

CS217: Artificial Intelligence and Machine Learning (associated lab: CS240)

Pushpak Bhattacharyya
CSE Dept.,
IIT Bombay

*Week3 of 20jan25, Monotone Restriction,
Perceptron*

Main points covered: week2 of
13jan25

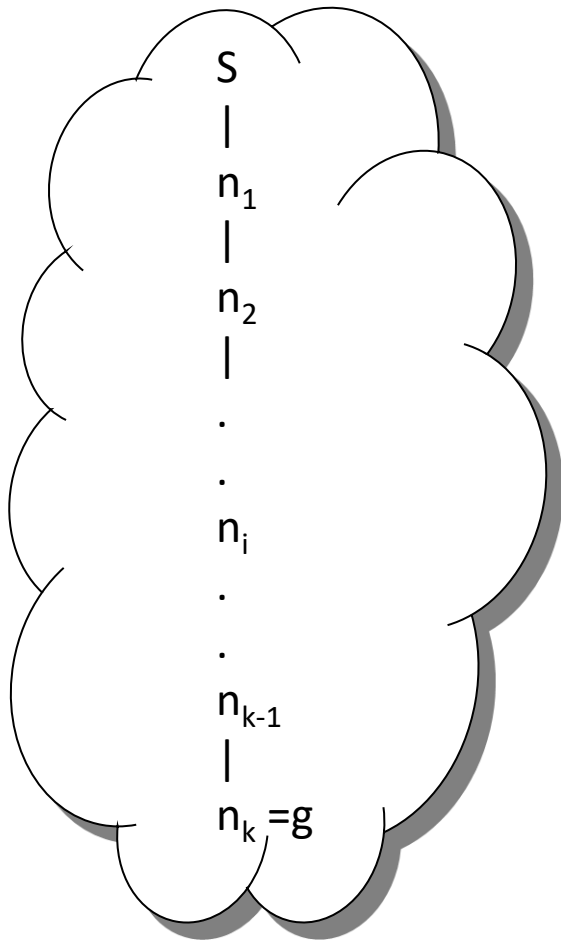
Key point about A* search

Statement:

Let $S - n_1 - n_2 - n_3 \dots n_i \dots - n_{k-1} - n_k (=G)$ be an optimal path.

At any time during the search:

1. There is a node n_i from the optimal path in the OL
2. For n_i all its ancestors $S, n_1, n_2, \dots, n_{i-1}$ are in CL
3. $g(n_i) = g^*(n_i)$



Admissibility of A*

- 1. A* algorithm halts
- 2. A* algorithm finds optimal path
- 3. If $f(n) < f^*(S)$ then node n has to be expanded before termination
- 4. If A* does not expand a node n before termination then $f(n) \geq f^*(S)$

Better heuristic performs better

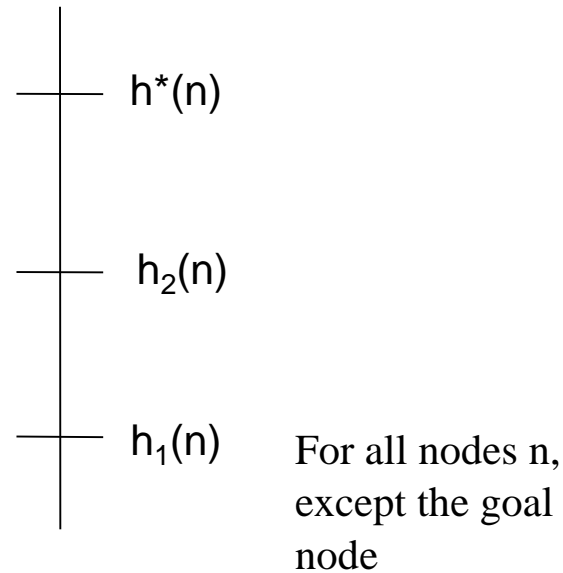
A version A_2^* of A^* that has a “better” heuristic than another version A_1^* of A^* performs at least “as well as” A_1^*

Meaning of “better”

$h_2(n) > h_1(n)$ for all n

Meaning of “as well as”

A_1^* expands at least all the nodes of A_2^*



Foundational Ideas

- Church Turing Hypothesis
- Physical Symbol System Hypothesis
- Uncomputability
- NP-completeness and NP-hardness
- AI is multidisciplinary
- Difference between brain computation and Turing Machine



End of main points

Monotonicity

Steps of GGS

(principles of AI, Nilsson,)

- 1. Create a search graph G , consisting solely of the start node S ; put S on a list called $OPEN$.
- 2. Create a list called $CLOSED$ that is initially empty.
- 3. Loop: if $OPEN$ is empty, exit with failure.
- 4. Select the first node on $OPEN$, remove from $OPEN$ and put on $CLOSED$, call this node n .
- 5. if n is the goal node, exit with the solution obtained by tracing a path along the pointers from n to s in G . (ointers are established in step 7).
- 6. Expand node n , generating the set M of its successors that are not ancestors of n . Install these memes of M as successors of n in G .

GGs steps (contd.)

- 7. Establish a pointer to n from those members of M that were not already in G (*i.e.*, not already on either *OPEN* or *CLOSED*). Add these members of M to *OPEN*. For each member of M that was already on *OPEN* or *CLOSED*, decide whether or not to redirect its pointer to n . For each member of M already on *CLOSED*, decide for each of its descendants in G whether or not to redirect its pointer.
- 8. Reorder the list *OPEN* using some strategy.
- 9. Go *LOOP*.

