RAMAPURAM SRM INSTITUTE OF SCIENCE & TECHNOLOGY (Deemed to be University u/s 3 of UGC Act, 1956)

UNIT – 4
18CSC305J / ARTIFICIAL INTELLIGENCE

What is Planning?

- The task of coming up with a sequence of actions that will achieve a goal is called planning.
- Planning Environments
  1. Classical Planning Environments
     - Fully observable, deterministic, finite, static and discrete.
  2. Non classical Planning Environments
     - Partially observable, stochastic with different algorithms and agent designs.

Planning (Contd..)

- Difficulty in problem-solving agent
  - Performs irrelevant actions
    - Have to explore all the states
    - Ex: buy book with 10-digit ISBN⍰ 10^{10} actions
  - Finding heuristics function
    - Problem-solving agent lacks autonomy because it depends on human to supply heuristic function for each new problem.
  - No problem decomposition
- **All these problems are overcome by planning agent by representing the goal as conjunction of subgoals.**

PLANNING PROBLEM

The planning problem is actually the question how to go to next state or the goal state from the current state. It involves two things 'how' and 'when'.

- The planning problem is defined with:

1. Domain model
2. Initial state
3. Goal state (next state)

The domain model defines the actions along with the objects. It is necessary to specify the operators too that actually describe the action. Along with this, information about actions and state constraints while acting should also be given. This entirely formulates the domain model.

The initial state is the state where any action is yet to take place (the stage when the exam schedule is put up!).

The final state or the goal state is the state which the plan is intended to achieve.

Planning Problem

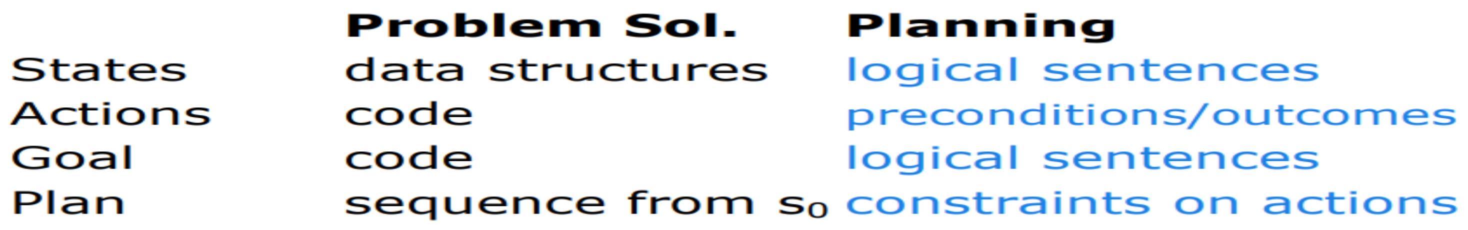
- Find **sequence of actions** - achieves a given **goal** from a given **initial world state**.
  - a set of operator descriptions
  - an initial state description, and
  - a goal state description or predicate,
- compute a plan
  - a sequence of operator instances,
  - executing them in the initial state will change the world to a state satisfying the goal-state description.
- Goals - specified as a conjunction of subgoals to be achieved

Simple Planning Agent

- An agent interacts with real world via perception and actions
- Perception - sense the world and assess the situation
- Actions - what the agent does in the domain.
- Planning involves reasoning about actions that the agent intends to carry out
- This reasoning involves the representation of the world that the agent has - representation of its actions.
- Hard constraints - objectives *have to* be achieved completely for success
- The objectives - soft constraints, or *preferences*, to be achieved as much as possible

Planning vs. Problem solving

- Planning agent is very similar to problem solving agent
  - Constructs plans to achieve goals, then executes them
- Planning is more powerful because of the representations and methods used
- Search - proceeds through *plan space* rather than *state space*
- Sub-goals - planned independently, it reduce the complexity of the planning problem



	Problem Sol.	**Planning**
States	data structures	logical sentences
Actions	code	preconditions/outcomes
Goal	code	logical sentences
Plan	sequence from s_0	constraints on actions

Planning Agents

- **problem-solving agents** are able to plan ahead - to consider the consequences of *sequences* of actions - before acting.
- **knowledge- based agents** can select actions based on explicit, logical representations of the current state and the effects of actions.
  - This allows the agent to succeed in complex, inaccessible environments that are too difficult for a problem-solving agent
- **Problem Solving Agents + Knowledge-based Agents = Planning Agents**

Simple Planning Agent

- A simple planning agent is very similar to problem-solving agents in that it constructs plans that achieve its goals, and then executes them.
- The limitations of the problem- solving approach motivates the design of planning systems.

To solve a planning problem using a state-space search approach we would let the:

- initial state = initial situation
- goal-test predicate = goal state description
- successor function computed from the set of operators
- once a goal is found, solution plan is the sequence of operators in the path from the start node to the goal node

Simple Planning Agent

- Planning can be viewed as a type of problem solving in which the agent uses beliefs about actions and their consequences to search for a solution over the more abstract space of plans, rather than over the space of situations.

Algorithm of a simple planning agent:

1. Generate a goal to achieve
2. Construct a plan to achieve goal from current state
3. Execute plan until finished
4. Begin again with new goal

- The agent first generates a goal to achieve, and then constructs a plan to achieve it from the current state. Once it has a plan, it keeps executing it until the plan is finished, then begins again with a new goal.

Key Ideas Behind Planning

- Planning emphasizes what is in operator and goal representations.

There are three key ideas behind planning:

- to "open up" the representations of state, goals, and operators so that a reasoner can more intelligently select actions when they are needed
- the planner is free to add actions to the plan wherever they are needed, rather than in an incremental sequence starting at the initial state
- most parts of the world are independent of most other parts which makes it feasible to take a conjunctive goal and solve it with a divide-and-conquer strategy

Planning Languages

- Languages must represent..
  - States
  - Goals
  - Actions
- Languages must be
  - Expressive for ease of representation
  - Flexible for manipulation by algorithms

State Representation

- A state is represented with a conjunction of positive literals
- Using
  - Logical Propositions: *Poor ∧ Unknown*
  - FOL literals: *At(Plane1,OMA) ∧ At(Plan2,JFK)*
- FOL literals must be ground & function-free
  - Not allowed: *At(x,y)* or *At(Father(Fred),Sydney)*
- Closed World Assumption
  - What is not stated are assumed false

Goal Representation

- Goal is a <u>partially</u> specified state
- A proposition satisfies a goal if it contains all the atoms of the goal and possibly others..
  - Example: Rich ∧ Famous ∧ Miserable satisfies the goal Rich ∧ Famous

Action Representation

- Action Schema
  - Action name
  - Preconditions
  - Effects
- Example

  Action(Fly(p,from,to),

  PRECOND: At(p,from) ∧ Plane(p) ∧ Airport(from) ∧ Airport(to)

  EFFECT: ¬At(p,from) ∧ At(p,to))
- Sometimes, Effects are split into ADD list and DELETE list

At(WHI,LNK),Plane(WHI),
Airport(LNK), Airport(OHA)

$\boxed{\text{Fly(WHI,LNK,OHA)}}$

At(WHI,OHA), ¬ At(WHI,LNK)

Languages for Planning Problems

- STRIPS
  - Stanford Research Institute Problem Solver
  - Historically important
- ADL
  - Action Description Languages
- PDDL
  - Planning Domain Definition Language

Planning languages

- A language is the one that should be expressive.
- Three Languages
  - I. Stanford Research Institute Problem Solver (STRIPS)
  - 2. Action Description Language (ADL)
  - 3. Planning Domain Description Language (PDDL)

1) Stanford Research Institute Problem Solver (STRIPS) :

Makes use of the first order predicates.

STRIPS allows function-free literals.

Example of a robot: The example involves a robot, a cup tea, guest and two rooms. We want the robot to get the tea and give it to the guest.

The planning with STRIPS is done as follows:

Let us begin with the STRIPS representation for this example.

Initial and final states are depicted in figure.

Example of a robot

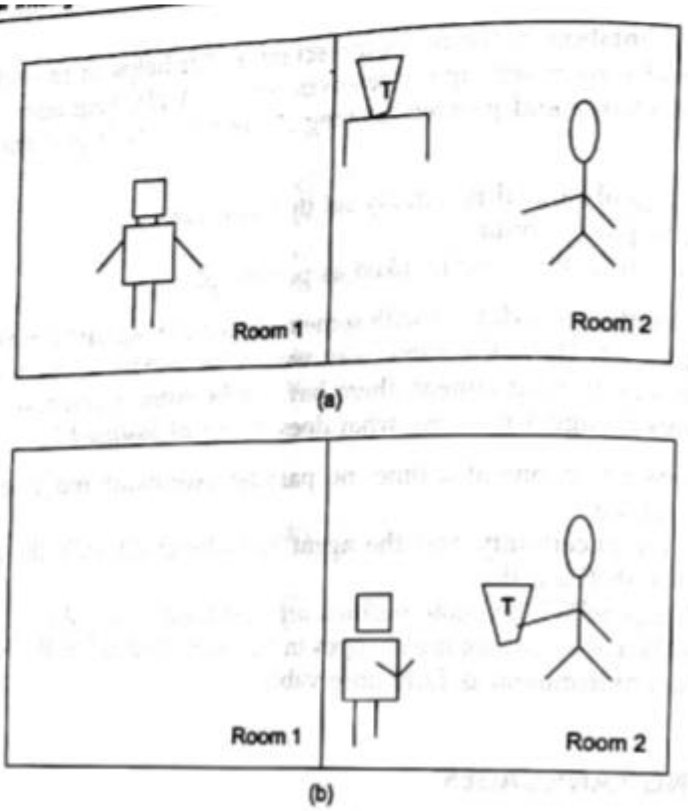


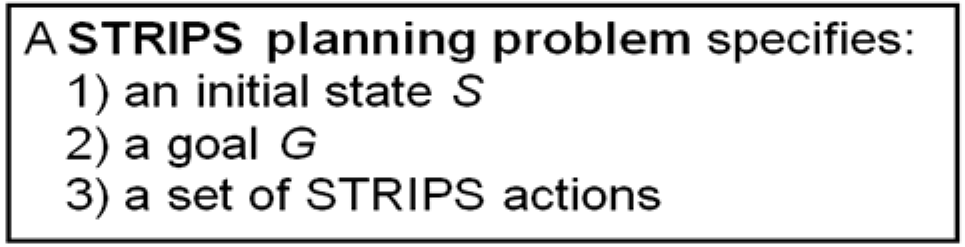
Figure 9.2 (a) Initial stage, (b) Goal state.

Block World

- There are ‘N’ number of Blocks resting on table with specified sequence.
- Goal is to arrange in desired sequence.
- Available moves
  - Put block on table
  - Put a block on another block top
- State is represented using sequence of blocks in current pos.

STRIPS

- STRIPS stands for "STanford Research Institute Problem Solver," was the planner used in Shakey, one of the first robots built using AI technology ,which is an action-centric representation ,for each action , specifies the effect of an action.



A **STRIPS planning problem** specifies:
1) an initial state *S*
2) a goal *G*
3) a set of STRIPS actions

The **STRIPS representation** for an action consists of

- the **precondition**, which is a set of assignments of values to features that must be true for the action to occur, and
- the **effect**, which is a set of resulting assignments of values to those primitive features that change as the result of the action.

