InfoLab

CR521
Neuroinformatics
Sequence Alignment

2013.5.13

Min-Soo Kim

DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

Why multiple alignment?

- **The simultaneous alignment of a number of DNA or protein sequences is one of the commonest tasks in bioinformatics**

- **Useful for :**
  - phylogenetic analysis (inferring a tree, estimating rates of substitution, etc.)
  - detection of homology between a newly sequenced gene and an existing gene family
  - prediction of protein structure
  - demonstration of homology in multigene families
  - determination of a consensus sequence (e.g., in assembly)
  - finding primers for multiple sequences

Example alignment (globin peptides)

```
                        10        20        30        40        50        60
                         .         .         .         .         .         .
Hbb_Human.pep   --------VHLTPEEKSAVTALWGKVN--VDEVGGEALGRLLVVYPWTQRFFESFGDLST
Hbb_Horse.pep   --------VQLSGEEKAAVLALWDKVN--EEEVGGEALGRLLVVYPWTQRFFDSFGDLSN
Hba_Human.pep   ---------VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLS--
Hba_Horse.pep   ---------VLSAADKTNVKAAWSKVGGHAGEYGAEALERMFLGFPTTKTYFPHFDLS--
Myg_Phyca.pep   ---------VLSEGEWQLVLHVWAKVEADVAGHGQDILIRLFKSHPETLEKFDRFKHLKT
Glb5_Petma.pep  PIVDTGSVAPLSAAEKTKIRSAWAPVYSTYETSGVDILVKFFTSTPAAQEFFPKFKGLTT
Lgb2_Luplu.pep  --------GALTESQAALVKSSWEEFNANIPKHTHRFFILVLEIAPAAKDLFSFLKGTSE
                         *.       .    *                          *  .    *

Hbb_Human.pep   PDAVMGNPKVKAHGKKVLGAFSDGLAHLD-----NLKGTFATLSELHCDKLHVDPENFRL
Hbb_Horse.pep   PGAVMGNPKVKAHGKKVLHSFGEGVHHLD-----NLKGTFAALSELHCDKLHVDPENFRL
Hba_Human.pep   ----HGSAQVKGHGKKVADALTNAVAHVD-----DMPNALSALSDLHAHKLRVDPVNFKL
Hba_Horse.pep   ----HGSAQVKAHGKKVGDALTLAVGHLD-----DLPGALSNLSDLHAHKLRVDPVNFKL
Myg_Phyca.pep   EAEMKASEDLKKHGVTVLTALGAILKKKG-----HHEAELKPLAQSHATKHKIPIKYLEF
Glb5_Petma.pep  ADQLKKSADVRWHAERIINAVNDAVASMDDT--EKMSMKLRDLSGKHAKSFQVDPQYFKV
Lgb2_Luplu.pep  VP--QNNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG-VADAHFPV
                        .. *    .                              *    *      .

Hbb_Human.pep   LGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKYH------
Hbb_Horse.pep   LGNVLVVVLARHFGKDFTPELQASYQKVVAGVANALAHKYH------
Hba_Human.pep   LSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLTSKYR------
Hba_Horse.pep   LSHCLLSTLAVHLPNDFTPAVHASLDKFLSSVSTVLTSKYR------
Myg_Phyca.pep   ISEAIIHVLHSRHPGDFGADAQGAMNKALELFRKDIAAKYKELGYQG
Glb5_Petma.pep  LAAVIADTVAAG---------DAGFEKLMSMICILLRSAY-------
Lgb2_Luplu.pep  VKEAILKTIKEVVGAKWSEELNSAWTIAYDELAIVIKKEMNDAA---
                .   .   .                          .
```

InfoLab

Multiple Sequence Alignment

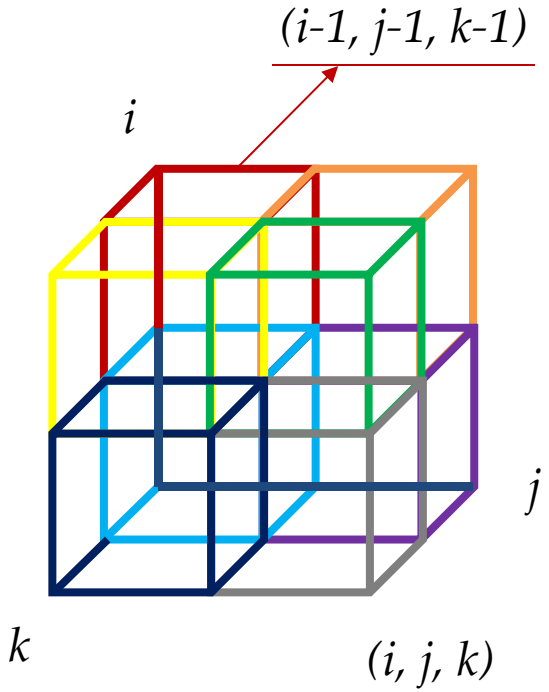
Exact methods

DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

Multidimensional DP (for 3 sequences)

- **We build a 3-dim DP Matrix**
- **Seven cases when filling matrix (with no affine gap)**
  - δ(x, y, z) : score of a column with letters x, y, and z
  - No case of δ(-,-,-)

