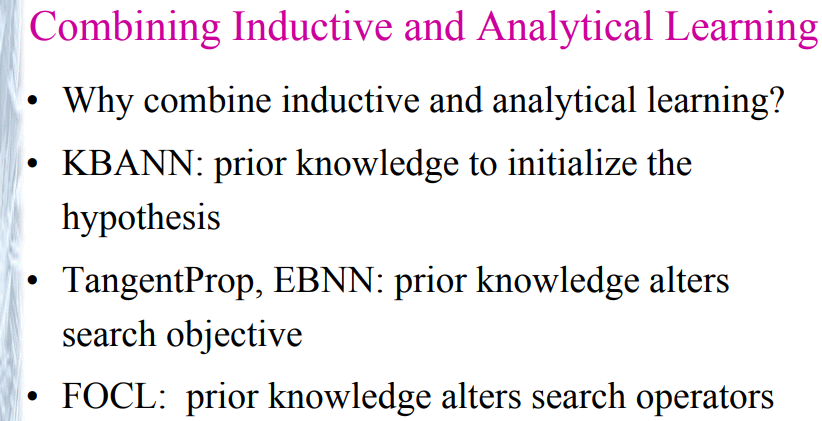
UNIT-5

Combining Inductive and Analytical Learning –

Motivation



**Inductive Learning**

* Goal: hypothesis fits data
* Justification: statistical inference
* Advantages: requires little prior knowledge
* Pitfalls: scarce data, incorrect bias

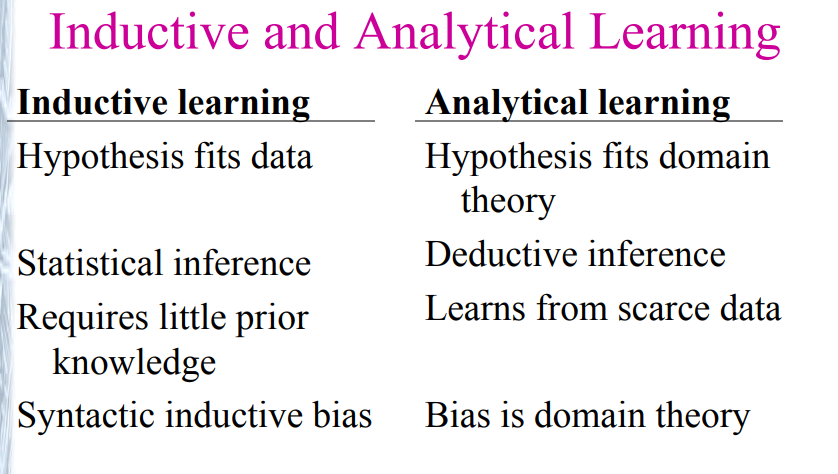
**Analytical Learning**

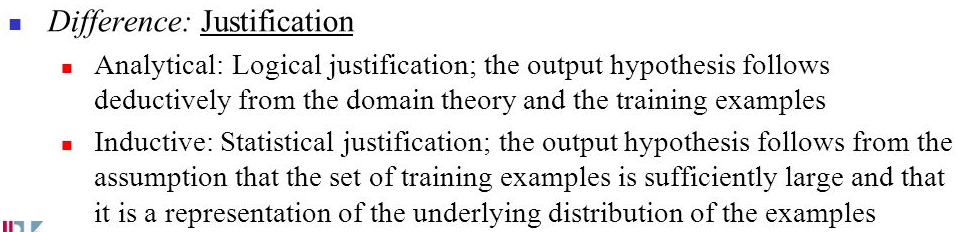
* Goal: hypothesis fits domain theory
* Justification: deductive inference
* Advantages: learns from scarce data
* Pitfalls: imperfect domain theory

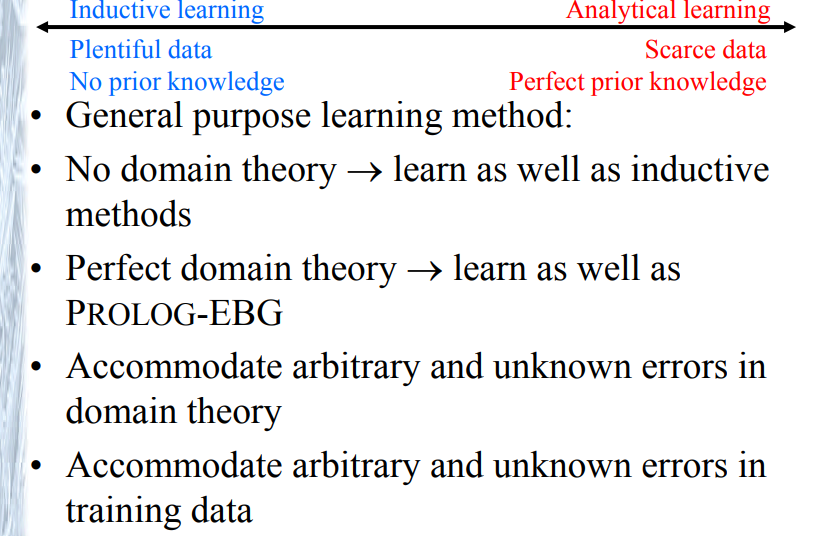
**Desirable Properties of a Combined System**