Illustration for CL parent pointer redirection recursively

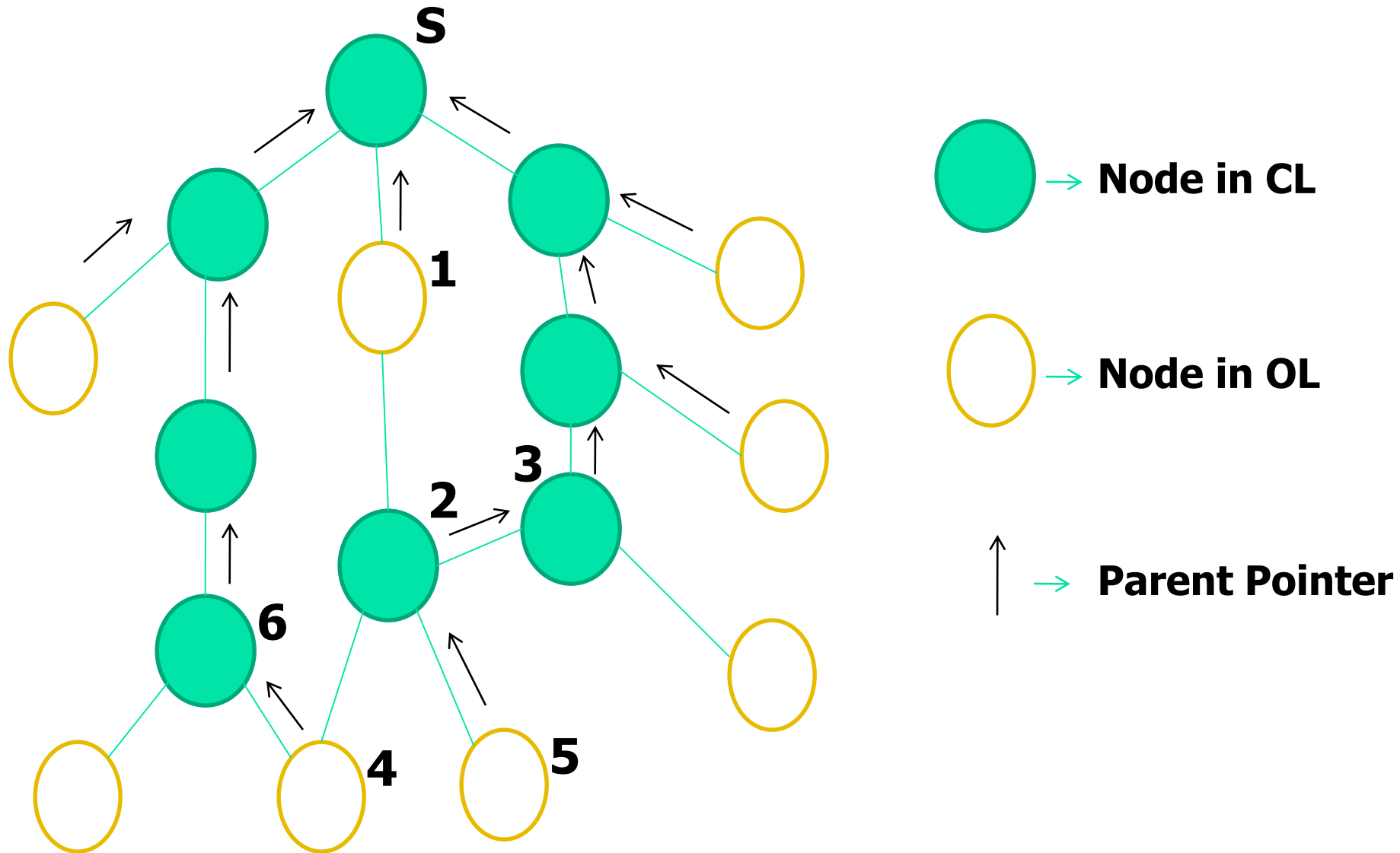
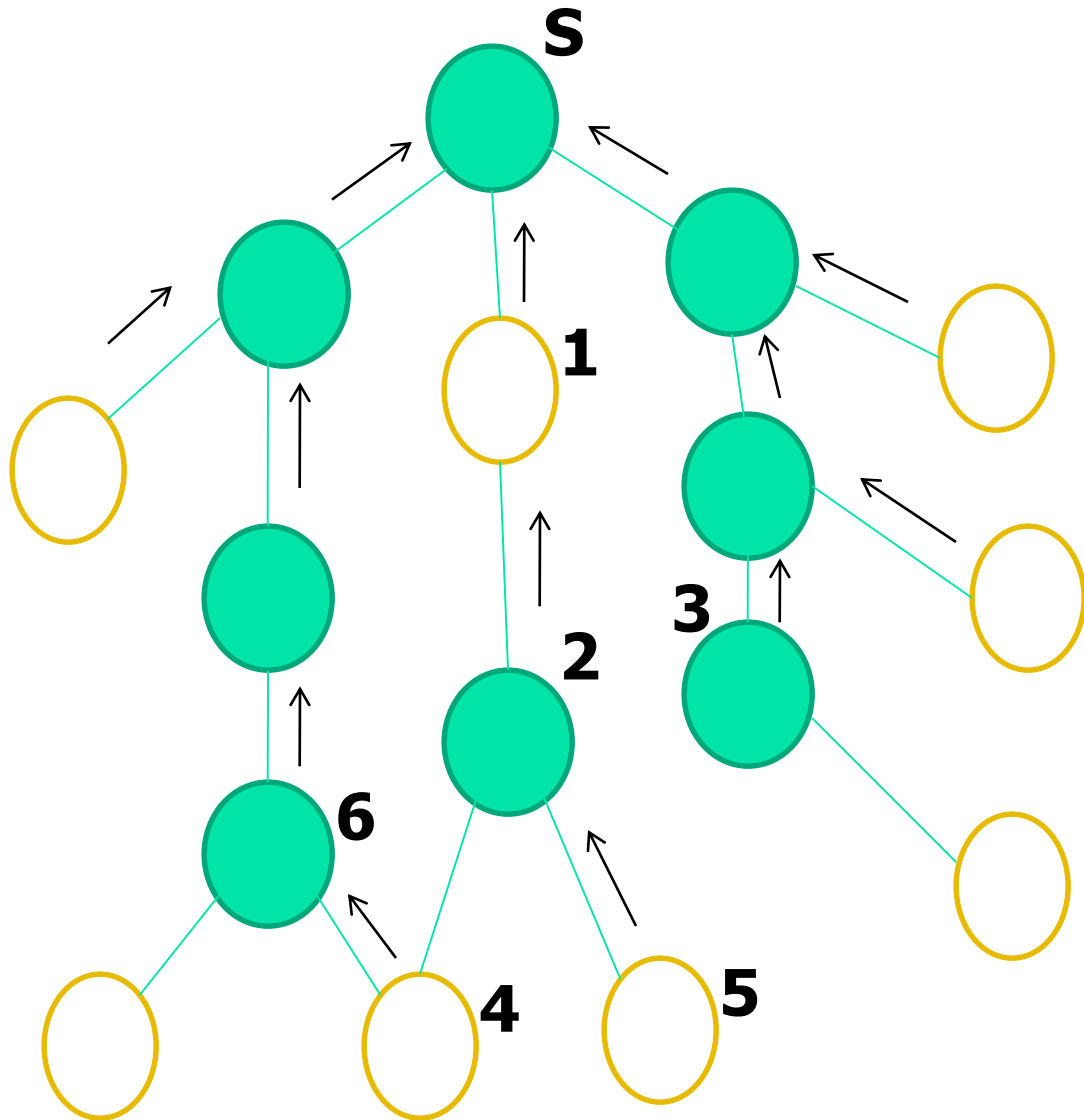


Illustration for CL parent pointer redirection recursively

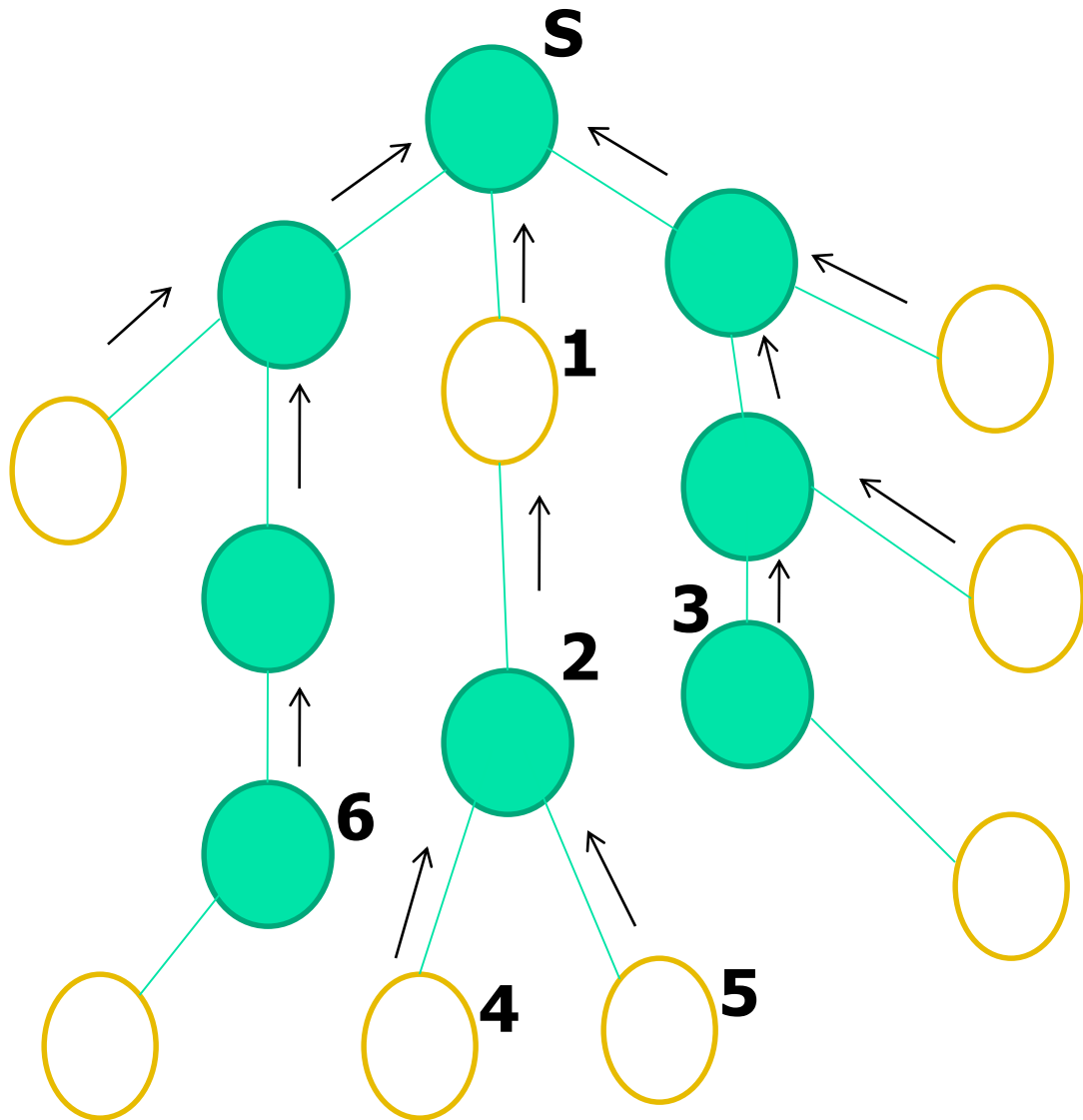


Stage 1 :

Parent Pointer change from

2 - > 3 (Cost = 4)
to
2 - > 1 (Cost = 2)