1. $(i-1, j, k)$ for score $\delta(u_i, -, -)$
2. $(i, j-1, k)$ for score $\delta(-, v_j, -)$
3. $(i, j, k-1)$ for score $\delta(-, -, w_k)$
4. $(i-1, j-1, k)$ for score $\delta(u_i, v_j, -)$
5. $(i-1, j, k-1)$ for score $\delta(u_i, -, w_k)$
6. $(i, j-1, k-1)$ for score $\delta(-, v_j, w_k)$
7. $(i-1, j-1, k-1)$ for score $\delta(u_i, v_j, w_k)$

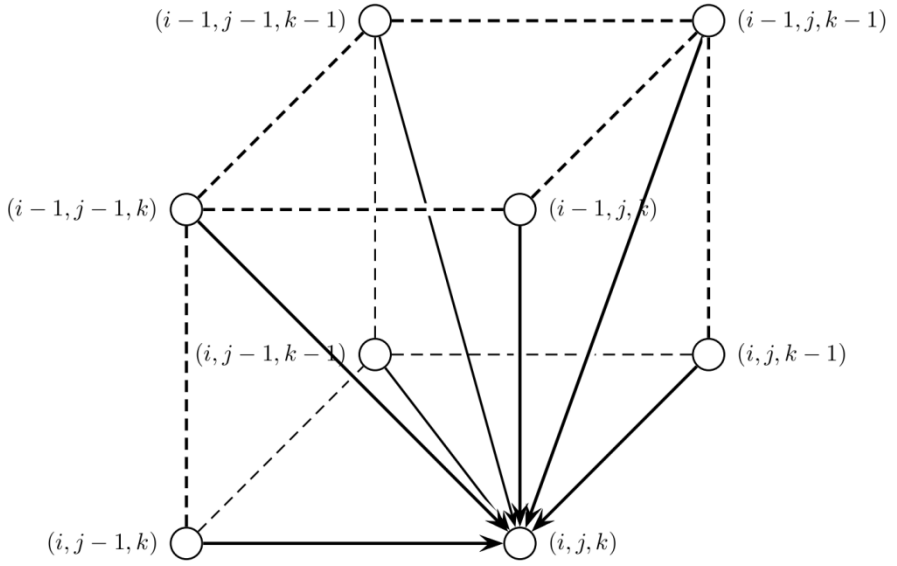


- **We choose the one with the max score**

$$s_{i,j,k} = \max \begin{cases} s_{i-1,j,k} & +\delta(v_i, -, -) \\ s_{i,j-1,k} & +\delta(-, w_j, -) \\ s_{i,j,k-1} & +\delta(-, -, u_k) \\ s_{i-1,j-1,k} & +\delta(v_i, w_j, -) \\ s_{i-1,j,k-1} & +\delta(v_i, -, u_k) \\ s_{i,j-1,k-1} & +\delta(-, w_j, u_k) \\ s_{i-1,j-1,k-1} & +\delta(v_i, w_j, u_k) \end{cases}$$

Time complexity :

$O(2^3n^3) = O((2n)^3)$



Multidimensional DP (for *N* sequences)

- $s_{i1, i2, \ldots, iN}$
  - max score of an alignment ending with $x^1_{i1}, x^2_{i2}, \ldots, x^N_{iN}$
- **Time complexity : $O(2^N n^N) = O((2n)^N)$**
  - computationally prohibitive when *N* increases

$$
s_{i1, i2, \cdots, iN} = max \begin{cases}
s_{i1-1, i2-1, \cdots, iN-1} + \delta(x^1_{i1}, x^2_{i2}, \ldots, x^N_{iN}) \\
s_{i1, \ i2-1, \cdots, iN-1} + \delta(-, x^2_{i2}, \ldots, x^N_{iN}) \\
s_{i1-1, i2, \cdots, iN-1} + \delta(x^1_{i1}, -, \ldots, x^N_{iN}) \\
\ldots \\
s_{i1-1, i2-1, \cdots, iN} + \delta(x^1_{i1}, x^2_{i2}, \ldots, -) \\
s_{i1, i2, \cdots, iN-1} + \delta(-, -, \ldots, x^N_{iN}) \\
\ldots \\
s_{i1, i2-1, \cdots, iN} + \delta(-, x^2_{i2}, \ldots, -) \\
\ldots
\end{cases}
$$

InfoLab

Multiple Sequence Alignment

Heuristic methods

DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

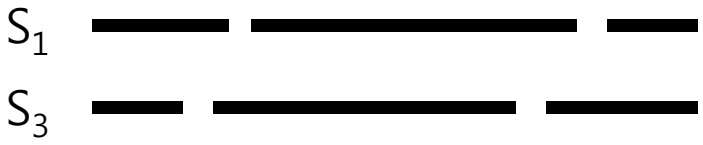
Progressive alignment (1987)

- Concept: apply pairwise alignment algorithm *iteratively*
  - align the most closely related pair of sequences
  - align the next most similar one to that pair
  - and so on

- Rule of “once a gap, always a gap” (preserve gaps)
  - the positions and lengths of gaps introduced between more similar pairs of sequences should not be affected by more distantly related ones