Block World Problem

- Action List:

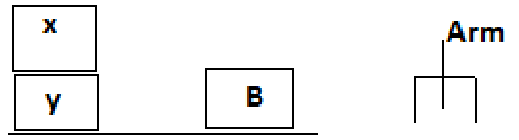
SNo	Action	Precondition	Effect

1	Pickup(x)	Arm Empty
		On(x, Table)
		Clear(x)
			Holding(x)

2	Putdown (x)	Holding(x)	Arm Empty
			On(x, Table)
			Clear(x)

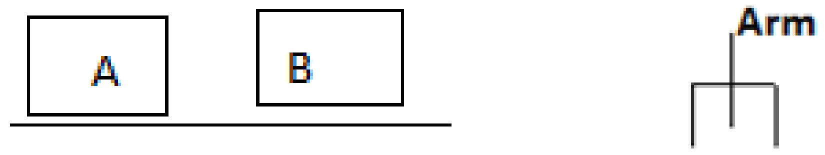
3	Stack(x,y)	Holding(x)
		Clear(y)
			On(x,y)
			Clear(x)
			Arm Empty

4	Unstack(x,y)	On(x,y)
		Clear(x)
		Arm Empty
			Holding(x)
			Clear(y)

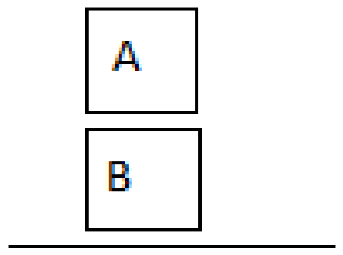


Block World Problem

- Start State:



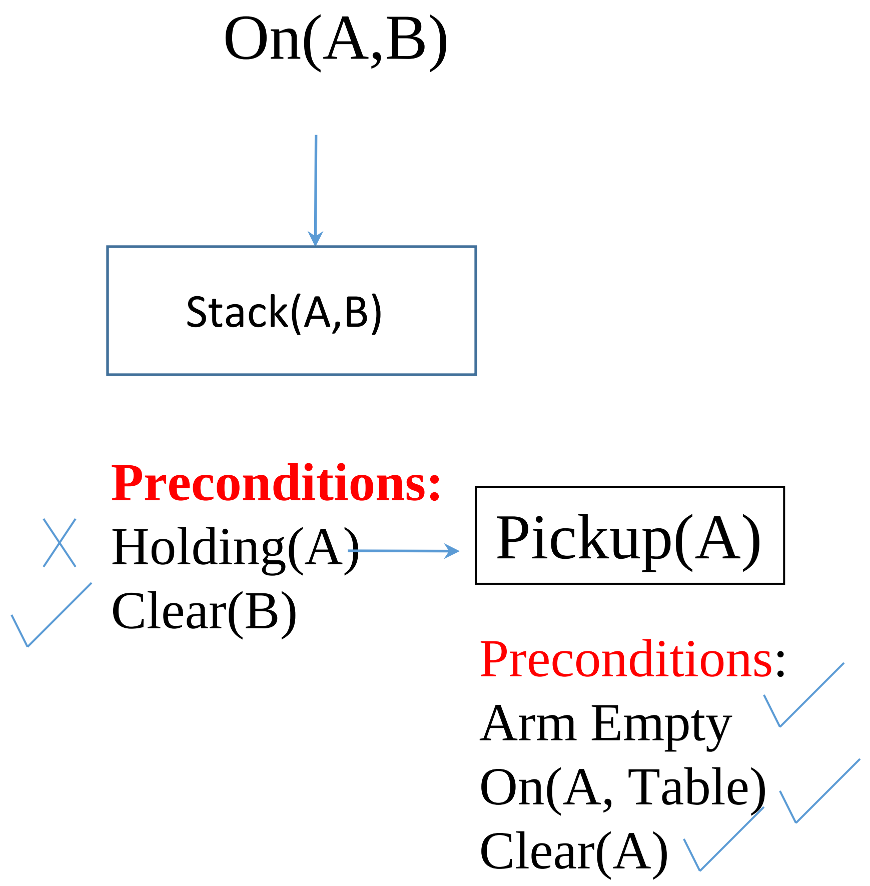
- Goal State:



Block World Problem

- Start State:
  - On(A, table)
  - On(B, table)
- Goal State:
  - On(A,B)

Solution:



Means - Ends Analysis

- Search strategies either reason forward of backward
- Mixed strategy - solve the major parts of problem first and solve the smaller problems that arise when combining them together.
- Such a technique is called "Means - Ends Analysis".
- Means-Ends Analysis is problem-solving techniques used in Artificial intelligence for limiting search in AI programs.
- It is a mixture of Backward and forward search technique.
- The means -ends analysis process centers around finding the difference between current state and goal state.

Means - Ends Analysis

How means-ends analysis Works:

- The means-ends analysis process can be applied recursively for a problem. It is a strategy to control search in problem-solving.

Following are the main Steps which describes the working of MEA technique for solving a problem.

1. First, evaluate the difference between Initial State and final State.
2. Select the various operators which can be applied for each difference.
3. Apply the operator at each difference, which reduces the difference between the current state and goal state.

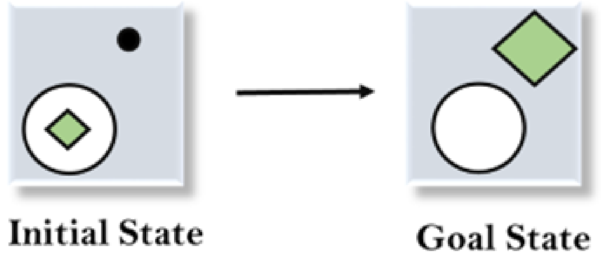
MEA Algorithm

- **Step 1:** Compare CURRENT to GOAL, if there are no differences between both then return Success and Exit.
- **Step 2:** Else, select the most significant difference and reduce it by doing the following steps until the success or failure occurs.
  a. Select a new operator O which is applicable for the current difference, and if there is no such operator, then signal failure.
  b. Attempt to apply operator O to CURRENT. Make a description of two states.
     i) O-Start, a state in which O?s preconditions are satisfied.
     ii) O-Result, the state that would result if O were applied In O-start.
  c. If
     (First-Part <------ MEA (CURRENT, O-START)
     And
     (LAST-Part <----- MEA (O-Result, GOAL), are successful, then signal Success and return the result of combining FIRST-PART, O, and LAST-PART.

Example of Mean-Ends Analysis:

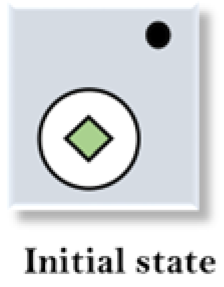
- Apply MEA to get the goal state.

- Solution:
- To solve the above problem, first find the differences between initial states and goal states, and for each diffe
- **Move**
- **Delete**
- **Expand**

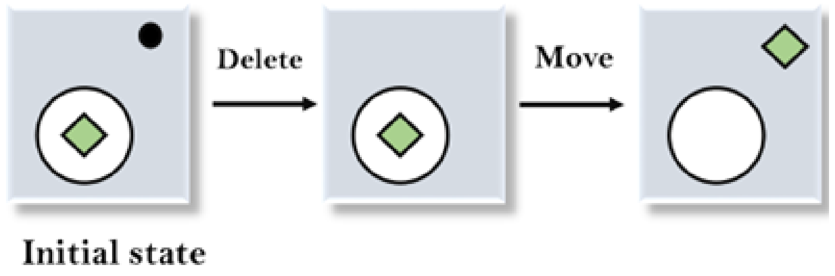


Example of Mean-Ends Analysis:

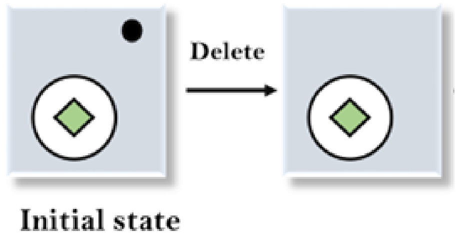
Step 1: Evaluate Initial State



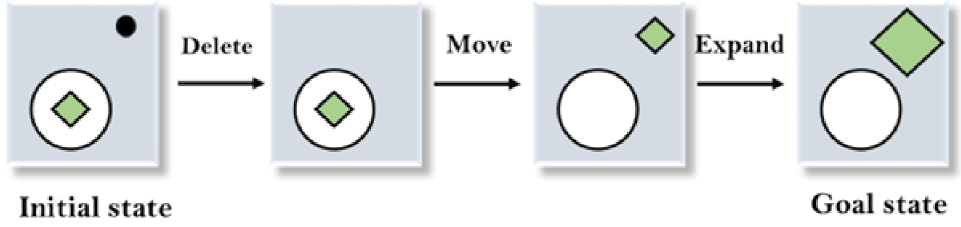
Step 3: Apply Move Operator



Step 2: Apply Delete Operator

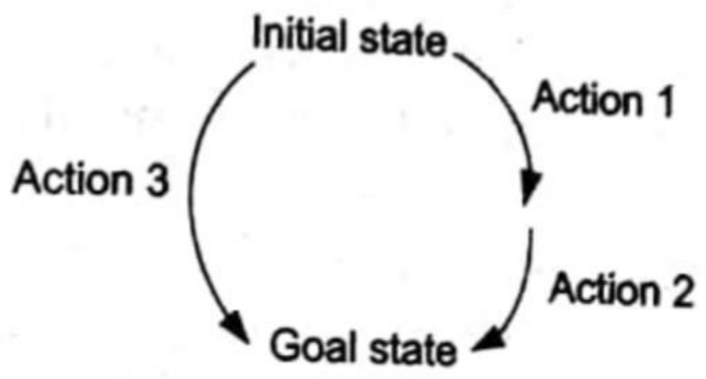


Step 1: Apply Expand Operator



NON-LINEAR PLANNING

- A plan that consists of sub-problems, which are solved simultaneously is said non-linear plan.
- In case of the goal stack planning there are some problems. To achieve any goal, it could have an impact on the one that has been achieved.



Non-linear planning.

NON-LINEAR PLANNING (Contd..)

There is a concept of constraint posting that comes with non-linear planning. The constraint posting states that the plan can be built by

- 1. Addition of operators or suggesting operators
- 2. Ordering them
- 3. Binding the variables to the operators

Conditional Planning

- Conditional planning has to work regardless of the outcome of an action.
- The outcome of actions cannot be determined so the environment is said to be nondeterministic.
- It’s a way to deal with uncertainty by checking what is actually happening in the environment at predetermined points in the plan. (Conditional Steps)

Example:

Check whether SFO airport (San Francisco International Airport) is operational. If so, fly there; otherwise, fly to some other place.

Conditional Planning

- Three kind of Environments

Fully Observable

- The agent always knows the current state

Partially Observable

- The agent knows only a certain amount about the actual state. (much more common in real world)

Unknown

- The agent knows nothing about the current state

Conditional Planning in Fully Observable Environments

- Agent used conditional steps to check the state of the environment to decide what to do next.
- Plan information stores in a library Ex: Action(Left) Clean v Right

Syntax: If then plan_A else plan_B

Reactive Planning

- Reactive planning is planning under uncertainty.
- Makes use of the if-then rules.
- The reactive planners are based on the concept that they should be able to handle an unknown situation too. So, the reaction rules are used that help them in doing so.
- A rule selection is based on the priority and a holding condition that maximises the priority.
- The rule which is at present in execution is said to be active whereas the and the ones with holding priority (we can call them possible competitors) are pre-active others are inactive.
- A B-tree structure is used in reactive planning, where the things are algorithm selects the rule. Sometimes, no rule can be selected. In such a case, dependent on the algorithm implementation for rule selection.

LEARNING

learning consists of various activities like understanding, memorisation, knowledge acquisition and inference.

Learning is a continuous process

Learning from observation is required to construct meaningful classification of observed objects and situations.

MACHINE LEARNING

ACTIVE LEARNING MECHANISM

Perceptual learning

– learning of new objects , categories , relations.

