Module 5

Planning and Learning		**12**	**CO4**
5.1	The planning problem, Planning Vs Searching, STRIPS and ADL, Planning with state space search, Partial order planning, Hierarchical planning, Contingent Planning		
	#Self learning : Multiagent planning		
5.2	Learning: Forms of Learning, Inductive Learning, Learning Decision Tree, applications of learning		
	#Self learning : Practical machine learning		
5.3	**Applications of AI:** Natural Language Processing(NLP):Language models, text classification, information retrieval, information extraction **Expert Systems:** Components of expert systems, ES vs Traditional System. Characteristics of expert systems, roles in ES implementation, ES implementation process, applications, advantages and limitations of ES		

What is planning in AI?

Planning in Artificial Intelligence is about the **decision making tasks performed by the robots** or computer programs to achieve a specific goal.

- The execution of planning is about **choosing a sequence of actions** with a high likelihood to complete the specific task.

Real-World Problems

- Planning: the task of coming up with a **sequence of action**s that will achieve a goal
  - Search-based problem-solving agent
  - Logical planning agent
  - Complex/large scale problems

- Consider only environments that are fully observable, deterministic, finite, static (change happens only when the agent acts), and discrete (in time, action, objects, and effects)
- **These are called classical planning environments.**
  - In contrast, non classical planning is for partially observable or stochastic environments and involves a different set of algorithms and agent designs

- **Assumption that most real-world problems are nearly decomposable.**
- **NEARLY DECOMPOSABLE** - Planner can **work on sub goals independently**, but might need to do some additional work to combine the resulting sub plan
- Key is to find a **language that is expressive enough** to describe a wide variety of problems, but restrictive enough to allow efficient algorithms to operate over it.
- Basic representation language of classical planners, known as the **STRIPS** language- STanford Research Institute Problem Solver.

STRIPS-STanford Research Institute Problem Solver

- A set of **states-** It is a conjunction of positive ground literals.
- A set of **goals-** partially specified state represented as a conjunction of positive literals, and
- A set of **actions-** For each action, there is a precondition that must be satisfied and an effect which reflects the impact the action has on the environment after it has been performed.

STRIPS

STRIPS (STanford Research Institute Problem Solver)

- a restrictive way to express states, actions and goals, but leads to more efficiency
- **States**: conjunctions of ground, function-free, and positive literals, such as At(Home) ^ Have(Banana)
- Closed-world assumption is used
- **Goals**: conjunctions of literals, may contain variables (existential), hence goal may represent more than one state
- E.g. At(Home) ^ ~ Have(Bananas)
- E.g. At(x) ^ Sells(x, Bananas)
- **Actions**: preconditions that must hold before execution and the effects after execution

STRIPS Action Schema

- An action schema includes:
- action name & parameter list (variables)

Example:

- Action: Buy (x)
- Precondition: At (p), Sells (p, x)
- Effect: Have(x)

Challenges of AI and Planning

- Closed world assumption: assumes that world model contains everything the robot needs to know: there can be no surprise
- Frame problem: how to represent real world situations in a manner that is computationally tractable

Planning with State-Space Search

- Planning algorithms:
- The most straightforward approach is to use state-space search
- Forward state-space search (Progression)
- Backward state-space search (Regression)

Problem Formulation for Progression

- Initial state:
  - Initial state of the planning problem
- Actions:
  - Applicable to the current state (actions' preconditions are satisfied)
- Goal test:
  - Whether the state satisfies the goal of the planning
- Step cost:
  - Each action is 1

PLANNING

- TOP – Total Order Planning
- POP- Partial Order Planning

Total-Order Planning

- Only explore linear sequences of actions from start to goal state
- They cannot take advantage of problem decomposition, i.e. splitting the problem into smaller sub-problems and solving them individually.

Partial-Order Planning

- Works on problem decomposition.
- Divide the problem into parts and achieve these sub goals independently.
- S**olves the sub problems with sub plans** and then **combines these sub plans and reorders them based on requirements.**
- Ordering of the actions is partial.
- Does not specify which action will come first out of the two actions which are placed in the plan.

POP Example

- Putting on a pair of shoes:
- Goal(RightShoeOn ^ LeftShoeOn)
- Init()
- Action: RightShoe
  - PRECOND: RightSockOn
  - EFFECT: RightShoeOn
- Action: RightSock
  - PRECOND: None
  - EFFECT: RightSockOn
- Action:LeftShoe
  - PRECOND: LeftSockOn
  - EFFECT: LeftShoeOn
- Action: LeftSock
  - PRECOND: None
  - EFFECT: LeftSockOn