Illustration for CL parent pointer redirection recursively

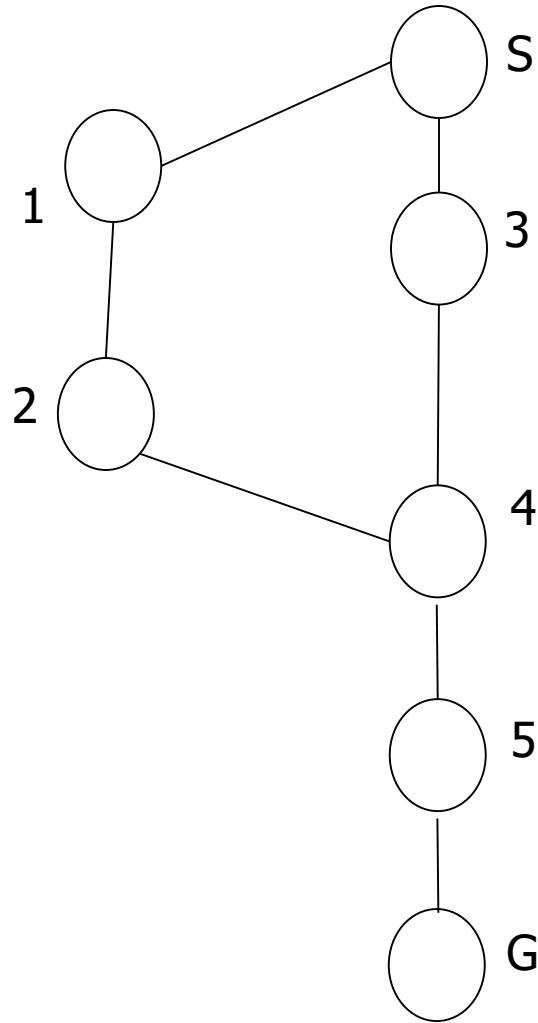


Stage 2 :

Parent Pointer change from

4 - > 6 (Cost = 4)
to
4 - > 2 (Cost = 3)

Another graph



Each arc cost
1 unit

$h=0$ for all nodes,
Except 3, which is
3, i.e., $h(3)=3$

3 violates MR
 $h(3)=3$
 $h(4)=0$

Sequence of expansions:
S-1-2-4-...

Definition of monotonicity

- A heuristic $h(p)$ is said to satisfy the monotone restriction, if for all ' p ',
 $h(p) \leq h(p_c) + cost(p, p_c)$, where ' p_c ' is the child of ' p '.

Theorem

- If monotone restriction (also called triangular inequality) is satisfied, then for nodes in the closed list, redirection of parent pointer is not necessary. In other words, if any node ' n ' is chosen for expansion from the open list, then $g(n)=g^*(n)$, where $g(n)$ is the cost of the path from the start node ' s ' to ' n ' at that point of the search when ' n ' is chosen, and $g^*(n)$ is the cost of the optimal path from ' s ' to ' n '

Grounding the Monotone Restriction

7	3	
1	2	4
8	5	6

n



7	3	4
1	2	
8	5	6

n'

1	2	3
4	5	6
7	8	

G

$h(n)$ -: number of displaced tiles

Is $h(n)$ monotone ?

$$h(n) = 8$$

$$h(n') = 8$$

$$C(n, n') = 1$$

Hence monotone

Monotonicity of # of Displaced Tile Heuristic

- $h(n) \leq h(n') + c(n, n')$
- Any move changes $h(n)$ by at most 1
- $c = 1$
- Hence, $h(\text{parent}) \leq h(\text{child}) + 1$
- If the empty cell is also included in the cost, then h need not be monotone (try!)

Monotonicity of Manhattan Distance Heuristic (1/2)

- *Manhattan distance* = *X-dist* + *Y-dist* from the target position
- Refer to the diagram in the first slide:
- $h_{mn}(n) = 1 + 1 + 1 + 2 + 1 + 1 + 2 + 1 = 10$
- $h_{mn}(n') = 1 + 1 + 1 + 3 + 1 + 1 + 2 + 1 = 11$
- $Cost = 1$
- Again, $h(n) \leq h(n') + c(n, n')$

Monotonicity of Manhattan Distance Heuristic (2/2)

- Any move can either increase the h value or decrease it by **at most 1**.
- Cost again is 1.
- Hence, this heuristic also satisfies Monotone Restriction
- If empty cell is also included in the cost then manhattan distance does not satisfy monotone restriction (try!)
- Apply this heuristic for Missionaries and Cannibals problem

Relationship between Monotonicity and Admissibility

- Observation:

Monotone Restriction \rightarrow Admissibility
but not vice-versa

- Statement: *If $h(n_i) \leq h(n_j) + c(n_i, n_j)$
for all i, j
then $h(n_i) \leq h^*(n_i)$ for all i*

Proof of Monotonicity \rightarrow admissibility

Let us consider the following as the optimal path starting with a node $n = n_1 - n_2 - n_3 \dots n_i - \dots n_m = g_i$

Observe that

$$h^*(n) = c(n_1, n_2) + c(n_2, n_3) + \dots + c(n_{m-1}, g_i)$$

Since the path given above is the optimal path from n to g_i

Now,

$$h(n_1) \leq h(n_2) + c(n_1, n_2) \text{ ----- Eq 1}$$

$$h(n_2) \leq h(n_3) + c(n_2, n_3) \text{ ----- Eq 2}$$

$$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$$

$$h(n_{m-1}) = h(g_i) + c(n_{m-1}, g_i) \text{ ----- Eq (m-1)}$$

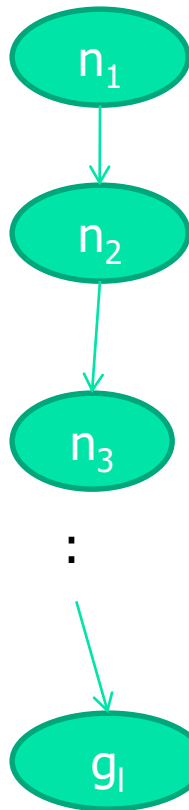
Adding Eq 1 to Eq (m-1) we get

$$h(n) \leq h(g_i) + h^*(n) = h^*(n)$$

Hence proved that $MR \rightarrow (h \leq h^*)$

Proof (continued...)

Counter example for vice-versa



$$h^*(n_1) = 3$$

$$h(n_1) = 2.5$$

$$h^*(n_2) = 2$$

$$h(n_2) = 1.2$$

$$h^*(n_3) = 1$$

$$h(n_3) = 0.5$$

\vdots

\vdots

$$h^*(g_1) = 0$$

$$h(g_1) = 0$$

$h < h^*$ everywhere but MR is not satisfied

Proof of MR leading to optimal path for every expanded node (1/2)

Let $S-N_1-N_2-N_3-N_4\ldots N_m\ldots N_k$ be an optimal path from S to N_k (all of which might or might not have been explored). Let N_m be the **last** node on this path which is on the open list, i.e., all the ancestors from S up to N_{m-1} are in the closed list.

For every node N_p on the optimal path,

$g^*(N_p)+h(N_p)\leq g^*(N_p)+C(N_p, N_{p+1})+h(N_{p+1})$, by monotone restriction
 $g^*(N_p)+h(N_p)\leq g^*(N_{p+1})+h(N_{p+1})$ on the optimal path
 $g^*(N_m)+h(N_m)\leq g^*(N_k)+h(N_k)$ by transitivity

Since all ancestors of N_m in the optimal path are in the closed list,

$g(N_m)=g^*(N_m)$.
 $\Rightarrow f(N_m)=g(N_m)+h(N_m)=g^*(N_m)+h(N_m)\leq g^*(N_k)+h(N_k)$

Proof of MR leading to optimal path for every expanded node (2/2)

Now if N_k is chosen in preference to N_m

$$\begin{aligned}f(N_k) &\leq f(N_m) \\g(N_k) + h(N_k) &\leq g(N_m) + h(N_m) \\&= g^*(N_m) + h(N_m) \\&\leq g^*(N_k) + h(N_k) \\g(N_k) &\leq g^*(N_k)\end{aligned}$$

But $g(N_k) \geq g^*(N_k)$, by definition

Hence $g(N_k) = g^*(N_k)$

This means that if N_k is chosen for expansion, the optimal path to this from S has already been found.

The key point here is that if MR is satisfied, nodes in an optimal path have to get expanded in the order of their distance from the start node.