Episodic Learning

– Learning of events like What , where and when

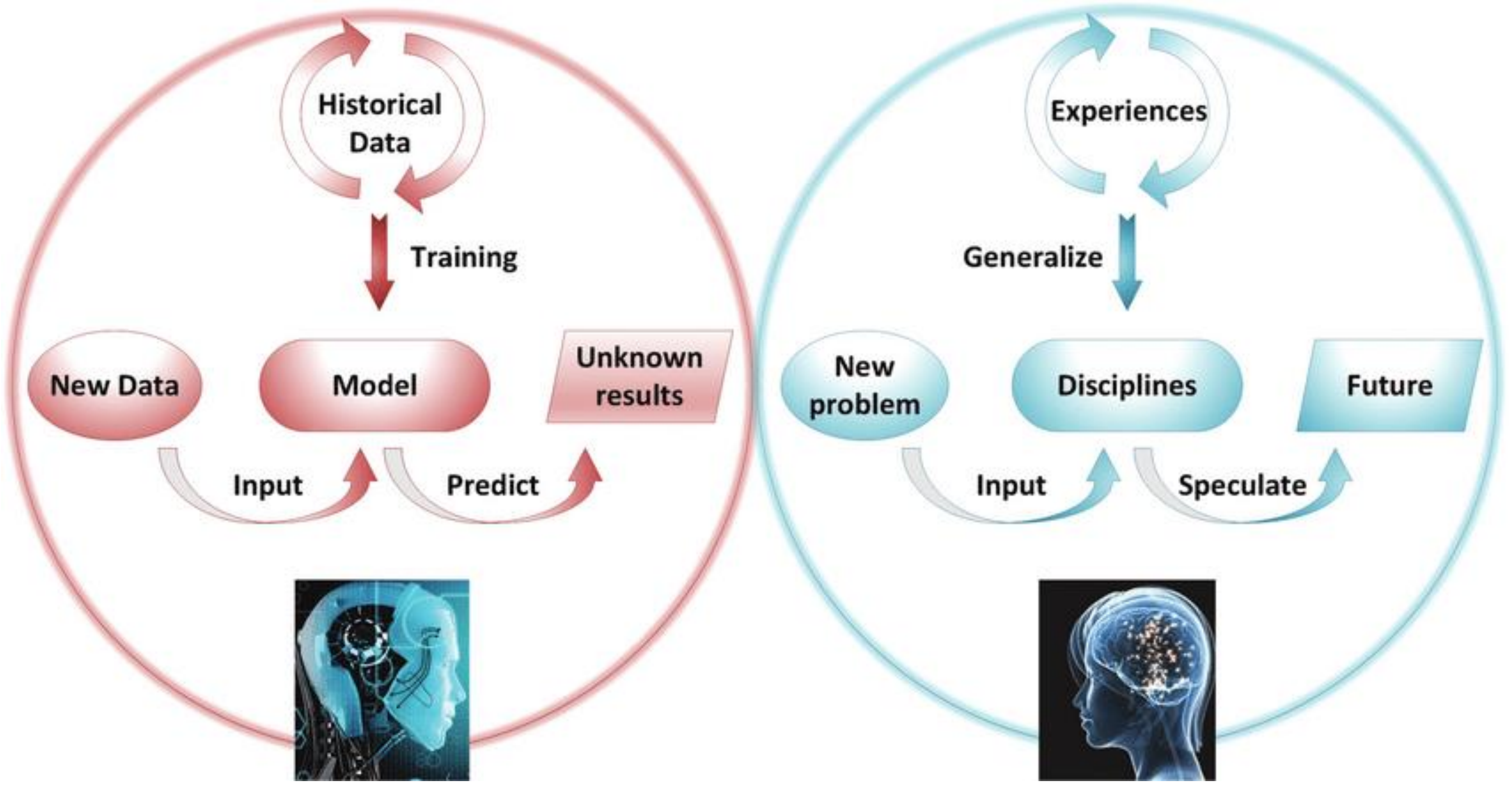
Procedural learning

– actions and their sequences to accomplish a task.

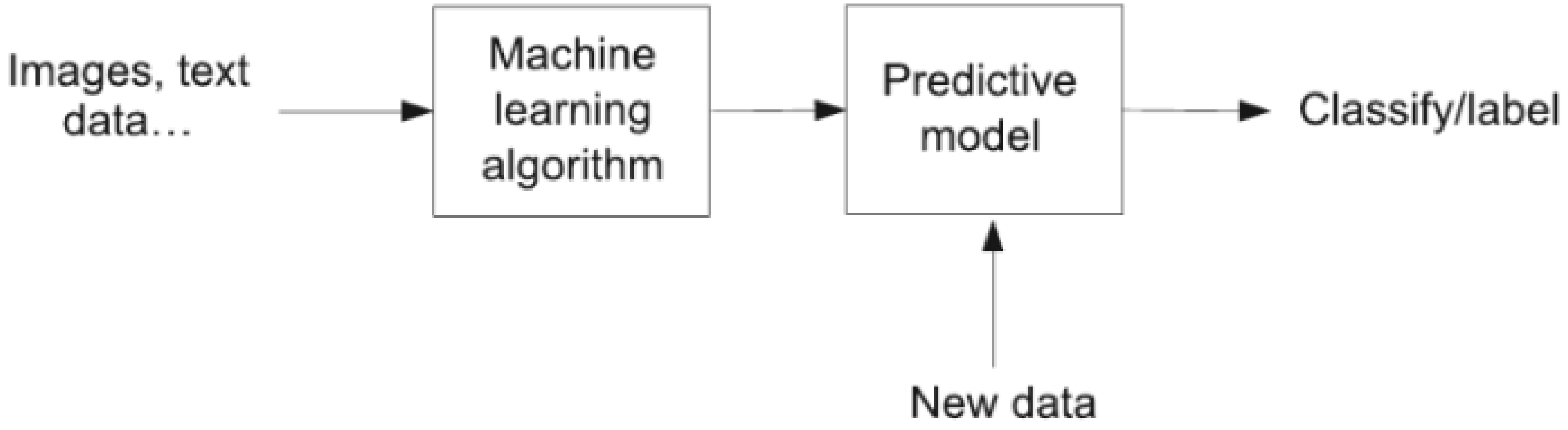
Machine Learning

Machine learning is building and exploring of methods for programming computer to make them Learn.

Machine learning Vs Human Learning



Machine Learning approach



Scope of machine learning

- Automatic Translation
- Traffic Prediction
- Web Search and Recommendation Engines
- Virtual Personal Assistants
- Online Fraud Detection
- Medical Diagnosis
- Text & Speech Recognition
- Email spam filtering
- Image Recognition

GOALS OF MACHINE LEARNING

- To produce learning algorithms with practical value.
- Development and enhancement of computer algorithms and models to meet the decision making requirements in practical scenarios.
- To facilitate in building intelligent systems (IS) that can be used in solving real time problems.

Challenges of Machine Learning

- Avaliability of limited learning data and unknown perspectives.
- Aquiring Accurate , compact and precise knowledge building.
- Require large working memory to store data.
- Focusing Too Much on Algorithms and Theories
- Monitoring and maintenance

Learning Concepts , methods and models

Computational structure used in Machine learning:

1. Functions.
2. Logic programs and rule sets
3. Finite state machines
4. Grammars
5. Problem solving system

Traditional paradigm

- Rote learning
- Learning from observations
- Supervised learning
- unsupervised learning
- semi-supervised learning
- ensemble learning
- Discovery based learning
- Learning by problem solving

Rote learning

- Rote learning is rudimentary form of learning, which focuses on memorization.
- Storing or memorizing the results improves the performance of a system
- saves a significant amount of time.
- Example selective paging catching

Learning from observations

Types of learning

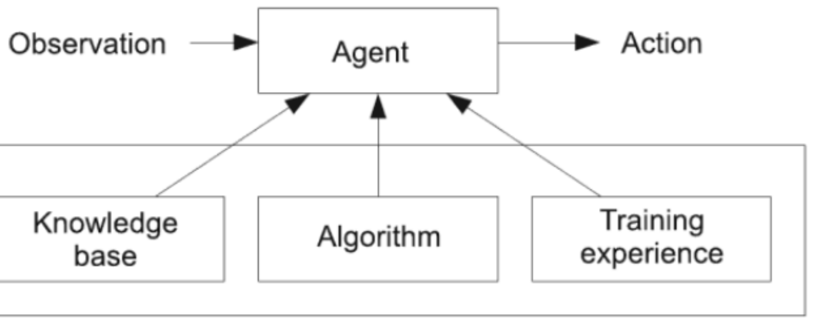
- learning from agents
- inductive learning
- decision tree learning

Learning from agents

•An agent is defined the computational entity which is capable of receiving the environment and can act based on the situation.

•The agent is composed of learning element performance element and a curiosity element.

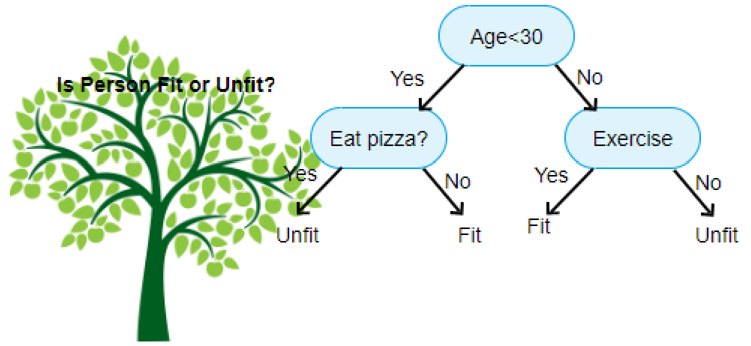
•Based on the coordination between these elements the outcome of the agents behaviour is measured.



Inductive Learning

- Inductive learning involves learning generalized rules from specific examples (can think of this as the “inverse” of deduction)
- Main task: given a set of examples, each classified as positive or negative produce a concept description that matches exactly the positive examples.
- The examples are coded in some representation language, e.g. they are coded by a finite set of real-valued features.
- The concept description is in a certain language that is presumably a superset of the language of possible example encodings.
- A “correct” concept description is one that classifies correctly ALL possible examples, not just those given in the training set.

Decision Tree Learning



•The learned function is represented by a decision tree.

•In terms of programming it is also represented as if then rules.

•Decision tree depicts the simple learning from the observation, method.

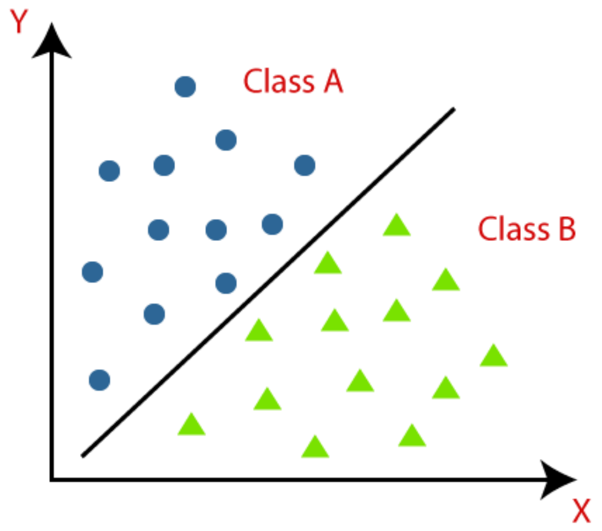
•Based on the observation, at every node, decision is taken.

Supervised learning

•Based on the labelled data set.

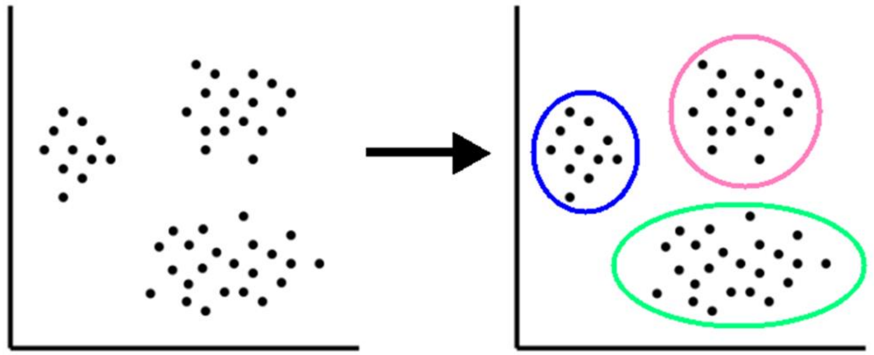
•supervised learning is the learning algorithm that is provided with the set of training data and the algorithm further induces the classifier to classify the unseen or new data.

• A line (hyperplane) which is generated after learning separating two classes class A and class the in two parts the classifier and the decision-making engine minimize the false positives and false negative.



