Artificial Intelligence

(Intro. Planning and ANN)

Dr. Priyadarshan S. Dhabe,
Ph.D, IIT Bombay,
Assistant Professor in Information Technology

Syllabus-

- **Planning-** Blocks world, STRIPS, Implementation using goal stack.
- **Introduction to Neural networks:-** basic, comparison of human brain and machine, biological neuron, general neuron model, activation functions, Perceptron learning rule, applications and advantages of neural networks.
- **Brief introduction to single layer and multiplayer networks.**

What is planning?

- **Ref:- Wikipedia**
- **Automated planning and scheduling**, sometimes denoted as simply **AI planning**, is a branch of artificial intelligence that concerns the realization of strategies or **action sequences**, typically for execution by intelligent agents, autonomous robots and unmanned vehicles.
- Planning is one of the aspect of human intelligence, we want to put in to machines.

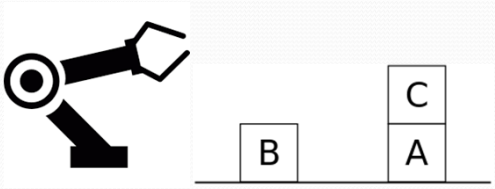
What is planning?

- **Planning definition-**
- Can be broadly defined as the process of computing **several steps** of a problem solving procedure before executing any of them.
- Planning decompose the problem into sub-problems/parts
- These sub-parts
  - interacts with each other
  - Are Ordered to solve big problem

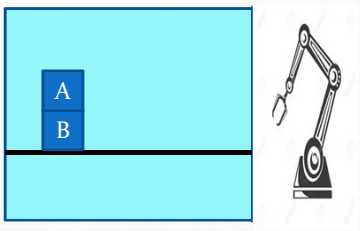
The Blocks world domain

- The domain considered for planning systems
  - There is s flat surface on which blocks can be placed.
  - Each block has exactly same size and identified with an alphabet. There are many such blocks.
  - At most one block can be stacked on the another block, since they are of same size.
  - There is a robot arm that can manipulate the blocks
  - Predicates are used to represent facts in the blocks world

$$ON(\mathrm{A},\mathrm{B}) \wedge ONTABLE(\mathrm{B})$$



The Blocks world domain



Robot arm can perform following actions

1. **UNSTACK(A,B)**- Pick up the block A from its current position on block B. The arm must be empty and no block should be on the top of A
2. **STACK(A,B)**- Place block A on block B. The arm must already be holding block A and the surface of B must be clear
3. **PICKUP(A)**- Pick up block A from the table and hold it. Precondition- arm must be empty and there must be nothing on the top of A

4. PUTDOWN(A)- Put block A down on the table. The arm must be holding block A.

The Blocks world domain

- **Following predicates describe the block world**

  $ON(A, B)$ – block A is on block B

  ONTABLE(B)- block B is on Table

  CLEAR(A)- There is nothing on the top of block A

  HOLDING(A)- Robot arm is currently holding block A

  ARMEMPTY- The arm is holding nothing

- **Various logical statements are true in the blocks world**

  $[\exists x : HOLDING(\mathrm{x})] \rightarrow \neg \mathrm{ARMEMPTY}$

  $\forall x : ONTABLE(x) \rightarrow \neg \exists y : ON(x, y)$

  $\forall x : [\neg \exists y : \mathrm{ON(y, x)}] \rightarrow \mathrm{CLEAR(x)}$

Components of a planning system

- Planning can also be viewed as searching of various operators that can be applied at current state to obtain new state .
- There must be a component to perform following functions
  1. Choose the best rule to apply next based on the best available heuristic information
  2. Apply the chosen rule to compute the new problem state that arises from its application
  3. Detect when a solution has been found
  4. Detect dead ends so that they can be abandoned and the system's effort directed in more fruitful directions.
  5. Detect when an almost correct solution has been found and employ special techniques to make it totally correct.

Components of a planning system

- **Choosing a rule to apply-**
  - if several rules are available, then find the best rule by exploiting the heuristic information. The chosen rule must allow to reduce the difference between current state and the goal state, in planning procedure.
- **Applying rules-**
- Each rule has a **precondition** and **consequent**
- Frame axioms – are axioms that describes the situation after application of the rule, are used.
- STRIPS- a robot problem solving system uses frame axioms

STRIPS-Stanford Research Institute Problem Solver

- **STRIPS** is developed in 1971, by Fikes and Nilsson
- Each robotic arm **operation** is described by list of **new predicates** that the operator causes to become true and a list of **old predicates** that it causes to become false.
- These **two list** are called as **ADD** and **DELETE** lists, respectively.
- A **third list** must also be specified for each operator called **PRECONDITION**. It lists predicates that must be true for the operator to be applied.
- Any predicate **not included** in either **ADD** or **DELETE** is assumed to be **unaffected** by it. Thus, **unrelated things** need not be considered.