TRY proving by induction on the length of optimal path

Monotonicity of $f(.)$ values

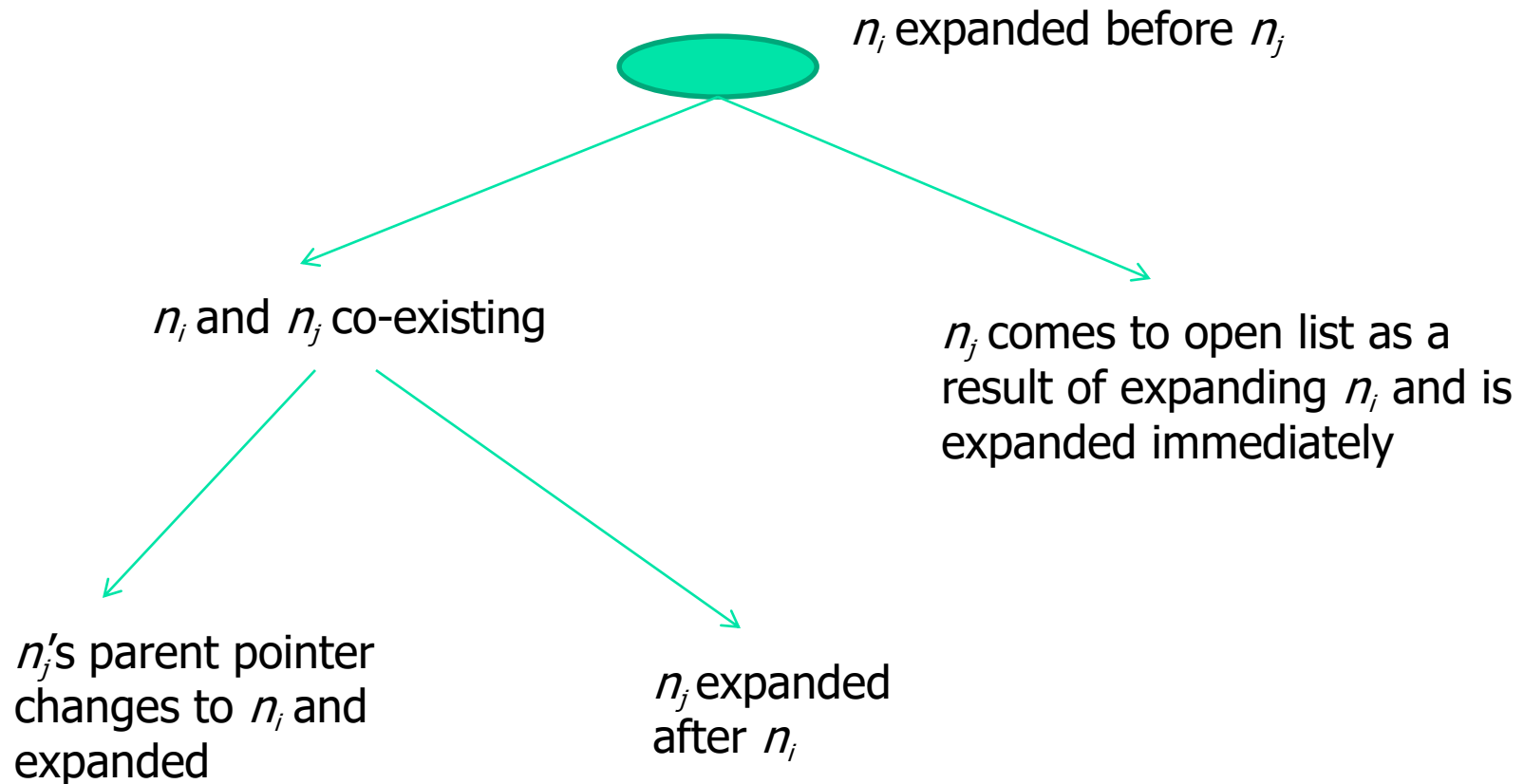
Statement:

f values of nodes expanded by A^* increase monotonically, if h is monotone.

Proof:

Suppose n_i and n_j are expanded with temporal sequentiality, *i.e.*, n_j is expanded after n_i

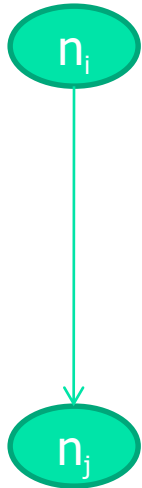
Proof (1/3)...



Proof (2/3)...

- All the previous cases are forms of the following two cases (think!)
- CASE 1:
 n_j was on open list when n_i was expanded
Hence, $f(n_i) \leq f(n_j)$ by property of A^*
- CASE 2:
 n_j comes to open list due to expansion of n_i

Proof (3/3)...



Case 2:

$$f(n_i) = g(n_i) + h(n_i) \quad (\text{Defn of } f)$$

$$f(n_j) = g(n_j) + h(n_j) \quad (\text{Defn of } f)$$

$$f(n_i) = g(n_i) + h(n_i) = g^*(n_i) + h(n_i) \quad \text{---Eq 1}$$

(since n_i is picked for expansion n_i is on optimal path)

With the similar argument for n_j we can write the following:

$$f(n_j) = g(n_j) + h(n_j) = g^*(n_j) + h(n_j) \quad \text{---Eq 2}$$

Also,

$h(n_i) \leq h(n_j) + c(n_i, n_j) \quad \text{---Eq 3 (Parent- child relation)}$

$$g^*(n_j) = g^*(n_i) + c(n_i, n_j) \quad \text{---Eq 4 (both nodes on optimal path)}$$

From Eq 1, 2, 3 and 4

$$f(n_i) \leq f(n_j)$$

Hence proved.

Better way to understand monotonicity of $f()$

- Let $f(n_1), f(n_2), f(n_3), f(n_4) \dots f(n_{k-1}), f(n_k)$ be the f values of k expanded nodes.
- The relationship between two consecutive expansions $f(n_i)$ and $f(n_{i+1})$ nodes always remains the same, *i.e.*, $f(n_i) \leq f(n_{i+1})$
- This is because
 - $f(n_i) = g(n_i) + h(n_i)$ and
 - $g(n_i) = g^*(n_i)$ since n_i is an expanded node (proved theorem) and this value cannot change
 - $h(n_i)$ value also cannot change Hence nothing after n_{i+1} 's expansion can change the above relationship.

Monotonicity of $f()$

$f(n_1), f(n_2), f(n_3), \dots, f(n_i), f(n_{i+1}), \dots, f(n_k)$

Sequence of expansion of $n_1, n_2, n_3 \dots n_i \dots n_k$

f values increase monotonically

$$f(n) = g(n) + h(n)$$

Consider two successive expansions - $> n_i, n_{i+1}$

Case 1:

n_i & n_{i+1} Co-existing in OL

n_i precedes n_{i+1}

By definition of A^*

$$f(n_i) \leq f(n_{i+1})$$

Monotonicity of $f()$

Case 2:

n_{i+1} came to OL because of expanding n_i and n_{i+1} is expanded

$$\begin{aligned} f(n_i) &= g(n_i) + h(n_i) \\ &\leq g(n_i) + c(n_i, n_{i+1}) + h(n_{i+1}) \\ &= g(n_i) + h(n_{i+1}) \\ &= f(n_{i+1}) \end{aligned}$$

Case 3:

n_{i+1} becomes child of n_i after expanding n_i and n_{i+1} is expanded. Same as case 2.

A list of AI Search Algorithms

- A*
 - AO*
 - IDA* (Iterative Deepening)
- Minimax Search on Game Trees
- Viterbi Search on Probabilistic FSA
- Hill Climbing
- Simulated Annealing
- Gradient Descent
- Stack Based Search
- Genetic Algorithms
- Memetic Algorithms

Foundational Points

Symbolic AI

Connectionist AI is contrasted with Symbolic AI

Symbolic AI - Physical Symbol System Hypothesis

Every intelligent system can be constructed by storing and processing symbols and nothing more is necessary.

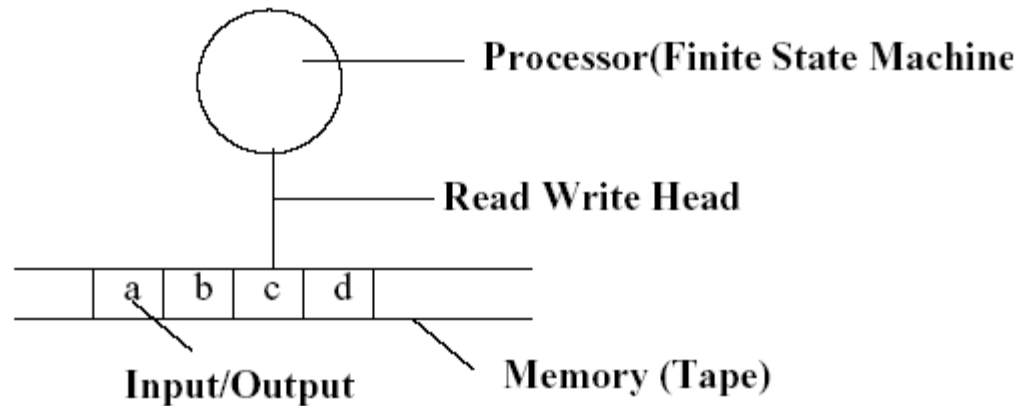
Symbolic AI has a bearing on models of computation such as

Turing Machine

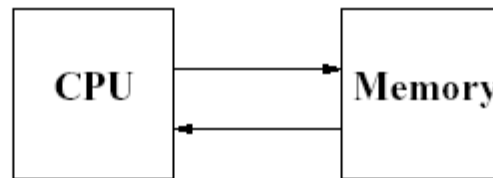
Von Neumann Machine

Lambda calculus

Turing Machine & Von Neumann Machine



Turing machine



VonNeumann Machine

Challenges to Symbolic AI

Motivation for challenging Symbolic AI

A large number of computations and information process tasks that living beings are comfortable with, are not performed well by computers!