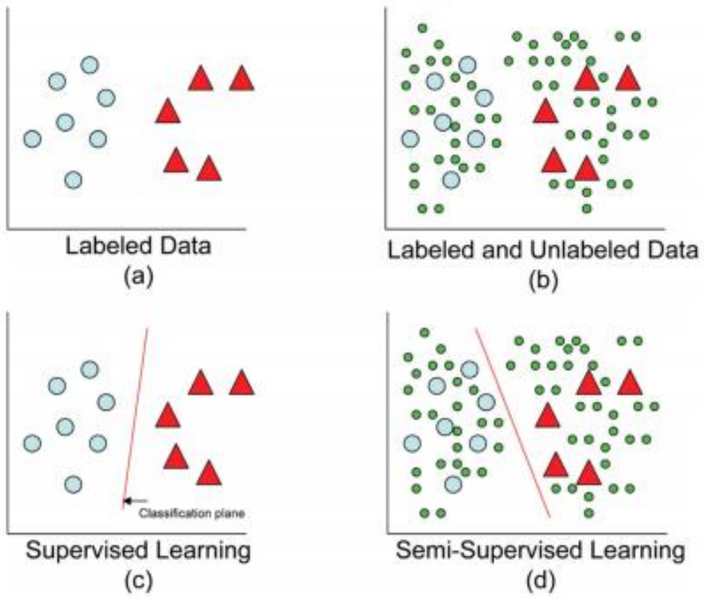
Unsupervised Learning

- Use unlabeled dataset.
- Learning is more based on similarities and differences which are visible. These differences and similarities are mathematically represented in unsupervised learning
- Grouping and categorisation of the objects is based on the understanding of similarities and visualisation of their relations
- Unsupervised learning performs hierarchical clustering.



Semi - Supervised Learning

- Semi supervised learning is developed to cope up with the issues of learning in supervised or unsupervised mode in isolation.
- Semi-supervised learning tries to learn from the labelled as well as unlabeled data.
- Let U be a set of unlabeled data and L be a set of labelled data.
- As the learning process the learning approach identifies the unlabeled data U with reference to a labelled data L and keeps on labelling the unlabeled data.
- This method is also called as self training in semi supervised learning.



Discovery based learning

On the basis of past experience and knowledge tries to discover the outcomes

Learning by Problem Solving

Various parameters related to solution and problem are considered. These parameters are used and effectively desirability of a particular outcome or decision is determined

Learning Methods

- Artificial neural network based learning-Back propagation
- Support vector machines
- Reinforcement learning
- Adaptive learning
- Multi_agent based learning
- Ensemble learning
- Learning for decision making
- Distributed learning
- Speedup learning

Artificial neural network based learning

ANN is a computational model that performs simulation of the human biological neurons.

The simulation is concerned with the functioning of neurons.

Artificial neurons have different inputs which are weighted as per the strength of signal and then computed by mathematical function which determines the activation of the neurons.

ANNs combined artificial neurons in order to process information the hidden layer is the one that Learns to recode for the input .

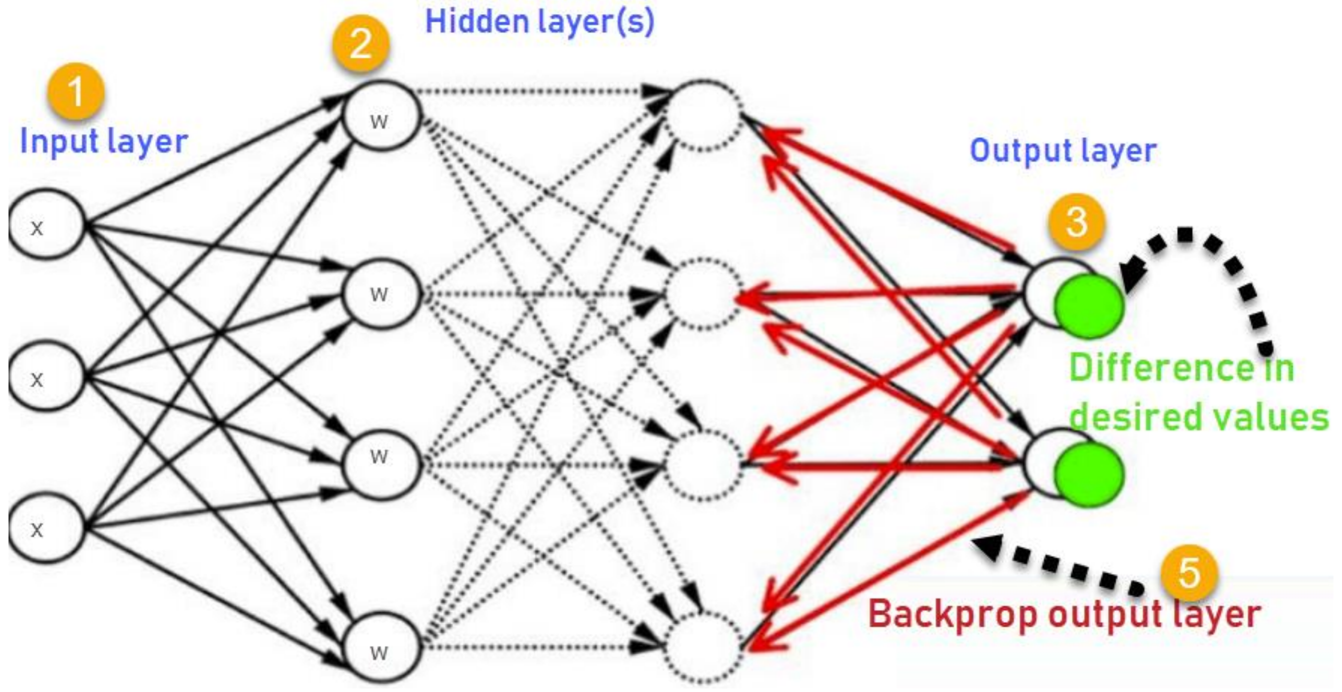
Back- propagation Algorithm

Back-propagation is the essence of neural net training.

It is the method of fine-tuning the weights of a neural net based on the error rate obtained in the previous epoch (i.e., iteration).

Proper tuning of the weights allows you to reduce error rates and to make the model reliable by increasing its generalization.

Backpropagation Algorithm



Backpropagation Algorithm

1. Inputs X, arrive through the preconnected path
2. Input is modeled using real weights W. The weights are usually randomly selected.
3. Calculate the output for every neuron from the input layer, to the hidden layers, to the output layer.
4. Calculate the error in the outputs.

```
ErrorB= Actual Output – Desired Output
```

5. Travel back from the output layer to the hidden layer to adjust the weights such that the error is decreased.

Keep repeating the process until the desired output is achieved.

Backpropagation Algorithm

Forward pass:

computes 'functional signal', feed forward propagation of input pattern signals through network

Backward pass phase:

computes 'error signal', propagates the error backwards through network starting at output units (where the error is the difference between actual and desired output values)

Support Vector Machines (SVM)

 Supervised learning methods for classification and regression

relatively new class of successful learning methods -

they can represent non-linear functions and they have an efficient training algorithm

 derived from statistical learning theory by Vapnik and Chervonenkis (COLT-92)

 SVM got into mainstream because of their exceptional performance in Handwritten Digit Recognition

- 1.1% error rate which was comparable to a very carefully constructed (and complex) ANN

Motivation for Support Vector Machines

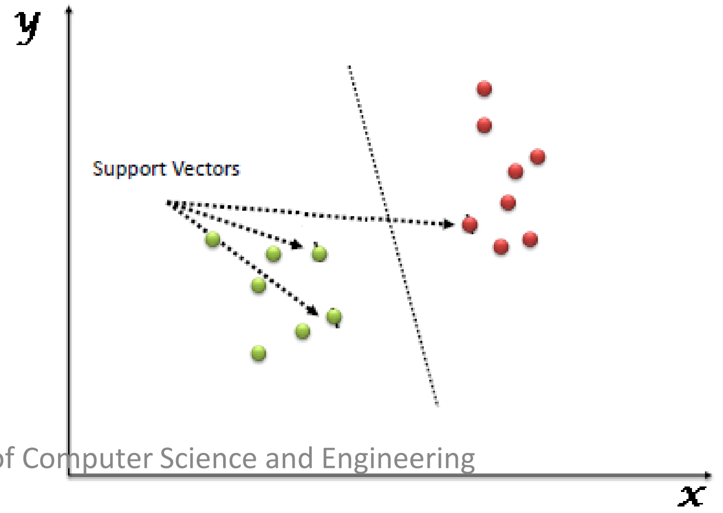
- The problem to be solved is one of the **supervised binary classification**. That is, we wish to categorize new unseen objects into two separate groups based on their properties and a set of known examples, which are already categorized.
- A good example of such a system is classifying a set of new *documents* into positive or negative sentiment groups, based on other documents which have already been classified as positive or negative.
- Similarly, we could classify new emails into spam or non-spam, based on a large corpus of documents that have already been marked as spam or non-spam by humans. SVMs are highly applicable to such situations.

Motivation for Support Vector Machines

- A Support Vector Machine models the situation by creating a *feature space*, which is a finite-dimensional vector space, each dimension of which represents a "feature" of a particular object. In the context of spam or document classification, each "feature" is the prevalence or importance of a particular word.
- The **goal of the SVM** is to train a model that assigns new unseen objects into a particular category.
- It achieves this by creating a linear partition of the feature space into two categories.
- Based on the features in the new unseen objects (e.g. documents/emails), it places an object "above" or "below" the separation plane, leading to a categorization (e.g. spam or non-spam). This makes it an example of a non-probabilistic linear classifier. It is non-probabilistic, because the features in the new objects fully determine its location in feature space and there is no stochastic element involved.

OBJECTIVES

- **Support vector machines (SVM)** are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.
- It is a **machine learning** approach.
- They analyze the large amount of data to identify patterns from them.
- SVMs are based on the idea of finding a hyperplane that best divides a dataset into two classes, as shown in the image below.

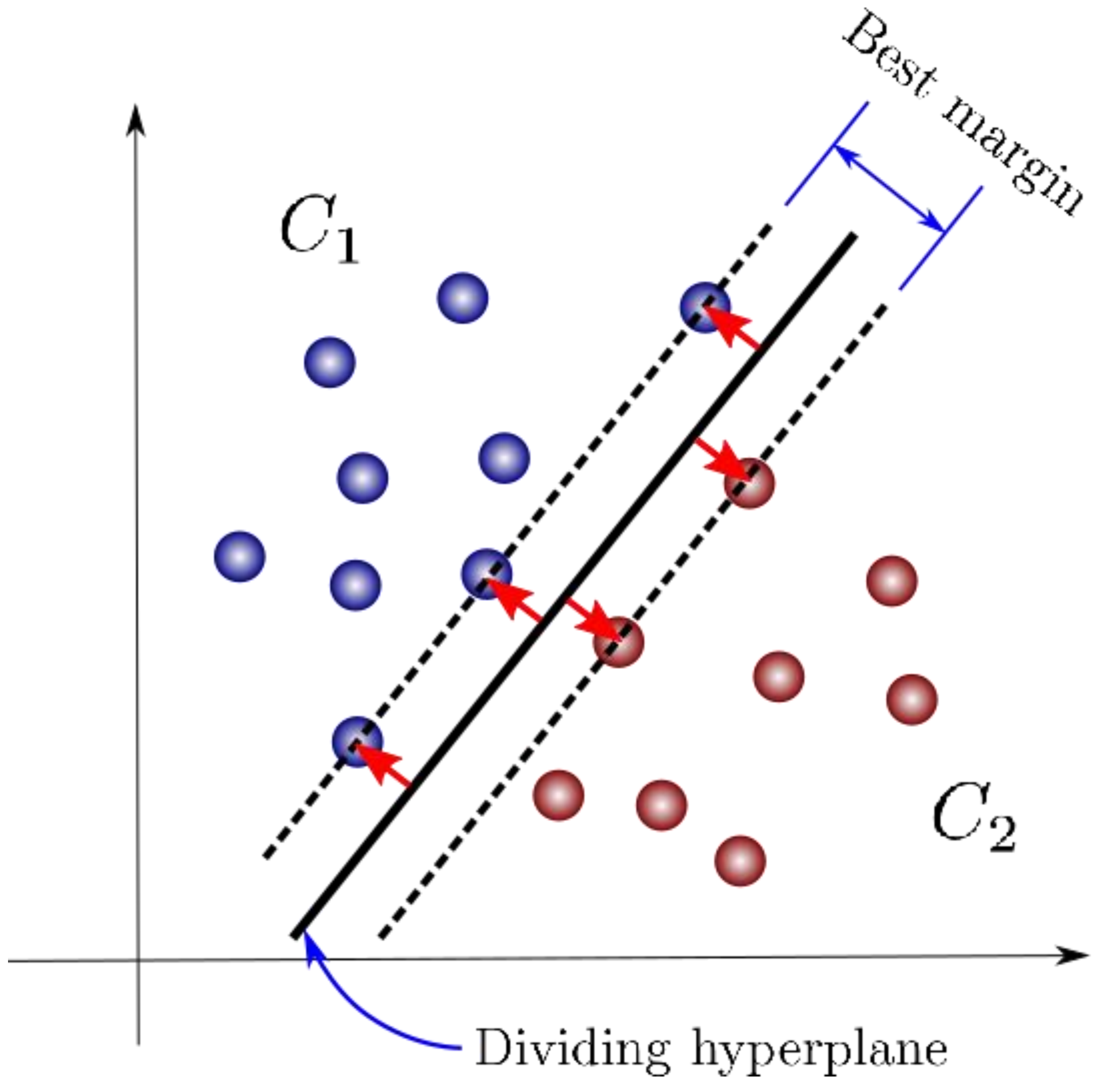


Support Vectors

- Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).
- Support vectors are the data points that lie closest to the decision surface (or hyperplane)
- They are the data points most difficult to classify
- They have direct bearing on the optimum location of the decision surface
- We can show that the optimal hyperplane stems from the function class with the lowest “capacity” (VC dimension).
- Support vectors are the data points nearest to the hyperplane, the points of a data set that, if removed, would alter the position of the dividing hyperplane. Because of this, they can be considered the critical elements of a data set.