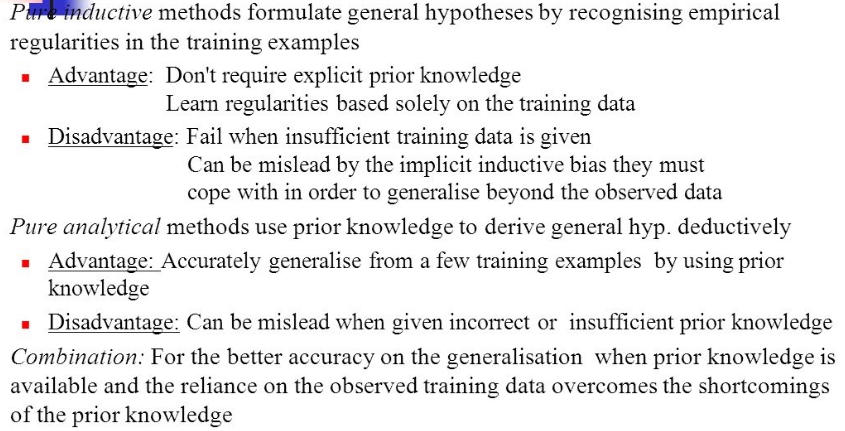
* Given no domain theory, it should learn at least as effectively as purely inductive methods
* Given a perfect domain theory, it should learn at least as effectively as analytical methods
* Given an imperfect domain theory and imperfect training data, it should combine the two to outperform either purely inductive or purely analytical methods
* It should accommodate an unknown level of data in the training data
* It should accommodate an unknown level of error in the domain theory

**Motivation**

****

****

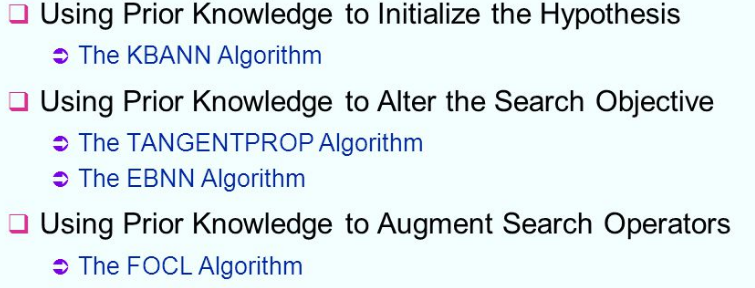
****

****

## 

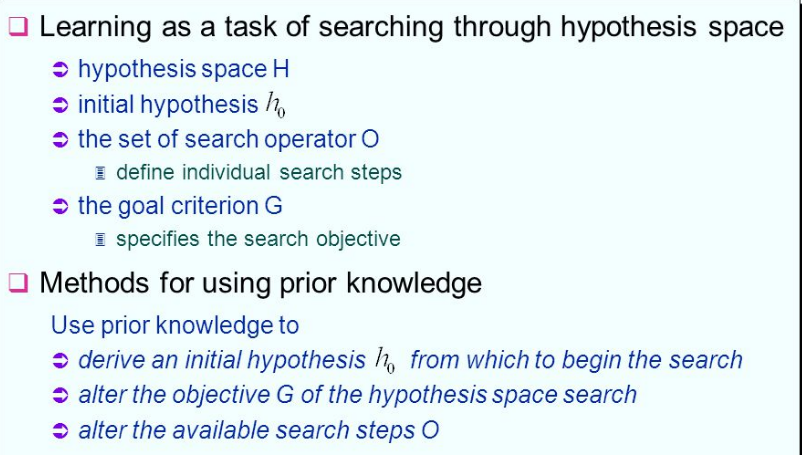
## Learning Framework

* Given D: the training data, possibly containing errors
* Given B: the domain theory, possibly containing errors
* Given H: the hypothesis space
* Determine h: a hypothesis that best fits the training data and domain theory
* How can we determine the best fit? One idea is to find argminh ∈ H kD errorD(h) + kB errorB(h)

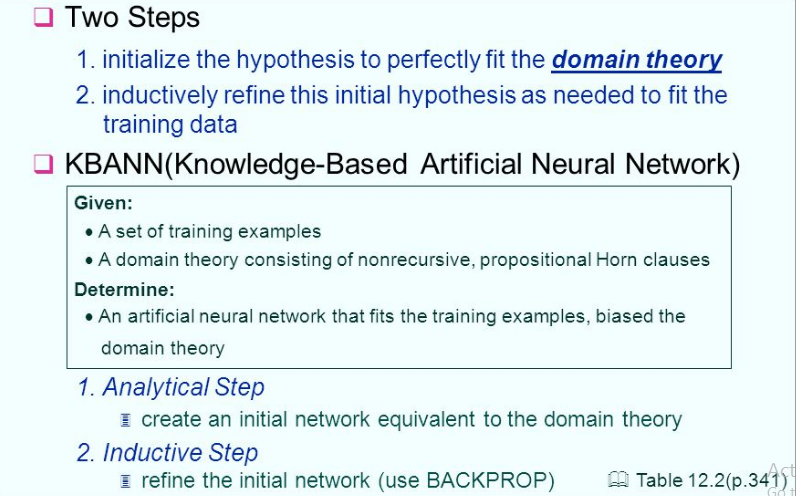


## Hypothesis Space Search Techniques

* Use prior knowledge to derive an initial hypothesis from which to begin the search - KBANN
* Use prior knowledge to alter the objective of the hypothesis space search - EBNN
* Use prior knowledge to alter the available search steps - FOCL