STRIPS- Style operators for blocks world domain

STACK(*x*, *y*)

P: CLEAR(y) ∧ HOLDING(x)

D: CLEAR(y) ∧ HOLDING(x)

A: ARMEMPTY ∧ ON(x,y)

UNSTACK(*x*, *y*)

P: ON(x,y) ∧ CLEAR(x) ∧ *ARMEMPTY*

D: ON(x,y) ∧ *ARMEMPTY*

A: HOLDING(x) ∧ CLEAR(y)

PICKUP(*x*)

P: CLEAR(x) ∧ *ONTABLE*(*x*) ∧ *ARMEMPTY*

D: *ONTABLE*(*x*) ∧ *ARMEMPTY*

A: HOLDING(x)

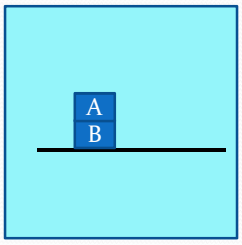
STRIPS- Style operators for blocks world domain

PUTDOWN(*x*)

P: HOLDING(*x*)

D: HOLDING(*x*)

A: ONTABLE(x) ∧ *ARMEMPTY*

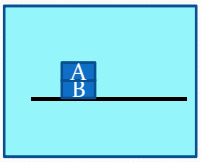


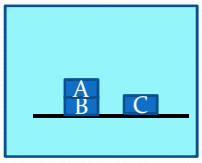
The situation in blocks world can be descrtibed as ON(A,B) ∧ ONTABLE(B) ∧ CLEAR(A)

After applying operator UNSTACK(A,B), the description of the world would be ONTABLE(B) ∧ CLEAR(B) ∧ HOLDING(A)

Components of a planning system

- **Detecting a solution-**
  - Planning procedures transforms a given state into another by application of operators. How will it know it has done?
  - If the states are described with the help of properties, the matter becomes difficult
  - Using predicate P(x), the goal state is described and it is tested for termination of planning procedures.
  - There can be multiple goal states e.g **ON(A,B)**





Components of a planning system

- **Detecting dead ends-**
  - Planning procedures searches for sequence of operators to solve a problem.
  - It must able to detect, when exploring a path that never lead to a solution (or at least unlikely to lead to one). This fact must be true in reasoning forward or backward

Components of a planning system

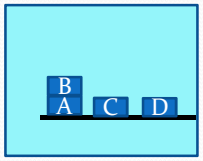
- **Repairing an almost correct solution-**
  - A bigger problem is decomposed into several sub-problems. But combining solutions of several sub-solutions may not yield the solution to original problem. Thus, following alternates are used.
    - 1. Throw that solution- wasted efforts
    - 2. If the difference between two sub-solutions A and B is greater than zero, then call problem solving system to eliminate this difference.
    - 3. Try more appropriate operators
    - 4. Keep the problems as it is till the last moment. Then when as much info as possible is available, then complete the solution (eliminate the difference). This approach is called least commitment strategy.

Goal stack planning

- This approach is used in STRIPS
- The problem solver makes use of a single stack that contains both goals and operators, that has been proposed to satisfy those goals.
- The problem solver relies on a KB that describes the current situation and a set of operators described as PRECONDITION, ADD, DELETE lists

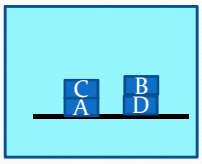
Goal stack planning-working

Start



ON(B,A) ∧
ONTABLE(A) ∧
ONTABLE(C) ∧
ONTABLE(D) ∧
ARMEMPTY

Goal



ON(C,A) ∧
ON(B,D) ∧
ONTABLE(A) ∧
ONTABLE(D)

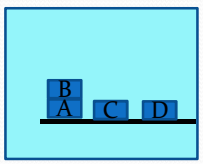
When we start solving this problem , the goal stack is simply

ON(C,A) ∧ *ON*(B,D) ∧ ONTABLE(A) ∧ ONTABLE(D)

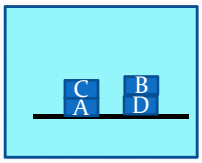
We want to separate this problem into 4 sub problems, one for each component of the original goal

Goal stack planning-working

Start



Goal



ON(C,A) ∧ *ON*(B,D) ∧ ONTABLE(A) ∧ ONTABLE(D)

*T*wo of the sub-problems ONTABLE(A) and ONTABLE(D) are already true in the initial state. Thus, we will work for remaining two

Depending on the order in which we want to solve sub-problems two **goal stacks** are possible.

Let OTAD=ONTABLE(A) ∧ ONTABLE(D)

ON(C,A)
ON(B,D)
ON(C,A) ∧ ON(B,D) ∧ OTAD

Stack 1

ON(B,D)
ON(C,A)
ON(C,A) ∧ ON(B,D) ∧ OTAD

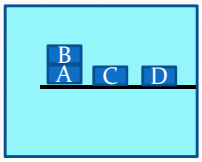
Stack 2

Goal stack planning-working

- At each step of the problem solving process, the top of the stack will be pursued.
- When a sequence of operators that satisfies it is found, that sequence is applied to the state description, yielding new description.
- The process continues till the goal stack is empty.
- The last check is the original goal is compared with the final state derived from the application of chosen operators.