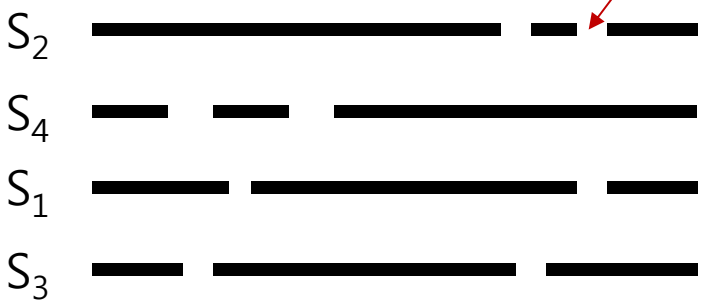
Gap propagation

align the most similar pair



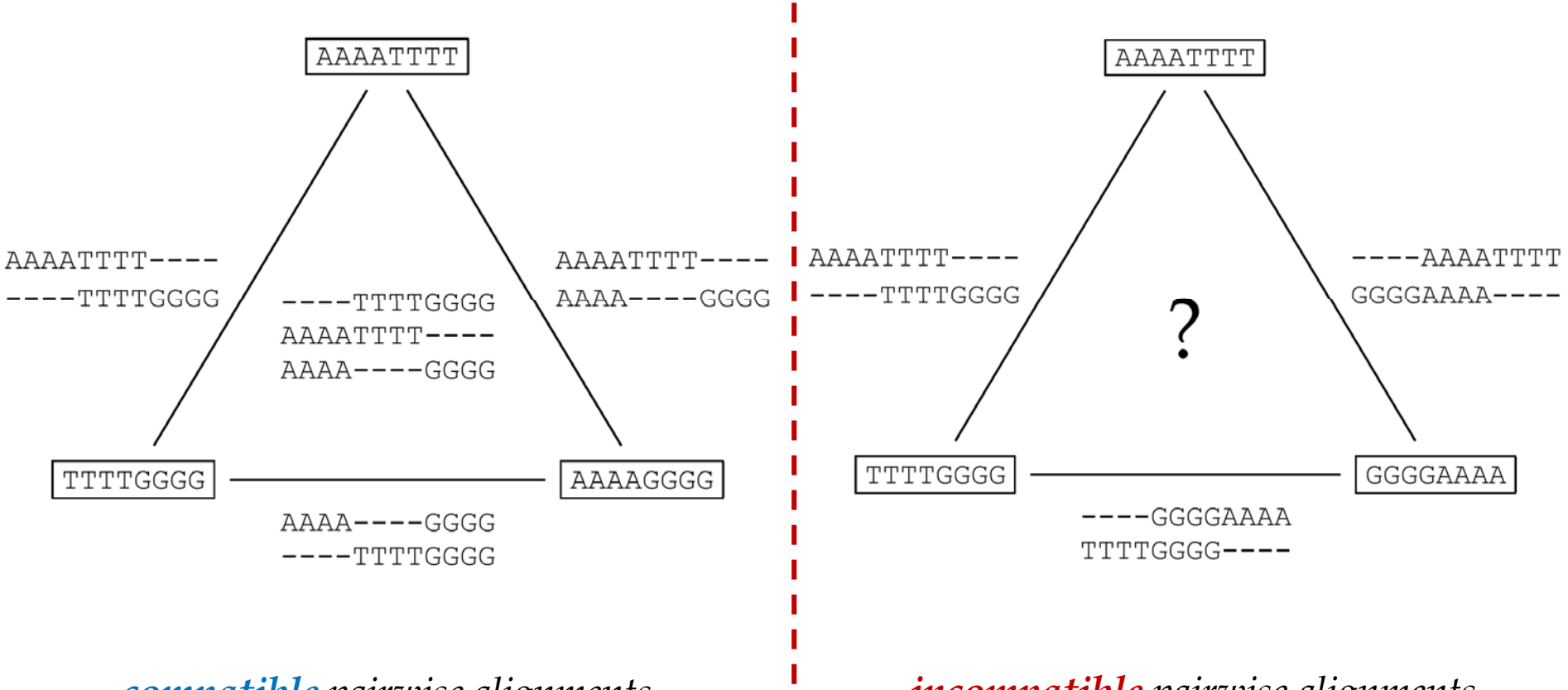
align the next most similar pair

new gap to optimize alignment of s_2s_4 with s_1s_3



align alignments – preserve gaps

Compatibility of pairwise alignments



compatible pairwise alignments

incompatible pairwise alignments

CLUSTAL W (1994)

- The most widely used progressive alignment algorithm

- Three basic steps (common in all progressive alignment algorithms)

  ① Calculate a matrix of pairwise distances based on pairwise alignments between the sequences

  ② Use the result of (1) to build a guide tree, which is an inferred phylogeny for the sequences

  ③ Use the tree from (2) to guide the progressive alignment of the sequences

(1) Calculating the pairwise distances

- A pair of sequences is aligned by the usual dynamic programming algorithm
- A similarity or distance for the pair is calculated using the aligned portion (gaps excluded)
  - e.g., percent identity
- CLUSTAL W does not correct these distances for multiple substitutions (*e.g.*, by the Jukes-Cantor formula)
  - a simple count of differences between two sequences does not tell the full story about their genetic distance
  - Jukes-Cantor model: a probabilistic model to correct the observed number of differences to account for the possibility of multiple substitutions

