**Well Posed Learning Algorithm**

**Well Posed Learning Problem –** A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks T, as measured by P, improves with experience E.

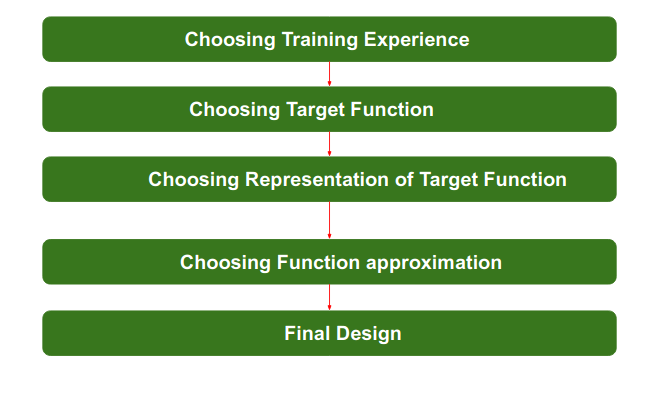
To have a Well Posed Learning Problem, three features must be identified:

1. Learning Task: The thing you want to learn
2. Performance Measure: Must know what you did bad and when you did good. Often called the critic
3. Training Experience: The source of training example

Example:

* A computer program that learns to play checkers might improve its performance as measured by its ability to win at the class of tasks involving playing checkers games, through experience obtained by playing games against itself.

**Steps for Designing Learning System are:**



1. **Choosing the Training Experience:**

* The very important and first task is to choose the training data or training experience which will be fed to the Machine Learning Algorithm.
* It is important to note that the data or experience that we fed to the algorithm must have a significant impact on the Success or Failure of the Model.
* So, Training data or experience should be chosen wisely.
* The attributes which will impact on Success and Failure of Data:
  + The training experience will be able to provide direct or indirect feedback regarding choices.
    - For example: While Playing chess the training data will provide feedback to itself like instead of this move if this is chosen the chances of success increases.
  + The degree to which the learner will control the sequences of training examples.
    - For example: when training data is fed to the machine then at that time accuracy is very less but when it gains experience while playing again and again with itself or opponent the machine algorithm will get feedback and control the chess game accordingly.

1. **Choosing target function:**

* The next important step is choosing the target function.
* It means according to the knowledge fed to the algorithm the machine learning will choose NextMove function which will describe what type of legal moves should be taken.
* For example: While playing chess with the opponent, when opponent will play then the machine learning algorithm will decide what be the number of possible legal moves taken in order to get success.

1. **Choosing Representation for Target function:**

* When the machine algorithm will know all the possible legal moves the next step is to choose the optimized move using any representation i.e, using linear Equations, Hierarchical Graph Representation, Tabular form etc.
* The NextMove function will move the Target move like out of these move which will provide more success rate.
* For Example: while playing chess machine have 4 possible moves, so the machine will choose that optimized move which will provide success to it.

1. **Choosing Function Approximation Algorithm:**

* An optimized move cannot be chosen just with the training data. The training data had to go through with set of examples and through these examples the training data will approximates which steps are chosen and after that machine will provide feedback on it.
* For Example: When a training data of Playing chess is fed to algorithm so at that time it is not machine algorithm will fail or get success and again from that failure or success it will measure while next move what step should be chosen and what is its success rate.

1. **Final Design:**

* The final design is created at last when system goes from number of examples, failures and success, correct and incorrect decision and what will be the next step etc.
* Example: DeepBlue is an intelligent computer which is ML-based won chess game against the chess expert Garry Kasparov, and it became the first computer which had beaten a human chess expert.

**Find S Algorithm**

* The find-S algorithm is a basic concept learning algorithm in machine learning. The find-S algorithm finds the most specific hypothesis that fits all the positive examples. We have to note here that the algorithm considers only those positive training example. The find-S algorithm starts with the most specific hypothesis and generalizes this hypothesis each time it fails to classify an observed positive training data. Hence, the Find-S algorithm moves from the most specific hypothesis to the most general hypothesis.
* **Important Representation:**
  + **?**indicates that any value is acceptable for the attribute.
  + specify a single required value (e.g., Cold) for the attribute.
  + **Φ** indicates that no value is acceptable.
  + The most **general hypothesis** is represented by: **{? , ?, ?, ?, ?, ?}**
  + The most **specific hypothesis** is represented by: **{ϕ, ϕ, ϕ, ϕ, ϕ, ϕ}**

**Algorithm:**

1. Initialize h to the most specific hypothesis in H
2. For each positive training instance x

For each attribute constraint a, in h

If the constraint a, is satisfied by x

Then do nothing

Else replace a, in h by the next more general constraint that is satisfied by x

1. Output hypothesis h

# Candidate Elimination Algorithm

The candidate elimination algorithm incrementally builds the version space given a hypothesis space H and a set E of examples. The examples are added one by one; each example possibly shrinks the version space by removing the hypotheses that are inconsistent with the example. The candidate elimination algorithm does this by updating the general and specific boundary for each new example.