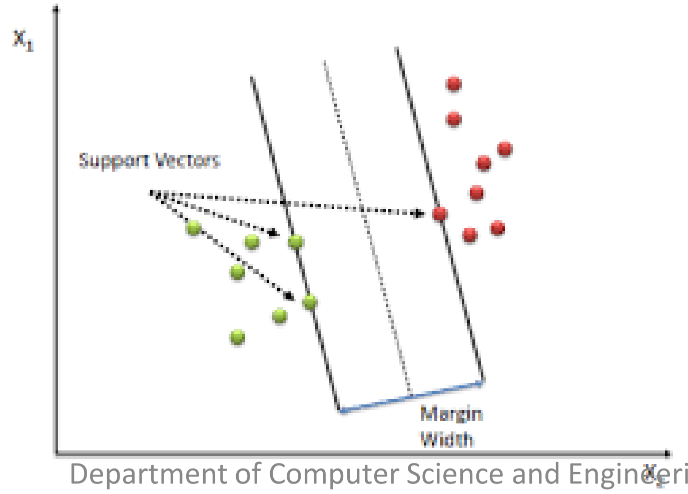
What is a hyperplane?

- As a simple example, for a classification task with only two features, you can think of a hyperplane as a line that linearly separates and classifies a set of data.
- Intuitively, the further from the hyperplane our data points lie, the more confident we are that they have been correctly classified. We therefore want our data points to be as far away from the hyperplane as possible, while still being on the correct side of it.
- So when new testing data are added, whatever side of the hyperplane it lands will decide the class that we assign to it.


Best margin
C_1
C_2
Dividing hyperplane

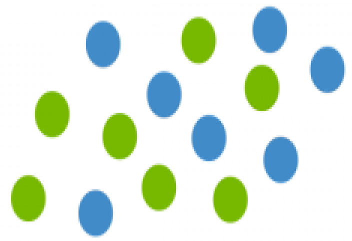
How do we find the right hyperplane?

- How do we best segregate the two classes within the data?
- The distance between the hyperplane and the nearest data point from either set is known as the **margin**. The goal is to choose a hyperplane with the greatest possible margin between the hyperplane and any point within the training set, giving a greater chance of new data being classified correctly. **There will never be any data point inside the margin.**

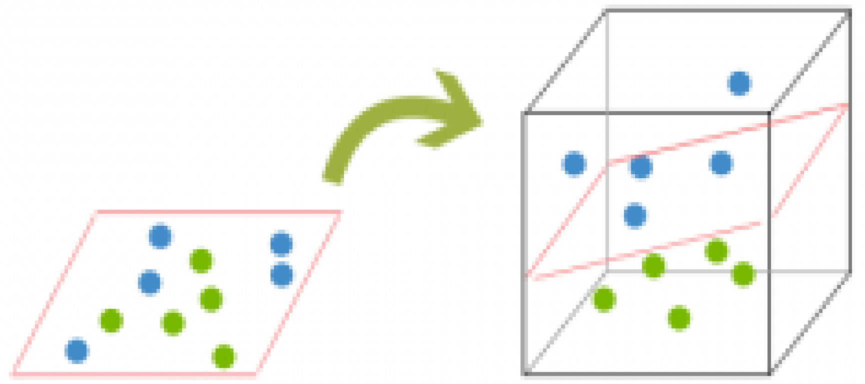


But what happens when there is no clear hyperplane?

- Data are rarely ever as clean as our simple example above. A dataset will often look more like the jumbled balls below which represent a linearly non separable dataset.



- In order to classify a dataset like the one above it’s necessary to move away from a 2d view of the data to a 3d view. Explaining this is easiest with another simplified example. Imagine that our two sets of colored balls above are sitting on a sheet and this sheet is lifted suddenly, launching the balls into the air. While the balls are up in the air, you use the sheet to separate them. This ‘lifting’ of the balls represents the mapping of data into a higher dimension. This is known as **kernelling**.



Because we are now in three dimensions, our hyperplane can no longer be a line. It must now be a plane as shown in the example above. The idea is that the data will continue to be mapped into higher and higher dimensions until a hyperplane can be formed to segregate it.

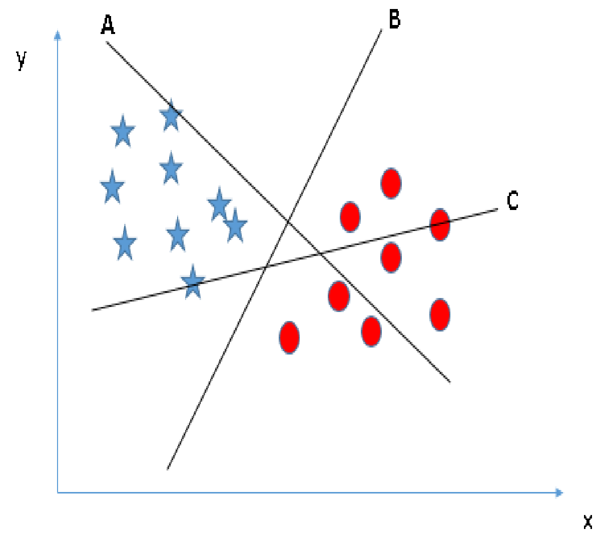
How does it work? How can we identify the right hyper-plane?

- You need to remember a thumb rule to identify the right hyper-plane:

“Select the hyper-plane which segregates the two classes better”.

Identify the right hyperplane (Scenario-1):

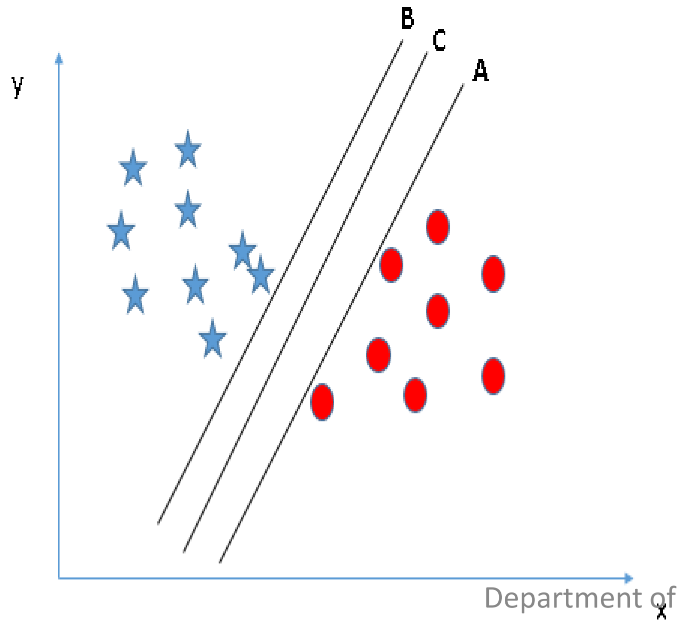
- Here, we have three hyperplanes (A, B and C). Now, identify the right hyperplane to classify star and circle.



- Hyperplane "B" has excellently performed this job.

Identify the right hyperplane (Scenario-2):

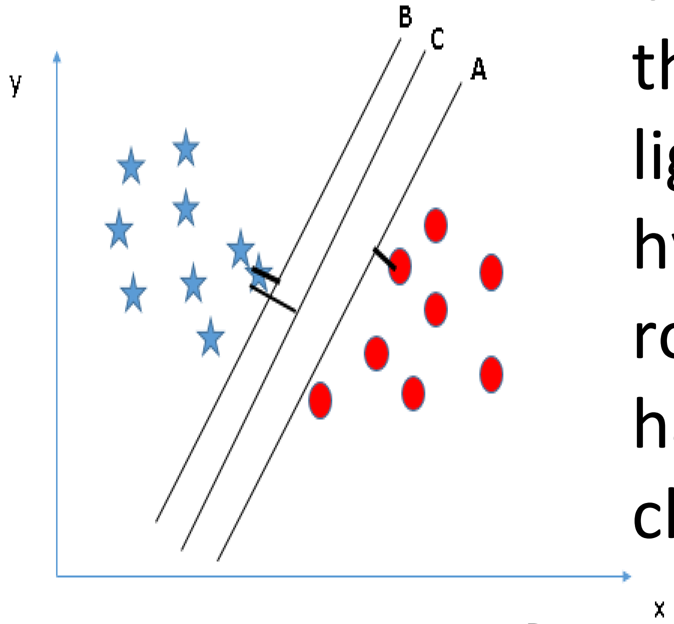
- Here, we have three hyperplanes (A, B and C) and all are segregating the classes well. Now, how can we identify the right hyperplane?



Here, maximizing the distances between nearest data point (either class) and hyperplane will help us to decide the right hyperplane.

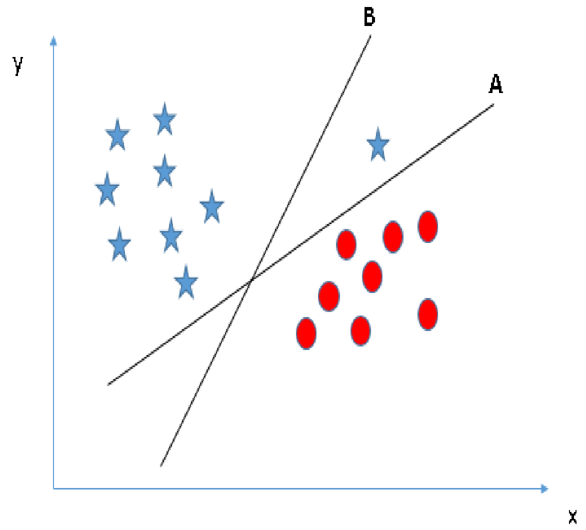
Scenario-2

This distance is called as **Margin**. Let’s look at the below snapshot:



We can see that the margin for hyperplane C is high as compared to both A and B. Hence, we name the right hyperplane as C. Another lightning reason for selecting the hyperplane with higher margin is robustness. If we select a hyperplane having low margin then there is high chance of missclassification.

