MACHINE LEARNING

[19CAT703] LECTURE NOTES

MCA II YEAR(2022-23)

DEPARTMENT OF COMPUTER APPLICATIONS

SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

19CAT703	MACHINE LEARNING	L	T	P	J	C
		3	0	0	0	3

UNIT-I FOUNDATIONS OF LEARNING 9

Components of learning –learning models –geometric models –probabilistic models –logic models – grouping and grading –learning versus design –types of learning –supervised –unsupervised – reinforcement –theory of learning –feasibility of learning–error and noise –training versus testing – theory of generalization –generalization bound –approximation-generalization tradeoff –bias and variance –learning curve

UNIT-II LINEAR MODELS 9

Linear classification –univariate linear regression –multivariate linear regression –regularized regression –Logistic regression –perceptrons –multilayer neural networks –learning neural networks structures –support vector machines –soft margin SVM –going beyond linearity –generalization and overfitting –regularization –validation

UNIT-III DISTANCE-BASED MODELS 9

Nearest neighbor models –K-means –clustering around medoids –silhouttes –hierarchical clustering –k-d trees –locality sensitive hashing–non-parametric regression –ensemble learning –bagging and random forests –boosting –meta learning

UNIT-IV TREE AND RULE MODELS 9

Decision trees –learning decision trees –ranking and probability estimation trees –regression trees – clustering trees –learning ordered rule lists –learning unordered rule lists –descriptive rule learning – association rule mining –first-order rule learning

UNIT-V MACHINE LEARNING WITH PYTHON'S SCIKIT-LEARN 9

Introduction to scikit-learn library – supervised learning – k nearest neighbors – linear regressions – support vector machines – support vector regression

L :45 T: 0 P: 0 J: 0 Total: 45 PERIODS

TEXT BOOKS

1 Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, ―Learning from Data, AML Book Publishers, 2012.

2 P. Flach, ―Machine Learning: The art and science of algorithms that make sense of data‖, Cambridge University Press, 2012.

REFERENCES

1 K. P. Murphy ―Machine Learning: A probabilistic perspective‖, MIT Press, 2012.

2 M. Mohri, A. Rostamizadeh, and A. Talwalkar ―Foundations of Machine Learning‖, MIT Press, 2012.

3 C. M. Bishop ―Pattern Recognition and Machine Learning‖, Springer, 2007.

4 D. Barber ―Bayesian Reasoning and Machine Learning‖, Cambridge University Press, 2012

COURSE OUTCOMES

At the end of the course student should be able to:

CO1 Able to understand various learning models of Machine Learning.

CO2 Develop algorithms to learn linear and non-linear models.

CO3 Apply data clustering algorithms on Analytical Problems.

CO4 Gain the knowledge on tree and rule-based models.

CO5 Apply reinforcement learning techniques for real life problems.

LESSON PLAN

Sl.No.	Topic	Method of Instruction	No. of Periods	Book to be referred	Page No
UNIT I - FOUNDATIONS OF LEARNING					
1.	Components of learning	Black board & ICT	1	T1	
2.	Learning models –geometric models – probabilistic models –logic models	ICT	1	T1,R4	
3.	Grouping and grading- learning versus design	ICT	1	T1	
4.	Types of learning –supervised – unsupervised –reinforcement	Role Play, ICT	1	T1,R1,R4	
5.	Theory of learning –feasibility of learning	Black board & ICT	1	T1	
6.	Error and noise - Training versus testing	Think pair share & ICT	1	T1	
7.	Theory of generalization –generalization bound	ICT	1	T1	
8.	Approximation-generalization tradeoff	ICT	1	T1	
9.	Bias and variance –learning curve	NPTELVideo	1	T1	
10.	Revision	Group Discussion	1	-	
		Unit I	10		

UNIT – I

FOUNDATION OF MACHINE LEARNING

Introduction:

Machine learning is programming computers to optimize a performance criterion using example data or past experience. We have a model defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience. The model may be predictive to make predictions in the future, or descriptive to gain knowledge from data, or both.

Arthur Samuel, an early American leader in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM. He defined machine learning as "the field of study that gives computers the ability to learn without being explicitly programmed." However, there is no universally accepted definition for machine learning. Different authors define the term differently.

What is Learning?

Definition: A computer program which learns from experience is called a machine learning program or simply a learning program.

Examples

i) Handwriting recognition learning problem

- Task T: Recognising and classifying handwritten words within images
- Performance P: Percent of words correctly classified
- Training experience E: A dataset of handwritten words with given classifications

ii) A robot driving learning problem

- Task T: Driving on highways using vision sensors
- Performance measure P: Average distance traveled before an error
- training experience: A sequence of images and steering commands recorded whileobserving a human driver

iii) A chess learning problem

- Task T: Playing chess
- Performance measure P: Percent of games won against opponents
- Training experience E: Playing practice games against itself

What is Machine Learning?