Init: Barefoot

Goal: RightShoeOn ^ LeftShoeOn

Action: 1. RightShoeOn

Precondition: RightSockOn

Effect: RightShoeOn

2. LeftShoeOn

Precondition: LeftSockOn

Effect: LeftShoeOn

3. LeftSockOn

Precondition: Barefoot

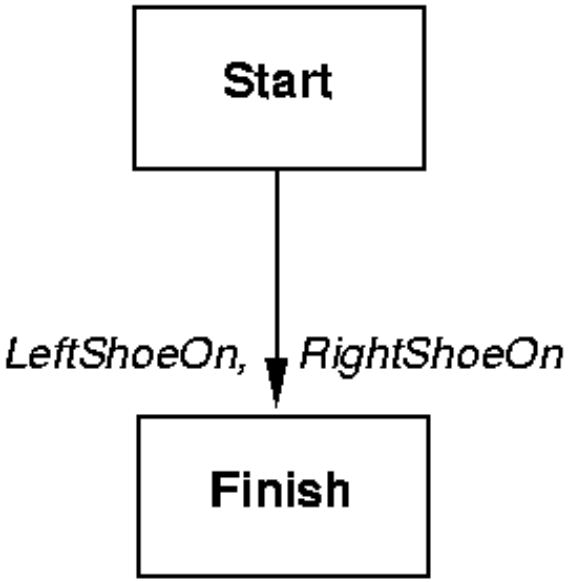
Effect: LeftSockOn

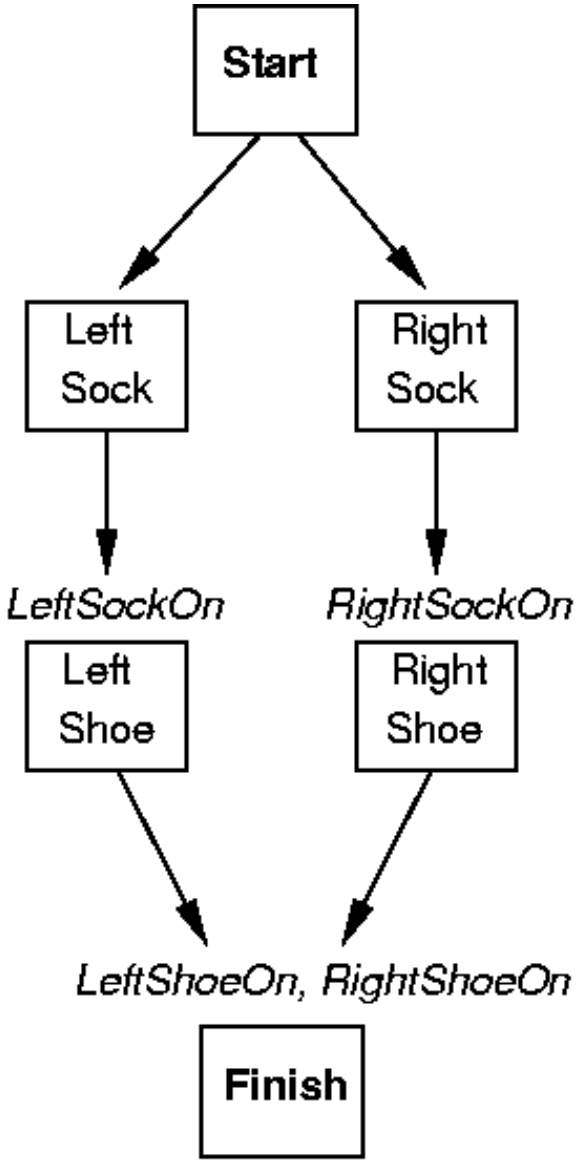
4. RightSockOn

Precondition: Barefoot

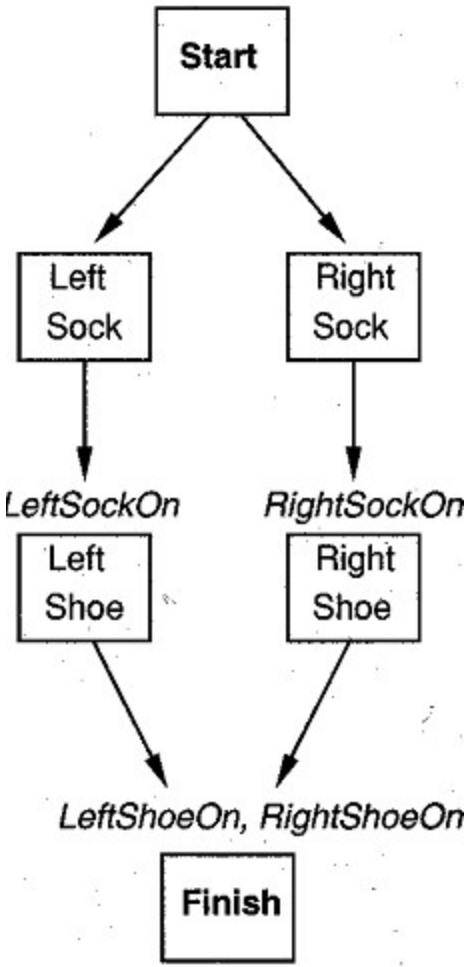
Effect: RightSockOn

POP Example

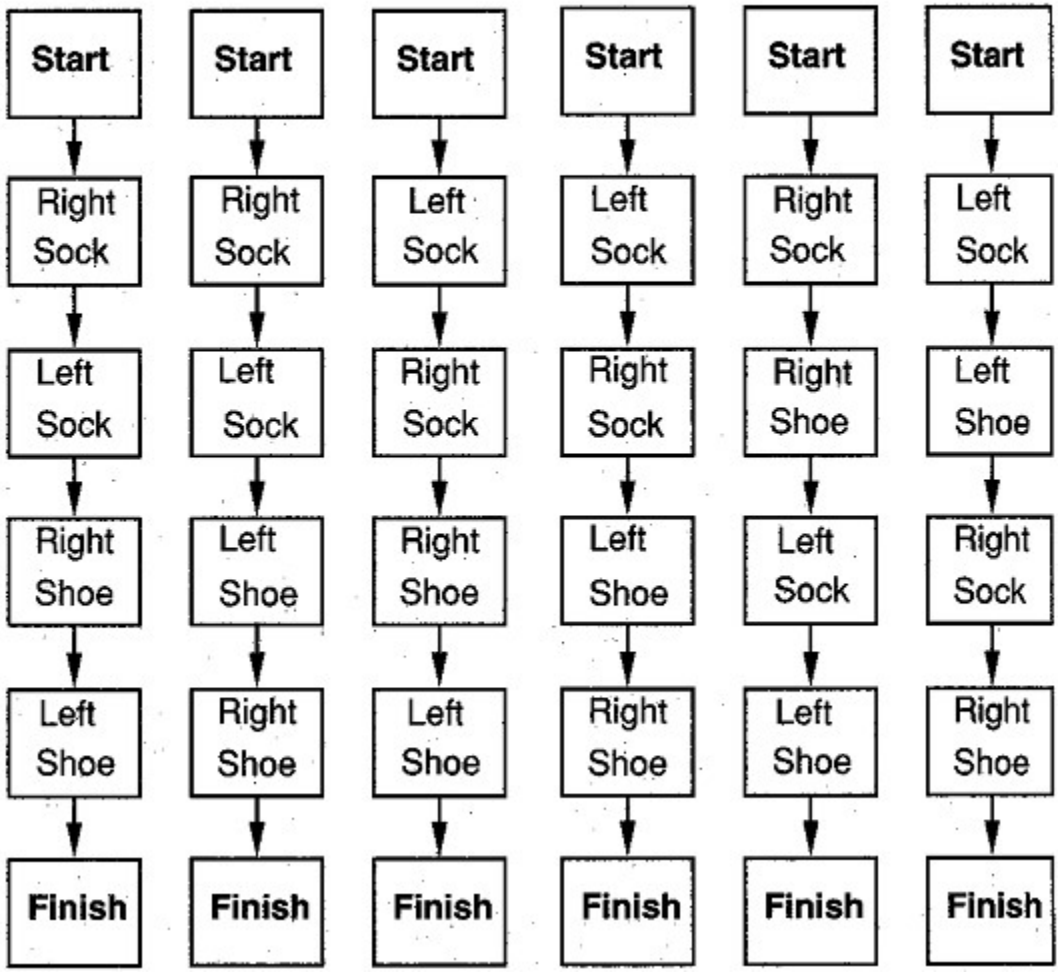

Start
LeftShoeOn, RightShoeOn
Finish


Start
Left Sock
Right Sock
LeftSockOn
RightSockOn
Left Shoe
Right Shoe
LeftShoeOn, RightShoeOn
Finish

Partial Order Plan:

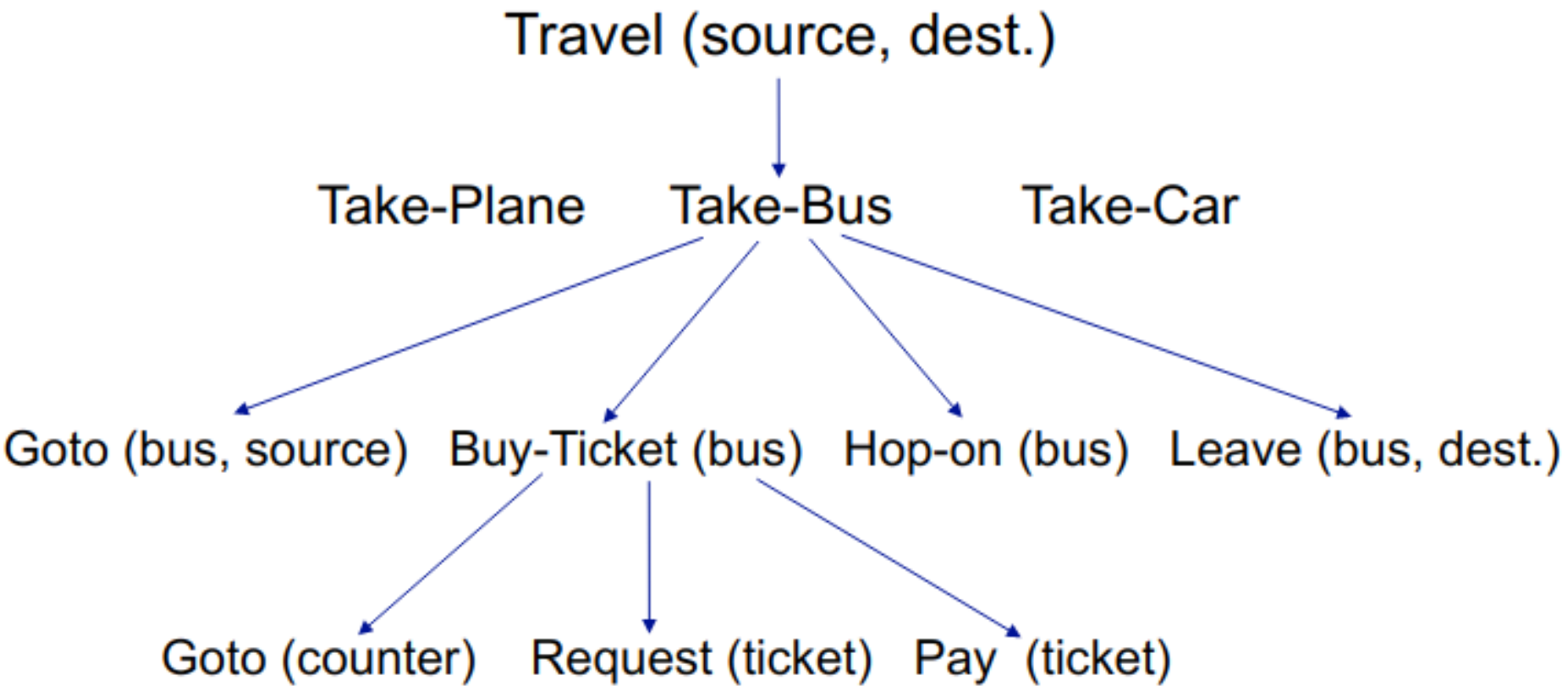


Total Order Plans:



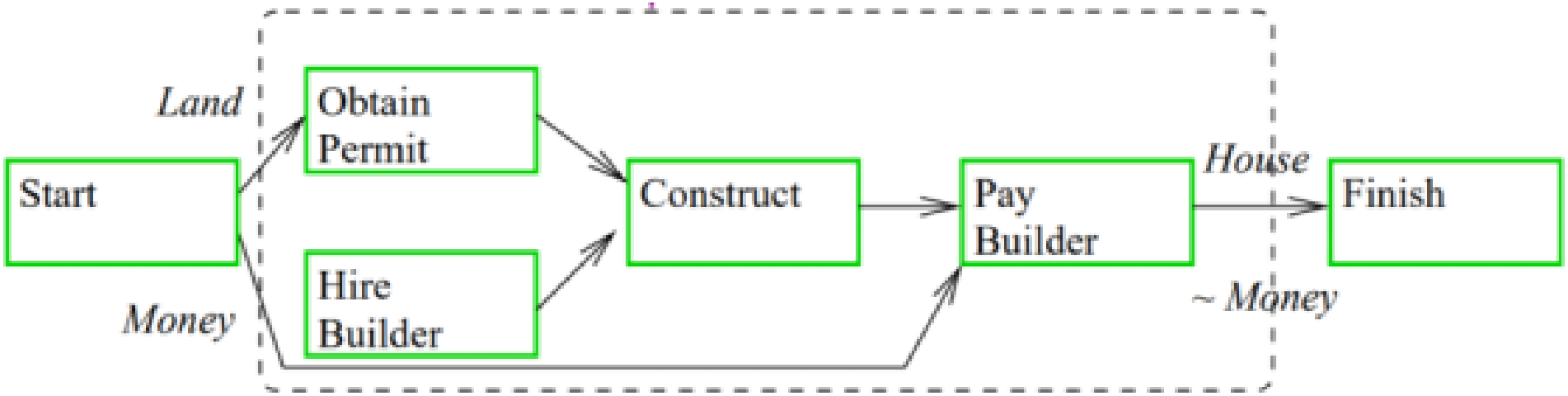
Hierarchical Planning

- Plans are organized in a hierarchical format.
- It works on plan decomposition.
- Complex actions are decomposed into simpler or primitive ones and it can be denoted with the help of links between various states at different levels of the hierarchy.


Travel (source, dest.)
Take-Plane
Take-Bus
Take-Car
Goto (bus, source)
Buy-Ticket (bus)
Hop-on (bus)
Leave (bus, dest.)
Goto (counter)
Request (ticket)
Pay (ticket)

For example

In case of building a house use of hierarchical planning



Conditional Planning

- Works regardless of the outcome of an action.
- Deals with uncertainty by inspecting what is happening in the environment at predetermined points in the plan.
- Fully observable and non-deterministic environments.
- Inputs actions and must be able to handle every outcome for the action taken.

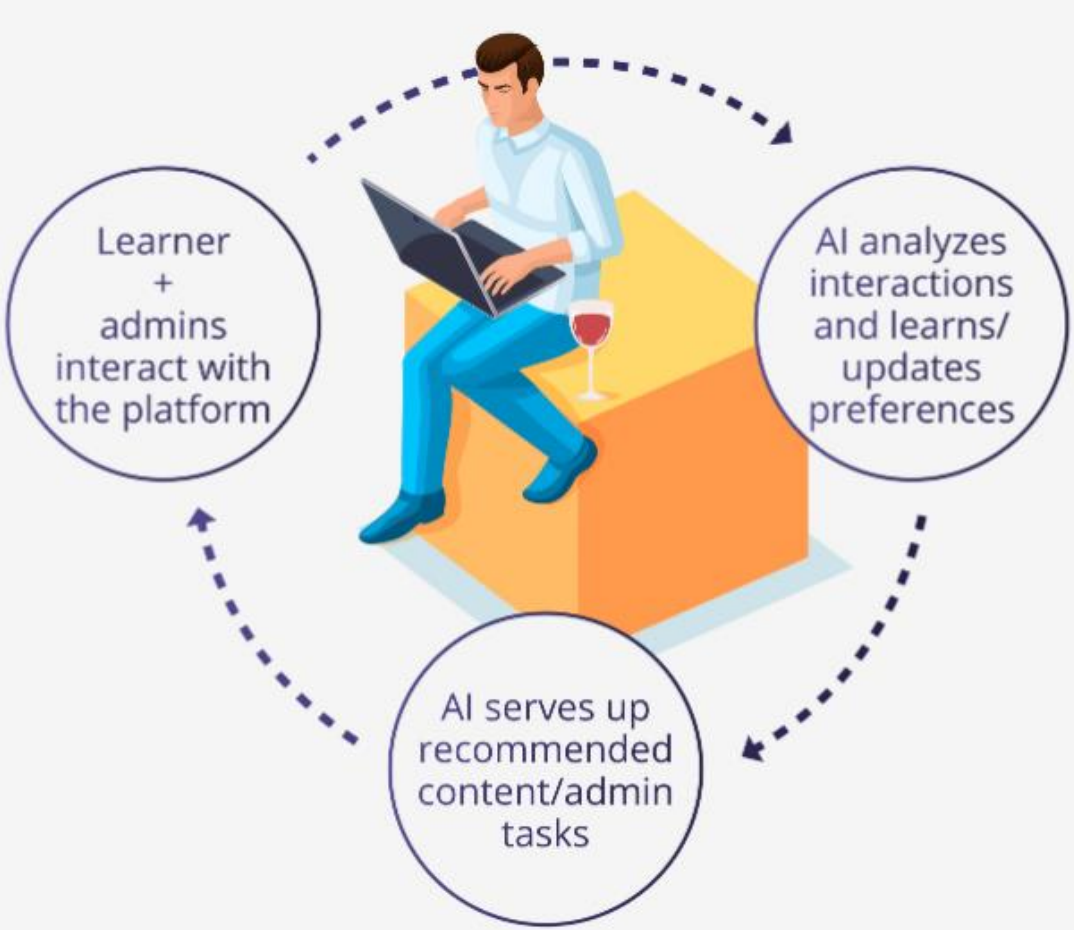
CONTINGENT PLANNING

Works under partial observability with sensing actions, agents actively use **sensing** to discover **meaningful facts about the world.**

SR. NO.	PARAMETERS	STRIPS	ADL
1	Stands for	**ST**anford **R**esearch Institute **P**roblem **S**olver	**A**ction **D**escription **L**anguage
2	Literals allowed in states	Only Positive. E.g.: Intelligent /\ Beautiful	Positive as well as Negative. E.g.: ~Stupid /\ ~Ugly
3	Assumption for Unmentioned Literals	Closed World Assumptions: Unmentioned literals are False	Open World Assumptions: Unmentioned literals are Unknown
4	Equality	No Support	Equality predicate (x=y) is Built-in
5	Types	No Support	Supported. Variables can have types -> p:Person
6	Effects	Effects are conjunctions	Conditional Effects allowed (when P:E)
7	Effect P /\ ~Q means:	Add P & Delete Q	Add P /\ ~Q Delete ~P /\ Q
8	Goals	Goals are conjunctions	Allows conjunctions & disjunctions: ~Poor /\ (Famous \/ Smart)
9	Goals	Only ground literals	Quantified Variables
10	Schema includes	Action name Parameter list Precondition Effect	Action name Parameter list (optional) Groups of clauses labelled: Precond, Add, Delete and Update (optional)
11	Modelling actions in real world applications	Not Suitable	Suitable (This inadequacy of STRIPS led to development of ADL)

Like A Fine Wine: AI Gets Better Over Time

The more you and your learners use and guide the AI-powered learning platform, the better the results it produces



LEARNING

- An **AGENT** is learning if it **improves its performance on future tasks** after making observations about the world.
  - Learning can range from the trivial → jotting down a phone number, to the profound → Albert Einstein, who inferred a new theory of the universe

Why would we want an agent to learn?

- If the **design** of the agent can be improved, why wouldn't the designers just program in that improvement to begin with?
- There are three main reasons

- **First**, the designers cannot anticipate all possible situations that the agent might find itself in.
  - For example, a robot designed to navigate mazes must learn the layout of each new maze it encounters.

- **Second**, the designers cannot anticipate all changes over time;
  - a program designed to predict tomorrow's stock market prices must learn to adapt when conditions change.

- **Third**, sometimes human programmers have no idea how to program a solution themselves.
  - For example, most people are good at recognizing the faces of family members, but even the best programmers are unable to program a computer to accomplish that task, except by using learning algorithms.