Goal stack planning-working

On the stack we have ON(C,A) which is **not true**. Thus, will find operators that could make it true. Out of 4 operators we are considering (stack, unstack, pickup, putdown) only STACK(C,A) can make it possible and thus replace ON(C,A) by STACK(C,A)



ON(C,A)
ON(B,D)
ON(C,A) ∧ ON(B,D) ∧ OTAD

Our new stack will be

STACK(C,A)
ON(B,D)
ON(C,A) ∧ ON(B,D) ∧ OTAD

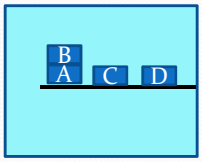
But, in order to apply operator STACK(C,A), its precondition must hold so we must establish them as sub goals

Thus, our new goal stack will be

CLEAR(A)
HOLDING(C)
CLEAR(A) ∧ HOLDING(C)
STACK(C,A)
ON(B, D)
ON(C,A) ∧ ON(B,D) ∧ OTAD

Goal stack planning-working

CLEAR(A)
HOLDING(C)
CLEAR(A) ∧ HOLDING(C)
STACK(C,A)
ON(B, D)
ON(C,A) ∧ ON(B,D) ∧ OTAD



Next we check that CLEAR(A) is not true and only operator that can make it true is UNSTACK(B,A) and thus, our new goal stack will be

UNSTACK(B,A)
HOLDING(C)
CLEAR(A) ∧ HOLDING(C)
STACK(C,A)
ON(B, D)
ON(C,A) ∧ ON(B,D) ∧ OTAD

Apply preconditions of UNSTACK(B,A) which are ON(B,A), CLEAR(B) and ARMEMPTY. Thus our goal stack will be

ON(B,A)
CLEAR(B)
ARMEMTY
HOLDING(C)
CLEAR(A) ∧ HOLDING(C)
STACK(C,A)
ON(B, D)
ON(C,A) ∧ ON(B,D) ∧ OTAD

Now we will compare top of stack ON(B,A) with our model and learn that it is satisfied so we will pop the stack till ARMEMPTY.

Goal stack planning-working

Our goal stack at this point will be

HOLDING(C)
CLEAR(A) ∧ HOLDING(C)
STACK(C,A)
ON(B, D)
ON(C,A) ∧ ON(B,D) ∧ OTAD

We now try to solve the goal HOLDING(C), but it is not true. It can be made true by operator PICKUP(C) and thus add preconditions of PICKUP(C) on stack, which are CLEAR(C)^ONTABLE(C)^ARMEMPTY

Our goal stack at this point will be

ONTABLE(C)
CLEAR(C)
ARMEMPTY
CLEAR(A) ∧ HOLDING(C)
STACK(C,A)
ON(B, D)
ON(C,A) ∧ ON(B,D) ∧ OTAD

ONTABLE(C) and CLEAR(C) are true, thus pop them. But ARMEMPTY is not satisfied since HOLDING(B) is true (due to UNSTACK(B,A)). TO make ARMEMPTY as true we will use STACK(B,D), since it is in the goal. Apply all the preconditions of STACK(B,D) to the stack

CLEAR(D)
HOLDING(B)
CLEAR(D)^HOLDING(B)
STACK(B,D)
CLEAR(A) ∧ HOLDING(C)
STACK(C,A)
ON(B, D)
ON(C,A) ∧ ON(B,D) ∧ OTAD

Goal stack planning-working

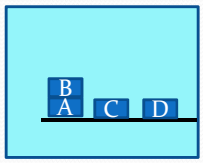
- At some stage our stack will look like

At this point problem solver will halt and produce a plan with following actions in the specified order

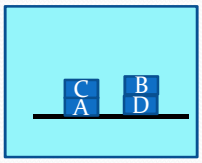
1. UNSTACK(B,A)
2. STACK(B,D)
3. PICKUP(C)
4.STACK(C,A)

CLEAR(D)
UNSTACK(B, A)
CLEAR(D)^HOLDING(B)
STACK(B, D)
ONTABLE(C) ∧ CLEAR(C) ∧ ARMEMPTY
PICKUP(C)
CLEAR(A) ∧ HOLDING(C)
STACK(C, A)
ON(B, D)
ON(C,A) ∧ ON(B,D) ∧ OTAD

Start



Goal



Introduction to Artificial Neural networks

Human Brain

Human brain is massively parallel processor

-Basic computing unit "Neuron"

- A neuron is connected to several other neurons
- The connection is called "synapse" and its property is "synaptic weight"
- Neurons communicates with electrical impulses of short duration with milli-electron-volt
- **Statistics**
- Neurons=10^11
- Synapse= Trillions= 10^18
- - 1 volt=10^-18 ev

Why to study ANN?

- Consider intelligence exhibited by some animals.
- Consider a problem of identifying a photo of a familiar person from a group photo requires considerable time or hours for a supercomputer (using fastest H/W and fastest algorithm). But the Dogs can recognize their masters in 10 m sec from a crowd.
- Who is intelligent?

Why to study ANN?

- Echo locating bat.

-It is a bird, who don't have "eyes" instead echo sensors.

-Mostly active in night.

-Eats insects near by the trees.

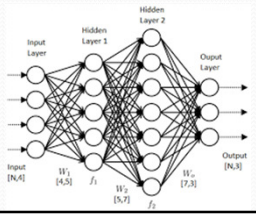
-How they catch their traget?.

-They capture reflection of an emitted rays from target to detect it.

Echo locating bats are several times intelligent than the well designed radar systems in terms of HIT ratios.

Why ANN?