Definition I: Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

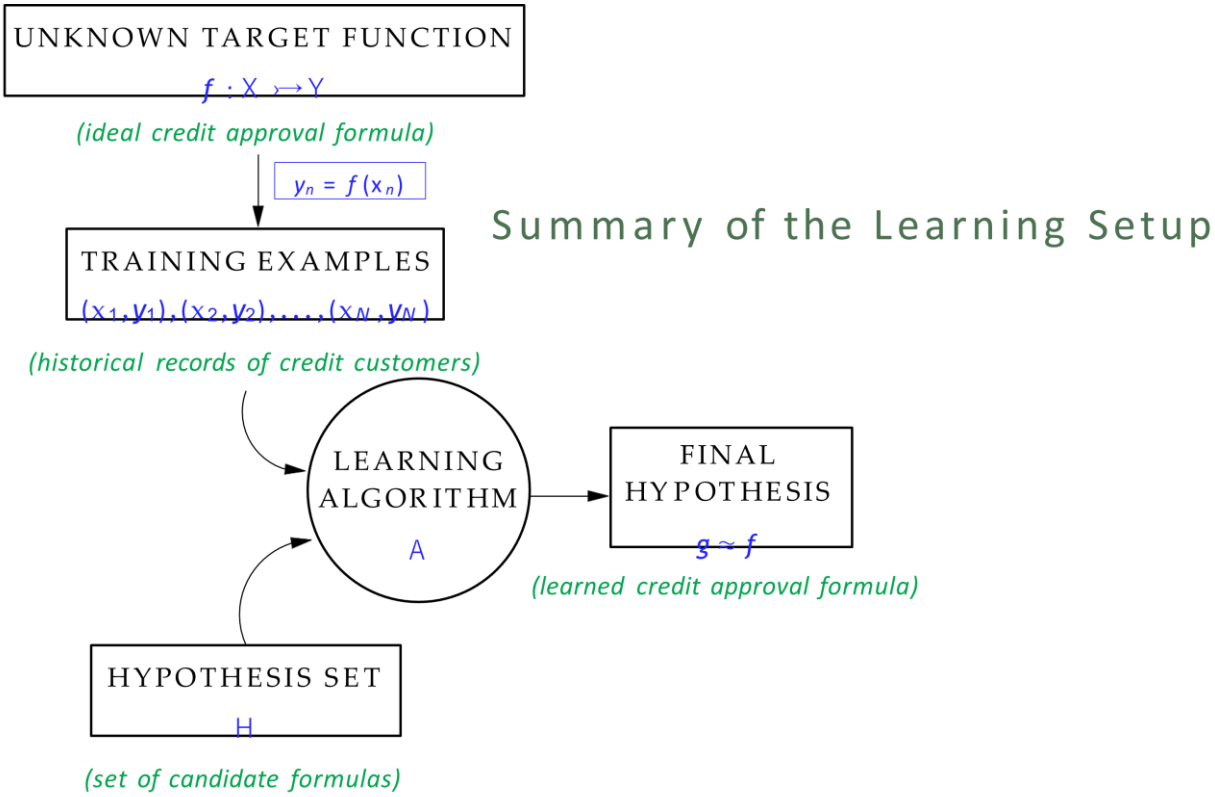
Definition II: Machine learning (ML) is **a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so**. Machine learning algorithms use historical data as input to predict new output values.

COMPONENTS OF LEARNING

Basic components of learning process

The learning process, whether by a human or a machine, can be divided into four components, namely:

1. Data storage
2. Abstraction
3. Generalization
4. Evaluation.



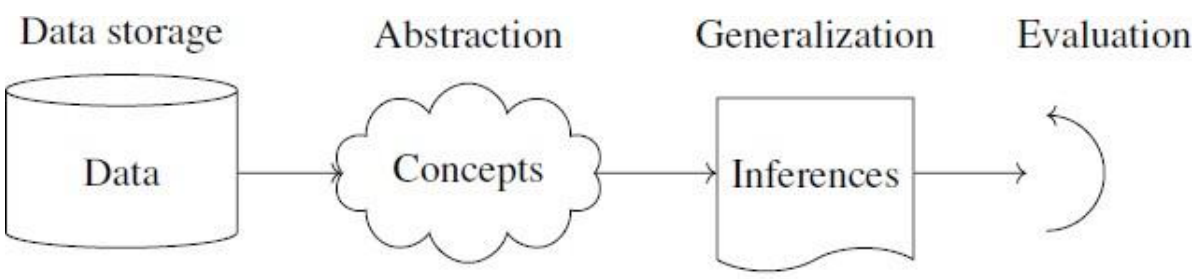


Figure 1.1: Components of learning process

1. Data storage

Facilities for storing and retrieving huge amounts of data are an important component of the learning process. Humans and computers alike utilize data storage as a foundation for advanced reasoning. In a human being, the data is stored in the brain and data is retrieved using electrochemical signals. Computers use hard disk drives, flash memory, random access memory and similar devices to store data and use cables and other technology to retrieve data.

2. Abstraction

Abstraction is the process of extracting knowledge about stored data. This involves creating general concepts about the data as a whole. The creation of knowledge involves application of known models and creation of new models. The process of fitting a model to a dataset is known as training. When the model has been trained, the data is transformed into an abstract form that summarizes the original information.

3. Generalization

The term generalization describes the process of turning the knowledge about stored data into a form that can be utilized for future action. These actions are to be carried out on tasks that are similar, but not identical, to those what have been seen before. In generalization, the goal is to discover those properties of the data that will be most relevant to future tasks.

4. Evaluation

It is the process of giving feedback to the user to measure the utility of the learned knowledge. This feedback is then utilized to effect improvements in the whole learning process

LEARNING MODELS –GEOMETRIC MODELS –PROBABILISTIC MODELS – LOGIC MODELS

Learning Models:

Machine learning is concerned with using the right features to build the right models that achieve the right tasks. The basic idea of Learning models has divided into three categories. For a given problem, the collection of all possible outcomes represents the **sample space or instance space**.

- Using a Logical expression. (**Logical models**)
- Using the Geometry of the instance space. (**Geometric models)**
- Using Probability to classify the instance space. (**Probabilistic models**)

1**. Logical models:** ***Tree models and Rule models***

Logical models use a logical expression to divide the instance space into segments and hence construct grouping models. A **logical expression** is an expression that returns a Boolean value, i.e., a True or False outcome.

Once the data is grouped using a logical expression, the data is divided into homogeneous groupings for the problem we are trying to solve.

For example, for a classification problem, all the instances in the group belong to one class.