FORMS OF LEARNING

- Any component of an agent can be improved by learning from data.
- Improvements and techniques used to make them, depend on four major factors:
- **1. Which component is to be improved.**
- **2. What prior knowledge the agent already has.**
- **3. What representation is used for the data and the component.**
- **4. What feedback is available to learn from**

Components to be learned

- The components of the agents include:

1. A **direct mapping** from conditions on the current state to actions.
2. A means to **infer relevant properties** of the world from the percept sequence.
3. Information about the way the **world evolves** and about the results of possible actions the agent can take.
4. **Utility information** indicating the desirability of world states.
5. **Action-value information** indicating the desirability of actions.
6. **Goals** that describe classes of states whose achievement maximizes the agent's utility.

Each of these components can be learned

Consider, for example, an agent training to become a taxi driver.

- Every time the instructor shouts "**Brake**!" the agent might learn a condition–action rule for when to brake (component 1);
- the agent also learns every time the instructor does not shout.
- By seeing many camera images that it is told contain buses, it can learn to recognize them (2).
- By trying actions and observing the results—for example, braking hard on a wet road—it can learn the effects of its actions (3).
- Then, when it receives no tip from passengers who have been thoroughly shaken up during the trip, it can learn a useful component of its overall utility function (4).

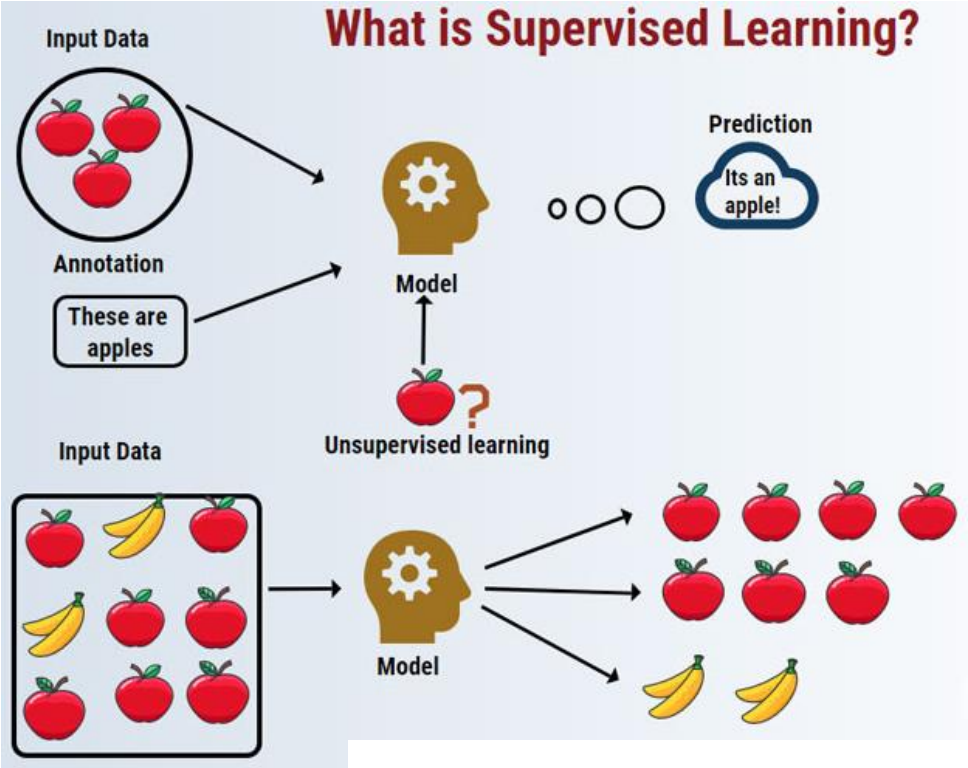
Feedback to learn from

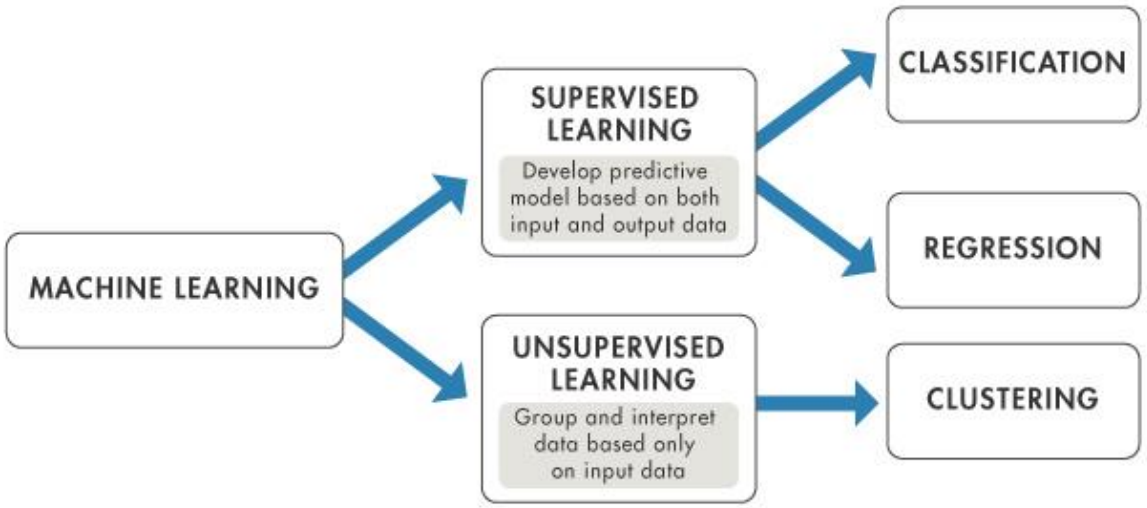
There are three types of feedback that determine the three main types of learning:

- **Supervised Learning**
- **Unsupervised Learning**
- **Reinforcement Learning**


Predicting the time you will reach home depends on
1) Weather Conditions
2) Time of Day
3) Holidays
4) Route Chosen
NW 62nd St
Dr Martin Luther
King Jr Blvd
EXIT ONLY
1/2

- **Supervised learning** tasks find patterns where we have a dataset of “right answers” to learn from.





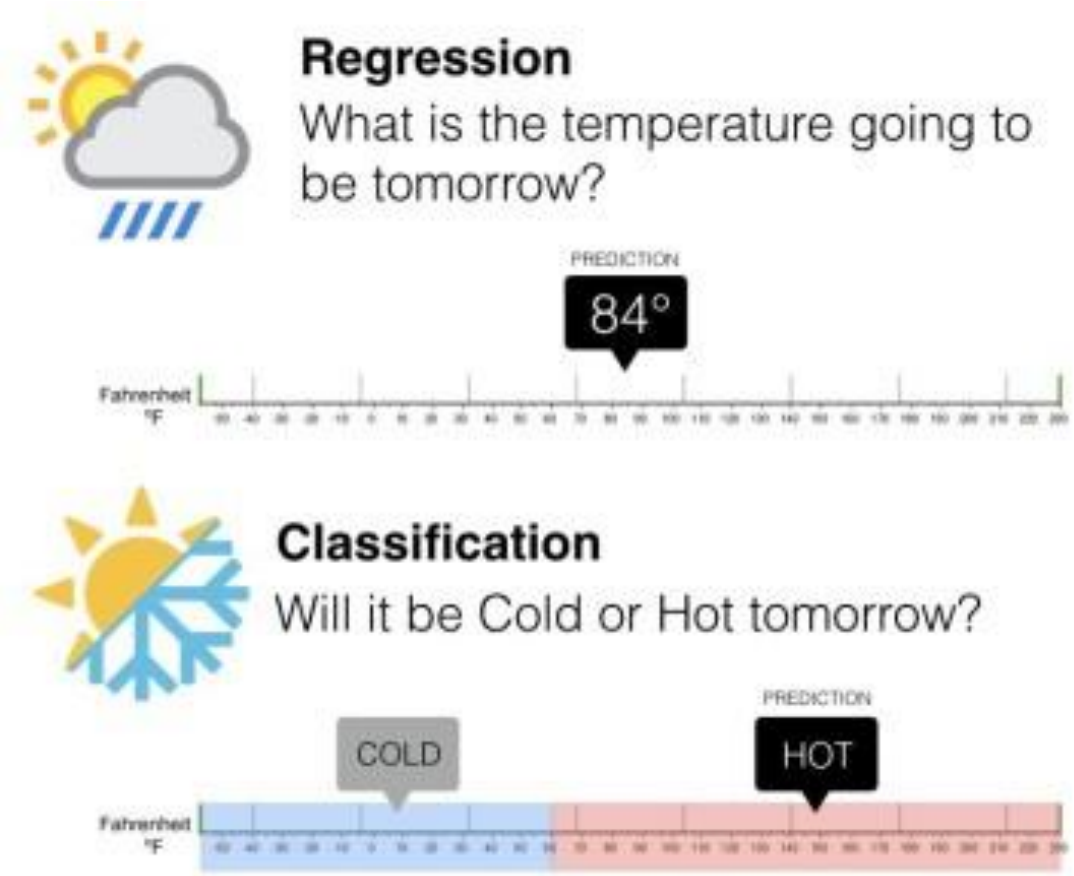
Supervised Learning

- Access to examples of **correct input-output pairs** that we can show to the machine during the training phase.
  - Common example of handwriting recognition is typically approached as a supervised learning task.