- **Visual and audible pattern recognition is found most difficult with the machines.**
- **Animals are intelligent than the computers, which are inspired from human brain.**
- **This fact motivated scientists to study the brain.**
- **ANN is outcome of such a sincere study.**
- **ANN is a new way of computation called neuro-computing as opposed to digital computing.**
- **ANN system works like human brain that has capability to learn and reason.**



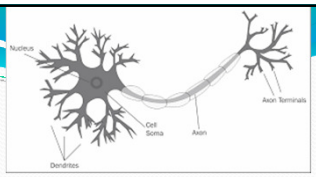
Human brain and Machine

Human Brain	Machine
1. Outperform in cognitive tasks.	1. **Outperform in precise calculatory task.**
2. Basic computation unit (**neuron**) is non-linear.	2. **Basic computational unit is linear.**
3. Switching speed is order of 10^{-3} (msec)	3. **Switching speed is 10^{-9} (pico sec)**
4. Massively parallel processor (10^{11})	4. **Few processor**
5. Consumes less power	5. **Consumes more power.**

Structure Of Human brain

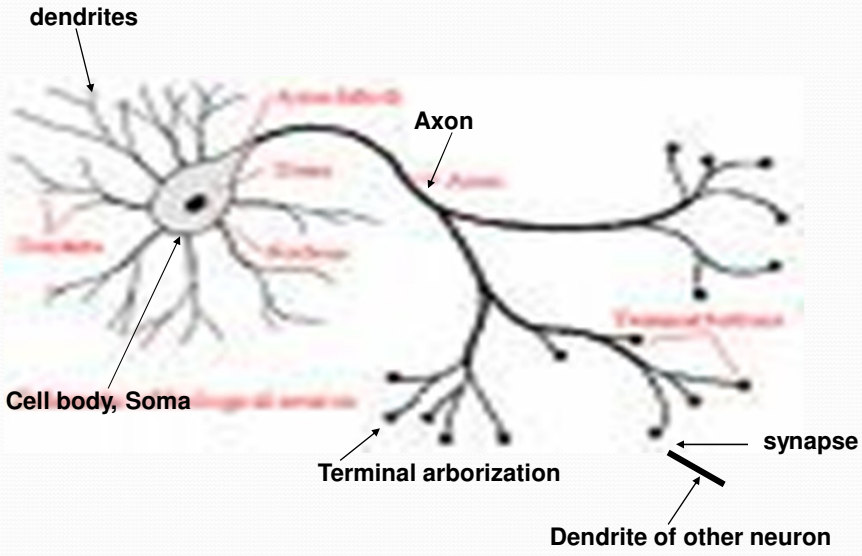
- Human brain consists of 10^{11} computational units called "*neurons*". (which is approx. the number of stars in Milky Way!!) Each neuron is connected to at least 10^4 other neuron with "*synaptic link*".
- Neurons communicate with each other in terms of *electrical impulses* generated by a biochemical process. Thus brain can be considered as densely connected network conditioned by bio-chemical process.

Biological neuron



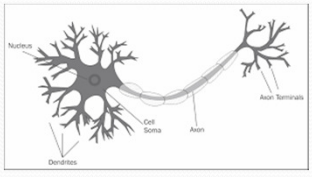
- The elementary nerve cell is called a neuron, which is the fundamental building block of the biological neural network.
- A neuron has three major regions 1) Dendrites 2) Axon and 3) Cell body
- Dendrites forms a dendritic tree, which is fine bush of thin fibers around neuron body (see and draw fig.).
- Dendrites receives information to the neuron and thus are acting as input channels.

Schematic of biological neuron model.

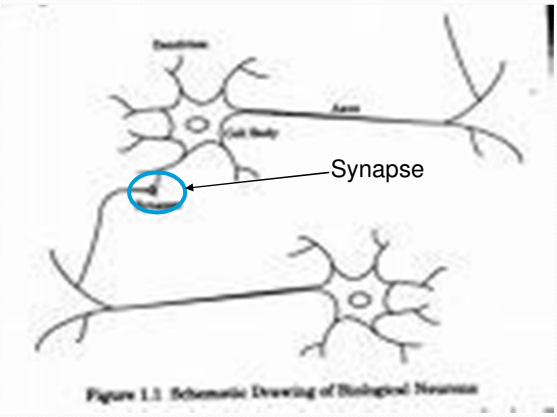


Biological neuron

- Neuron throws its output on Axon, which can be treated as output channel of neuron.
- Dendrites of a neuron are connected to axons of another neighboring neurons.
- End part of Axon splits into fine arborization (arbo-means tree in Latin)
- The axon-dendrite contact organ is called a synapse. The synapse is where the neuron introduces its signal to another neuron.