There are mainly two kinds of logical models:

1. Tree models

2. Rule models.

Tree models

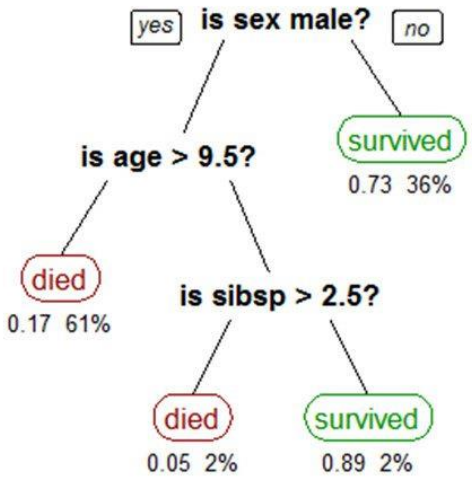
Tree models can be seen as a particular type of rule model where the if-parts of the rules are organised in a tree structure.

Rule models.

Rule models consist of a collection of implications or IF-THEN rules. For tree-based models, the 'if-part' defines a segment and the 'then-part' defines the behaviour of the model for this segment. Rule models follow the same reasoning.

Both Tree models and Rule models use the same approach to supervised learning.

The model can be summarised as: User chances of survival were good if user were (i) a female or (ii) a male younger than 9.5 years with less than 2.5 siblings.



Logical models and Concept learning

To understand logical models further, we need to understand the idea of **Concept Learning**. A Formal Definition for Concept Learning is "***The inferring of a Boolean-valued function from training examples of its input and output.***" In concept learning, we only learn a description for the positive class and label everything that doesn't satisfy that description as negative.

The following example explains this idea in more detail.

A Concept Learning Task – Enjoy Sport
Training Examples

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

ATTRIBUTES CONCEPT

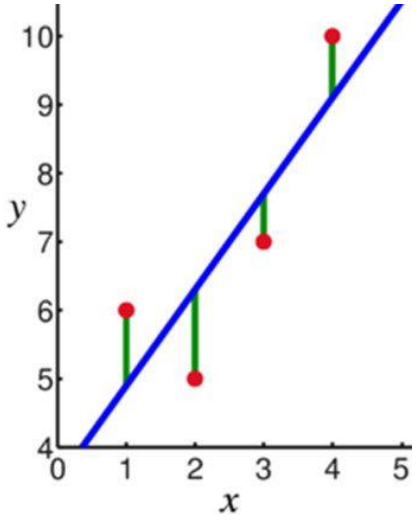
2. Geometric models

In Geometric models, features could be described as points in two dimensions (x- and y-axis) or a three-dimensional space (x, y, and z). Even when features are not intrinsically geometric, they could be modelled in a geometric manner (for example, temperature as a function of time can be modelled in two axes). In geometric models, there are two ways we could impose similarity.

- We could use geometric concepts like **lines or planes to segment (classify)** the instance space. These are called **Linear models**.
- Alternatively, we can use the geometric notion of distance to represent similarity. In this case, if two points are close together, they have similar values for features and thus can be classed as similar. We call such models as **Distance-based models**.

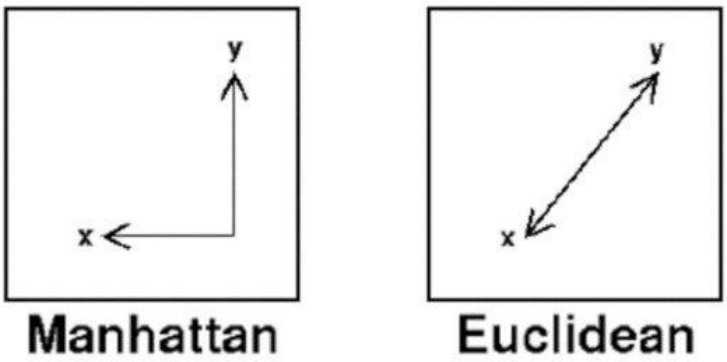
2.1 Linear models

Linear models are relatively simple. In this case, the function is represented as a linear combination of its inputs. Thus, if x_1 and x_2 are two scalars or vectors of the same dimension and a and b are arbitrary scalars, then $ax_1 + bx_2$ represents a linear combination of x_1 and x_2. In the simplest case where $f(x)$ represents a straight line, we have an equation of the form $f(x) = mx + c$ where c represents the intercept and m represents the slope.



2.2 Distance-based models

Distance-based models are the second class of Geometric models. Like Linear models, distance-based models are based on the geometry of data. As the name implies, distance-based models work on the concept of distance. In the context of Machine learning, the concept of distance is not based on merely the physical distance between two points. Instead, we could think of the distance between two points considering the **mode of transport** between two points. The distance metrics commonly used are **Euclidean**, **Minkowski**, **Manhattan**, and **Mahalanobis**.



Distance is applied through the concept of **neighbours and exemplars**. Neighbours are points in proximity with respect to the distance measure expressed through exemplars. Exemplars are either **centroids** that find a centre of mass according to a chosen distance metric or **medoids** that find the most centrally located data point. The most commonly used centroid is the arithmetic mean, which minimises squared Euclidean distance to all other points.

Notes:

- The **centroid** represents the geometric centre of a plane figure.
- **Medoids** are similar in concept to means or centroids.

Examples of distance-based models include the **nearest-neighbour** models, which use the training data as exemplars – for example, in classification. The **K-means clustering** algorithm also uses exemplars to create clusters of similar data points.

3. Probabilistic models

The third family of machine learning algorithms is the probabilistic models. We have seen before that the k-nearest neighbour algorithm uses the idea of distance (e.g., Euclidian distance) to classify entities, and logical models use a logical expression to partition the instance space. In this section, we see how the **probabilistic models use the idea of probability to classify new entities.**

Probabilistic models see features and target variables as random variables. The process of modelling represents and **manipulates the level of uncertainty** with respect to these variables. There are two types of probabilistic models: **Predictive and Generative**.

Predictive probability models use the idea of a **conditional probability** distribution $P(Y|X)$ from which Y can be predicted from X.

Generative models estimate the **joint distribution** $P(Y, X)$. Once we know the joint distribution for the generative models, we can derive any conditional or marginal distribution involving the same variables.

Naïve Bayes is an example of a probabilistic classifier.