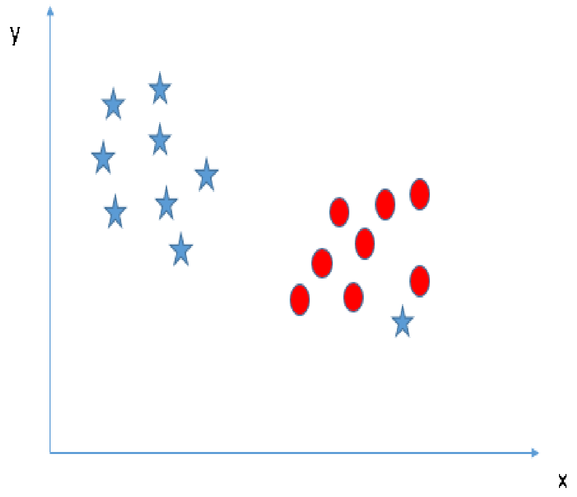
Identify the right hyperplane (Scenario-3)

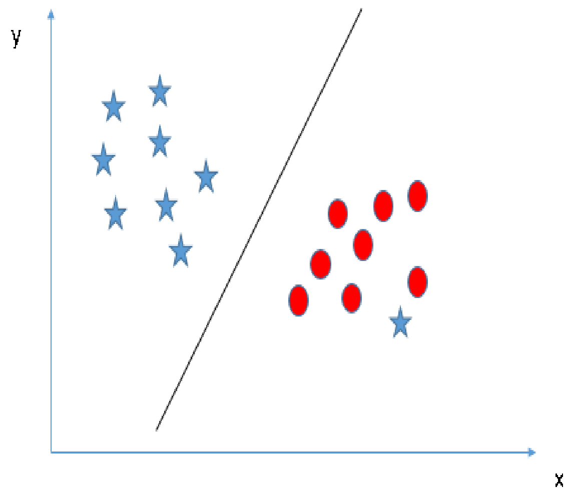


- Some of you may have selected the hyper-plane **B** as it has higher margin compared to **A.** But, here is the catch, SVM selects the hyperplane which classifies the classes accurately prior to maximizing margin. Here, hyperplane B has a classification error and A has classified all correctly. Therefore, the right hyperplane is **A.**

Can we classify two classes (Scenario-4)?

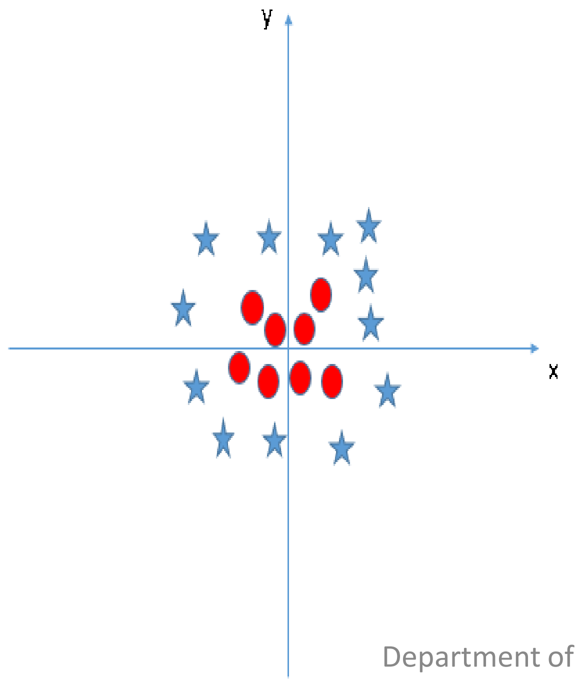
- We are unable to segregate the two classes using a straight line, as one of star lies in the territory of other (circle) class as an outlier.
- One star at other end is like an outlier for star class. SVM has a feature to ignore outliers and find the hyperplane that has maximum margin. Hence, we can say, SVM is robust to outliers.





Find the hyperplane to segregate to classes (Scenario-5)

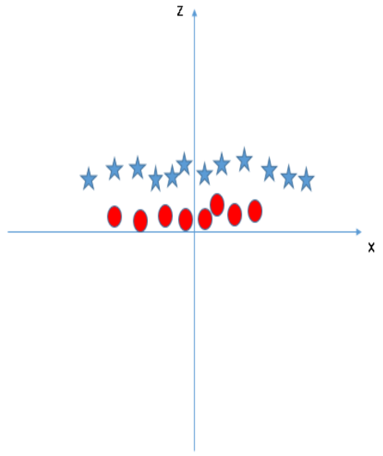
- In the scenario below, we can't have linear hyperplane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyperplane.



SVM can solve this problem. It solves this problem by introducing additional feature. Here, we will add a new feature $z=x^2+y^2$.

Scenario-5

- Now, let's plot the data points on axis x and z:



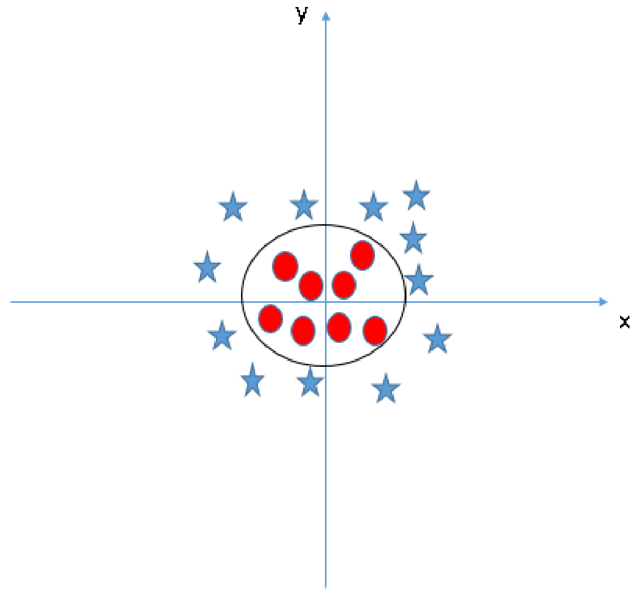
- In above plot, points to consider are:
  - All values for z would be positive always because z is the squared sum of both x and y
  - In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

Scenario-5

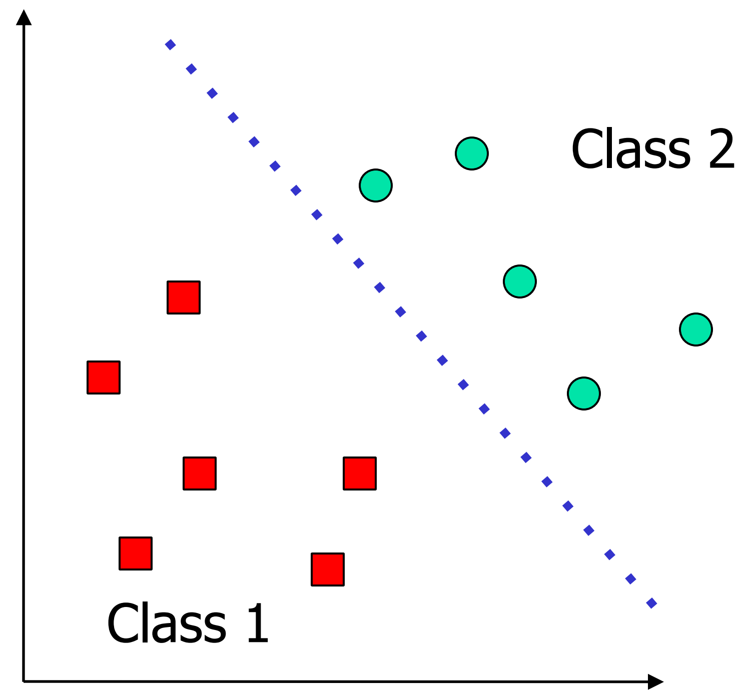
- In SVM, it is easy to have a linear hyperplane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyperplane. No, SVM has a technique called the **kernel trick**.
- These are functions which takes low dimensional input space and transform it to a higher dimensional space i.e. it converts not separable problem to separable problem, these functions are called **kernels**. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then find out the process to separate the data based on the labels or outputs you've defined.

Scenario-5

- When we look at the hyperplane in original input space it looks like a circle:

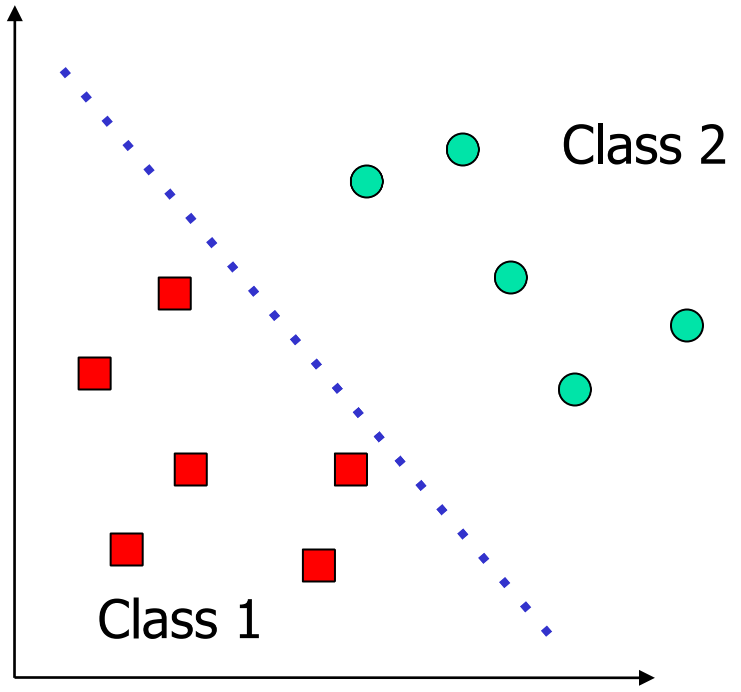


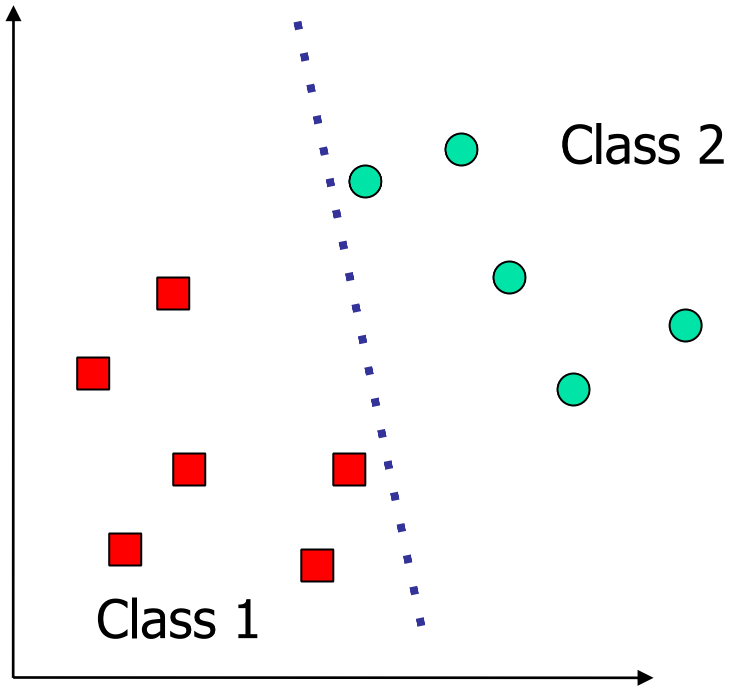
Two Class Problem: Linear Separable Case



- Many decision boundaries can separate these two classes
- Which one should we choose?