  We show the computer a number of images of handwritten digits along with the correct labels for those digits, and the computer learns the patterns that relate images to their labels.

Classification of Supervised Learning

- REGRESSION
- CLASSIFICATION



Classification of Supervised Learning

➢ Regression

- **P**roblem of **estimating or predicting** a continuous quantity.
  - What will be the value of the S&P 500 one month from today?
  - How tall will a child be as an adult?
  - How many of our customers will leave for a competitor this year?

Gather past examples of “right answer” input/output pairs that deal with the same problem.

For the inputs, we would identify **features** that we believe would be predictive of the outcomes that we wish to predict.

Classification of Supervised Learning

- **CLASSIFICATION** deals with assigning observations into **discrete categories,** rather than estimating continuous quantities.
- Ex: two possible categories;→**binary classification**.
- Many important questions can be framed in terms of binary classification.
  - Will a given customer leave us for a competitor?
  - Does a given patient have cancer?
  - Does a given image contain a hot dog?
    - For instance, a simple solution to the handwriting recognition problem is to simply train a bunch of binary classifiers: a 0-detector, a 1-detector, a 2-detector, and so on, which output their certainty that the image is of their respective digit. The classifier just outputs the digit whose classifier has the highest certainty.

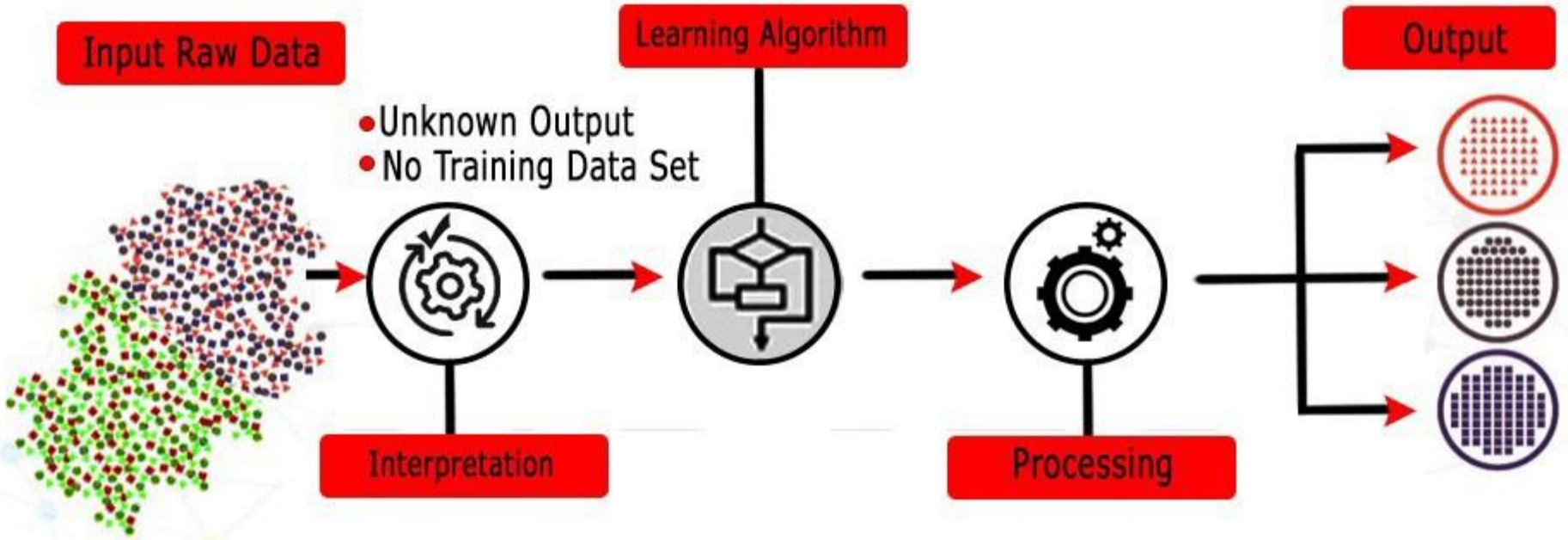
Algorithms commonly used in supervised learning

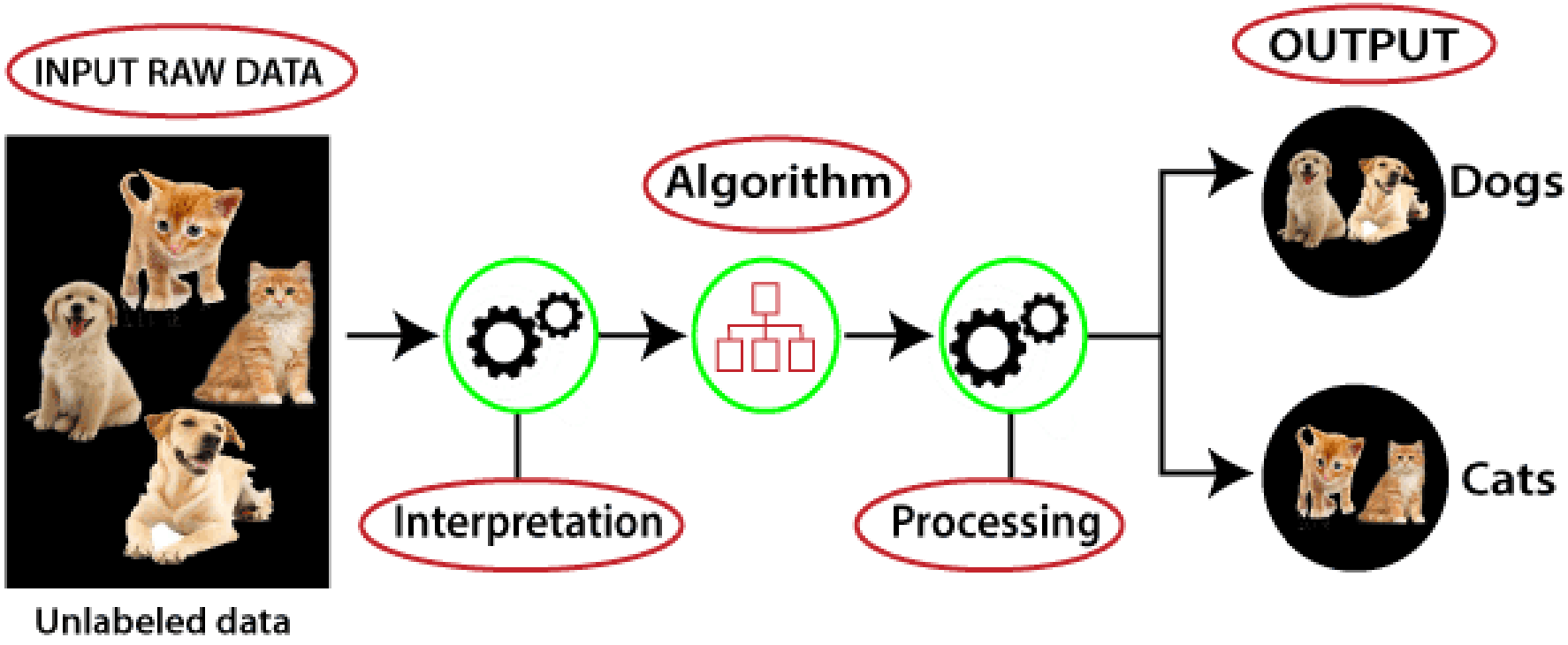
- Linear Regression
- Logistic Regression
- Neural Networks
- Linear Discriminant Analysis
- Decision Trees
- Similarity Learning
- Bayesian Logic
- Support Vector Machines (SVM)
- Random Forests

UNSUPERVISED LEARNING

- Agent **learns patterns in the input** even though no explicit feedback is supplied.
  - Most common unsupervised learning task→Clustering-detecting potentially useful clusters of input examples.
    - For example, a taxi agent might gradually develop a concept of "good traffic days" and "bad traffic days" without ever being given labeled examples of each by a teacher.