Biological neuron



Connected neurons

General neuron model

In 1943 Warren S. McCulloch, a neuroscientist, and Walter Pitts, a logician, published "A logical calculus of the ideas immanent in nervous activity" in the Bulletin of Mathematical Biophysics 5: 115-133. In this paper McCulloch and Pitts tried to understand how the brain could produce highly complex patterns by using many basic cells that are connected together. These basic brain cells are called neurons, and McCulloch and Pitts gave a **highly simplified model** of a neuron in their paper.

http://www.mind.ilstu.edu/curriculum/modOverview.php?modGUI=212

General neuron Model

- In, general neuron model input is given as vector **x**, weights are represented with vector **w** and output **o** is function of net.
- The **net input, net** or **internal activity level** of neuron is calculated as

$$net = \sum_{i=1}^{n} w_i x_i$$

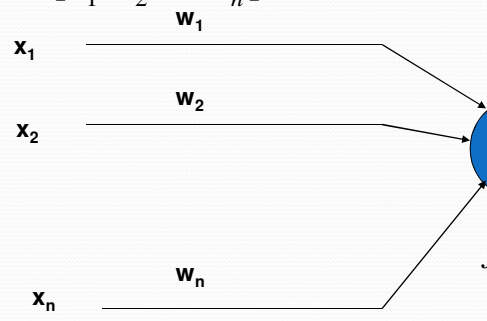
General neuron Model

$w = [w_1, w_2, \ldots, w_n]$ weight vector

$x = [x_1, x_2, \ldots, x_n]$ Input vector

$w_i = any \quad value$

$for \quad i = 1, 2, \ldots, n$



Transfer function

$f(.) - Activation\ function$

$$net = \sum_{i=1}^{n} w_i x_i$$

$$O = f(net)$$

Activation functions

- Based on sign of output they are of two type
  - Unipolar (either positive or negative)
  - Bi-ploar (both positive and negative)
- Based on nature of function they are of two type
  - Continuous
  - Discrete
- Thus unipolar continuous, bipolar continuous, unipolar discrete and bipolar discrete are possible types.

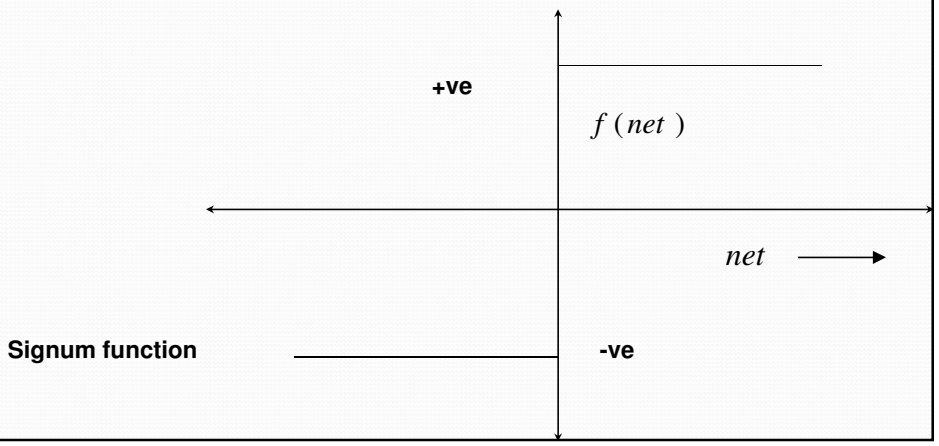
Activation functions

- General neuron model is commonly used in ANN literature but can use different activation functions.
- Some of the activation functions are shown below.

Bipolar Binary-Activation function

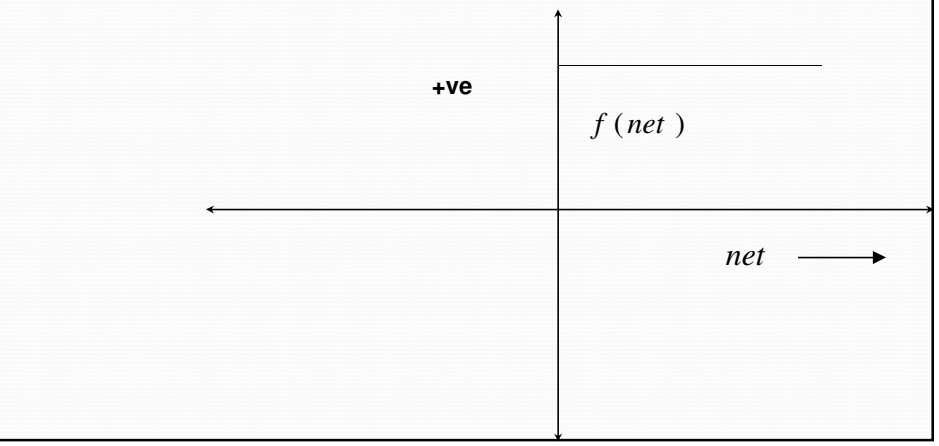
$$f(net) = \mathrm{sgn}(net) = \begin{cases} +1 & if\ net > 0 \\ -1 & if\ net < 0 \end{cases}$$