The goal of any probabilistic classifier is given a set of features (x_0 through x_n) and a set of classes (c_0 through c_k), we aim to determine the probability of the features occurring in each class, and to return the most likely class. Therefore, for each class, we need to calculate $P(c\_i \mid x\_0, \ldots, x\_n)$.
We can do this using the **Bayes rule** defined as

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

The Naïve Bayes algorithm is based on the idea of **Conditional Probability. Conditional probability is based on finding the** probability that something will happen, *given that something else* has already happened. The task of the algorithm then is to look at the evidence and to determine the likelihood of a specific class and assign a label accordingly to each entity.

Some broad categories of models:

Geometric models:

1. K-Nearest Neighbors
2. Linear regression
3. Support vector Machine
4. Logistic Regression

Probabilistic models:

1. Naïve Bayes
2. Gaussian Process Regression
3. Conditional Random Field

Logical Models:

1. Decision Tree
2. Random Forest

GROUPING AND GRADING

Grading vs grouping is an orthogonal categorization to geometric-probabilistic-logical-compositional.

1. Grouping models break the instance space up into groups or segments and in each segment apply a very simple method (such as majority class).
    - E.g. decision tree, KNN.
2. Grading models form one global model over the instance space.
    - E.g. Linear classifiers – Neural networks

LEARNING VERSUS DESIGN

The design choices will be to decide the following key components:

1. **Type of training experience**
2. **Choosing the Target Function**
3. **Choosing a representation for the Target Function**
4. **Choosing an approximation algorithm for the Target Function**
5. **The final Design**

Type of training experience

1. Direct or Indirect training experience.
2. Teacher or Not
3. Is the training experience good

Choosing the Target Function

1. During the direct experience
2. When there is an indirect experience

Choosing a representation for the Target Function

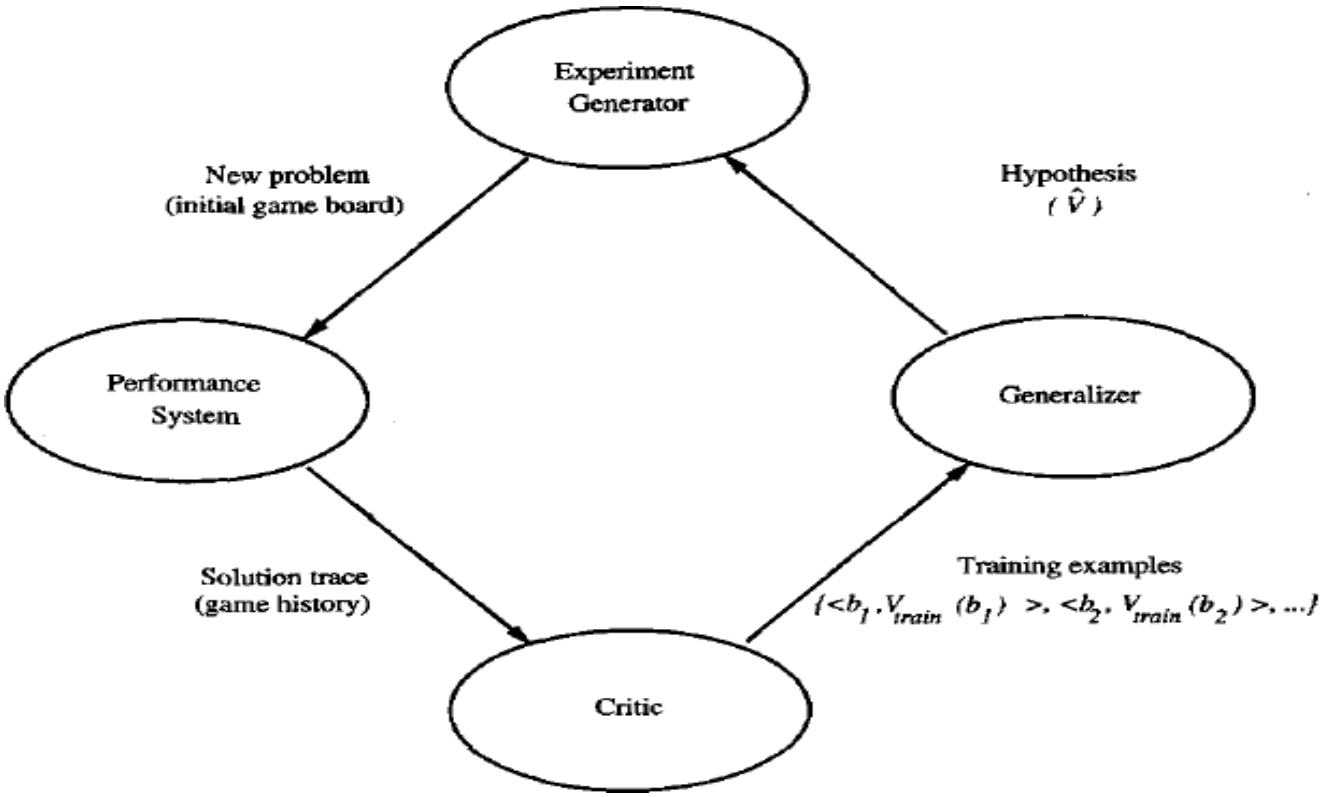
1. Specification of the Machine Learning Problem at this time