Unsupervised Learning




INPUT RAW DATA
Algorithm
OUTPUT
Dogs
Cats
Interpretation
Processing
Unlabeled data

- **Unsupervised learning** tasks find patterns where we don't.
  - The "right answers" are unobservable, or infeasible to obtain, or maybe for a given problem, there isn't even a "right answer" per se.
- Unsupervised learning is used against data without any historical labels.
- The system is **not given** a pre-determined set of outputs or correlations between inputs and outputs or a "correct answer."
- The algorithm must figure out what it is seeing by itself, it has no storage of reference points.
- The goal is to explore the data and find some sort of **patterns of structure.**

Clustering

sample

Cluster/group

Clustering

- Large subclass of unsupervised tasks is the problem of **clustering**.
  - Clustering refers to **grouping observations** together in such a way that members of a common group are similar to each other, and different from members of other groups.
- Common application →marketing, where we wish to identify segments of customers or prospects with similar preferences or buying habits.
- **Major challenge**→ Difficult /Impossible to know how many clusters should exist, or how the clusters should look.

Unsupervised Algorithm

- K-means clustering
- KNN (k-nearest neighbors)
- Hierarchal clustering
- Anomaly detection
- Neural Networks
- Principle Component Analysis
- Independent Component Analysis
- Apriori algorithm
- Singular value decomposition

REINFORCEMENT LEARNING

- Learns by interacting with its environment.
- It receives rewards by performing correctly and penalties for doing so incorrectly.
- Learns without having to be directly taught by a human – **it learns by seeking the greatest reward and minimising penalty.**
  - Learning is tied to a **context** because what may lead to maximum reward in one situation may be directly associated with a penalty in another.

REINFORCEMENT LEARNING

- Reinforcement learning the agent learns from a series of reinforcements—rewards or punishments.
  - For example, the lack of a tip at the end of the journey gives the taxi agent an indication that it did something wrong.
  - The two points for a win at the end of a chess game tells the agent it did something right.
  - It is up to the agent to decide which of the actions prior to the reinforcement were most responsible for it.

- Reinforcement learning tends to be used for gaming, robotics and navigation.
- The algorithm discovers which steps lead to the maximum rewards through a process of trial and error.
- When this is repeated, the problem is known as a **Markov Decision Process.**

Example

Facebook's News Feed is an example most of us will be able to understand.

Facebook uses machine learning to personalise people's feeds. If you frequently read or "like" a particular friend's activity, the News Feed will begin to bring up more of that friend's activity more often and nearer to the top. Should you stop interacting with this friend's activity in the same way, the data set will be updated and the News Feed will consequently adjust

Semi supervised learning

- Semi-supervised learning falls somewhere in the middle of supervised and unsupervised learning.
- It is used because many problems that AI is used to solving require a balance of both approaches.

Machine learning models cheat sheet

Supervised learning

Data scientists provide input, output and feedback to build model (as the definition)

EXAMPLE ALGORITHMS:

Linear regressions
- sales forecasting
- risk assessment

Support vector machines
- image classification
- financial performance comparison

Decision tree
- predictive analytics
- pricing

Unsupervised learning

Use deep learning to arrive at conclusions and patterns through unlabeled training data.

EXAMPLE ALGORITHMS:

Apriori
- sales functions
- word associations
- searcher

K-means clustering
- performance monitoring
- searcher intent

Semi-supervised learning

Builds a model through a mix of labeled and unlabeled data, a set of categories, suggestions and exampled labels.

EXAMPLE ALGORITHMS:

Generative adversarial networks
- audio and video manipulation
- data creation

Self-trained Naïve Bayes classifier
- natural language processing

Reinforcement learning

Self-interpreting but based on a system of rewards and punishments learned through trial and error, seeking maximum reward.

EXAMPLE ALGORITHMS:

Q-learning
- policy creation
- consumption reduction

Model-based value estimation
- linear tasks
- estimating parameters

Inductive Learning

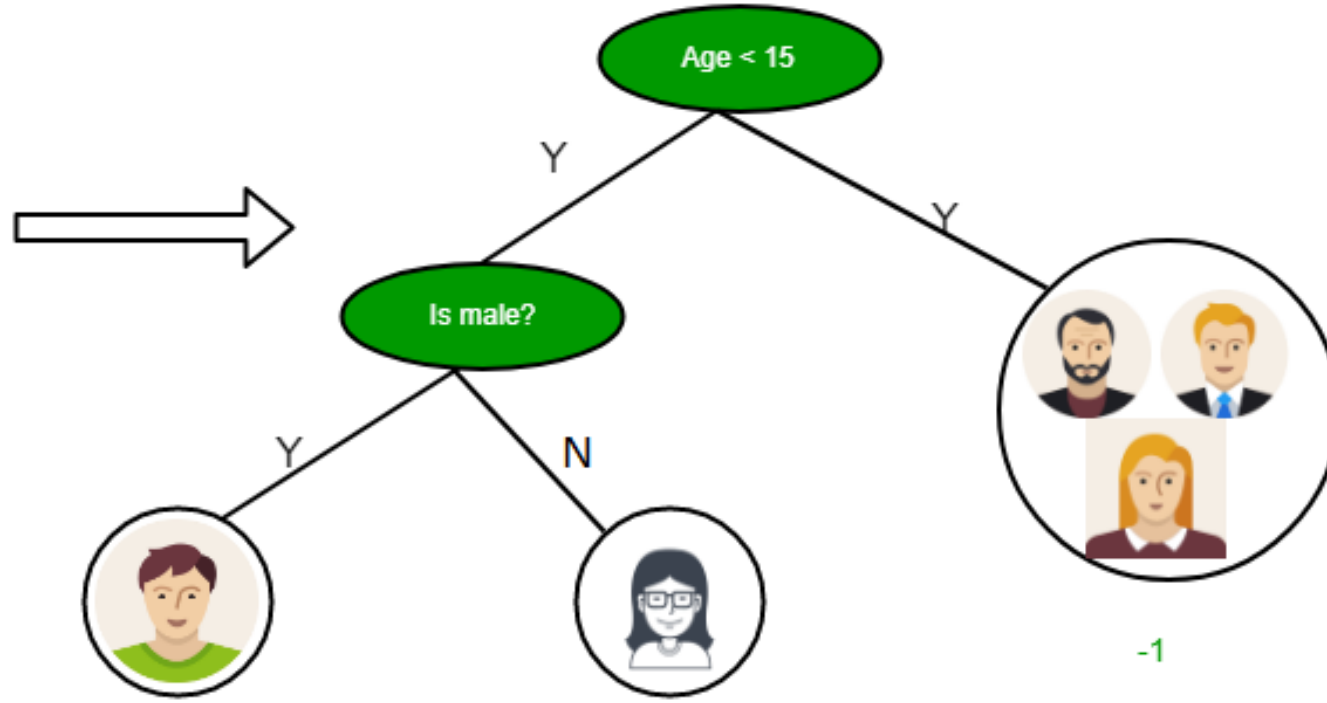
- This involves the process of ***learning by example*** -- where a system tries to **induce a general rule from a set of observed instances.**
- This involves classification -- assigning, to a particular input, the name of a class to which it belongs.
- Classification is important to many problem solving tasks.
- A learning system has to be capable of evolving its own class descriptions
  - Initial class definitions may not be adequate.
  - The world may not be well understood or rapidly changing.
- The task of constructing class definitions is called *induction* or *concept learning*

LEARNING DECISION TREE

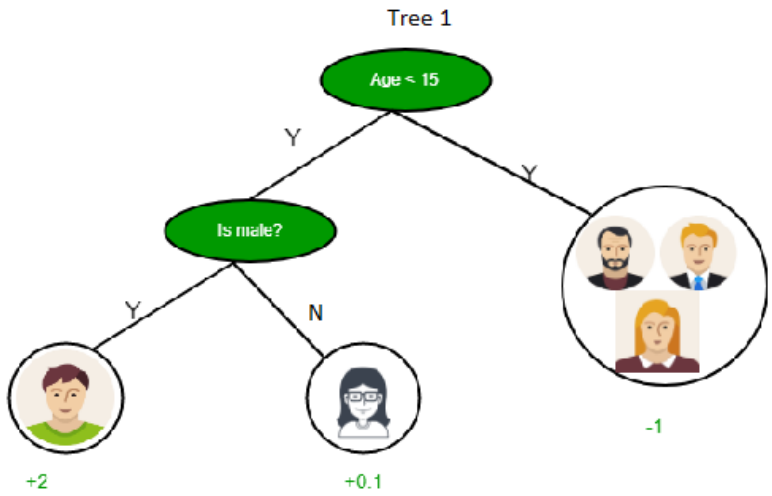
- A decision tree is a simple representation for classifying examples.
- Decision tree learning is one of the most successful techniques for supervised classification learning.
  - Assume that all of the features have finite discrete domains, and there is a single target feature called the **classification**.
  - Each element of the domain of the classification is called a **class**.

