2 The joint distribution function of the sample, denoted by $F_{\Xi}(\xi)$, is not known exactly;

3 The sample ξ is used to infer some characteristics of $F_{\Xi}(\xi)$;

4 A model for Ξ is used to make inferences, where a model is simply a set of joint distribution functions to which $F_{\Xi}(\xi)$ is assumed to belong.

In a conditional model, the sample ξ is partitioned into inputs and outputs: Where, y denotes the vector of outputs and x the vector of inputs.

$$\xi = [y\ x]$$

$$F_{Y|X=x}(y)$$

The object of interest is the conditional distribution function of the outputs given the inputs and specifying a conditional model means specifying a set of conditional distribution functions to which $F_{Y|X=x}(y)$ is assumed to belong. In other words, in a conditional model, the problem of model specification is simplified by narrowing the focus of the statistician's attention on the conditional distribution of the outputs and by ignoring the distribution of the inputs. This can be seen, for example, in the case in which both inputs and outputs are continuous random variables. In such a case, specifying an unconditional model is equivalent to specifying a joint probability density function $f_{X,Y}(x,y)$ for the inputs and the outputs. But a joint density can be seen as the product of a marginal and a conditional density: $f_{X,Y}(x,y) = F_{Y|X=x}(y)f_X(x)$. So, in an unconditional model we explicitly or implicitly specify both the marginal probability density function $f_X(x)$ and the conditional probability density function $f_{X,Y}(x,y)$. On the other hand, in a conditional model, we specify only the conditional $f_{X,Y}(x,y)$ and we leave the marginal $f_X(x)$ unspecified.

Regression and classification

The following distinction is often made, especially in the field of machine learning:

1 If the output is a continuous random variable, then a conditional model is called a regression model;

2 If the output is a discrete random variable, taking finitely many values (typically few), then a conditional model is called a classification model.

13.4.1 Linear regression model

The linear regression model is probably the oldest, best understood and most widely used conditional model. In the linear regression model, the response variables y are assumed to be a linear function of the inputs x: $y_i = x_i\beta + \varepsilon_i$. A linear regression model is specified by making assumptions about the error term ε_i. For example, ε_i is often assumed to have a normal distribution with zero mean and to be independent of x_i. In such a case, we have that, conditional on the inputs x_i, the output y_i has a normal distribution with mean $x_i\beta$. As a consequence, the conditional density of y_i is:

$$f_{Y_i|X_i=x_i}(y_i) = \frac{1}{\sqrt{2\pi}}\frac{1}{\sigma}e^{\left(-\frac{1}{2}\frac{(y_i-x_i\beta)^2}{\sigma^2}\right)}$$

where σ^2 is the variance of ε_i. The parameters β and σ^2 are usually unknown and need to be estimated. So, we have a different conditional distribution for each of the values of β and σ^2 that are deemed plausible by the statistician before observing the sample. The set of all these conditional distributions (associated to the different parameters) constitutes the conditional model for (y_i, x_i).

13.4.2 Logistic classification model

In the logistic classification model, the response variable is a Bernoulli random variable. It can take only two values, either 1 or 0. It is assumed that the conditional probability mass function of y_i is a non-linear function of the inputs x_i:

$$P_{Y_i|X_i=x_i}(y_i) = \begin{cases} sigm(x_i\beta) & \text{if } y_i = 1 \\ 1 - sigm(x_i\beta) & \text{if } y_i = 0 \\ 0 & otherwise \end{cases}$$

where x_i is a $1 \times K$ vector of inputs, β is a $K \times 1$ vector of constants and $sigm(t)$ is the logistic function defined by

$$sigm(t) = \frac{1}{1 - e^{-t}}$$

13.5 Generative Model

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability. Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.
In probability and statistics, a generative model is a model for randomly generating observable data values, typically given some hidden parameters. It specifies a joint probability distribution over observation and label sequences. Generative models are used in machine learning for either modeling data directly (i.e., modeling observations drawn from a probability density function, or as an intermediate step to forming a conditional probability density function. A conditional distribution can be formed from a generative model through Bayes' rule.
Generative models contrast with discriminative models, in that a generative model is a full probabilistic model of all variables, whereas a discriminative model provides a model only for the target variable(s) conditional on the observed variables. Thus a generative model can be used, for example, to simulate (i.e. generate) values of any variable in the model, whereas a discriminative model allows only sampling of the target variables conditional on the observed quantities. Despite the fact that discriminative models do not need to model the distribution of the observed variables, they cannot generally express more complex relationships between the observed and target variables. They don't necessarily perform better than generative models at classification and regression tasks. In modern applications the two classes are seen as complementary or as different views of the same procedure. Examples of generative models include:

- Gaussian mixture model
- Hidden Markov model
- Probabilistic context free grammar
- Naive Bayes
- Averaged one dependence estimators
- Latent Dirichlet allocation
- Restricted Boltzmann machine
- Generative adversarial networks