Choosing an approximation algorithm for the Target Function

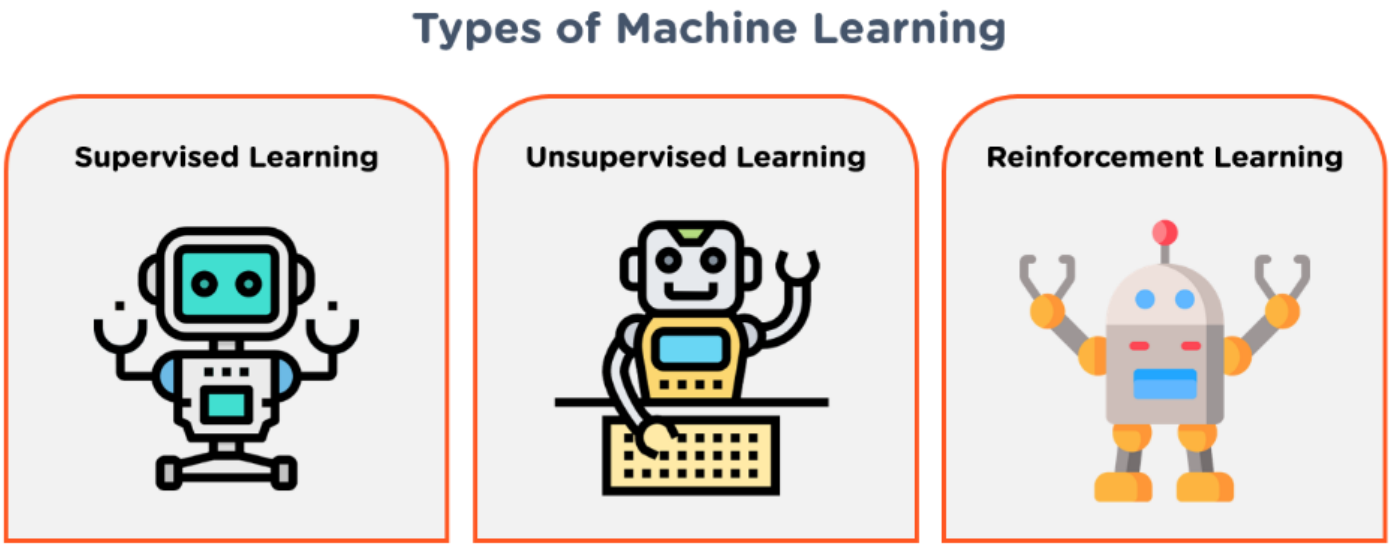
1. Generating training data

2. Adjusting the weights

Final Design for Checkers Learning system



Types of learning – Supervised – Unsupervised – Reinforcement

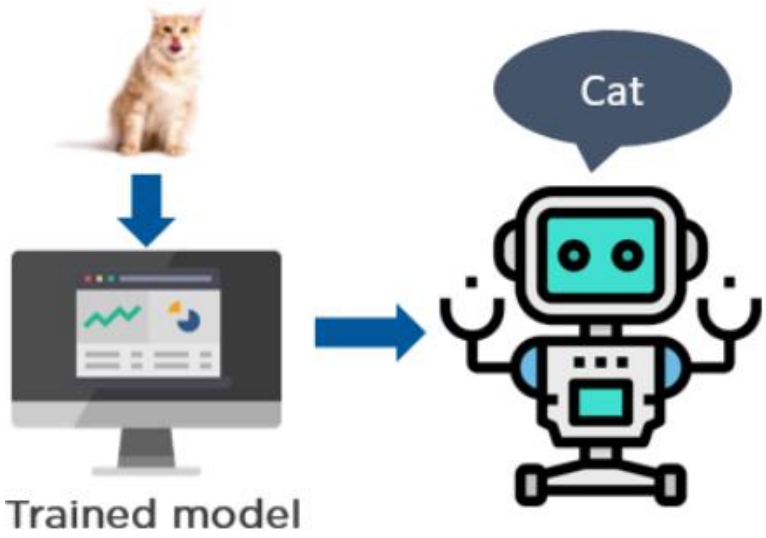


Supervised Learning

Overview:

Supervised learning is a type of machine learning that uses labeled data to train machine learning models. In labeled data, the output is already known. The model just needs to map the inputs to the respective outputs.

1. An example of supervised learning is to train a system that identifies the image of an animal.
2. Attached below, you can see that we have our trained model that identifies the picture of a cat.



Algorithms:

Some of the most popularly used supervised learning algorithms are:

- Linear Regression
- Logistic Regression
- Support Vector Machine
- K Nearest Neighbor
- Decision Tree
- Random Forest
- Naive Bayes

Working:

Supervised learning algorithms take labeled inputs and map them to the known outputs, which means you already know the target variable.

Now, let's focus on the training process for the supervised learning method.

Supervised Learning methods need external supervision to train machine learning models. Hence, the name supervised. They need guidance and additional information to return the desired result.

Applications:

Supervised learning algorithms are generally used for solving classification and regression problems.

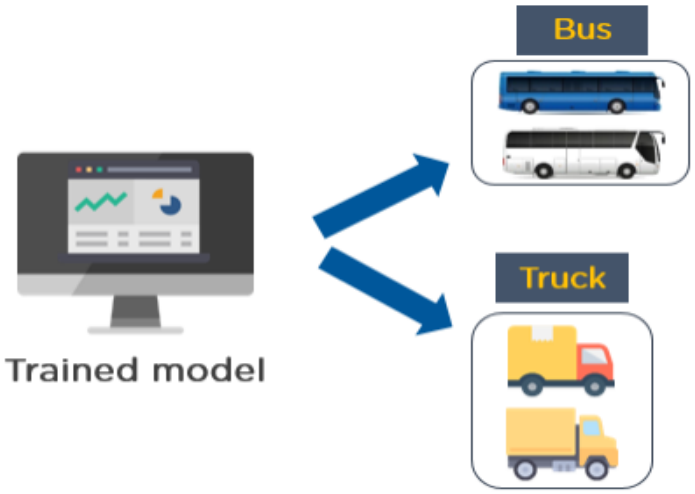
Few of the top supervised learning applications are weather prediction, sales forecasting, stock price analysis.

Now that you understand what Supervised learning is, let's see the next type of machine learning.

Unsupervised Learning
Overview:

Unsupervised learning is a type of machine learning that uses unlabeled data to train machines. Unlabeled data doesn't have a fixed output variable. The model learns from the data, discovers the patterns and features in the data, and returns the output.