Input: Age, Gender, Occupation, . .

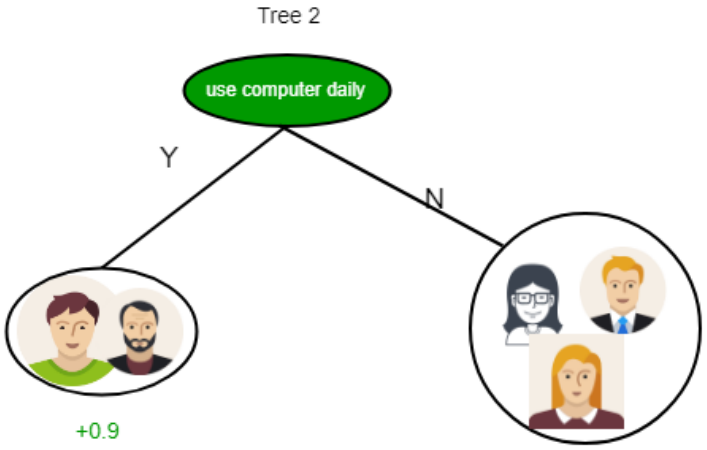
Does the person likes computer games



Prediction score in each leaf ⟶ +2 +0.1



If ( ) = 2 + 0.9 = 2.9

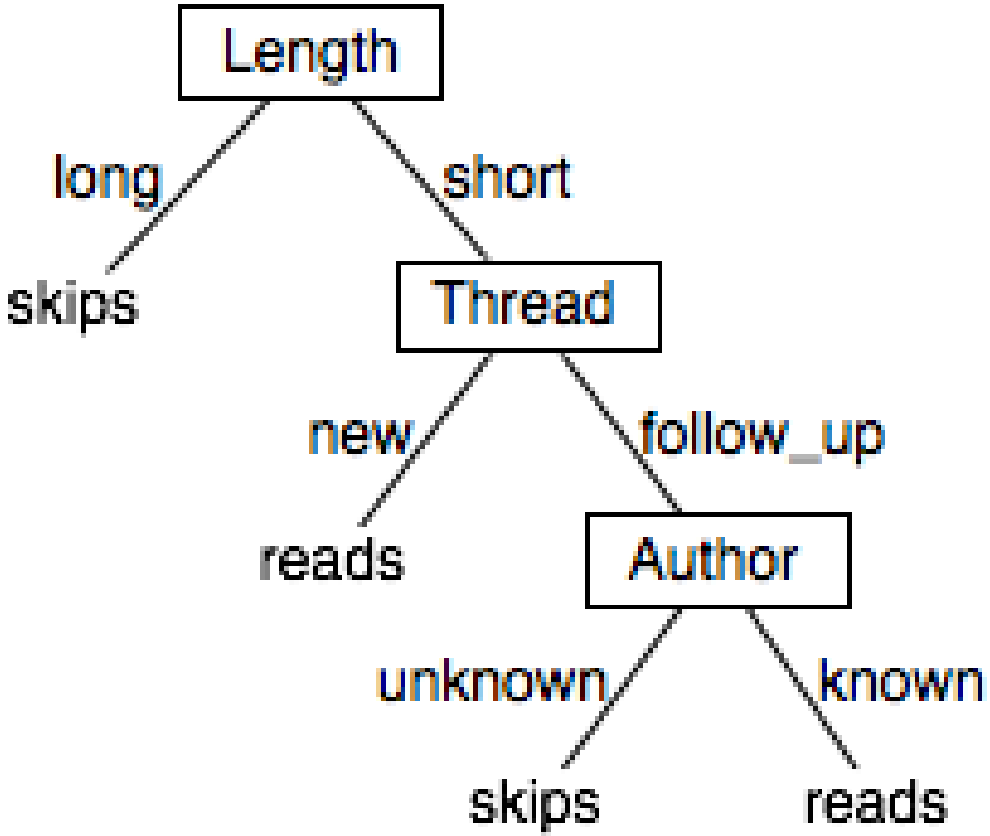


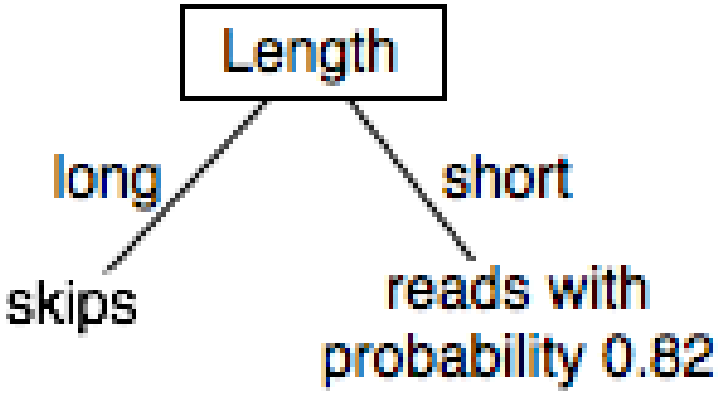
If ( ) = -1 + 0.9 = -0.1

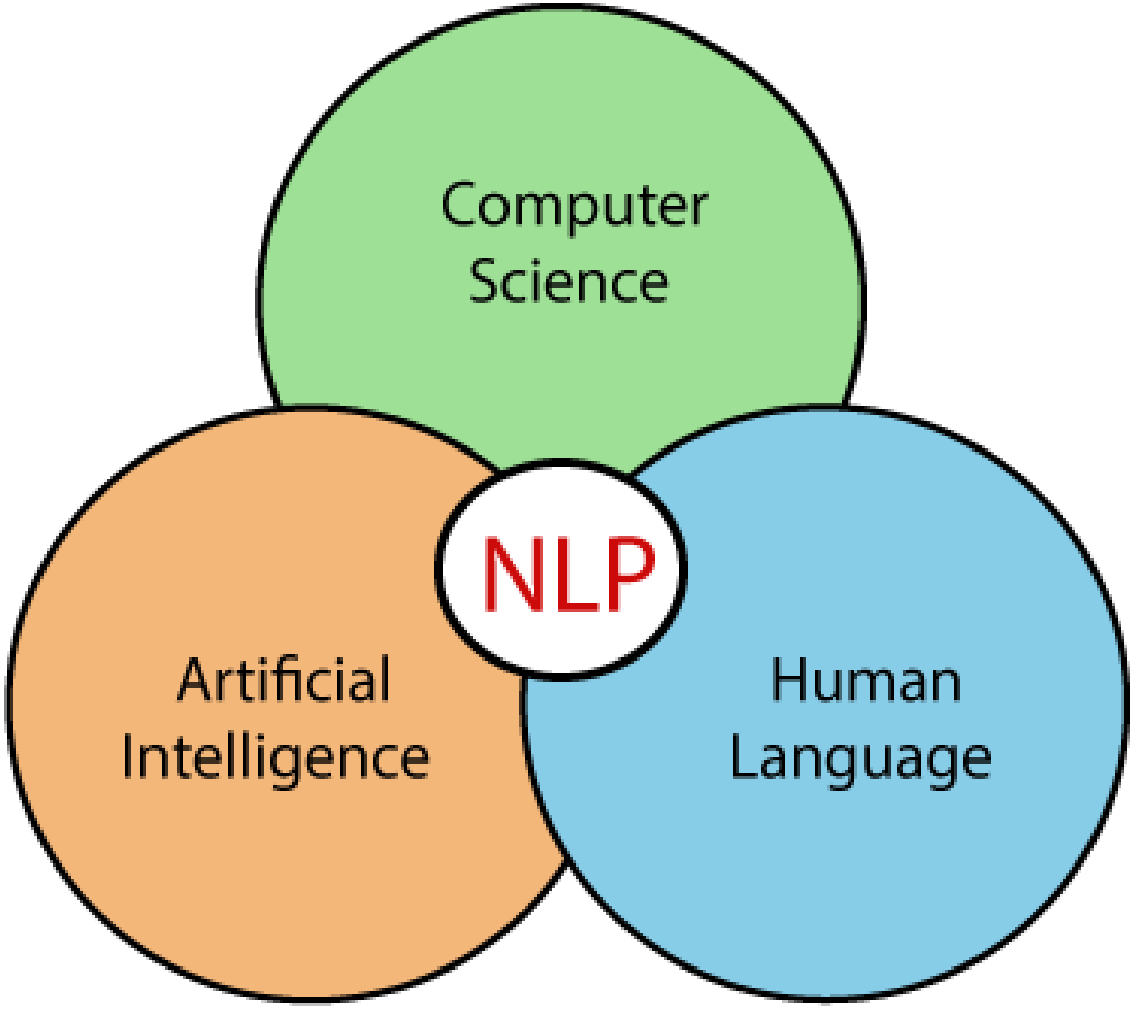
Example

Shows possible decision trees for the example where each decision tree can be used to classify examples according to the user's action. To classify a new example using the tree on the left, first determine the length. If it is long, predict skips. Otherwise, check the thread. If the thread is new, predict reads. Otherwise, check the author and predict read only if the author is known. This decision tree can correctly classify

The tree on the right makes probabilistic predictions when the length is short. In this case, it predicts reads with probability 0.82 and so skips with probability 0.18.