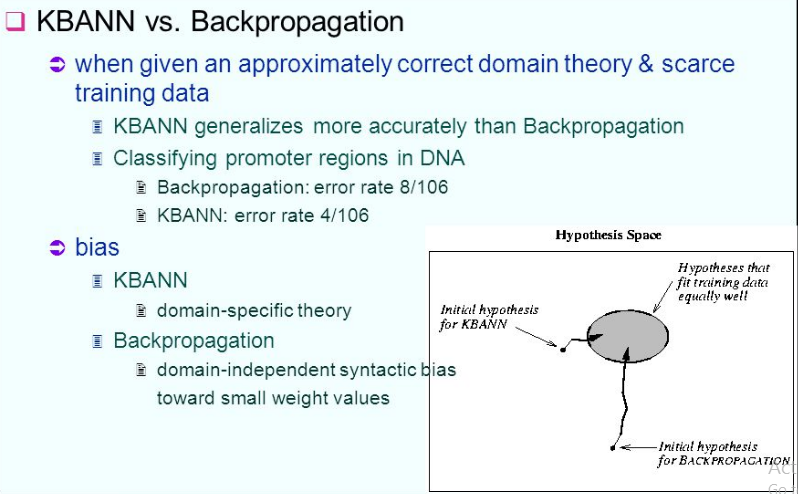


**Using Prior Knowledge to Initialize the Hypothesis**



## KBANN

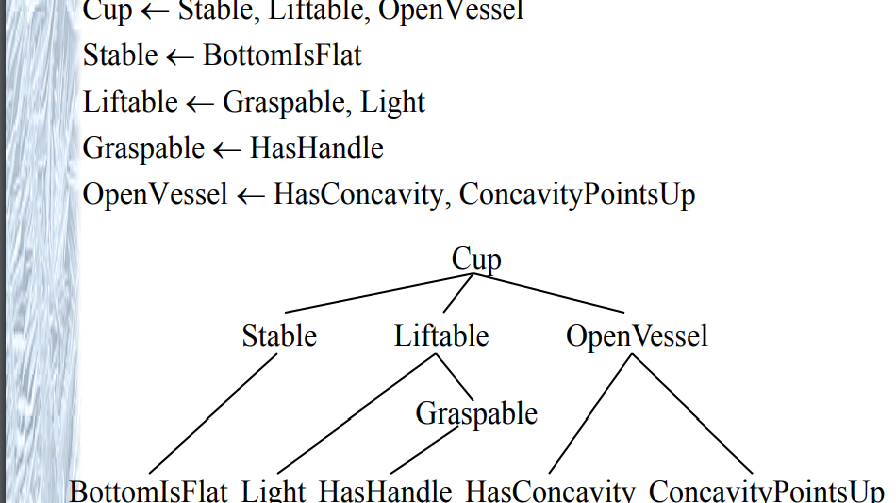
* Knowledge-Based Artificial Neural Network
* KBANN's bias comes from the domain-specific theory used to initialize the weights
* Input: D
* Input: B, a domain theory consisting of nonrecursive, propositional Horn clauses
* First, construct a neural network that classifies D according to the domain theory (see below)
* Second, employ backpropagation to fit D
* KBANN typically generalizes more accurately than standard backpropagation. However, B must be fairly accurate and B is limited to a particular syntactic form.

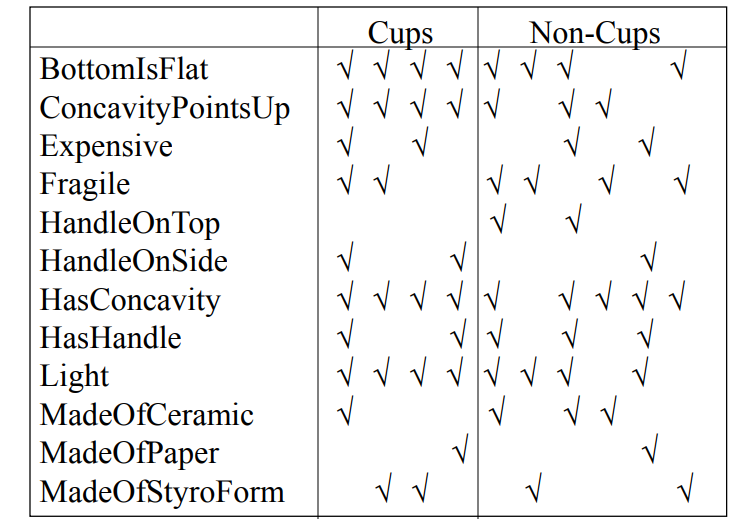


### Constructing the Initial KBANN Network

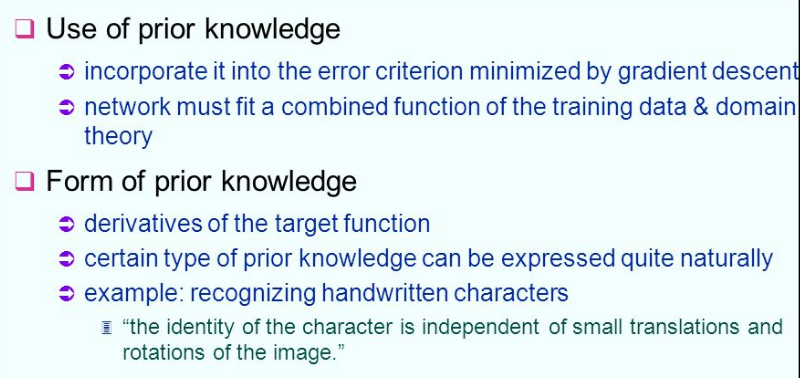
1. For each instance attribute create a network input
2. For each Horn clause in B, create a network unit as follows
   * Connect the antecedents of the Horn clause to the consequent, creating new network nodes as needed
   * For each non-negated antecedent of the clause, assign a weight of W to the corresponding sigmoid unit input
   * For each negated antecedent of the clause, assign a weight of -W to the corresponding sigmoid unit input
   * Set the threshold weight w0 for this unit to -(n - .5)W where n is the number of non-negated antecedents of the clause
3. Add additional connections among the network units, connecting each network unit at depth i from the input layer to all network units at depth i+1. Assign random near-zero weights to these additional connections