The Differences

Brain computation in living beings computers

Pattern Recognition

Learning oriented

**Distributed & parallel processing
processing**

Content addressable

TM computation in

Numerical Processing

Programming oriented

Centralized & serial

Location addressable

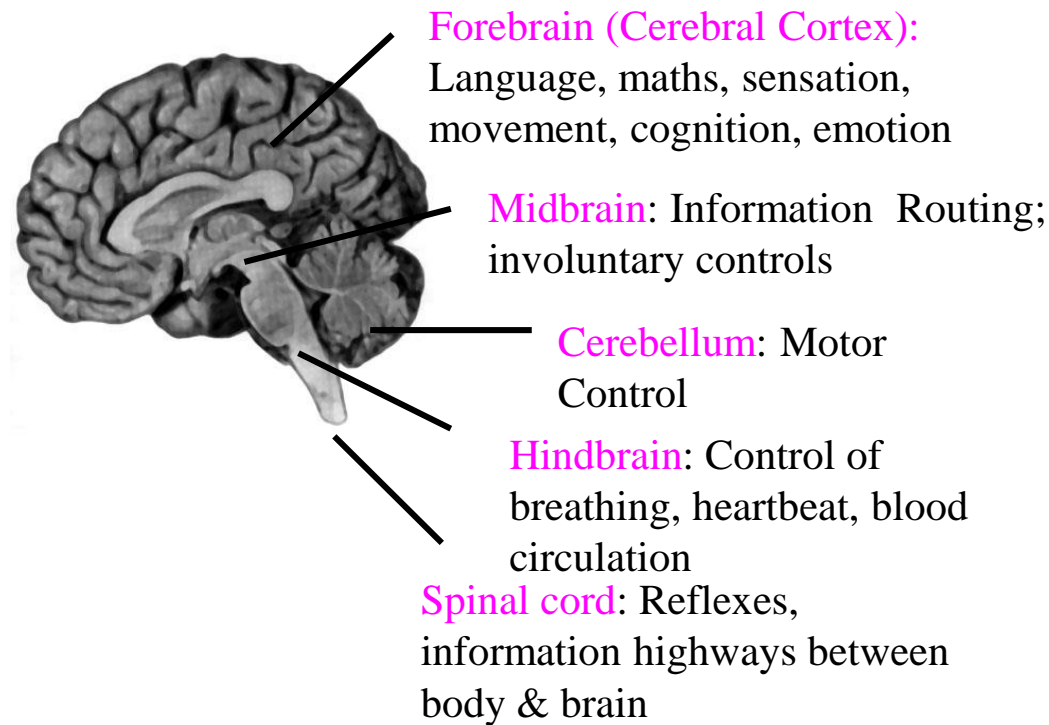
The human brain



Seat of consciousness and cognition

Perhaps the most complex information processing machine in nature

Beginner's Brain Map



Brain : a computational machine?

Information processing: brains vs computers

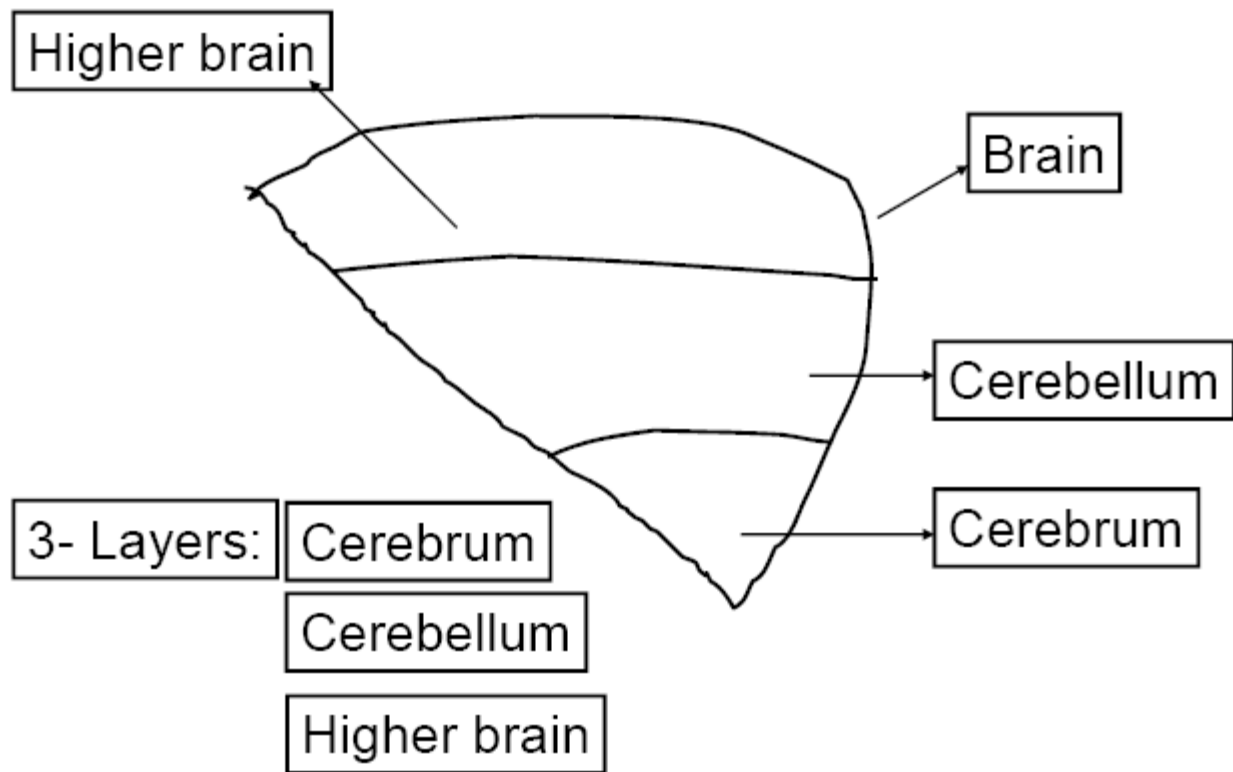
- brains better at perception / cognition
- slower at numerical calculations
- parallel and distributed Processing
- associative memory

Brain : a computational machine? *(contd.)*

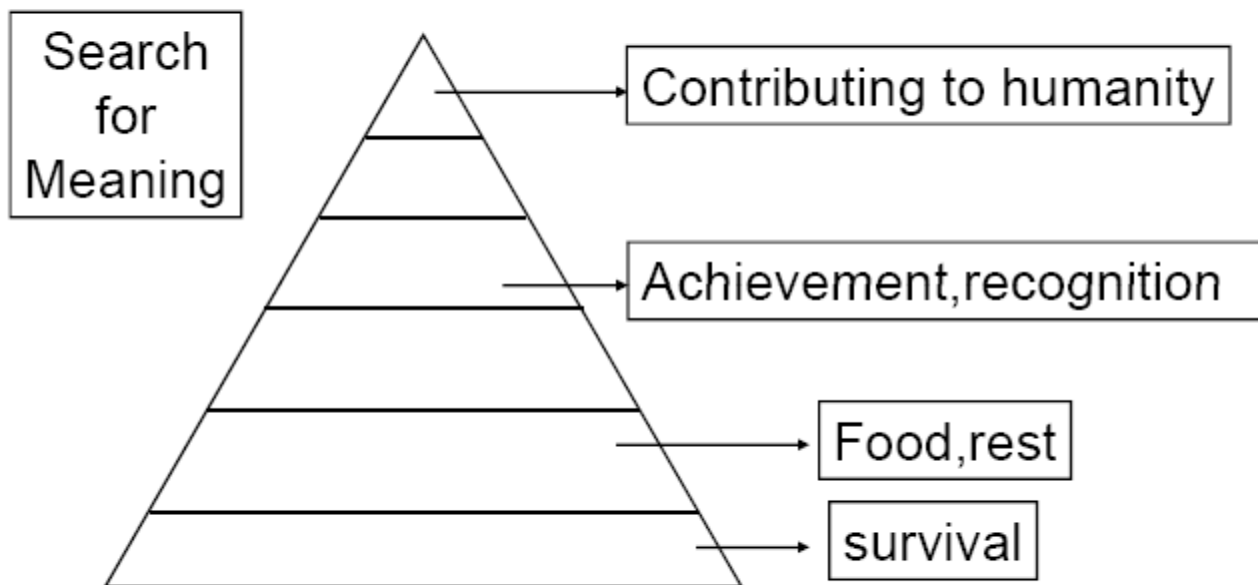
- Evolutionarily, brain has developed algorithms most suitable for survival
- Algorithms unknown: the search is on
- Brain astonishing in the amount of information it processes
 - Typical computers: 10^9 operations/sec
 - Housefly brain: 10^{11} operations/sec