Undefined for net=0



Unipolar Binary-Activation function

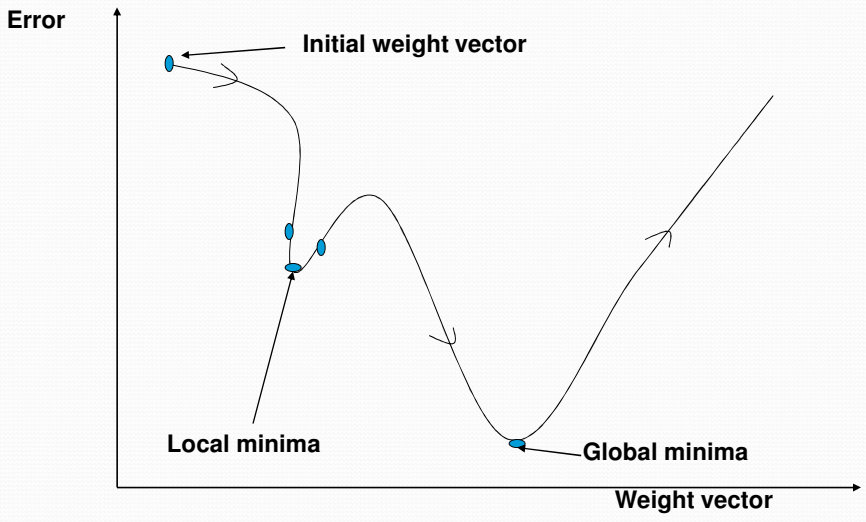
$$f(net) = \begin{cases} +1 & if\ net > 0 \\ 0 & if\ net < 0 \end{cases}$$



How neuron learns?

- It learns by adjustment of its weights. Weights are adjusted to reduce the error in desired output and actual output of a neuron.
- Various learning rules are used to update the weights based on mode of learning i.e supervised or unsupervised learning

Philosophy of Learning



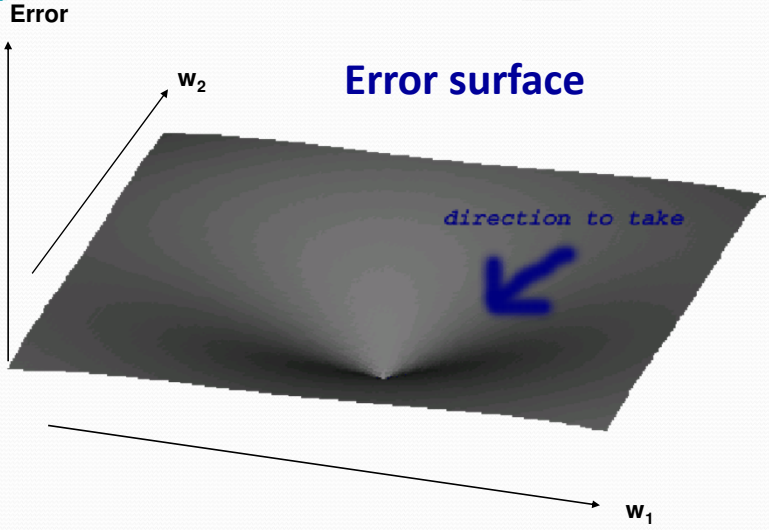


Figure : Hypothetical 2-D error surface

Perceptron Learning Rule

- Invented by Rosenblatt (A psychologist) in 1958. It is the first neural network.
- It is supervised.
- Learning signal (r) is the difference between desired output and actual output. $r=(d_k-o_k)$
- This rule can be stated as

$$W^{k+1} = W^k + \Delta W$$

$$W^{k+1} = W^k + c * r * x^k$$

$$\boxed{W^{k+1} = W^k + c * (d_k - o_k) * x^k}$$

Perceptron Learning Rule

$$W^{k+1} = W^k + \Delta W$$

$$W^{k+1} = W^k + c * r * x^k$$

$$W^{k+1} = W^k + c * (d_k - o_k) * x^k$$

C- learning rate >0, user defined parameter

Extreme cases of C

1. C too large- we will not get the optimal weight vector
2. C too less- We will require more training time

Extreme values of C and learning

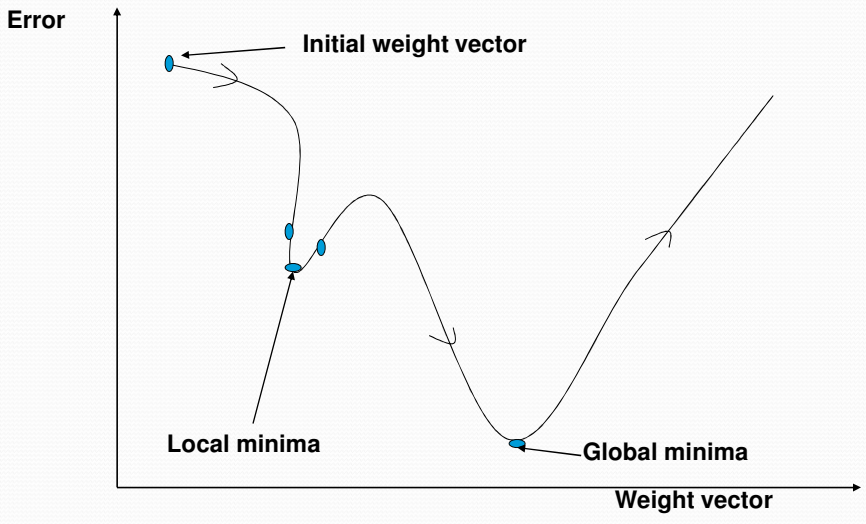
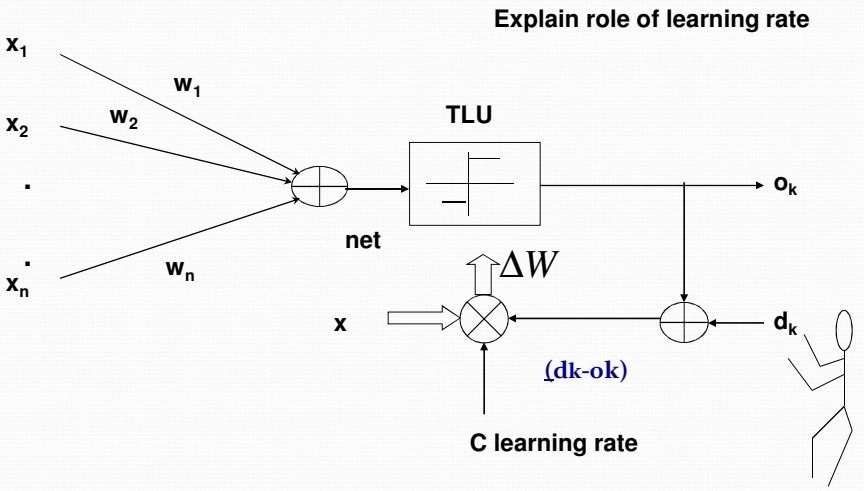