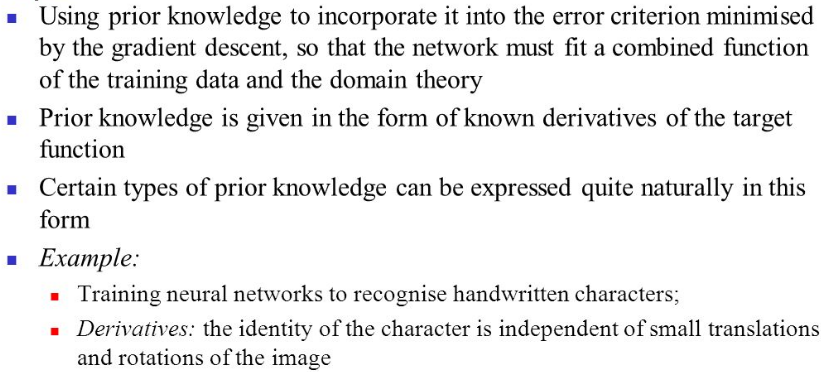
#### Domain theory example



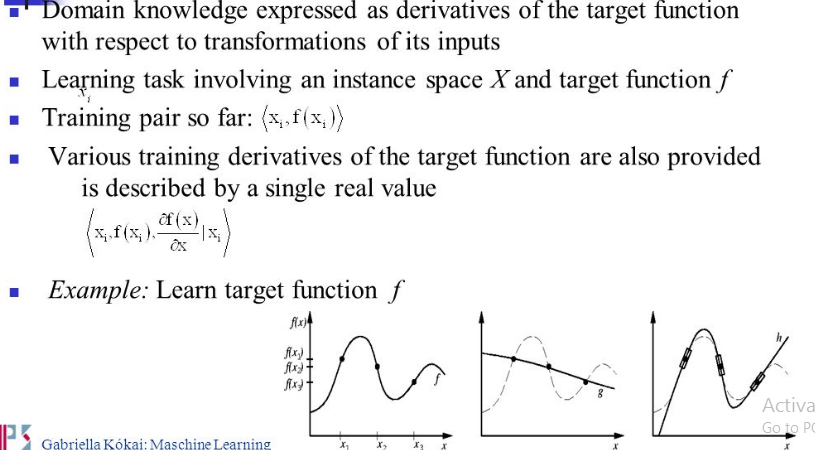


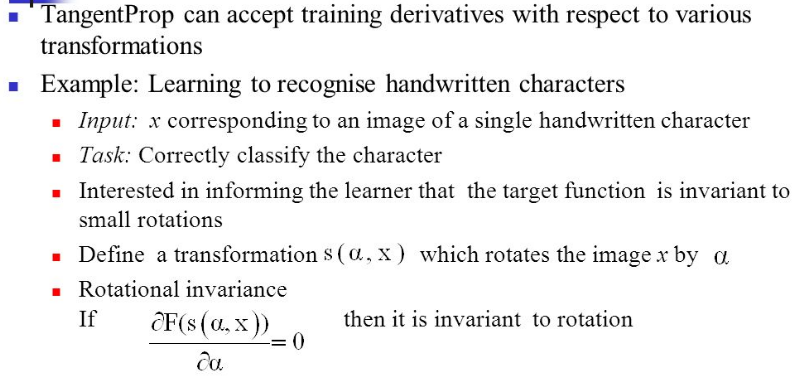
**Using Prior Knowledge to Alter the Search Objective**