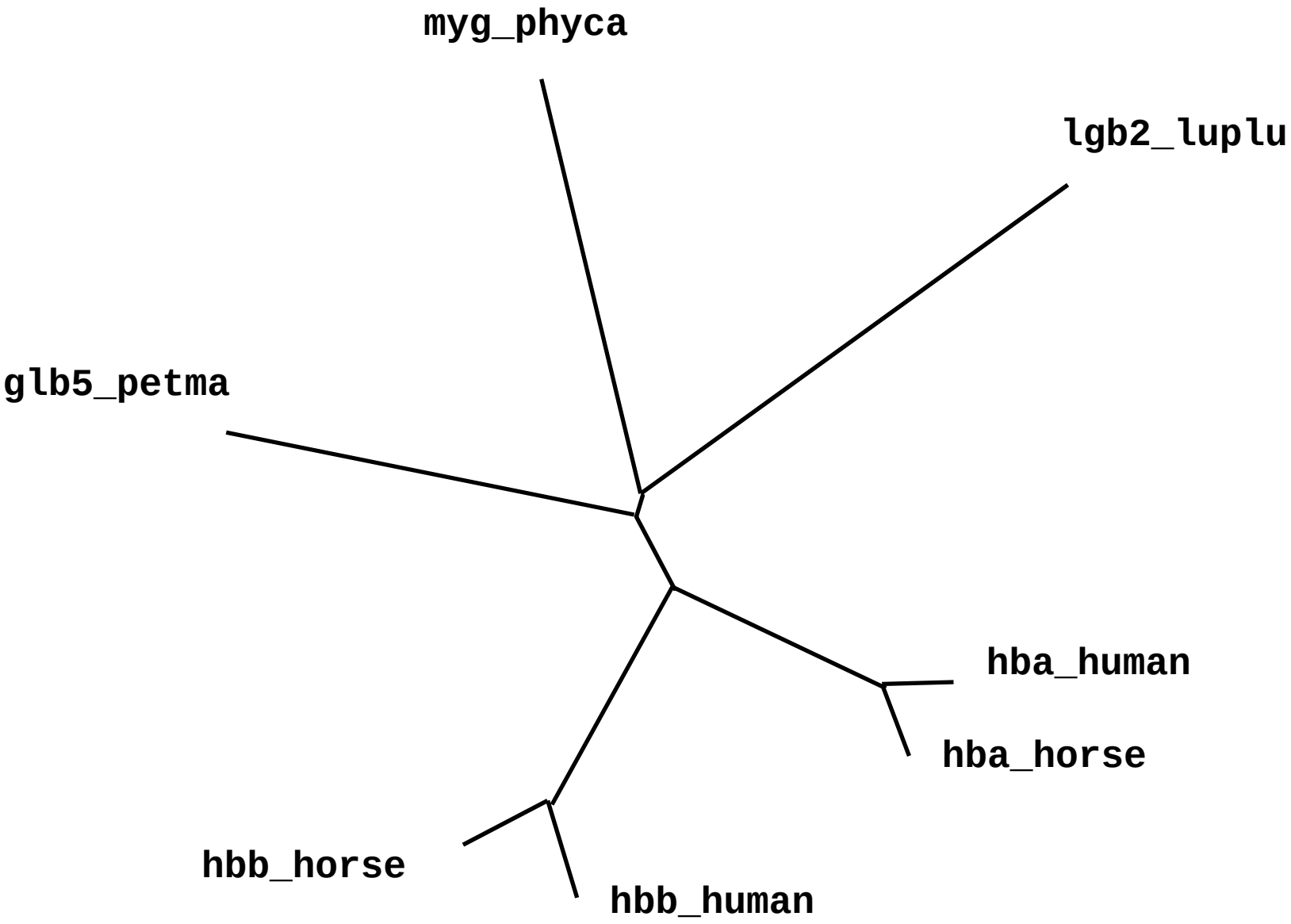
Example: Globin

```
 DISTANCES between protein sequences:

 Calculated over: 1 to 167
 Correction method: Simple distance (no corrections)
 Distances are: observed number of substitutions per 100 amino acids
Symmatrix version 1
Number of matrices: 1

//
Matrix 1, dimension: 7

Key for column and row indices:

  1 hba_human
  2 hba_horse
  3 hbb_human
  4 hbb_horse
  5 glb5_petma
  6 myg_phyca
  7 lgb2_luplu

 Matrix 1: Part 1

                1        2        3        4        5        6        7
_____________________________________________________________________ ..
|     1   |     0.00    12.06    54.68    55.40    64.12    71.74    83.57
|     2   |              0.00    55.40    53.96    64.89    72.46    82.86
|     3   |                       0.00    16.44    74.26    73.94    82.52
|     4   |                                0.00    75.74    73.94    81.12
|     5   |                                         0.00    75.91    82.61
|     6   |                                                  0.00    80.95
|     7   |                                                           0.00
```