Depicted below is an example of an unsupervised learning technique that uses the images of vehicles to classify if it's a bus or a truck. The model learns by identifying the parts of a vehicle, such as a length and width of the vehicle, the front, and rear end covers, roof hoods, the types of wheels used, etc. Based on these features, the model classifies if the vehicle is a bus or a truck.



Algorithms:

Selecting the right algorithm depends on the type of problem you are trying to solve. Some of the common examples of unsupervised learning are:

- K Means Clusterin
- Hierarchical Clustering
- DBSCAN
- Principal Component Analysis

Working:

Unsupervised learning finds patterns and understands the trends in the data to discover the output. So, the model tries to label the data based on the features of the input data.

The training process used in unsupervised learning techniques does not need any supervision to build models. They learn on their own and predict the output.

Applications:

Unsupervised learning is used for solving clustering and association problems.

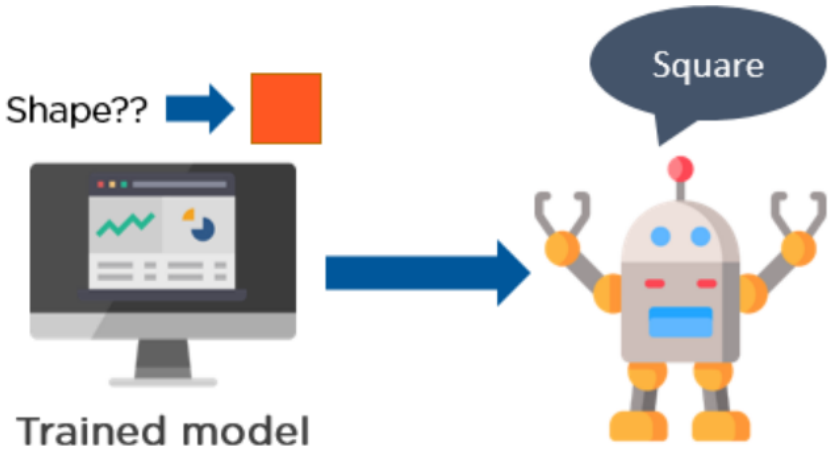
One of the applications of unsupervised learning is customer segmentation. Based on customer behavior, likes, dislikes, and interests, you can segment and cluster similar customers into a group. Another example where unsupervised learning algorithms are used is used churn rate analysis.

Let’s see the third type of machine learning, i.e., reinforcement learning.

Reinforcement Learning
Overview

Reinforcement Learning trains a machine to take suitable actions and maximize its rewards in a particular situation. It uses an agent and an environment to produce actions and rewards. The agent has a start and an end state. But, there might be different paths for reaching the end state, like a maze. In this learning technique, there is no predefined target variable.

An example of reinforcement learning is to train a machine that can identify the shape of an object, given a list of different objects. In the example shown, the model tries to predict the shape of the object, which is a square in this case.



Algorithms

Some of the important reinforcement learning algorithms are:

1. Q-learning
2. Sarsa
3. Monte Carlo
4. Deep Q network

Working

Reinforcement learning follows trial and error methods to get the desired result. After accomplishing a task, the agent receives an award. An example could be to train a dog to catch the ball. If the dog learns to catch a ball, you give it a reward, such as a biscuit.

Reinforcement Learning methods do not need any external supervision to train models.

Reinforcement learning problems are reward-based. For every task or for every step completed, there will be a reward received by the agent. If the task is not achieved correctly, there will be some penalty added.

Now, let's see what applications we have in reinforcement learning.

Applications

Reinforcement learning algorithms are widely used in the gaming industries to build games. It is also used to train robots to do human tasks.

Error and noise dataset

1) the ***residual*** is the difference between the true phenomenon being studied and the model being employed to describe it.

2) ***noise*** is that part of the ***residual*** which is in-feasible to model by any other means than a purely statistical description. note that such modelling limitations also arise due to limitations of the measurement device (e.g. finite bandwidth & resolution).

3) ***error*** is that component of the ***residual*** that remains after accounting for the noise.

according to the above definitions:

a) ***noise*** and ***error*** are uncorrelated

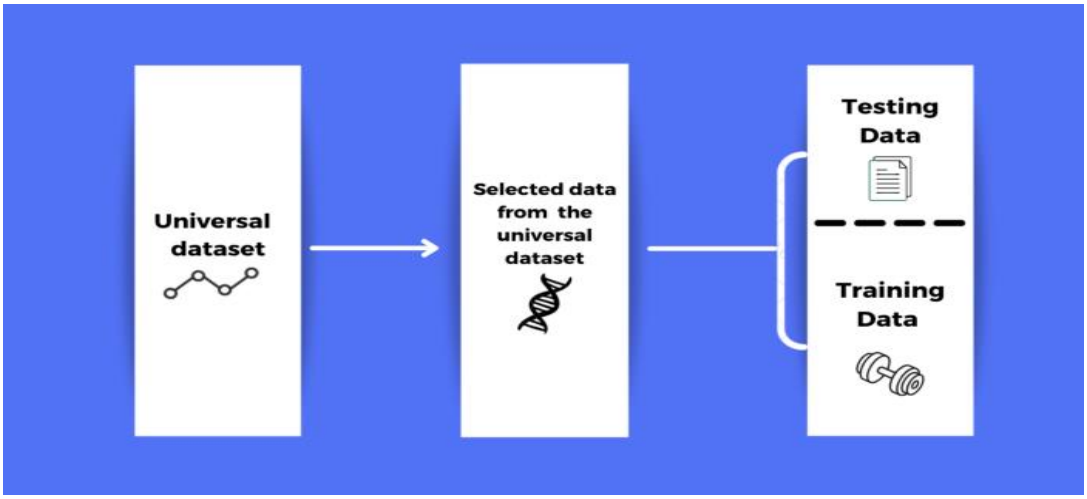
b) ***residual*** may be reduced by either reducing ***noise*** or by reducing ***error***

c) these definitions are compatible with the intuitive statements that "*noise does not introduce bias*" and "*bias is a class of error*".

finally note that ***error*** can only be reduced by improving the model (either of the phenomenon or of the measurement process). however ***noise*** may be reduced by either improving the measurement device, or by improving the model fidelity.