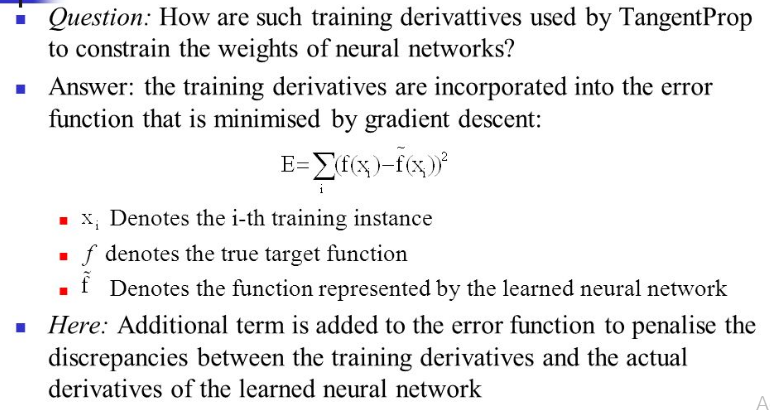


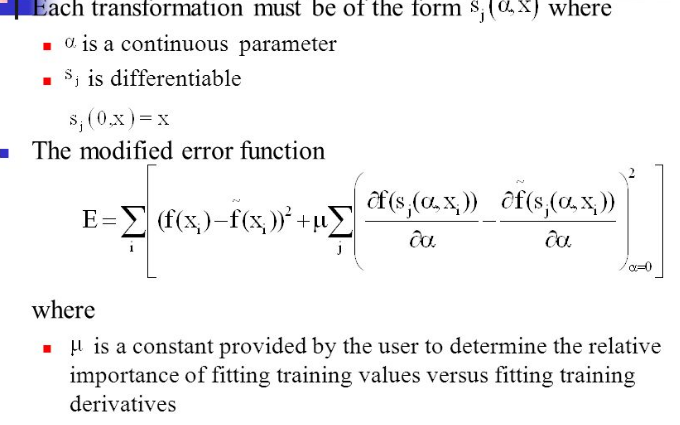


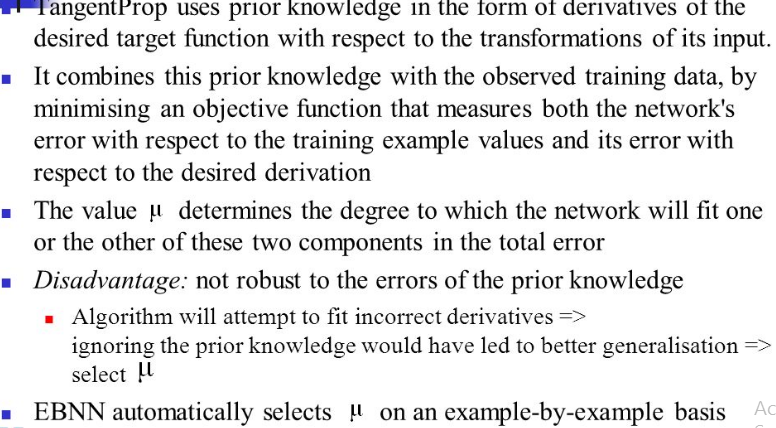
**The TANGENTPROP Algorithm**



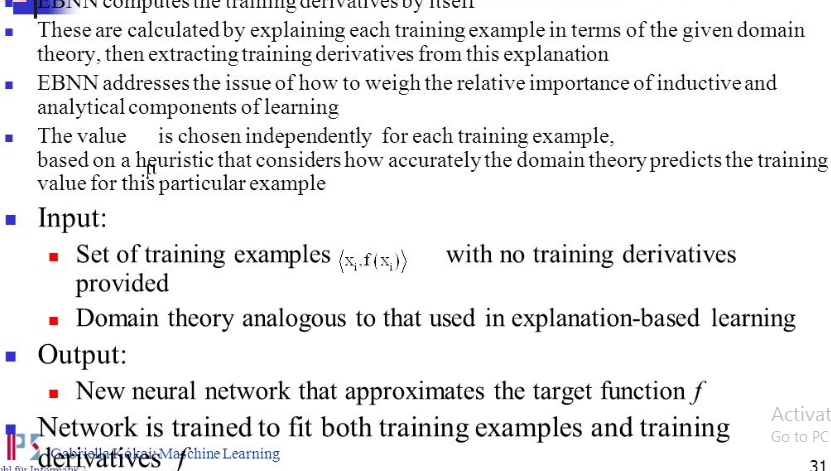


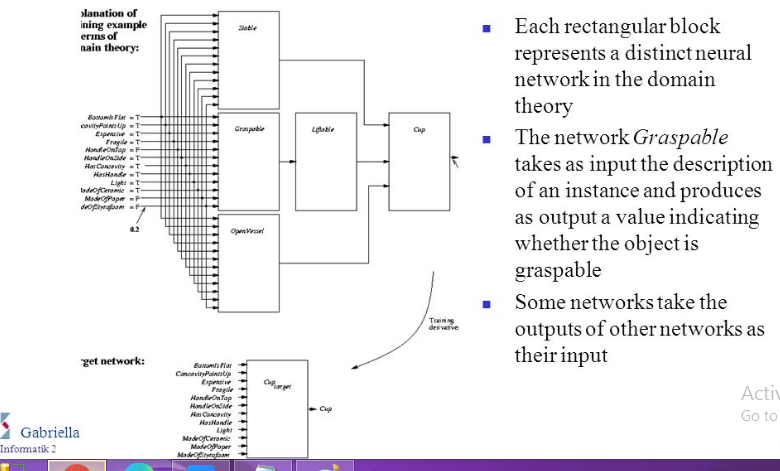


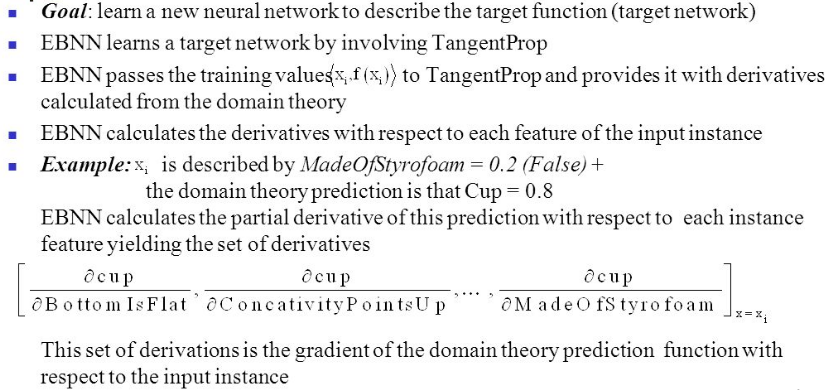


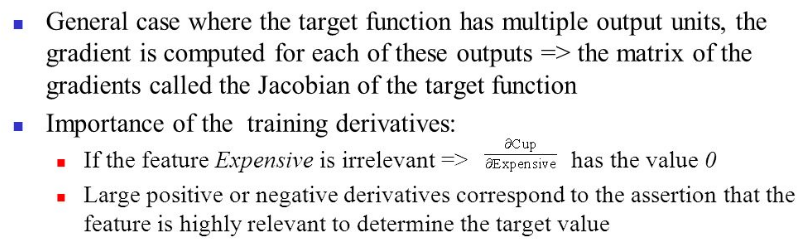