* You can consider this as an extended form of Find-S algorithm.
* Consider both positive and negative examples.
* Actually, positive examples are used here as Find-S algorithm.
* While the negative example is specified from generalize form.

**Algorithm:**

1. Initialize General Hypothesis and Specific Hypothesis.
2. For each training example
3. If example is positive example

if attribute\_value == hypothesis\_value:

Do nothing

else:

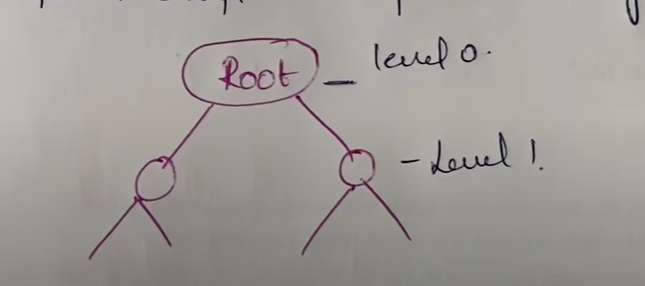
replace attribute value with '?' (Basically generalizing it)

1. If example is Negative example

Make generalize hypothesis more specific.

**HYPOTHESIS SPACE SEARCH IN DECISION TREE LEARNING**

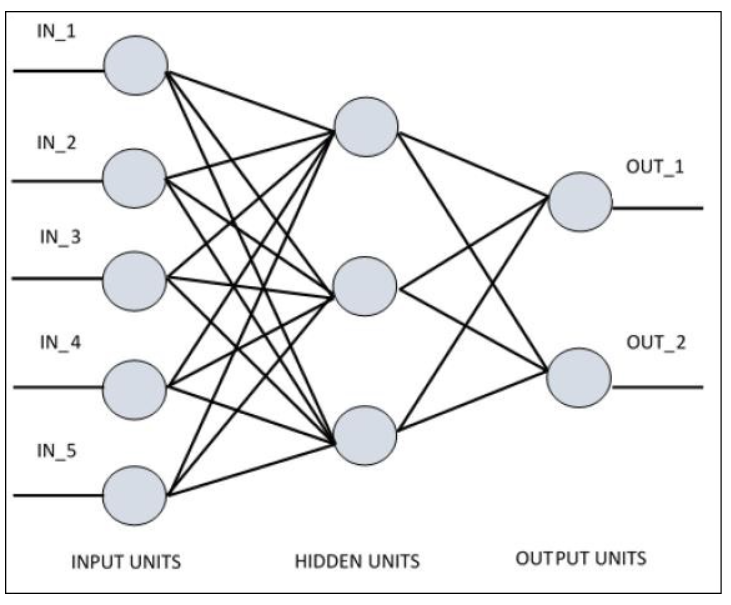
* Applying Concept Learning, Candidate-Elimination algorithm.
* Among all the decision trees available which decision tree should be picked is designed by this concept.
* ID3 can be characterized as searching a space of hypotheses for one that fits the training examples.
* ID3 will search a set of possible decision trees from available hypothesis.
* ID3 performs simple to complex searching, it means first it starts with simple and then goes on increases its complexity based on training examples.

****

* Initially starts with root at each level the complexity gets increased.
  + 1. ID3’s hypothesis space of all decision trees is a complete space of finite discrete-valued functions. Because every finite discrete-valued function can be represented by some decision tree, ID3 avoids one of the major risks of methods that search incomplete hypothesis spaces: that the hypothesis space might not contain the target function.
    2. ID3 maintains only a single current hypothesis as it searches through the space of decision trees.
       - For example, with the earlier version space candidate-eliminate method, which maintains the set of all hypotheses consistent with the available training examples.
    3. ID3 in its pure form performs no backtracking in its search. Once it, selects an attribute to test at a particular level in the tree, it never backtracks to reconsider this choice.
       - By maintaining a single hypothesis it can’t travel to the previous decision tree
       - In the case of ID3, a locally optimal solution corresponds to the decision tree it selects along the single search path it explores.
    4. ID3 uses all training examples at each step in the search to make statistically based decisions regarding how to refine its current hypothesis.
       - This contrasts with methods that make decisions incrementally, based on individual training examples (e.g., FIND-S or CANDIDATE-ELIMINATION).