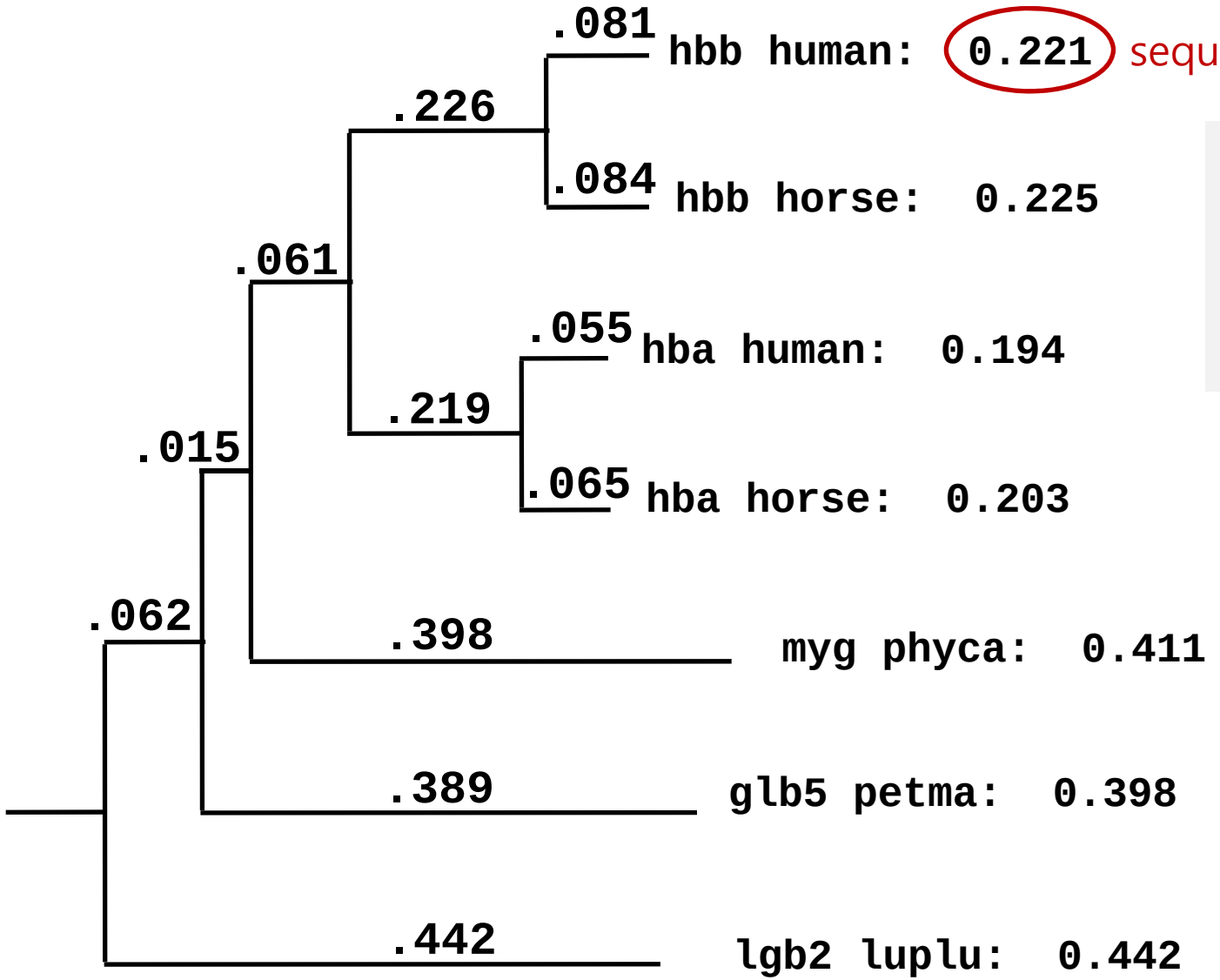
(2) Building the guide tree

- Many ways of building a tree from a matrix of pairwise distances
  - CLUSTAL W uses Neighbour-Joining(NJ) method
  - branch lengths are proportional to estimated divergence
- Root of the tree is determined by mid-point method
  - giving equal means for the branch lengths on either side of the root
- W in CLUSTAL W stands for Weights
  - weight is important feature of CLUSTAL W
    - normal progressive alignment algorithms use equal weights
  - rooted tree is used to derive a weight for each sequence
  - e.g., 0.221 = 0.081 + 0.226/2 + 0.061/4 + 0.015/5 + 0.062/6

Example: Unrooted NJ globin tree



Example: Rooted NJ tree (guide tree)



0.221 = .081
+ 0.226 / 2
+ 0.061 / 4
+ 0.015 / 5
+ 0.062 / 6

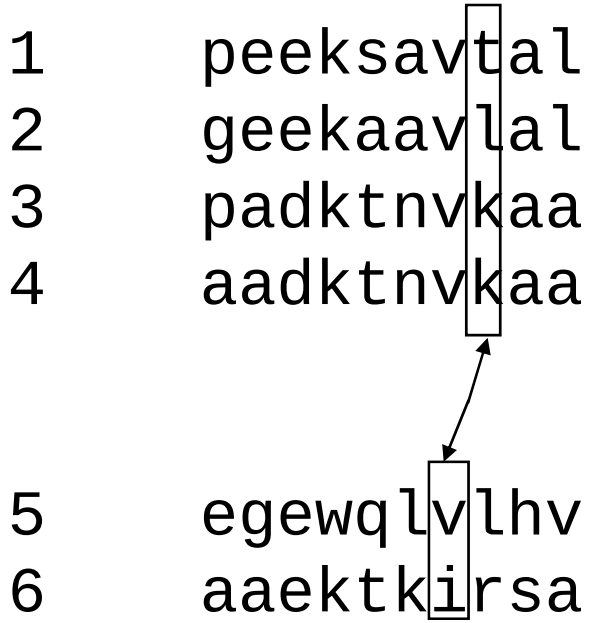
(3) Progressive alignment

- **Use a series of pairwise alignments to align larger and larger groups of sequences**
  - following the branching order of the guide tree
  - proceeding from the tips of the rooted tree towards the root