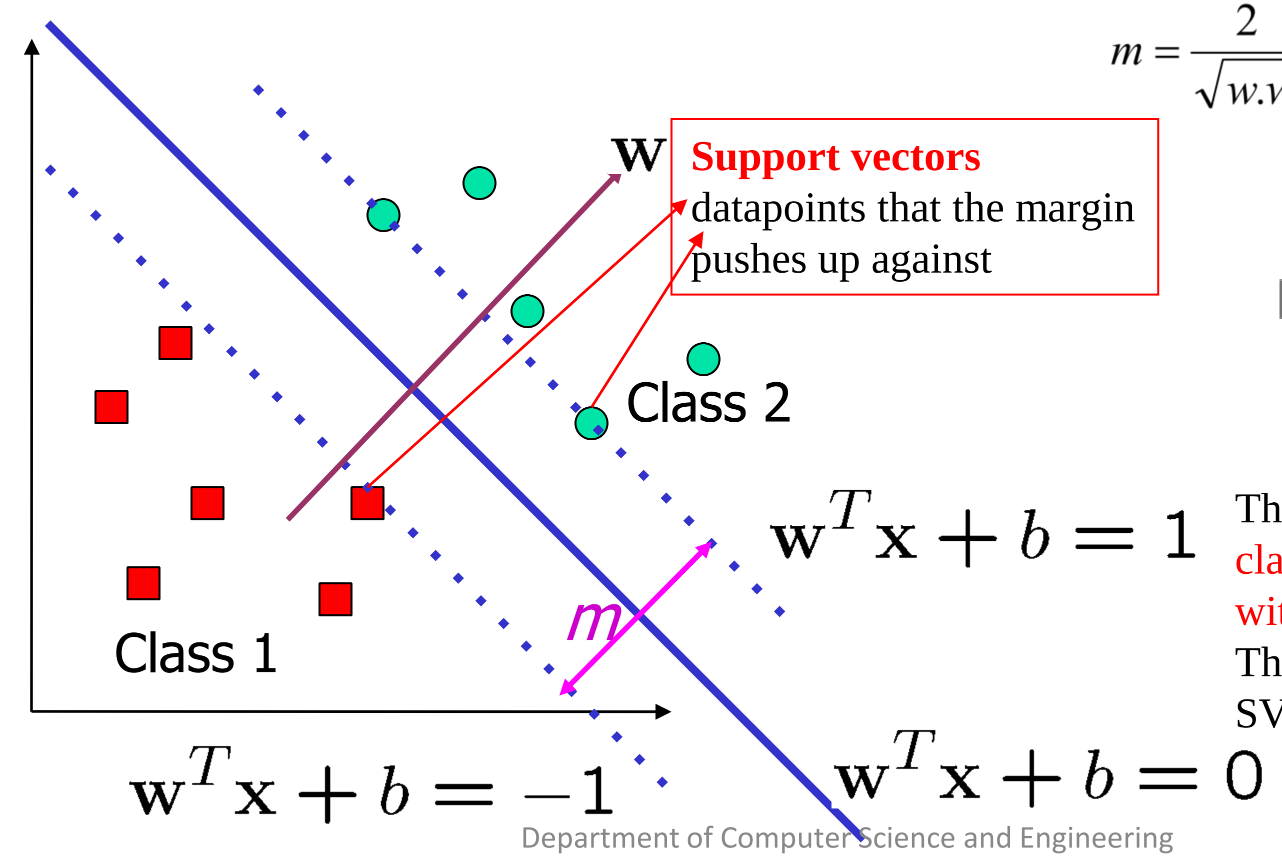
Example of Bad Decision Boundaries





Good Decision Boundary: Margin Should Be Large

- The decision boundary should be as far away from the data of both classes as possible
  - We should maximize the margin, *m*



$\mathbf{w}$
Class 2
Class 1
m
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}^T\mathbf{x} + b = -1$
$\mathbf{w}^T\mathbf{x} + b = 0$

Support vectors
datapoints that the margin pushes up against

$$m = \frac{2}{\sqrt{w.w}} \qquad m = \frac{2}{||\mathbf{w}||}$$

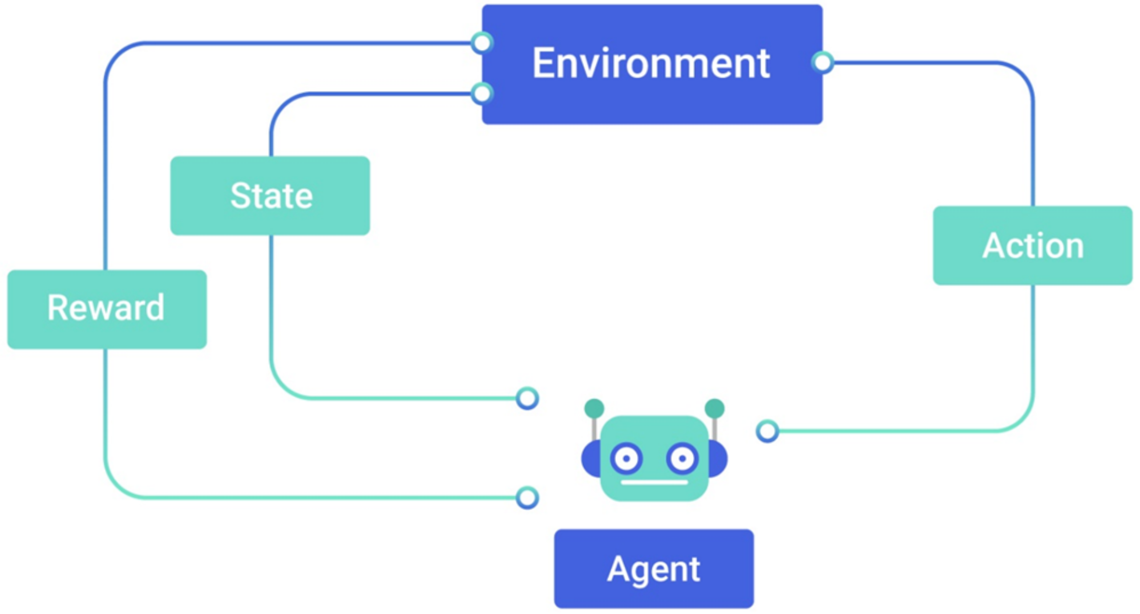
$$\|\mathbf{x}\| := \sqrt{x_1^2 + \cdots + x_n^2}.$$

The maximum margin linear classifier is the linear classifier with the maximum margin. This is the simplest kind of SVM (Called an Linear SVM)

What is Reinforcement Learning?

- Learning from interaction with an environment to achieve some long-term goal that is related to the state of the environment
- The goal is defined by reward signal, which must be maximised.
- Agent must be able to partially/fully sense the environment state and take actions to influence the environment state
- The state is typically described with a feature-vector

Reinforcement learning



Exploration versus Exploitation

- We want a reinforcement learning agent to earn lots of reward
- The agent must prefer past actions that have been found to be effective at producing reward
- The agent must exploit what it already knows to obtain reward
- The agent must select untested actions to discover reward-producing actions
- The agent must explore actions to make better action selections in the future
- Trade-off between exploration and exploitation

Reinforcement Learning Systems

- Reinforcement learning systems have 4 main elements:
  - Policy
  - Reward signal
  - Value function
  - Optional model of the environment

Policy

- A policy is a mapping from the perceived states of the environment to actions to be taken when in those states
- A reinforcement learning agent uses a policy to select actions given the current environment state

On-policy versus Off-policy

- An on-policy agent learns only about the policy that it is executing
- An off-policy agent learns about a policy or policies different from the one that it is executing

Reward Signal

- The reward signal defines the goal
- On each time step, the environment sends a single number called the reward to the reinforcement learning agent
- The agent’s objective is to maximise the total reward that it receives over the long run
- The reward signal is used to alter the policy

Value Function (1)

- The reward signal indicates what is good in the short run while the value function indicates what is good in the long run
- The value of a state is the total amount of reward an agent can expect to accumulate over the future, starting in that state
- Compute the value using the states that are likely to follow the current state and the rewards available in those states
- Future rewards may be time-discounted with a factor in the interval [0, 1]

Value Function (2)

- Use the values to make and evaluate decisions
- Action choices are made based on value judgements
- Prefer actions that bring about states of highest value instead of highest reward
- Rewards are given directly by the environment
- Values must continually be re-estimated from the sequence of observations that an agent makes over its lifetime

Model-free versus Model-based

- A model of the environment allows inferences to be made about how the environment will behave
- Example: Given a state and an action to be taken while in that state, the model could predict the next state and the next reward
- Models are used for planning, which means deciding on a course of action by considering possible future situations before they are experienced
- Model-based methods use models and planning. Think of this as modelling the dynamics p(s' | s, a)
- Model-free methods learn exclusively from trial-and-error (i.e. no modelling of the environment)
- This presentation focuses on model-free methods

Advantages of Reinforcement Learning

- It can solve higher-order and complex problems. Also, the solutions obtained will be very accurate.
- The reason for its perfection is that it is very similar to the human learning technique.
- Due to it’s learning ability, it can be used with neural networks. This can be termed as **deep reinforcement learning**.
- Since the model learns constantly, a mistake made earlier would be unlikely to occur in the future.
- Various problem-solving models are possible to build using reinforcement learning.
- When it comes to creating simulators, object detection in automatic cars, robots, etc., reinforcement learning plays a great role in the models.
- The best part is that even when there is no training data, it will learn through the experience it has from processing the training data.
- For various problems, which might seem complex to us, it provides the perfect models to tackle them.

Disadvantages of Reinforcement Learning

- The usage of reinforcement learning models for solving simpler problems won't be correct. The reason being, the models generally tackle **complex** problems.
- We will be wasting unnecessary processing power and space by using it for simpler problems.
- We need lots of data to feed the model for computation. Reinforcement Learning models require a lot of training data to develop accurate results.
- This consumes time and lots of computational power.
- When it comes to building models on real-world examples, the maintenance cost is very high.
- Like for building driverless vehicles, robots, we would require a lot of maintenance for both hardware and software.
- Excessive training can lead to overloading of the states of the model. This will result in the model for getting the result.
- This may happen if too much memory space goes out in processing the training data.

Adaptive learning

- No learning method is complete in itself. So
- Need to select the learning method based on the requirements.
- Need to develop a combination of some of the existing methods based on requirements.
- Adaptive machine learning algorithms are the machine learning models, where the changes in the environment help in selecting the algorithm or learning method.

Adaptive learning (Contd..)

- As per the scenario, most suitable algorithm is selected.
- Moreover the development of especially fast adapting algorithms poses many different issues like selection of choices, handling equilibrium states and so on.
- The adaptive learning solves some of the complex problems for which a single learning method is not enough.
- This method is even more appropriate when the environment is continuously changing and real time response is expected.
-

What is a Multi-Agent System?

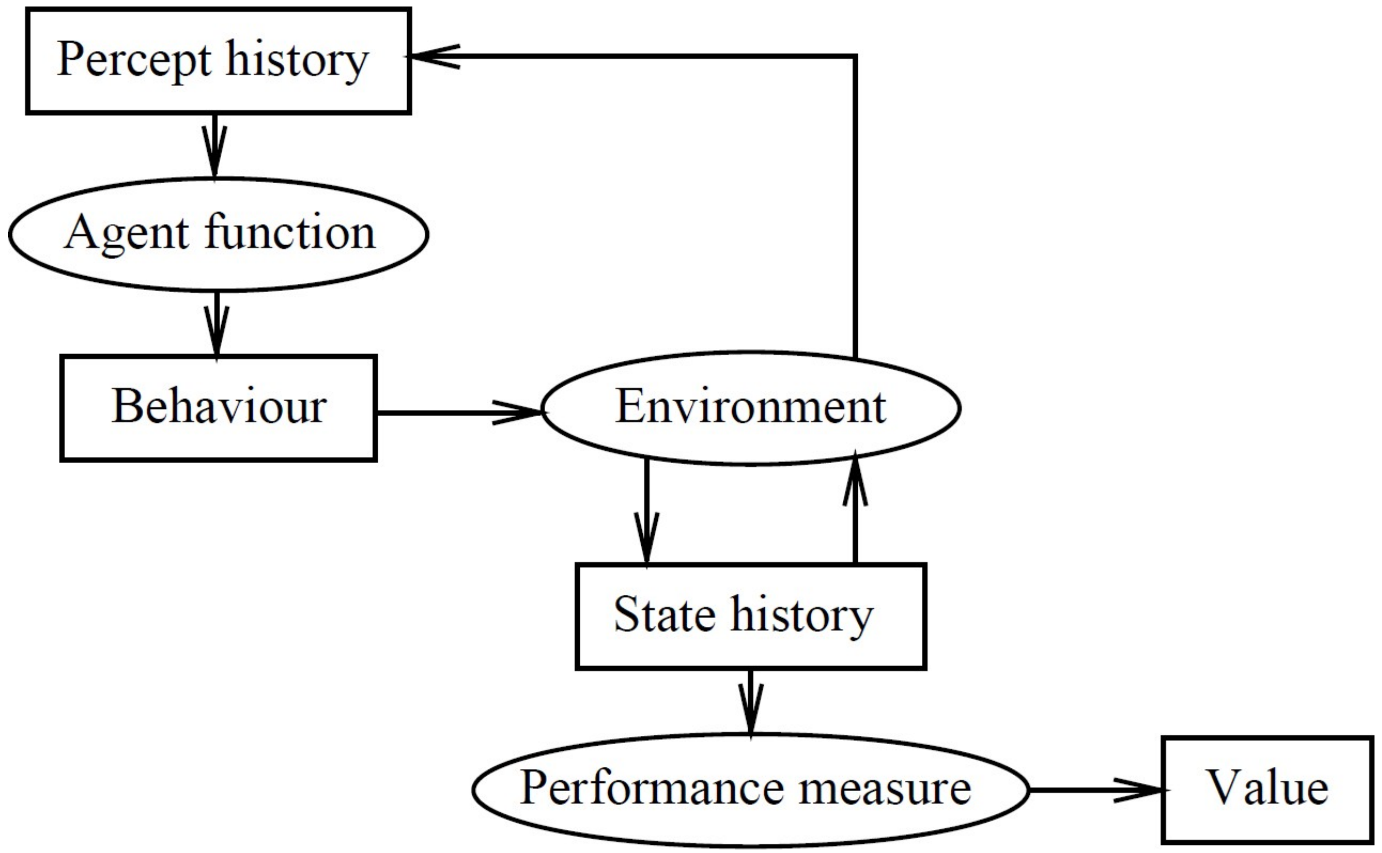
- A system with multiple autonomous entities, with distributed information, computational ability, and possibly divergent interests.