Length
long
short
skips
Thread
new
follow_up
reads
Author
unknown
known
skips
reads


Length
long
short
skips
reads with
probability 0.82


Computer
Science
NLP
Artificial
Intelligence
Human
Language

Natural Language Processing (NLP)

- NLP refers to AI method of communicating with an intelligent systems using a natural language such as English.
- Used in →Intelligent system like robot to perform as per your instructions, when you want to hear decision from a dialogue based clinical expert system, etc.
- Involves making computers to perform useful tasks with the natural languages humans use.
- The input and output of an NLP system can be −
  - Speech
  - Written Text

Components of NLP

1. NATURAL LANGUAGE UNDERSTANDING (NLU)

- Helps the machine to understand and analyse human language by extracting the **metadata from content such as concepts, entities, keywords**, emotion, relations, and semantic roles.
  - Used in Business applications to understand the customer's problem in both spoken and written language.
- NLU involves the following tasks -
  - Maps the given input into useful representation.
  - Analyzes different aspects of the language.

2. NATURAL LANGUAGE GENERATION (NLG)

- Natural Language Generation (NLG) acts as a **translator** that **converts the computerized data into natural language** representation.
- It mainly involves Text planning, Sentence planning, and Text Realization.

Steps to build an NLP

- Step1: **Sentence Segmentation-** Paragraph→ Sentences
- Step2: **Word Tokenization- Sentences** → Words/Tokens
- Step3: **Stemming-**Normalize words[intelligence, intelligent,& intelligently→root word "intelligen.“]
- Step 4: L**emmatization-**Grouping [Stemming]
- Step 5: **Identifying Stop Words-** removing is , and , a , the…
- Step 6: **Dependency Parsing**- relation of words in the sentence
- Step 7: **POS** tags-Noun, verb, adverb, and Adjective
- Step 8: **Named Entity Recognition (NER)**-detecting the named entity such as person name, movie name, organization name, or location.; **Steve Jobs** introduced I phones
- Step 9: **Chunking**-collect the individual piece of information and grouping them into bigger pieces of sentences.

Applications of NLP

Question Answering

- Question Answering focuses on building systems that automatically answer the questions asked by humans in a natural language.

Spam Detection

- Spam detection is used to detect unwanted e-mails getting to a user's inbox.

Sentiment Analysis

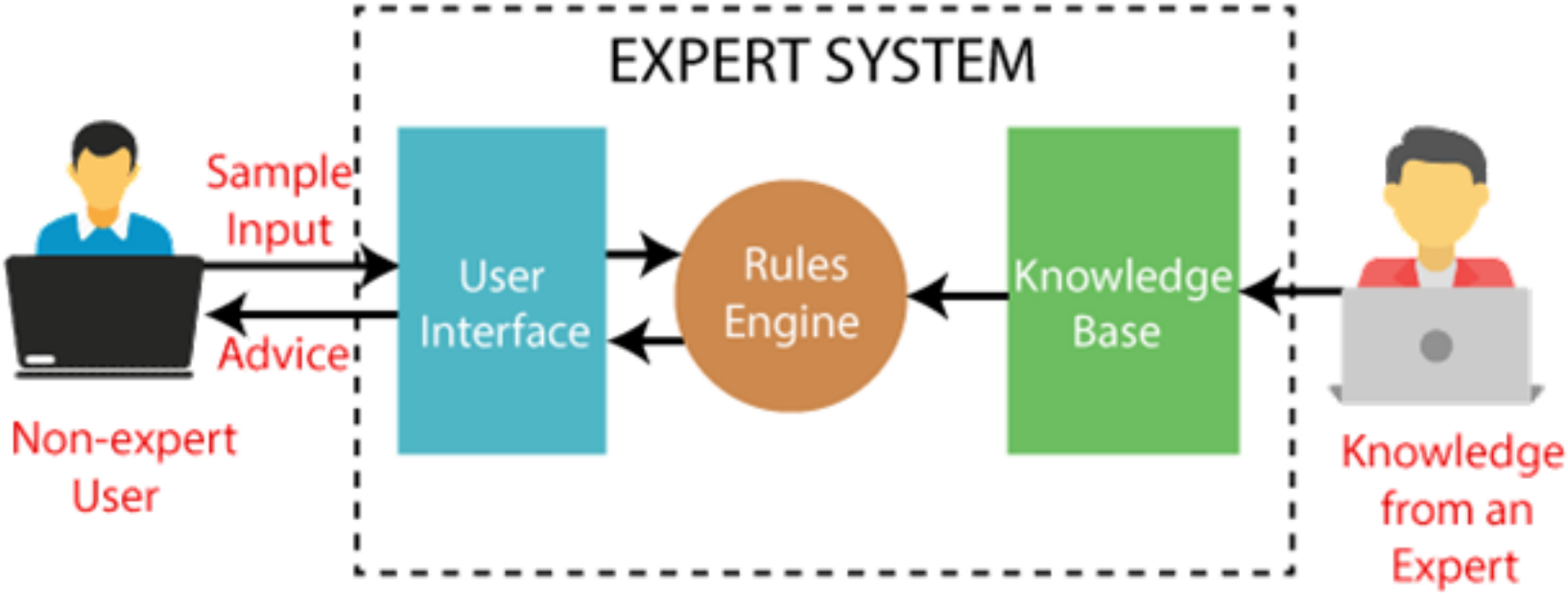
- Sentiment Analysis is also known as **opinion mining**
  - Analyse the attitude, behaviour, and emotional state of the sender.
- **Google Translator, Spelling correction, Speech Recognition , ChatBot,etc…..**

Expert systems (ES)

- One of the prominent research domains of AI.
- It is introduced by the researchers at Stanford University, Computer Science Department.

What are Expert Systems?

- The expert systems are the computer applications developed to solve complex problems in a particular domain, at the level of extra-ordinary human intelligence and expertise.


EXPERT SYSTEM
Sample Input
Advice
Non-expert User
User Interface
Rules Engine
Knowledge Base
Knowledge from an Expert

Characteristics of Expert System

- **High Performance:** The expert system provides high performance for solving any type of complex problem of a specific domain with high efficiency and accuracy.
- **Understandable:** It responds in a way that can be easily understandable by the user. It can take input in human language and provides the output in the same way.
- **Reliable:** It is much reliable for generating an efficient and accurate output.
- **Highly responsive:** ES provides the result for any complex query within a very short period of time.

Components of Expert System

- An expert system mainly consists of three components:
  - **User Interface**
  - **Inference Engine-Rules(Fwd chaining&Bckward chaining)**
  - **Knowledge Base**

Benefits of Expert Systems

- **Availability** − They are easily available due to mass production of software.
- **Less Production Cost** − Production cost is reasonable. This makes them affordable.
- **Speed** − They offer great speed. They reduce the amount of work an individual puts in.
- **Less Error Rate** − Error rate is low as compared to human errors.
- **Reducing Risk** − They can work in the environment dangerous to humans.
- **Steady response** − They work steadily without getting motional, tensed or fatigued.

Expert Systems Limitations

- No technology can offer easy and complete solution.
- Large systems are costly, require significant development time, and computer resources.
- ESs have their limitations which include −
  - Limitations of the technology
  - Difficult knowledge acquisition
  - ES are difficult to maintain
  - High development costs