- **Use full dynamic programming algorithm at each stage**
  - with a residue scoring matrix (e.g., a PAM or a BLOSUM matrix)
  - with penalties for opening and extending gaps

- **Make gaps in older alignments remain fixed**
  - new gaps introduced at each stage get full opening and extension penalties

Scoring scheme for two positions from two alignments



with sequence weights

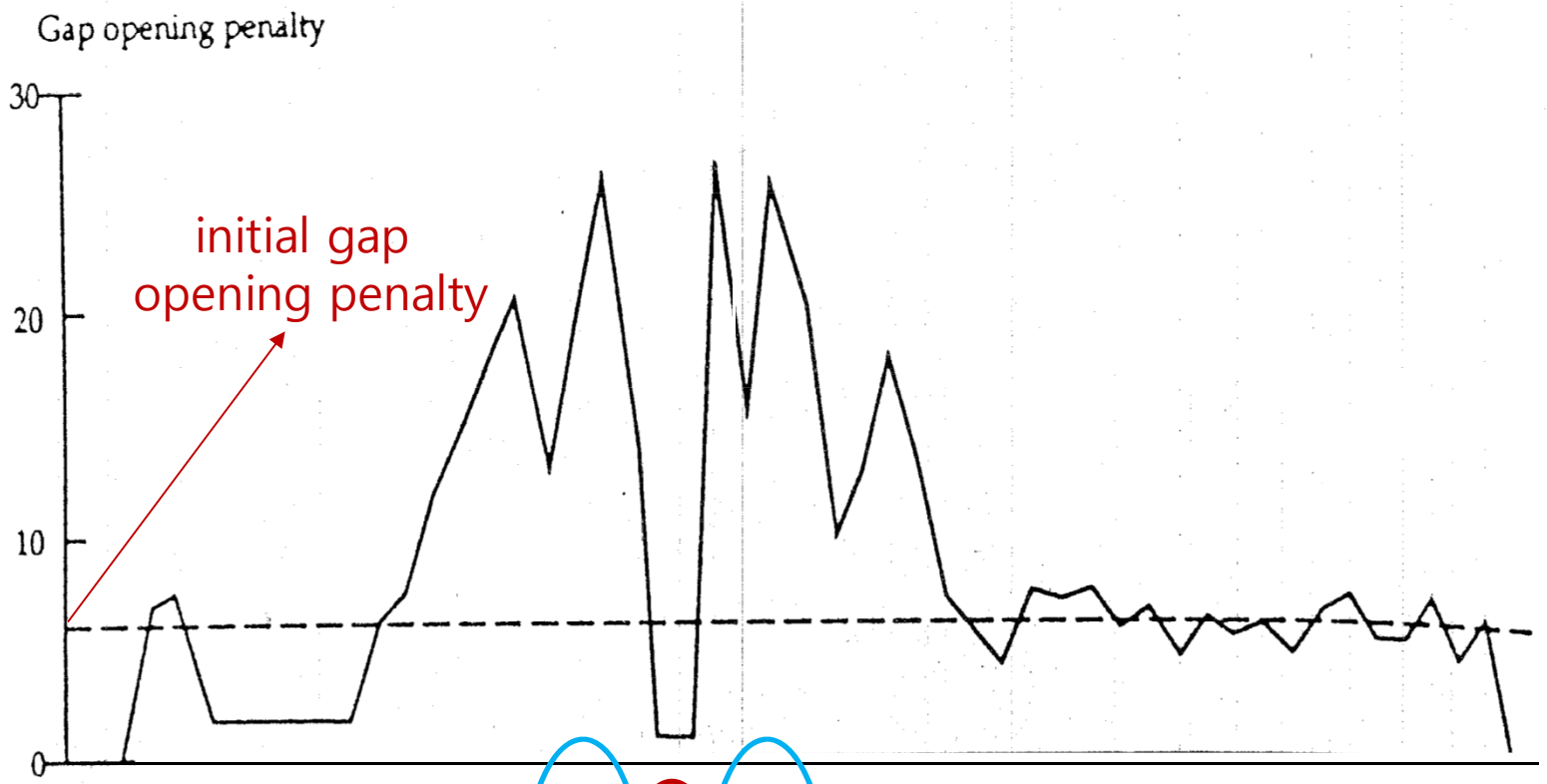
$$\text{score} = [\, M(t, v) \times w_1 \times w_5 + M(t, i) \times w_1 \times w_6 + M(l, v) \times w_2 \times w_5 + M(l, i) \times w_2 \times w_6 + M(k, v) \times w_3 \times w_5 + M(k, i) \times w_3 \times w_6 + M(k, v) \times w_4 \times w_5 + M(k, i) \times w_4 \times w_6 \,] \;/\; 8$$