Training Vs Testing Dataset

Testing data and training data are two of the main pillars of the machine learning process. Without one there cannot be the other. In machine learning, an unknown universal dataset is assumed to exist, which contains all the possible data pairs as well as their probability distribution of appearance in the real world. *When we are dealing* with real applications, what we observe is only a subset of the universal dataset. This acquired dataset is called the training set and used to learn the properties and knowledge of the universal dataset.



In machine learning, what we desire is that these learned "properties" can not only explain the training set, but also be used to predict unseen samples or future events. In order to examine the performance of learning, another dataset may be reserved for testing, called the test set.

For example, before final exams, the teacher may give students several questions for practice (training set), and the way he judges the performances of students is to examine them with another problem set (test set). That is why you must split your data set into a training and a testing data set.

What are training data and testing data

Training Data: Training data is a set of samples (such as a collection of photos or videos) used to train machine learning models. Training datasets are fed to machine learning algorithms, in order to learn. They are necessary to teach the algorithm how to make accurate predictions in accordance with the goals of an AI project.

Just like people learn better from examples, machines also need to start isolating patterns in data. Unlike human beings, however, computers need a lot more examples because they do not think in the same way as humans do. They do not see objects in the pictures or can not recognize people in the photos as we can. They speak their own programming languages.

Testing Data: Testing data, as the name suggests, helps you validate the progress of the algorithm's training and adjust or optimize it for improved results.The testing dataset is a subset of the training initial one, and it is "shown" to the model just after it has completed its training. It is very important to keep the test dataset separate from the training one. It is used to provide an unbiased evaluation of the performance of a model and ensure that it can generalise well to new, unseen data.

Why do we need train and test sample

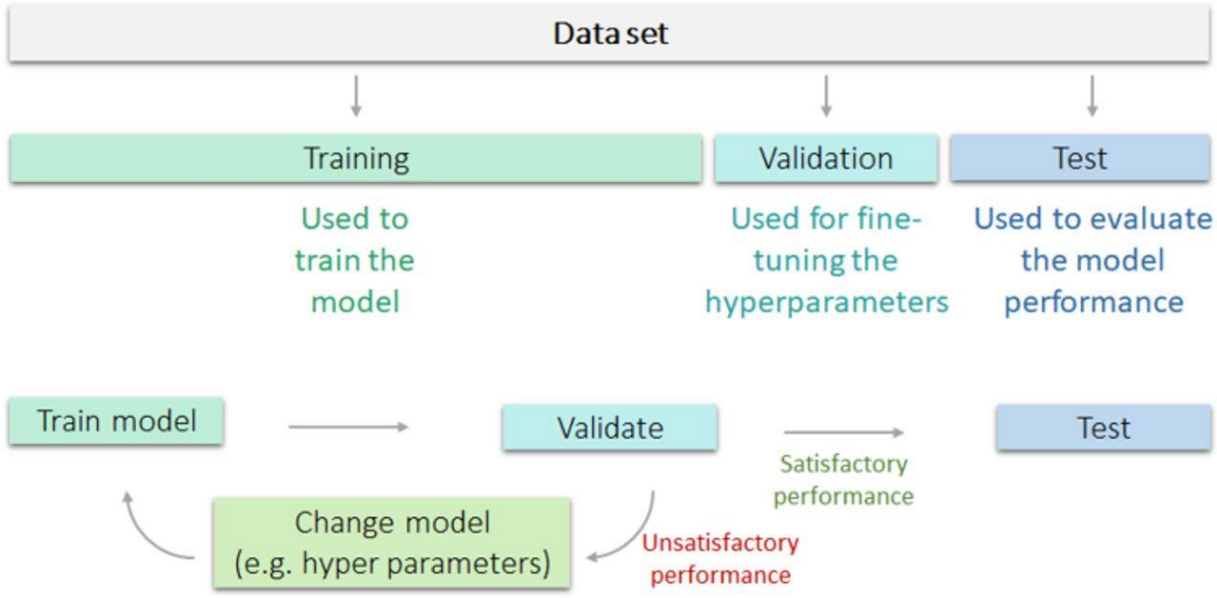
Creating different data samples for training and testing the model is the most common approach that can be used to identify these sorts of issues. The simplest way to split the modelling dataset into training and testing sets is to assign two thirds of the data to the former and the remaining one-third to the latter. Therefore, we train the model using the training set and then apply the model to the test set.

In this way, we can evaluate the performance of our model. For instance, if the training accuracy is extremely high while the testing accuracy is poor then this is a good indicator that the model is probably overfitted.

Overfitting and underfitting: what they are

A very common issue when training a model is overfitting. This phenomenon occurs when a model performs really well on the data that we used to train it but it fails to generalise well to new, unseen data. There are numerous reasons why this can happen — it could be due to the noise in data or it could be that the model learned to predict specific inputs rather than the predictive parameters that could help it make correct predictions. Typically, the higher the complexity of a model the higher the chance that it will be overfitted.

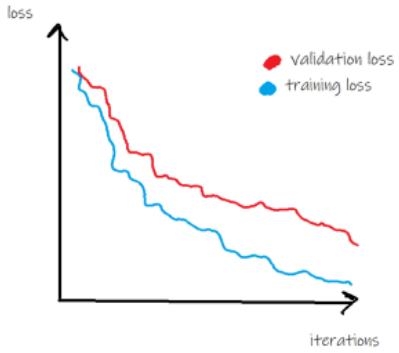
On the other hand, underfitting occurs when the model has poor performance even on the data that was used to train it. In most cases, underfitting occurs because the model is not suitable for the problem you are trying to solve. In general, an underfit model will be less flexible and cannot account for the data.