- Agents :: artificial or human, cooperative or self-interested

Multiagent Systems, a Definition

- A multiagent system is one that consists of a number of agents, which *interact* with one-another
- In the most general case, agents will be acting on behalf of users with different goals and motivations
- To successfully interact, they will require the ability to *cooperate*, *coordinate*, and *negotiate* with each other, much as people do

Overview of an agent



Learning in Multiagent Systems

- Intersection of DAI and ML

- Why bring them together?
  - There is a strong need to equip Multiagent systems with learning abilities
  - The extended view of ML as Multiagent learning is qualitatively different from traditional ML and can lead to novel ML techniques and algorithms

Need for Multi agent Learning

- A single agent cannot handle learning in case of applications
- A team or group of agents possesses the potential to overcome the limitation of single agent and work in in coordination to accomplish a task.
- This can be two cases in multi agent based learning:

Multi agent based learning

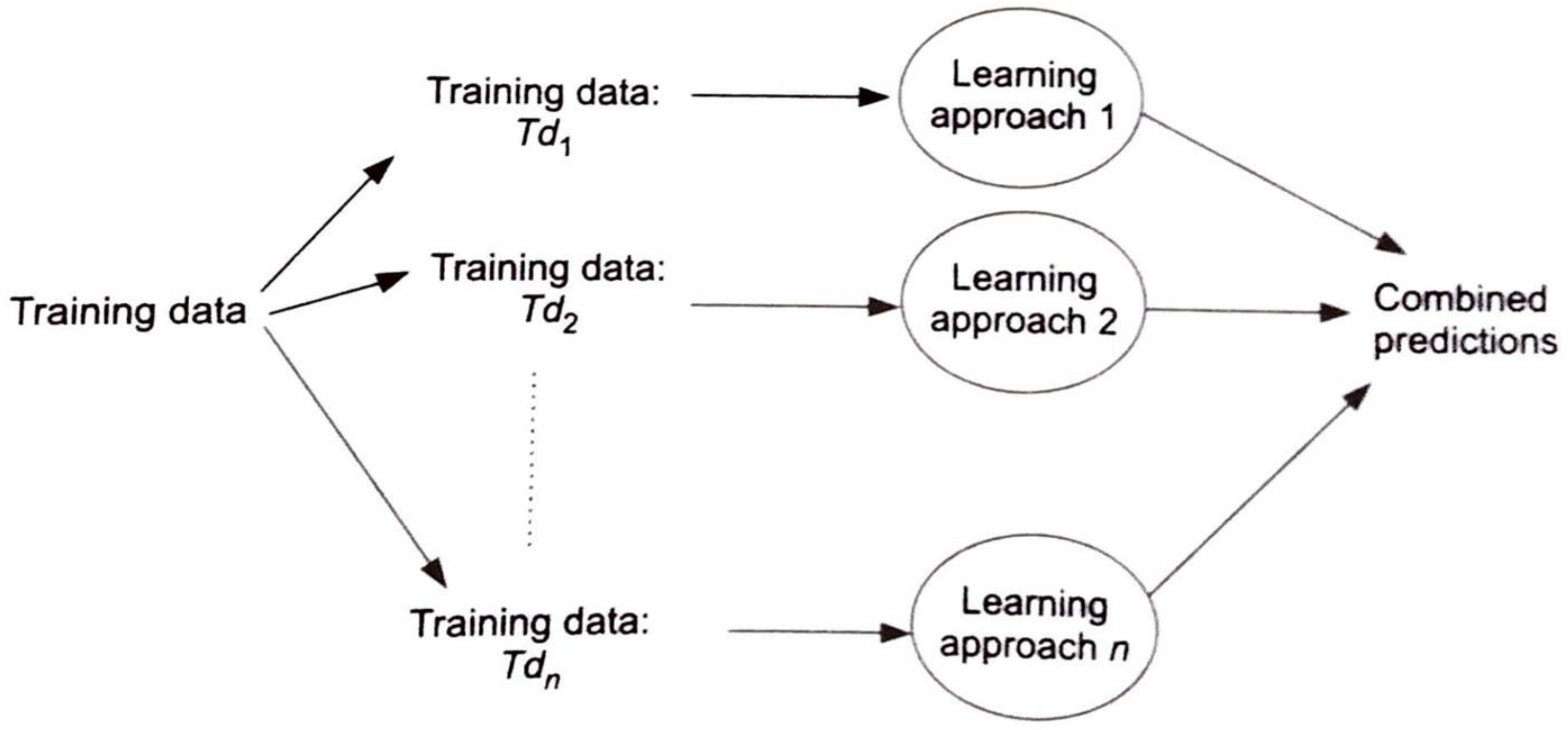
1) Where the agent tries to maximize its own utility

- Eg: Consider a manufacturing industry domain. The task are built and assigned where each agent works in co-operation to build the end product. This is the case to achieve a common goal.

2) Where they worked in collaboration to achieve some common goals.

- Eg: Game playing. In that gaming environment, multiple agents are in Operation to select the best strategy.
- Can be related with the reinforcement learning, Where for each strategy of the agent some reward is achieved this is where each agent tries to maximize his own utility function.
-

Ensemble Learning



Ensemble method: The concept.

Ensemble Learning (Contd…)

Ensemble learning method is the one where multiple learners or learning algorithms are trained.

- In most of the learning algorithms a single hypothesis drives the learning.
- In ensemble learning method the whole collection or ensemble of hypothesis is selected from the hypothesis space and their predictions are combined.
- In this approach, the learners or referred to as base learners.
- The most commonly used ensemble learning methods are

1) Boosting
2) Bagging.

Ensemble Learning (Contd..)

Boosting:

- Boosting can probably be defined as the method for generating accurate predictions by combining the rules that are comparatively inaccurate.
- Boosting works on the weighted training sets. The weights of the training example reflects the importance of training examples.

Bagging:

- In Bagging, the training data is resampled. This is referred to as ***bootstrap sampling***, where the training data with replacement is taken in the learning approaches.

Learning for decision making

- It is observed from different learning mechanisms that Capability to take decisions is increased.
- speaking about the supervisor and unsupervised methodologies the decisions taken are not sequential in nature.
- That is, if the system make a mistake on one decision, this has no bearing on the subsequent decisions.
- To cope up with this dynamic situation there is a need to understand the perspective of decision making.
- Another aspect is environment and system Learning, which also needs to be looked upon during decision making. While taking decisions **one specific learning approach may not be suitable**.
- The learning approach is dependent on decision scenario.

Distributed learning

- In distributed learning the task of learning is distributed.
- **Need for distributed learning** - Arises due to large data sets and time constraints.
- More than one agent in different parts of the data set. There will be **distributed learning algorithms** taken part in each partition to get the desired outcome, which would then be combined.
- **Efficiency** of distributed learning is affected to look at. it is extremely important that outcome of distributed learning matches with the ones acheived under the absence of distributed environment.
- Multi agent systems can be thought of as a **subset of distributed learning.**
-

Speedup learning

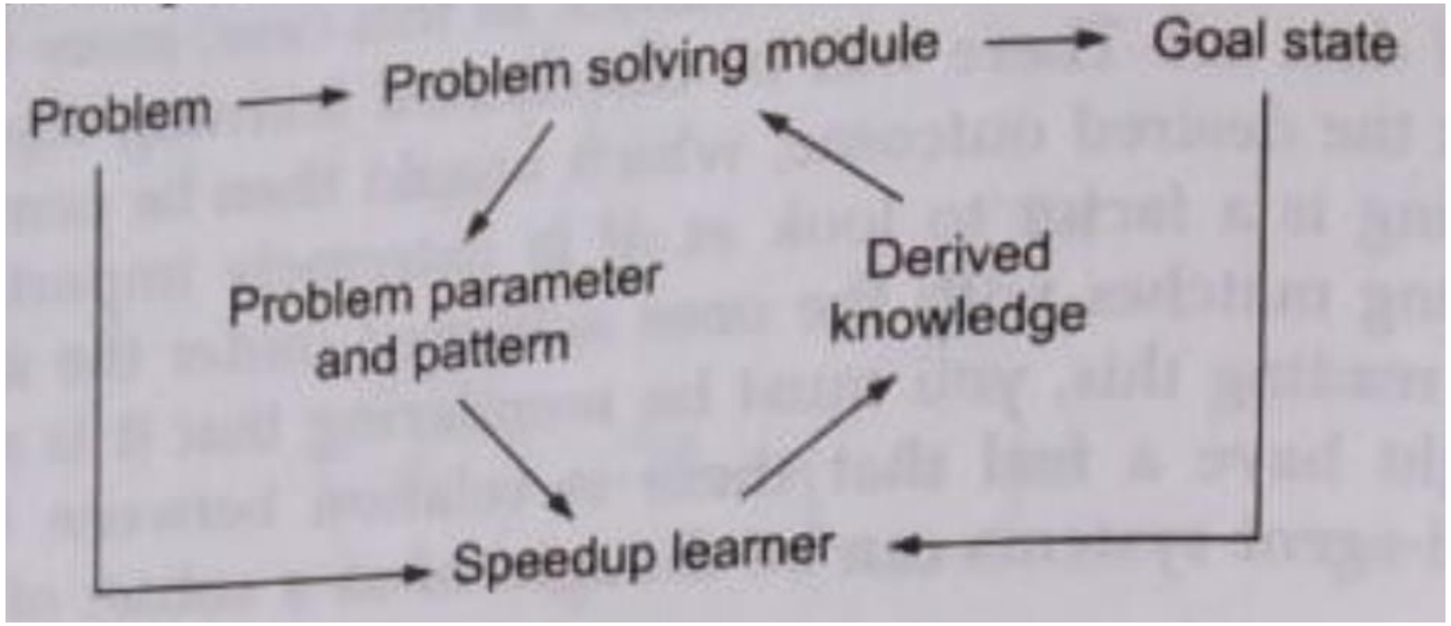
- Speed learning typically deals with speeding up problem solving by effective use of problem solving experience. Hence, Prayer problem solving experience is an input for speed of learning.
- In this learning,

1) There is no option with the environment.

2) New problems cannot be solved.

So, speed up learning accelerates the process experiences
And prior observations.

Speedup learning (Contd..)



Speedup learning modules.

Generalized learning

- Another dimension to the speed up learning is generalized learning. this is also known as explanation based learning.
- There are a number of issues with explanation based learning, as it is implemented and embedded in system to solve real life problems and is suitable for a particular set of problems where the sequential processes once developed can be used again and again.
- But this is not the case in many real life problems, where dynamic change in environment demands the improvement in the established scenarios and even there is a need to keep on learning based on the new findings
-