- w_i : weight for sequence i
- $M(x, y)$: weight matrix entry for x versus y

(3) Progressive alignment – gaps

- **Treatment of gap penalties is complicated**
  - Gap opening and extension penalties are calculated depending on
    1) weight matrix
    2) sequence similarity
    3) sequence length
    4) difference in the lengths of the sequences
    5) position of gaps (*see figure in the next slide*)
    6) residues at gaps (*see figure in the next slide*)
- **Motivation of 5) and 6) is as follows:**
  - if one knew the positions of all secondary structure elements in all or some of the sequences
  - one could increase the gap penalties inside and decrease outside them, forcing gaps to occur most often in loop regions

Position and residue-specific gap opening penalties



Final CLUSTALW alignment

```
                         10        20        30        40        50        60
                          .         .         .         .         .         .
Hbb_Human.pep    --------VHLTPEEKSAVTALWGKVN--VDEVGGEALGRLLVVYPWTQRFFESFGDLST
Hbb_Horse.pep    --------VQLSGEEKAAVLALWDKVN--EEEVGGEALGRLLVVYPWTQRFFDSFGDLSN
Hba_Human.pep    ---------VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLS--
Hba_Horse.pep    ---------VLSAADKTNVKAAWSKVGGHAGEYGAEALERMFLGFPTTKTYFPHFDLS--
Myg_Phyca.pep    ---------VLSEGEWQLVLHVWAKVEADVAGHGQDILIRLFKSHPETLEKFDRFKHLKT
Glb5_Petma.pep   PIVDTGSVAPLSAAEKTKIRSAWAPVYSTYETSGVDILVKFFTSTPAAQEFFPKFKGLTT
Lgb2_Luplu.pep   --------GALTESQAALVKSSWEEFNANIPKHTHRFFILVLEIAPAAKDLFSFLKGTSE
                          *.       .    *                       *  .    *

Hbb_Human.pep    PDAVMGNPKVKAHGKKVLGAFSDGLAHLD-----NLKGTFATLSELHCDKLHVDPENFRL
Hbb_Horse.pep    PGAVMGNPKVKAHGKKVLHSFGEGVHHLD-----NLKGTFAALSELHCDKLHVDPENFRL
Hba_Human.pep    ----HGSAQVKGHGKKVADALTNAVAHVD-----DMPNALSALSDLHAHKLRVDPVNFKL
Hba_Horse.pep    ----HGSAQVKAHGKKVGDALTLAVGHLD-----DLPGALSNLSDLHAHKLRVDPVNFKL
Myg_Phyca.pep    EAEMKASEDLKKHGVTVLTALGAILKKKG-----HHEAELKPLAQSHATKHKIPIKYLEF
Glb5_Petma.pep   ADQLKKSADVRWHAERIINAVNDAVASMDDT--EKMSMKLRDLSGKHAKSFQVDPQYFKV
Lgb2_Luplu.pep   VP--QNNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG-VADAHFPV
                        .. *    .                          *    *       .

Hbb_Human.pep    LGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKYH------
Hbb_Horse.pep    LGNVLVVVLARHFGKDFTPELQASYQKVVAGVANALAHKYH------
Hba_Human.pep    LSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLTSKYR------
Hba_Horse.pep    LSHCLLSTLAVHLPNDFTPAVHASLDKFLSSVSTVLTSKYR------
Myg_Phyca.pep    ISEAIIHVLHSRHPGDFGADAQGAMNKALELFRKDIAAKYKELGYQG
Glb5_Petma.pep   LAAVIADTVAAG---------DAGFEKLMSMICILLRSAY-------
Lgb2_Luplu.pep   VKEAILKTIKEVVGAKWSEELNSAWTIAYDELAIVIKKEMNDAA---
                 .   .   .                          .
```

Other methods

- **CLUSTAL X (1997)**
- **T-Coffee (2000)**
  - slow
  - accurate
- **MAFFT (2002)**
  - fast
  - useful for aligning large numbers of sequences
- **CLUSTAL Omega (2011)**
  - fast
  - accurate