13.6 Nearest Neighbor

The k-nearest-neighbors algorithm is a classification algorithm, and it is supervised: it takes a bunch of labelled points and uses them to learn how to label other points. To label a new point, it looks at the labelled points closest to that new point (those are its nearest neighbors), and has those neighbors vote, so whichever label the most of the neighbors have is the label for the new point (the "k" is the number of neighbors it checks). K-Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection. It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data). In pattern recognition, the k-nearest neighbors' algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

- In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.
- In k-NN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors.

k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms. Both for classification and regression, a useful technique can be used to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common

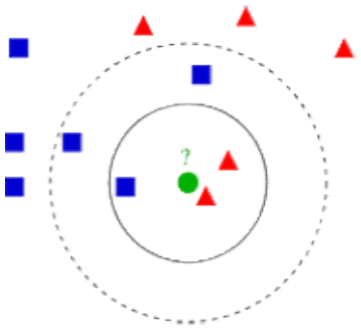

Fig. 10 k-NN classification

weighting scheme consists in giving each neighbor a weight of 1/d, where d is the distance to the neighbor. The neighbors are taken from a set of objects for which the class (for k-NN classification) or the object property value (for k-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

Algorithm

Let m be the number of training data samples. Let p be an unknown point.

1 Store the training samples in an array of data points arr[]. This means each element of this array represents a tuple (x, y).

2 for i=0 to m:

3 Calculate Euclidean distance d(arr[i], p).

4 Make set S of K smallest distances obtained. Each of these distances correspond to an already classified data point.

5 Return the majority label among S.

Example of k-NN classification Fig. **??**. The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If k = 3 it is classified to the second class because there are 2 triangles and only 1 square inside the inner circle. If k = 5 it is classified to first class (3 squares vs. 2 triangles inside the outer circle). The training examples are vectors in a multidimensional feature space. The space is partitioned into regions by locations and labels of the training samples. A point in the space is assigned to the class c if it is the most frequent class label among the k nearest training samples. Usually Euclidean distance is used. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples. In the actual classification phase, the test sample (whose class is not known) is represented as a vector in the feature space. Distances from the new vector to all stored vectors are computed and k closest samples are selected.

There are a number of ways to classify the new vector to a particular class, one of the most used technique is to predict the new vector to the most common class amongst the K nearest neighbors. A major drawback to use this technique to classify a new vector to a class is that the classes with the more frequent examples tend to dominate the prediction of the new vector, as they tend to come up in the K nearest neighbors when the neighbors are computed due to their large number. One of the ways to overcome this problem is to take into account the distance of each K nearest neighbors with the new vector that is to be classified and predict the class of the new vector based on these distances.

Parameter selection

The best choice of k depends upon the data; generally, larger values of k reduces effect of the noise on the classification, but make boundaries between classes less distinct. A good k can be selected by various heuristic techniques (see hyperparameter optimization). The special case where the class is predicted to be the class of the closest training sample (i.e. when k = 1) is called the nearest neighbor algorithm. The accuracy of the k-NN algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance. Much research effort has been put into selecting or scaling features to improve classification. A particularly popular approach is the use of evolutionary algorithms to optimize feature scaling. Another popular approach is to scale features by the mutual information of the training data with the training classes. In binary (two class) classification problems, it is helpful to choose k to be an odd number as this avoids tied votes. One popular way of choosing the empirically optimal k in this setting is via bootstrap method.

The 1-nearest neighbor classifier

The most intuitive nearest neighbour type classifier is the one nearest neighbour classifier that assigns a point x to the class of its closest neighbour in the feature space. As the size of training data set approaches infinity, the one nearest neighbour classifier guarantees an error rate of no worse than twice the Bayes error rate (the minimum achievable error rate given the distribution of the data).

A few Applications and Examples of KNN

- Credit ratings-collecting financial characteristics vs. comparing people with similar financial features to a database. By the very nature of a credit rating, people who have similar financial details would be given similar credit ratings. Therefore, they would like to be able to use this existing database to predict a new customer's credit rating, without having to perform all the calculations.
- Should the bank give a loan to an individual? Would an individual default on his or her loan? Is that person closer in characteristics to people who defaulted or did not default on their loans?
- In political science-classing a potential voter to a "will vote" or "will not vote", or to "vote Democrat" or "vote Republican".

- More advance examples could include handwriting detection (like OCR), image recognition and even video recognition.

Some pros and cons of KNN :
Pros:

- No assumptions about data - useful, for example, for non-linear data
- Simple algorithm - to explain and understand/interpret
- High accuracy (relatively) - it is pretty high but not competitive to better supervised learning models
- Versatile - useful for classification or regression

Cons:

- Computationally expensive - because the algorithm stores all of the training data
- High memory requirement
- Stores all (or almost all) of the training data
- Prediction stage might be slow (with big N)
- Sensitive to irrelevant features and the scale of the data