Brain facts & figures

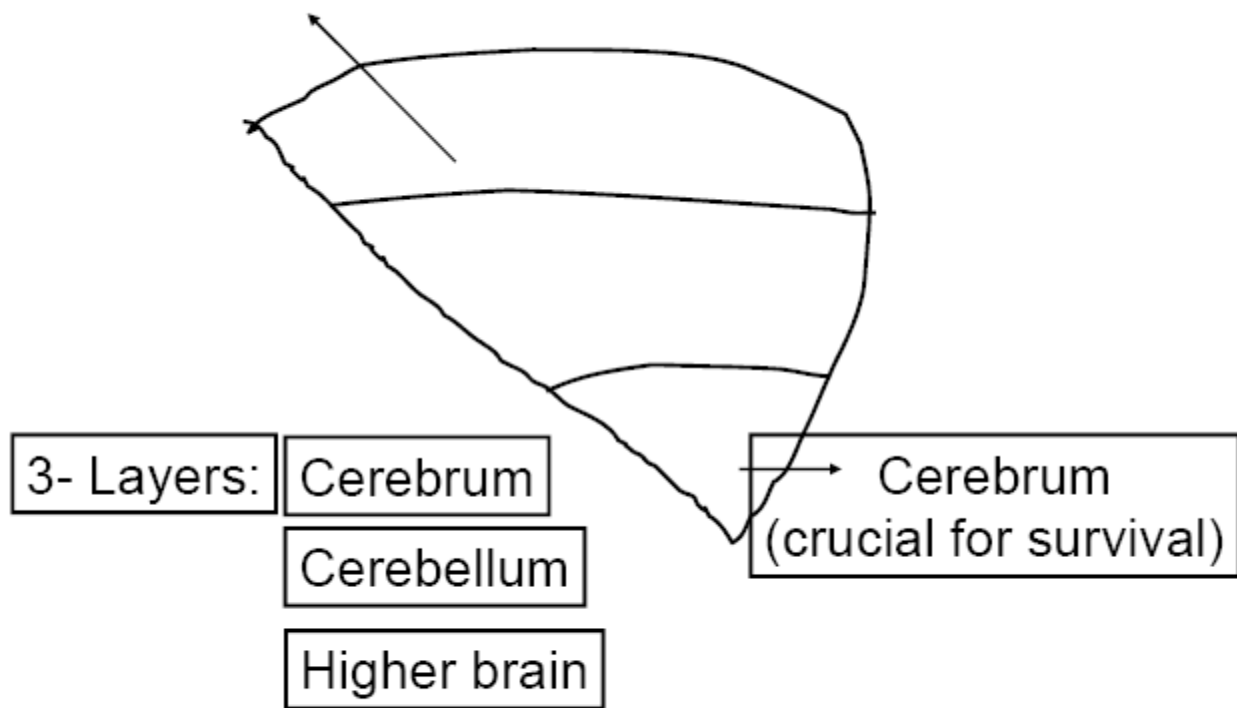
- Basic building block of nervous system: nerve cell (neuron)
- $\sim 10^{12}$ neurons in brain
- $\sim 10^{15}$ connections between them
- Connections made at “synapses”
- The speed: events on millisecond scale in neurons, nanosecond scale in silicon chips



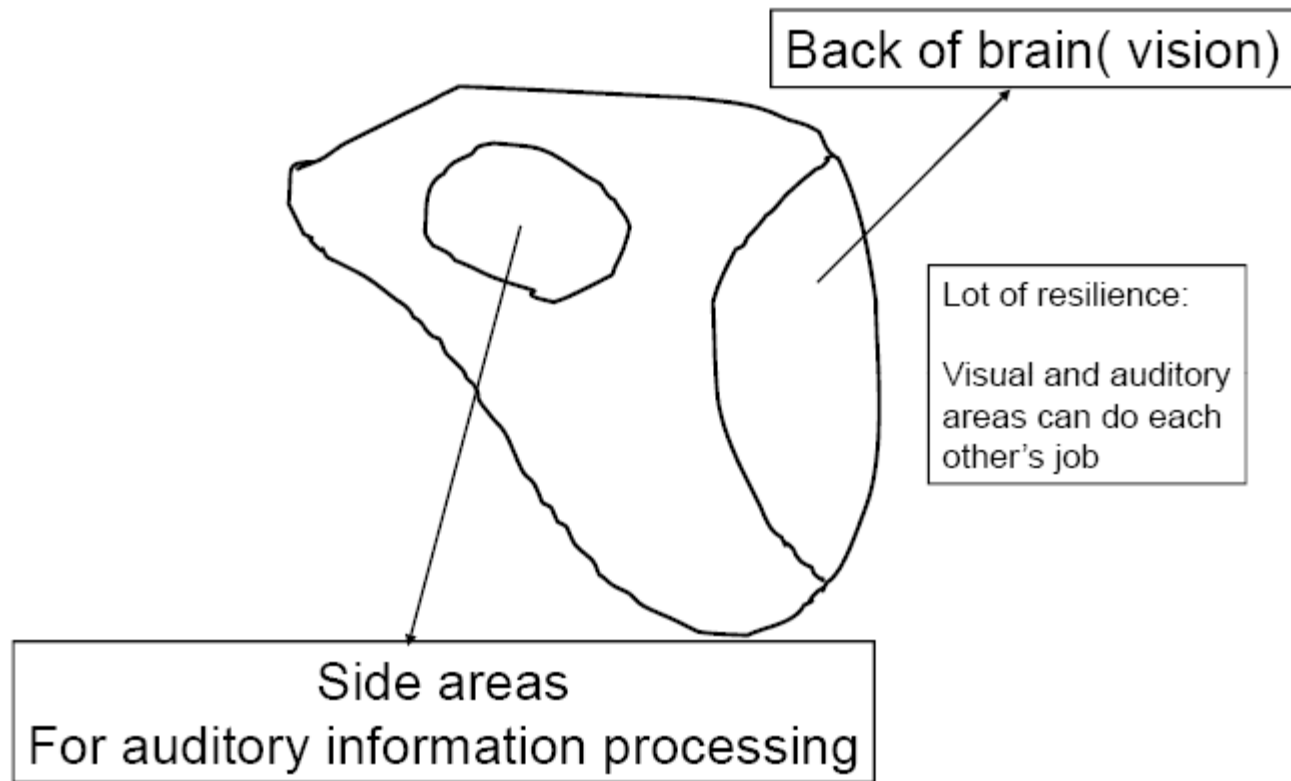
Maslow's hierarchy



Higher brain (responsible for higher needs)