**Single Perceptron**

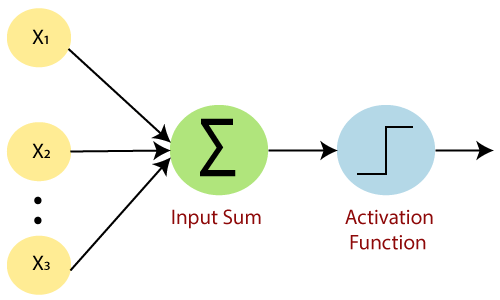
* The perceptron is a single processing unit of any neural network.
* Perceptron, is a simple neuron which is used to classify its input into one or two categories.
* Perceptron is a linear classifier, and is used in supervised learning. It helps to organize the given input data.
* Perceptron is mainly used to classify the data into two parts. Therefore, it is also known as **Linear Binary Classifier.**

****

* Perceptron uses the step function that returns +1 if the weighted sum of its input 0 and -1.
* The activation function is used to map the input between the required value like (0, 1) or (-1, 1).

### **The perceptron consists of 4 parts.**

* **Input value or One input layer**
* **Weights and Bias**
* **Net sum**
* **Activation Function**
* A Single artificial neuron that compute its weight input and uses a threshold activation function
* It is called as Threshold Logic Unit.



* Our goal is to find a linear decision function measured by the weight vector w and the bias parameter b.
* If bias value is 0 then input sum doesn’t work and if it’s 1 then it works.
* The computation of a single layer perceptron is performed over the calculation of sum of the input vector each with the value multiplied by corresponding element of vector of the weights.

Perceptron is a function that maps its input “x,” which is multiplied with the learned weight coefficient; an output value ”f(x)”is generated.

Perceptron_6.

In the equation given above:

* “w” = vector of real-valued weights
* “b” = bias (an element that adjusts the boundary away from origin without any dependence on the input value)
* “x” = vector of input x values

Perceptron_7.

* “m” = number of inputs to the Perceptron

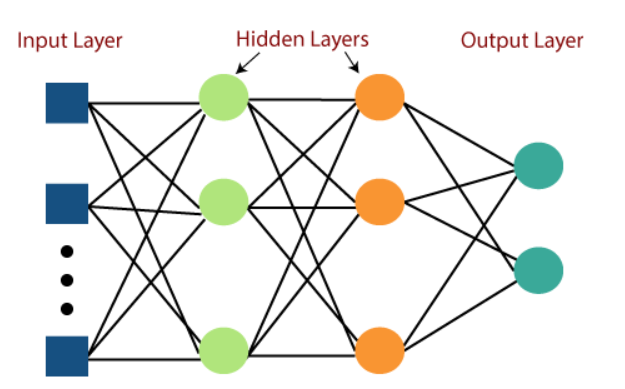
The output can be represented as “1” or “0.”  It can also be represented as “1” or “-1” depending on which activation function is used.

**Back Propagation Algorithm**

* Backpropagation, short for “backward propagation of errors”, is a mechanism used to update the weights using gradient descent.
* When error occurs we go in backward direction.
* Steps involved in back propagation:
  + Step – 1: Forward Propagation
  + Step – 2: Backward Propagation
  + Step – 3: Putting all the values together and calculating the updated weight value

**Multilayer Neural network**

* A multi-layer neural network contains more than one layer of artificial neurons or nodes.
* So basic purpose of multilayer neural network is to control model complexity or capacity.
* The model with less complexity cannot understand training dataset properly means it will underfit (Dumb model) and we get higher error. So, to extract more complex, meaningful, insightful features from training dataset it needs to increase model complexity. So, by using MLP we can easily come up with any complex function to solve our classification or regression problem.
* The complexity of a neural network can be controlled by two aspects of the model:
  + Number of Neurons.
  + Number of Layers.
* The model with more number of neurons per layer and more number of layer has higher representational capacity, in turn, is capable to map any complicated function OR Model is capable to extract any level complex features from out input datasets.
* The number of neurons per layer of model are referred as **Width.**



* MLP networks are used for supervised learning format. A typical learning algorithm for MLP networks is also called **back propagation's algorithm**.