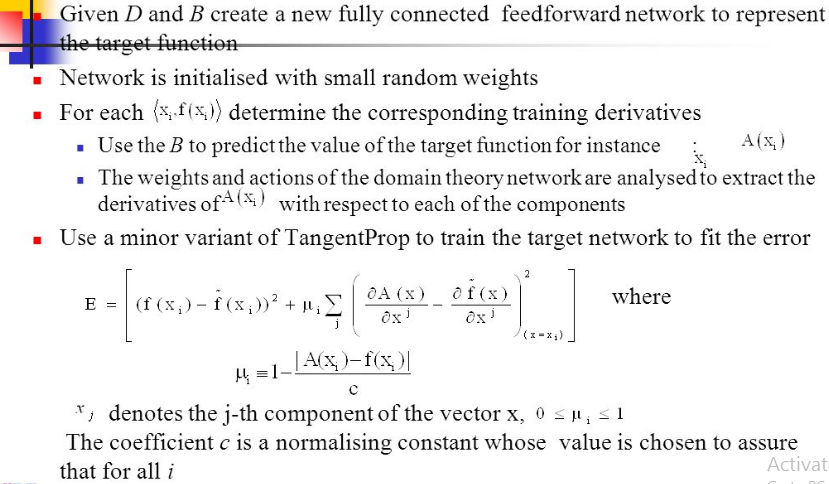
**EBNN Algorithm**





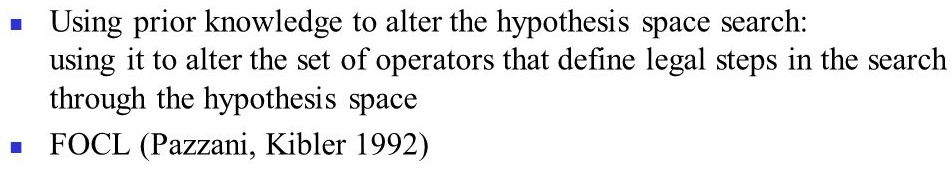




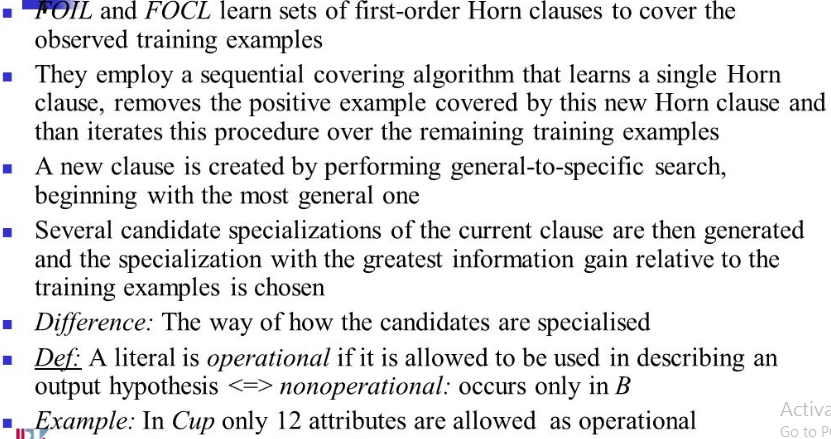


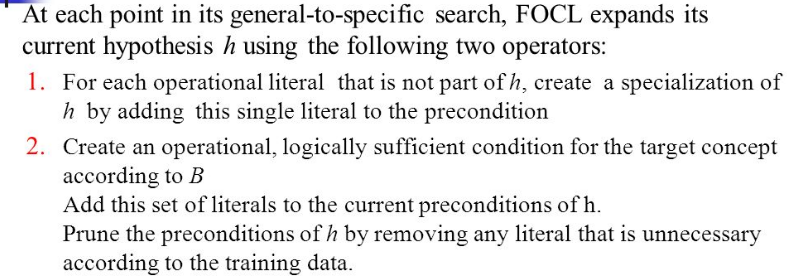


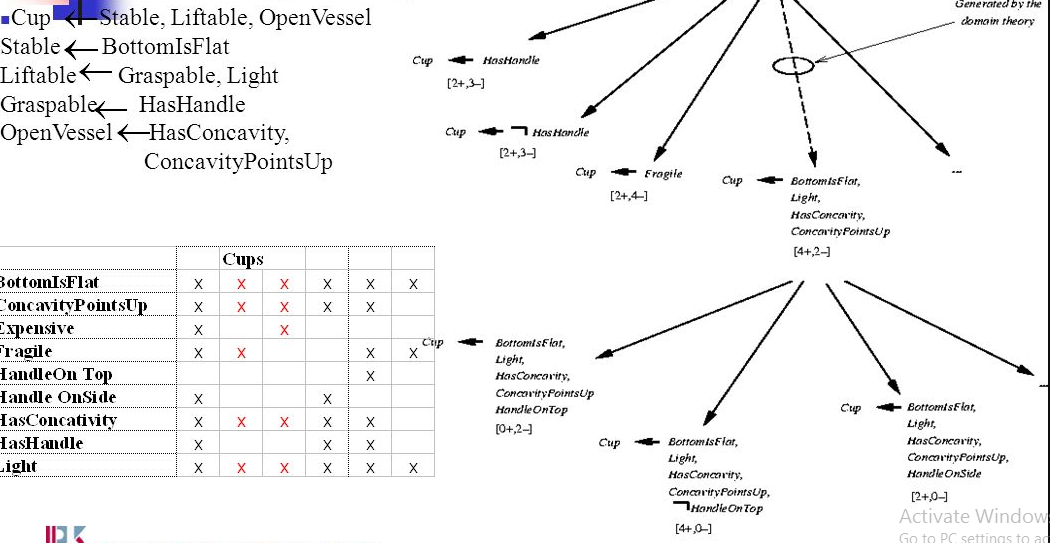
**Using Prior Knowledge to Augment Search Operators**