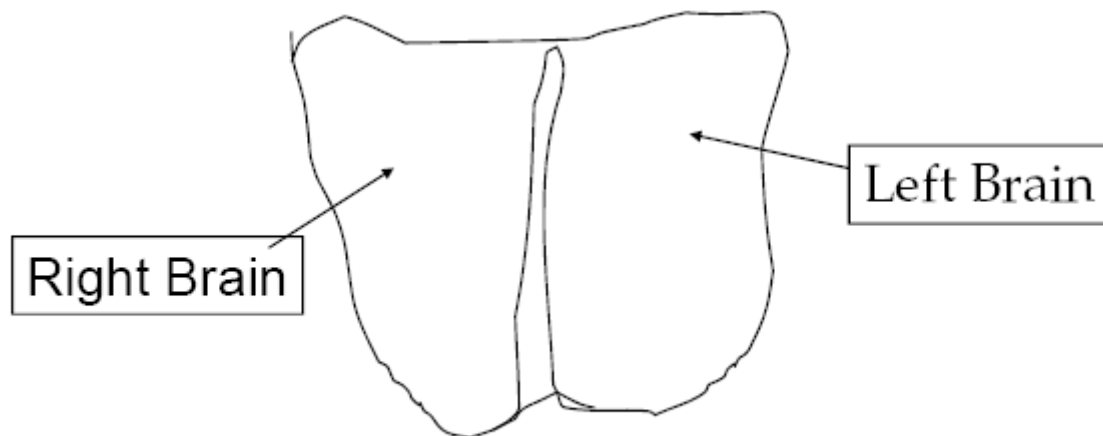


Mapping of Brain



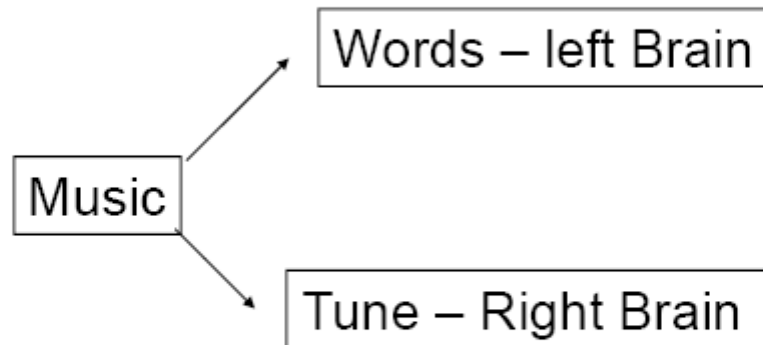
Left Brain and Right Brain

Dichotomy



Left Brain – Logic, Reasoning, Verbal ability

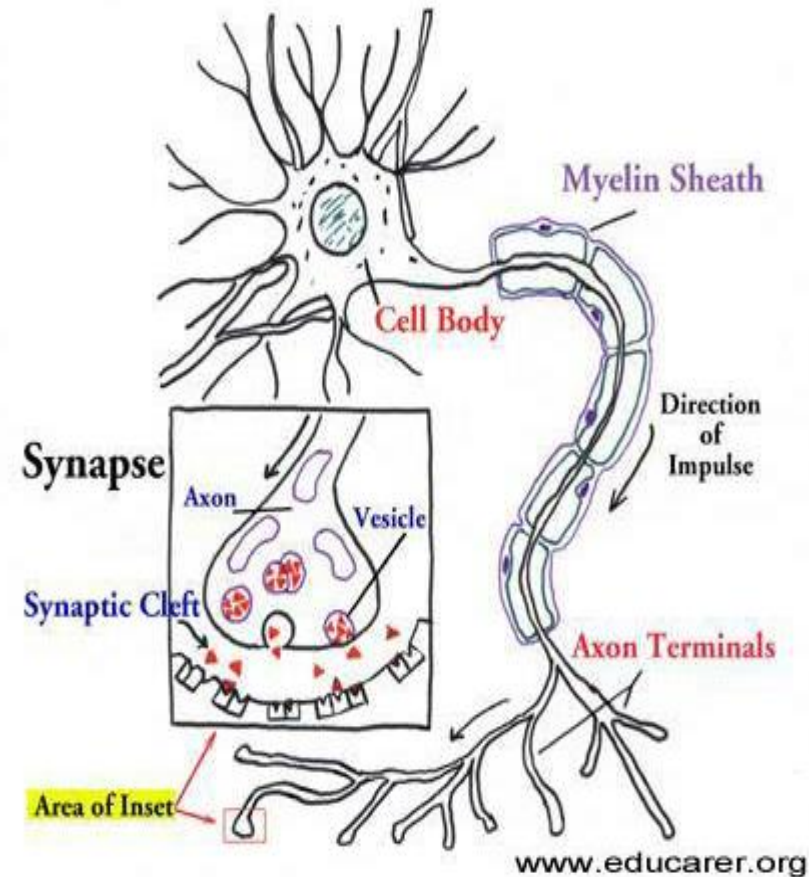
Right Brain – Emotion, Creativity



Maps in the brain. Limbs are mapped to brain

Neuron - “classical”

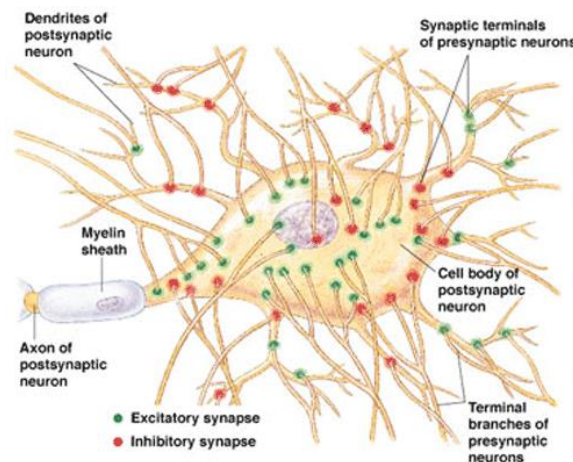
- Dendrites
 - Receiving stations of neurons
 - Don't generate action potentials
- Cell body
 - Site at which information received is integrated
- Axon
 - Generate and relay action potential
 - Terminal
 - Relays information to next neuron in the pathway



<http://www.educarer.com/images/brain-nerve-axon.jpg>

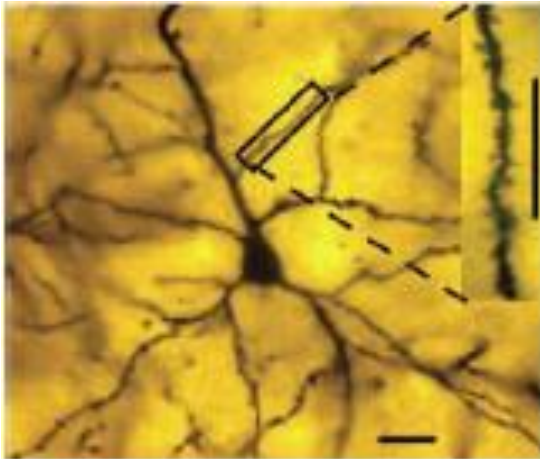
Computation in Biological Neuron

- Incoming signals from synapses are summed up at the soma
- Σ , the biological “inner product”
- On crossing a threshold, the cell “fires” generating an action potential in the axon hillock region

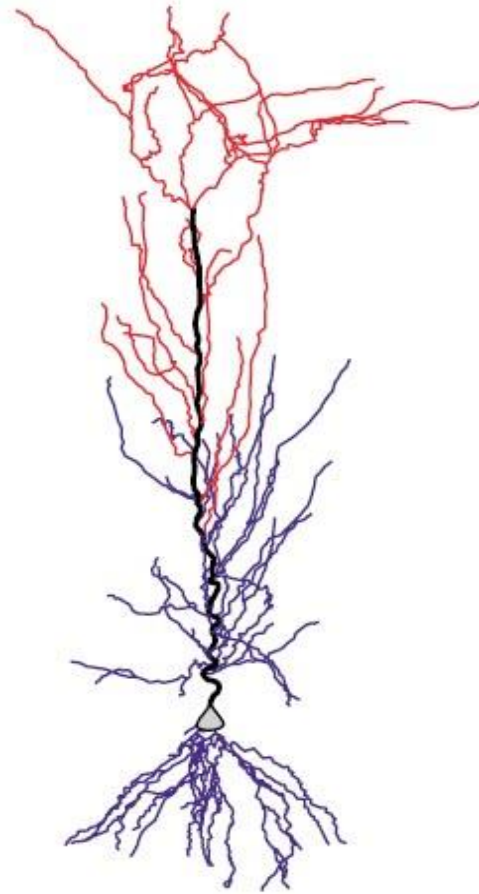


**Synaptic inputs:
Artist's conception**

The biological neuron



Pyramidal neuron, from
the amygdala (Rupshi
et al. 2005)