Theory of generalization –generalization bound

What is generalization?

I f training loss in fact does decrease as expected, it doesn't automatically mean that whatever the model has learned is also useful. This is where the validation loss comes into play. Things look good if the validation loss decreases alongside the training loss. In that case, the learned patterns seem to generalize to the unseen validation data. The validation loss will typically be higher than the training loss, however, since not all patterns generalize, as you can see in the following graphic.



If validation loss decreases as well, the learned patterns seem to generalize.

Bias

Bias is defined as the average squared difference between predictions and true values. It's a measure of how good your model fits the data. Zero bias would mean that the model captures the true data generating process perfectly. Both your training and validation loss would go to zero. That is unrealistic, however, as data is almost always noisy in reality, so some bias is inevitable — called the *irreducible error*.

Anyway, if losses do not decrease as expected, it probably signals that the model is not a good fit for the data. It would happen, for example, if you tried to fit an exponential relationship with a linear model — it can simply not adequately capture that relationship. Just try a different, more flexible model in that case.

You may also call this *underfitting*, with a slightly different connotation, though. Unlike bias, underfitting would imply that the model has still capacity to learn, so you would simply train for more iterations or collect more data.

Importantly, biases may also be hidden in the training data — which is easily overlooked. Your training loss may decrease as usual in that case. Only testing on real data can reveal any such bias.

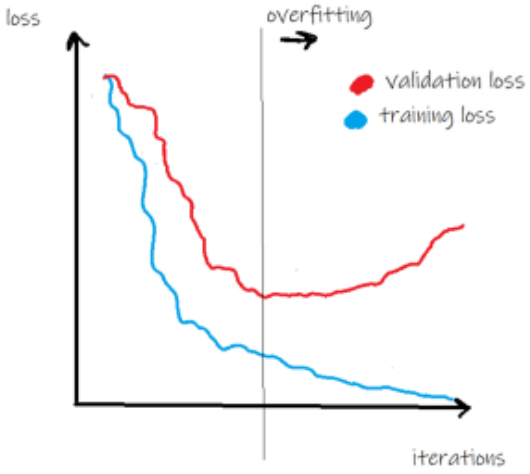
Variance

A model is said to have high variance if its predictions are sensitive to small changes in the input. In other words, you can think of it as the surface between the data points not being smooth but very wiggly. That is usually not what you want. High variance often means *overfitting* because the model seems to have captured random noise or outliers.

Like high bias and underfitting, high variance and overfitting are related as well but are still not totally equivalent in meaning. See below.

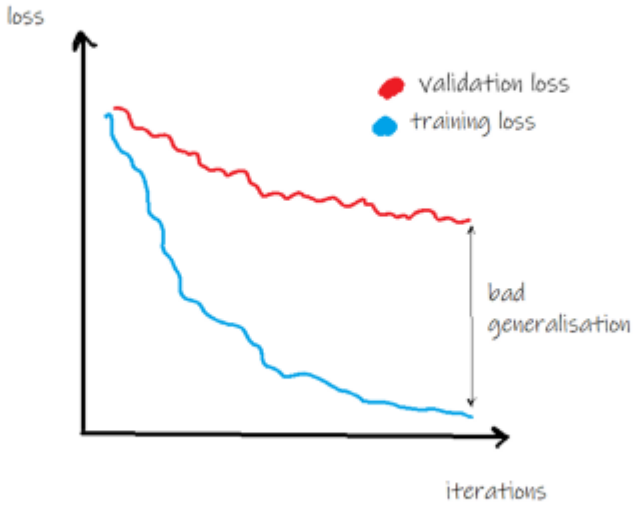
Overfitting

At some point during the training of a model, the validation loss usually levels out (and sometimes even starts to increase again) while the training loss continues to decrease. That signals overfitting. In other words, the model is still learning patterns but they do not generalize beyond the training set (see graphic below). Overfitting is particularly typical for models that have a large number of parameters, like deep neural networks.



Overfitting can happen after a certain number of training iterations.

A large gap between training and validation loss is a hint that the model does not generalize well and you may want to try to narrow that gap (graphic below). The simplest solution to overfitting is early-stopping, that is to stop the training loop as soon as validation loss is beginning to level off. Alternatively, regularization may help (see below). Underfitting, on the other hand, may happen if you stop too early.

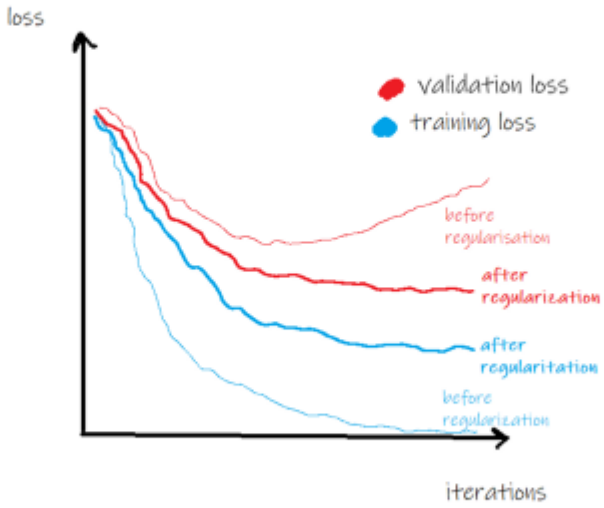


Generalization is low if there is large gap between training and validation loss.

Regularization

Regularization is a method to avoid high variance and overfitting as well as to increase generalization. Without getting into details, regularization aims to keep coefficients close to zero. Intuitively, it follows that the function the model represents is simpler, less unsteady. So predictions are smoother and overfitting is less

likely (graphic below). Regularization can be as simple as shrinking or penalizing large coefficients — often called *weight decay*. L1 and L2 regularization are two widely used methods. But you may also encounter different forms, such as dropout regularization in neural networks.



Regularization can help avoid high variance and overfitting.