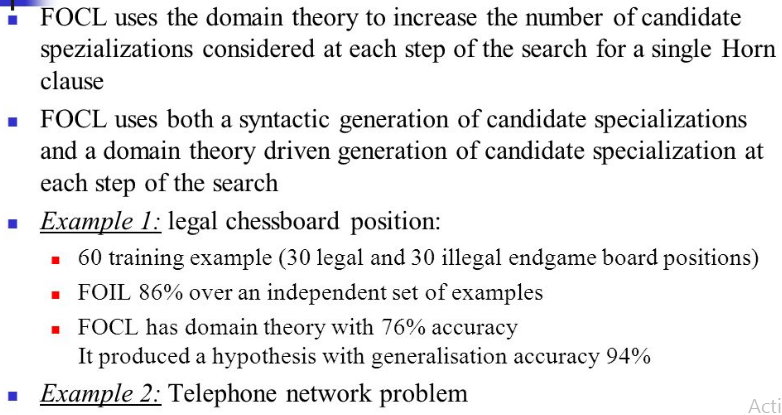


**FOCL Algorithm**









# Reinforcement learning

Reinforcement learning is an area of Machine Learning. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behavior or path it should take in a specific situation. Reinforcement learning differs from the supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of a training dataset, it is bound to learn from its experience.

**Example:** The problem is as follows: We have an agent and a reward, with many hurdles in between. The agent is supposed to find the best possible path to reach the reward. The following problem explains the problem more easily.

* Input: The input should be an initial state from which the model will start
* Output: There are many possible output as there are variety of solution to a particular problem
* Training: The training is based upon the input, The model will return a state and the user will decide to reward or punish the model based on its output.
* The model keeps continues to learn.
* The best solution is decided based on the maximum reward.

**Difference between Reinforcement learning and Supervised learning:**

| Reinforcement learning | Supervised learning |
| --- | --- |
| Reinforcement learning is all about making decisions sequentially. In simple words we can say that the output depends on the state of the current input and the next input depends on the output of the previous input  In Reinforcement learning decision is dependent, So we give labels to sequences of dependent decisions  Ex:Chess game | In Supervised learning the decision is made on the initial input or the input given at the start  Supervised learning the decisions are independent of each other so labels are given to each decision.  Ex:Object recognition |

**Types of Reinforcement:** There are two types of Reinforcement:

1. **Positive –**  
   Positive Reinforcement is defined as when an event, occurs due to a particular behavior, increases the strength and the frequency of the behavior. In other words, it has a positive effect on behavior.

Advantages of reinforcement learning are:

* + Maximizes Performance
  + Sustain Change for a long period of time

Disadvantages of reinforcement learning:

* + Too much Reinforcement can lead to overload of states which can diminish the results

**2 Negative –**  
Negative Reinforcement is defined as strengthening of a behavior because a negative condition is stopped or avoided.

Advantages of reinforcement learning:

* + Increases Behavior
  + Provide defiance to minimum standard of performance

Disadvantages of reinforcement learning:

* + It Only provides enough to meet up the minimum behavior

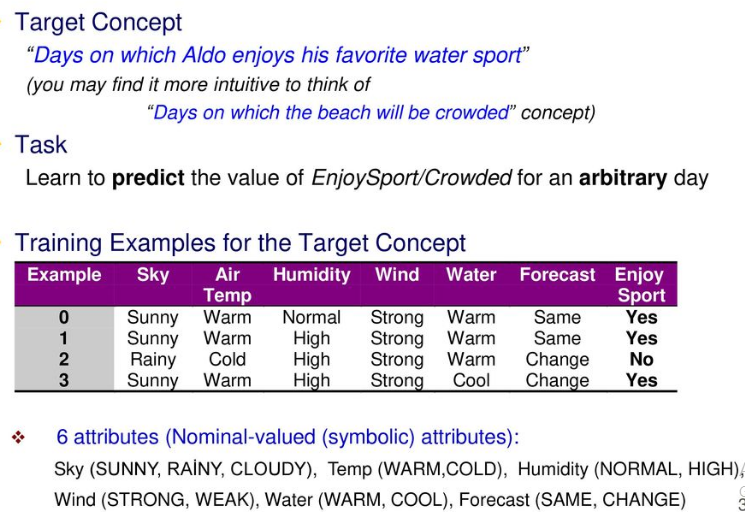
**Various Practical applications of Reinforcement Learning –**

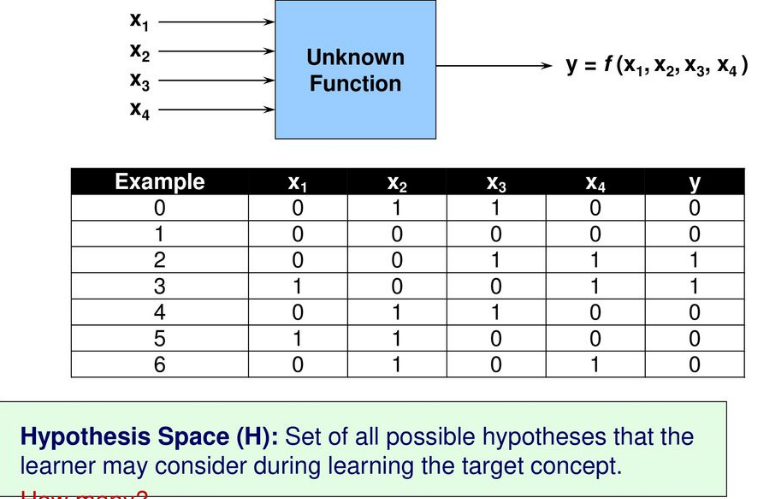
* RL can be used in robotics for industrial automation.
* RL can be used in machine learning and data processing
* RL can be used to create training systems that provide custom instruction and materials according to the requirement of students.