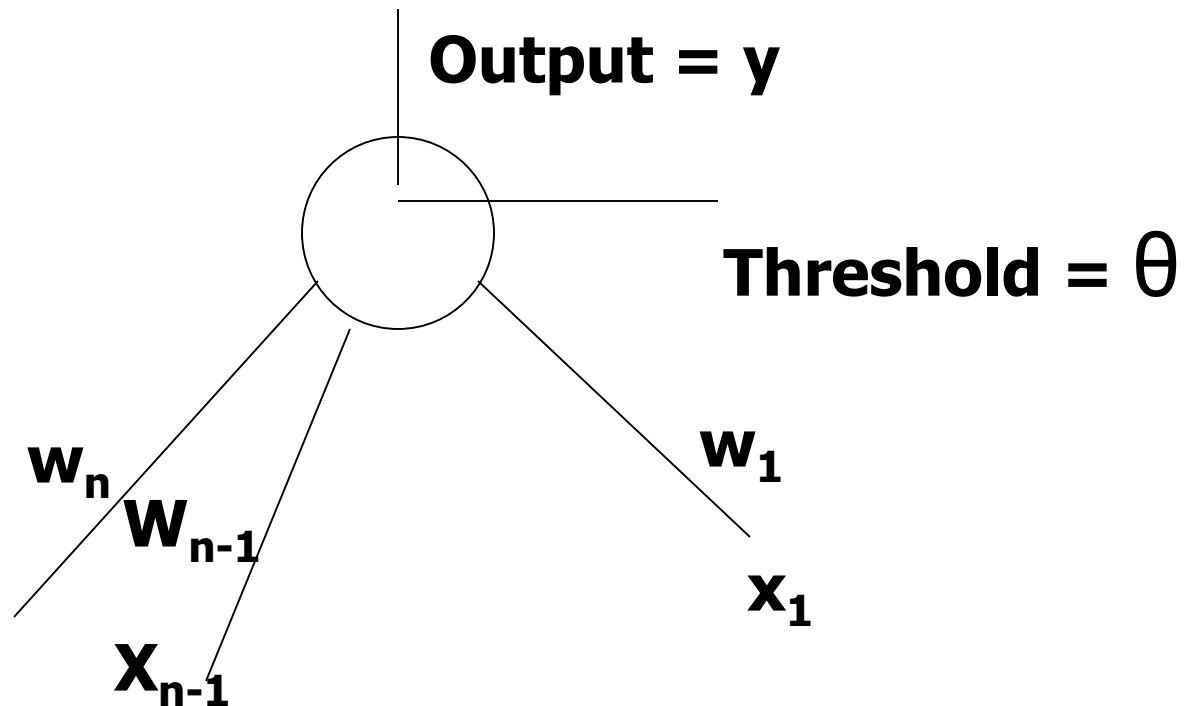


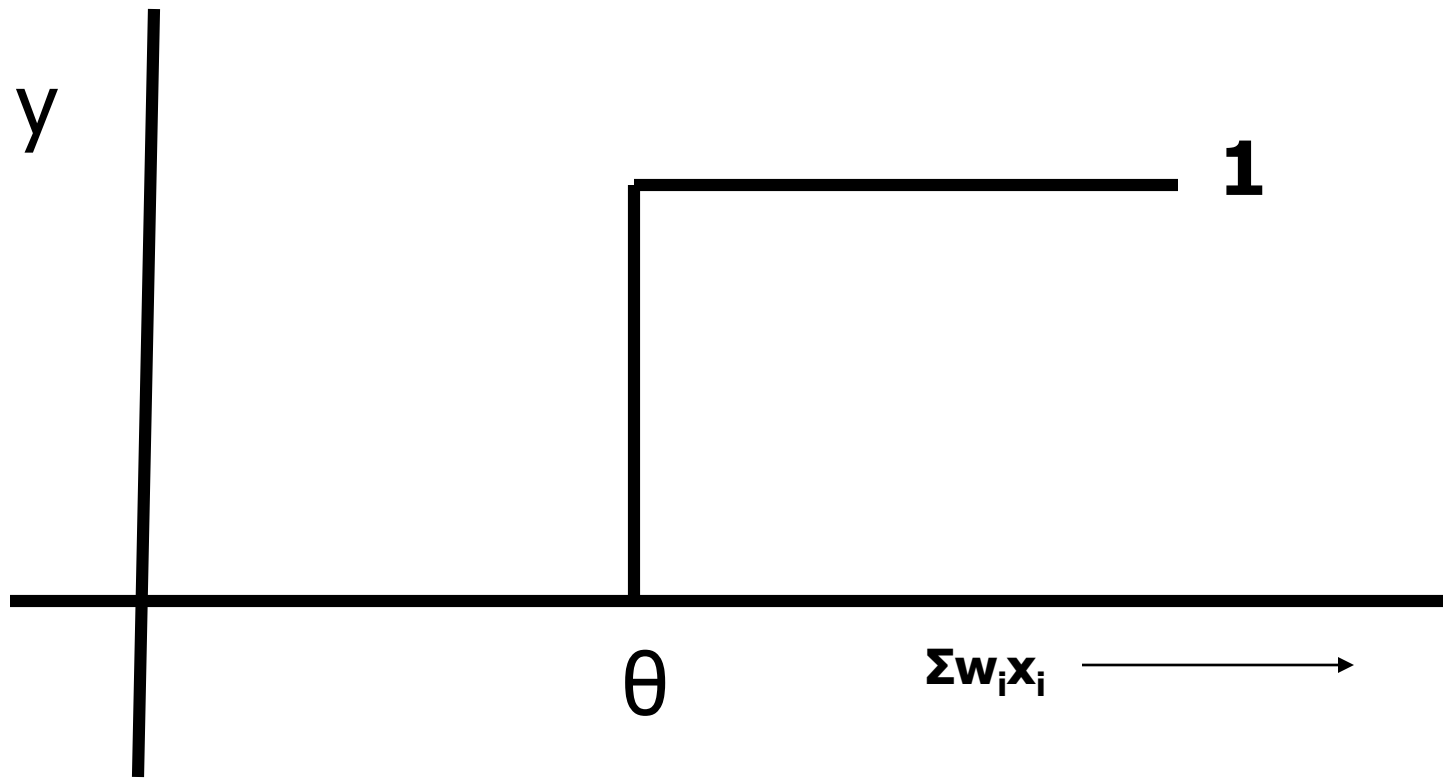
A CA1 pyramidal neuron (Mel *et al.* 2004)

Perceptron

The Perceptron Model

A perceptron is a computing element with input lines having associated weights and the cell having a threshold value. The perceptron model is motivated by the biological neuron.





Step function / Threshold function

**y $= 1$ for $\sum w_i x_i \geq \theta$
 $= 0$ otherwise**

Features of Perceptron

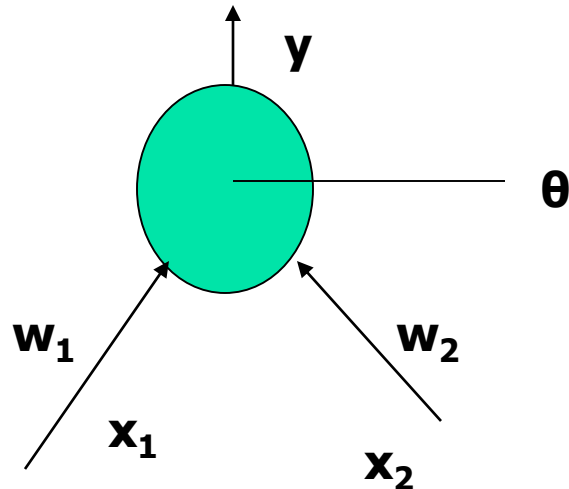
- Input output behavior is discontinuous and the derivative does not exist at $\sum w_i x_i = \theta$
- $\sum w_i x_i - \theta$ is the net input denoted as net
- Referred to as a linear threshold element - linearity because of \mathbf{x} appearing with power **1**
- $\mathbf{y} = \mathbf{f}(\mathbf{net})$: Relation between y and net is non-linear

Computation of Boolean functions

AND of 2 inputs

X1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1

The parameter values (weights & thresholds) need to be found.



Computing parameter values

$$w_1 * 0 + w_2 * 0 \leq \theta \rightarrow \theta \geq 0; \text{ since } y=0$$

$$w_1 * 0 + w_2 * 1 \leq \theta \rightarrow w_2 \leq \theta; \text{ since } y=0$$

$$w_1 * 1 + w_2 * 0 \leq \theta \rightarrow w_1 \leq \theta; \text{ since } y=0$$

$$w_1 * 1 + w_2 * 1 > \theta \rightarrow w_1 + w_2 > \theta; \text{ since } y=1$$
$$w_1 = w_2 = 0.5$$

satisfy these inequalities and find parameters to be used for computing AND function.

Other Boolean functions

- **OR can be computed using values of $w1 = w2 = 1$ and $\theta = 0.5$**

- **XOR function gives rise to the following inequalities:**

$$w1 * 0 + w2 * 0 \leq \theta \rightarrow \theta \geq 0$$

$$w1 * 0 + w2 * 1 > \theta \rightarrow w2 > \theta$$

$$w1 * 1 + w2 * 0 > \theta \rightarrow w1 > \theta$$

$$w1 * 1 + w2 * 1 \leq \theta \rightarrow w1 + w2 \leq \theta$$

No set of parameter values satisfy these inequalities.

Threshold functions

n	# Boolean functions (2^{2^n})	#Threshold Functions
1	4	4
2	16	14
3	256	128
4	64K	1008

- **Functions computable by perceptrons - threshold functions**
- **#TF becomes negligibly small for larger values of #BF.**
- **For $n=2$, all functions except XOR and XNOR are computable.**

Perceptrons and their computing power

Threshold functions

n	# Boolean functions (2^{2^n})	#Threshold Functions
1	4	4
2	16	14
3	256	128
4	64K	1008

- **Functions computable by perceptrons - threshold functions**
- **#TF becomes negligibly small for larger values of #BF.**
- **For $n=2$, all functions except XOR and XNOR are computable.**

[Link](#)