Diagram of Perceptron LR



Features of Perceptron LR

- Learning:-Supervised.
- Weight initialized:- ANY
- Neuron type:- Binary (uni and bipolar)
- Used:- Classification and recognition

Single Perceptron Learning problem

Initial weight vector $W^0=[1,-1,0,0.5]$

Use Perceptron Learning rule to update weights for following three inputs use c=1 and *signum* function if the desired outputs are 1,-1 and 1, respectively

$$f(net)=\mathrm{sgn}(net)=\begin{cases}+1 & if\ net>0\\ -1 & if\ net<0\end{cases}$$

$$net=\sum_{i=1}^{n} w_i x_i$$

$$O=f(net)$$

$X_1=[1,-2,1.5,0]$

$X_2=[1,-0.5,-2,-1.5]$

$X_3=[0,1,-1,1.5]$

$$W^{k+1}=W^k+c*(d_k-o_k)*x^k$$

Answer

1. W1=[1,-1,0,0.5]
2. W2=[-1,0,4.0, 3.5]
3. W3=[-1.0,0,4.0, 3.5]

Applications of neural network

- Pattern classification/ Recognition/clustering
- Prediction and forecasting
- Fault diagnosis.
- Correlation.

Advantages of neural networks

1.Capability to learn and learn on fly.

2. Superior for cognitive task.

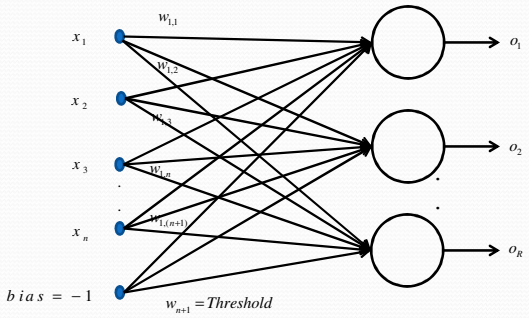
3. Fault tolerant ability.

Reference Book:-

Title:- "*An introduction to Artificial neural systems*" – By Jacek M. Zurada, Jaico Publication

Single layer Perceptron Classifiers

$w_{i,j}$ - weight connecting j^{th} input to the i^{th} neuron



Representation of pattern classes

1. Local :- each neuron represents a class
2. Global:- a specific output of all the neurons represents a class

Weight matrix W is of size R x (n+1)

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,(n+1)} \\ w_{2,1} & w_{2,2} & \dots & w_{2,(n+1)} \\ .. & .. & .. & .. \\ w_{R,1} & w_{R,2} & \dots & w_{R,(n+1)} \end{bmatrix}$$

Desired output d_i is of size (R x 1)

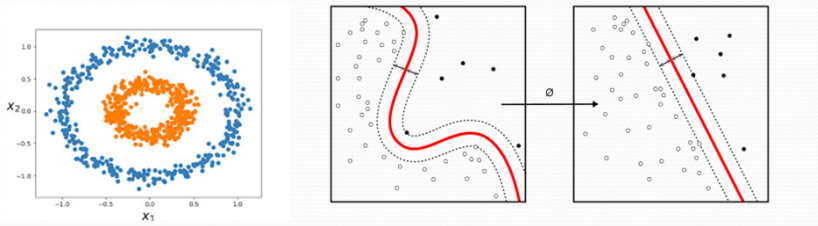
$$d_i = \begin{bmatrix} d_{i,1} \\ d_{i,2} \\ .. \\ d_{i,R} \end{bmatrix}$$

R- class Single layer discrete Perceptron training algorithm

- Given P training pairs $\{x_1, d_1,\ x_2, d_2,\ ..., x_p, d_p\}$ where x_i is $(n \times 1)$, d_i is $(R \times 1)$, $i = 1, ..., P$
- The augmented input vectors are $y_i = \begin{bmatrix} x_i \\ -1 \end{bmatrix}$, for $i = 1, ..., P$
- In the following, k is training step and p is step counter within training cycle.
  1. Choose $c > 0$
  2. Initialized weights at small random values, $W = [w_{ij}]$ is $R \times (n+1)$
  3. Initialize counters and error: $k \longleftarrow 1,\ p \longleftarrow 1, E \longleftarrow 0$
  4. Training cycle begins here. Set $y \longleftarrow y_p, d \longleftarrow d_p,\ o_i = sgn(w_i^T y)$ for $i = 1, ..., R$ (sgn is sign function)
  5. Update weights $w_i \longleftarrow w_i + \frac{1}{2}c(d_i - o_i)y$ for $i = 1, ..., R$
  6. Find error: $E \longleftarrow \frac{1}{2}(d_i - o_i)^2 + E$ for $i = 1, ..., R$
  7. If $p < P$ then $p \longleftarrow p + 1, k \longleftarrow k + 1$, go to step 4, otherwise, go to step 8.
  8. If $E = 0$ the training is terminated, otherwise $E \longleftarrow 0, p \longleftarrow 1$ go to step 4 for new training cycle