RL can be used in large environments in the following situations:

1. A model of the environment is known, but an analytic solution is not available;
2. Only a simulation model of the environment is given (the subject of simulation-based optimization)
3. The only way to collect information about the environment is to interact with it.

**The Learning Task**





# Learning

**Q-Learning** is a basic form of Reinforcement Learning which uses Q-values (also called action values) to iteratively improve the behavior of the learning agent.

* **Q-Values or Action-Values:** Q-values are defined for states and actions. Rendered by QuickLaTeX.com is an estimation of how good is it to take the action Rendered by QuickLaTeX.com at the state Rendered by QuickLaTeX.com. This estimation of Rendered by QuickLaTeX.com will be iteratively computed using the **TD- Update rule** which we will see in the upcoming sections.
* **Rewards and Episodes:** An agent over the course of its lifetime starts from a start state, makes a number of transitions from its current state to a next state based on its choice of action and also the environment the agent is interacting in. At every step of transition, the agent from a state takes an action, observes a reward from the environment, and then transits to another state. If at any point of time the agent ends up in one of the terminating states that means there are no further transition possible. This is said to be the completion of an episode.
* **Temporal Difference or TD-Update:**

The Temporal Difference or TD-Update rule can be represented as follows :



This update rule to estimate the value of Q is applied at every time step of the agents interaction with the environment. The terms used are explained below. :

* Rendered by QuickLaTeX.com : Current State of the agent.
* Rendered by QuickLaTeX.com : Current Action Picked according to some policy.
* Rendered by QuickLaTeX.com : Next State where the agent ends up.
* Rendered by QuickLaTeX.com : Next best action to be picked using current Q-value estimation, i.e. pick the action with the maximum Q-value in the next state.
* Rendered by QuickLaTeX.com : Current Reward observed from the environment in Response of current action.
* Rendered by QuickLaTeX.com(>0 and <=1) : Discounting Factor for Future Rewards. Future rewars are less valuable than current rewards so they must be discounted. Since Q-value is an estimation of expected rewards from a state, discounting rule applies here as well.
* Rendered by QuickLaTeX.com : Step length taken to update the estimation of Q(S, A).
* **Choosing the Action to take using**Rendered by QuickLaTeX.com**-greedy policy:**
* Rendered by QuickLaTeX.com-greedy policy of is a very simple policy of choosing actions using the current Q-value estimations. It goes as follows :
* With probability Rendered by QuickLaTeX.com choose the action which has the highest Q-value.
* With probability Rendered by QuickLaTeX.com choose any action at random.

Now with all the theory required in hand let us take an example. We will use OpenAI’s gym environment to train our Q-Learning model.

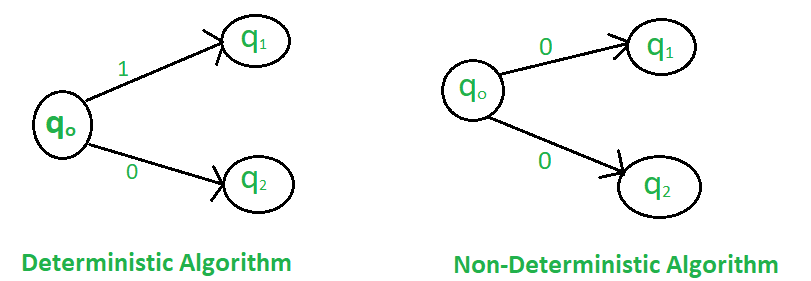
Command to Install **gym**–

pip install gym

Before starting with example, you will need some helper code in order to visualize the working of the algorithms. There will be two helper files which need to be downloaded in the working directory. One can find the files .

# Difference between Deterministic and Non-deterministic Algorithms

In **deterministic algorithm**, for a given particular input, the computer will always produce the same output going through the same states but in case of **non-deterministic algorithm**, for the same input, the compiler may produce different output in different runs. In fact non-deterministic algorithms can’t solve the problem in polynomial time and can’t determine what is the next step. The non-deterministic algorithms can show different behaviors for the same input on different execution and there is a degree of randomness to it



**Some of the terms related to the non-deterministic algorithm are defined below**:

* **choice(X) :** chooses any value randomly from the set X.
* **failure() :** denotes the unsuccessful solution.
* **success() :** Solution is successful and current thread terminates.

Example:

***Problem Statement :****Search an element x on A[1:n] where n>=1, on successful search return j if a[j] is equals to x otherwise return 0.*

***Non-deterministic Algorithm for this problem :***

*1.j= choice(a, n)*

*2.if(A[j]==x) then*

*{*

*write(j);*

*success();*

*}*

1. *write(0); failure();*

| **Deterministic Algorithm** | **Non-deterministic Algorithm** |
| --- | --- |
| For a particular input the computer will give always same output. | For a particular input the computer will give different output on different execution. |
| Can solve the problem in polynomial time. | Can’t solve the problem in polynomial time. |
| Can determine the next step of execution. | Cannot determine the next step of execution due to more than one path the algorithm can take. |