R- class Single layer discrete Perceptron training algorithm

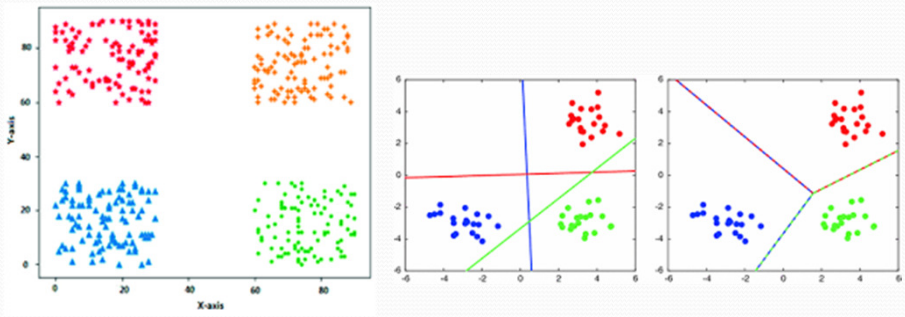
- Pattern classes must be pairwise linearly separable



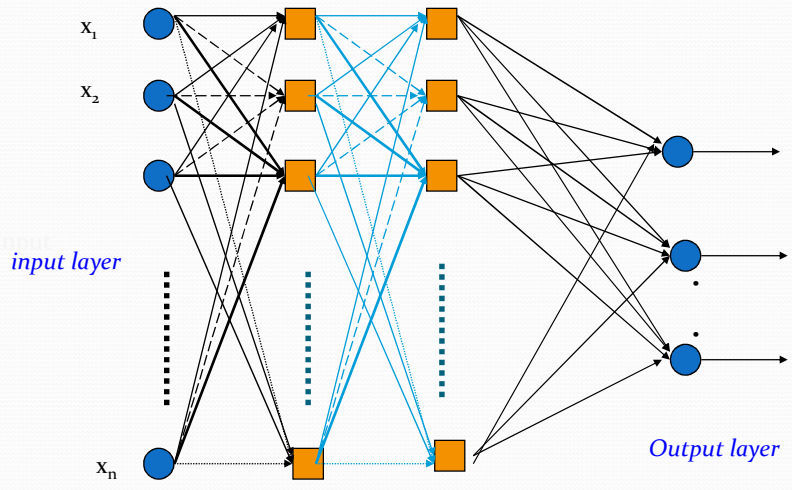
Non-linearly separable **Linearly separable**

- For non-linearly separable patterns we need multi-layer neural networks

Example scatterplots of pairwise linearly separable classes



Architecture of Multi-layer perceptron



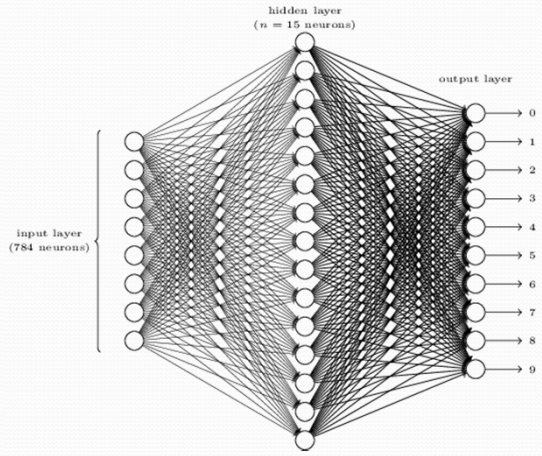
Hidden layers

Error back propagation training algorithm is used

Reference- http://neuralnetworksanddeeplearning.com/chap2.html

58

A sample 3 –layer neural network architecture for handwritten digit recognition



Reference:- http://neuralnetworksanddeeplearning.com/chap1.html

Deep Learning Neural networks

- Neural networks/Architectures with thousands of hidden layers are called "**deep architectures**" and their learning is called "**deep Learning**" (proposed in **1967**)
- They need considerable computing power and thus generally implemented on **high performance computing platform (>1 TFLOPS)** with many CPUs and GPUs
- Human brain is itself a deep architecture which is motivation behind deep learning.
- Deep learning machines exhibited **super human** capabilities for a specific **cognitive tasks**. (NVIDIA DIGITS)
- **Tensor Processing Unit (TPU)** is designed specifically for deep learning architectures with library "**Tensor Flow**". TPU is highly optimized for tensor operations.
- Some new NVIDIA GPUs also provide "**Tensor Cores**" for accelerating tensor operations on GPU (e.g **Tesla V100**, has 640 tensor cores and provides 100 TFLOPS)