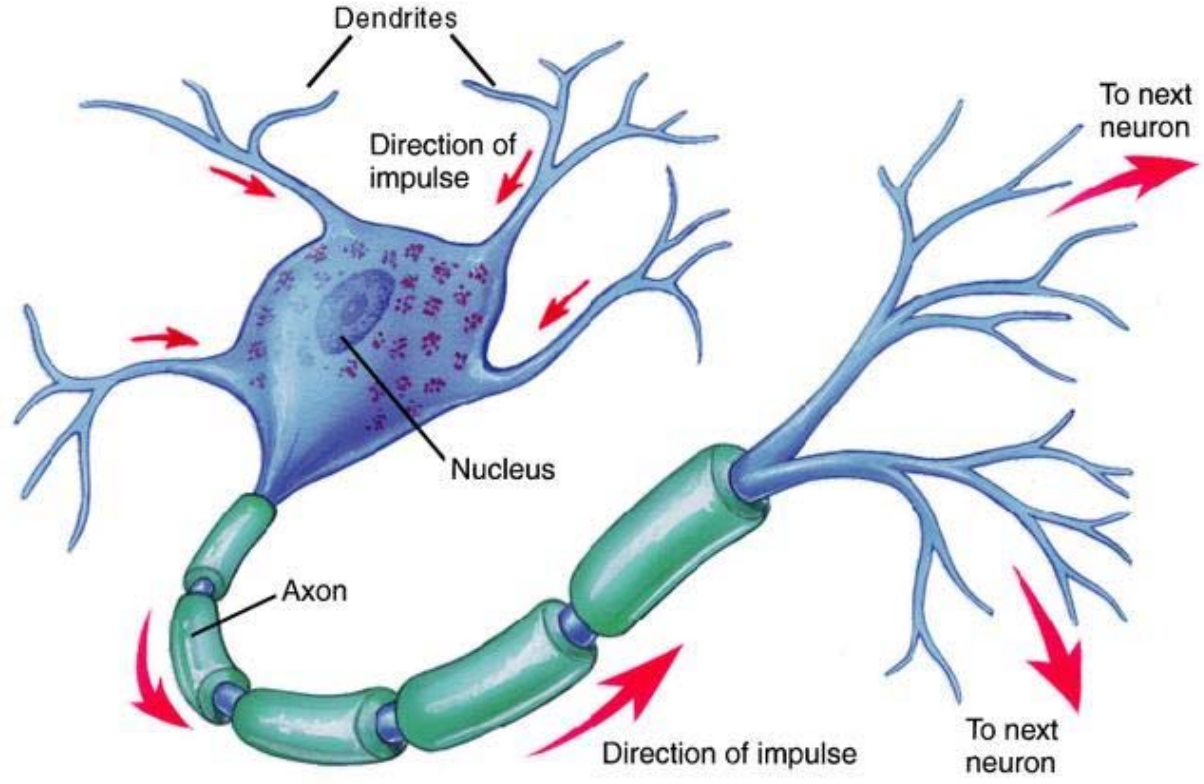
InfoLab

CR521
Neuroinformatics
Artificial Neural Networks

DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

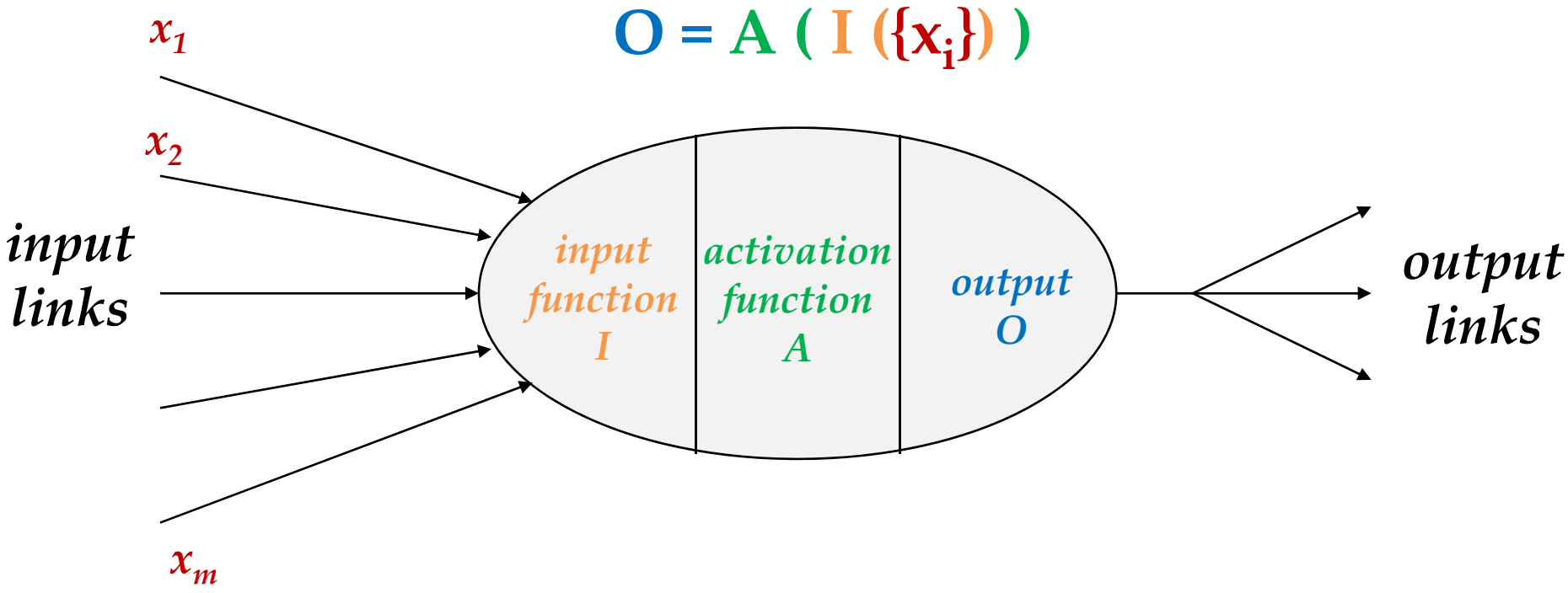
Biological neuron network

- **Unit of network: biological neuron**



Artificial neural network

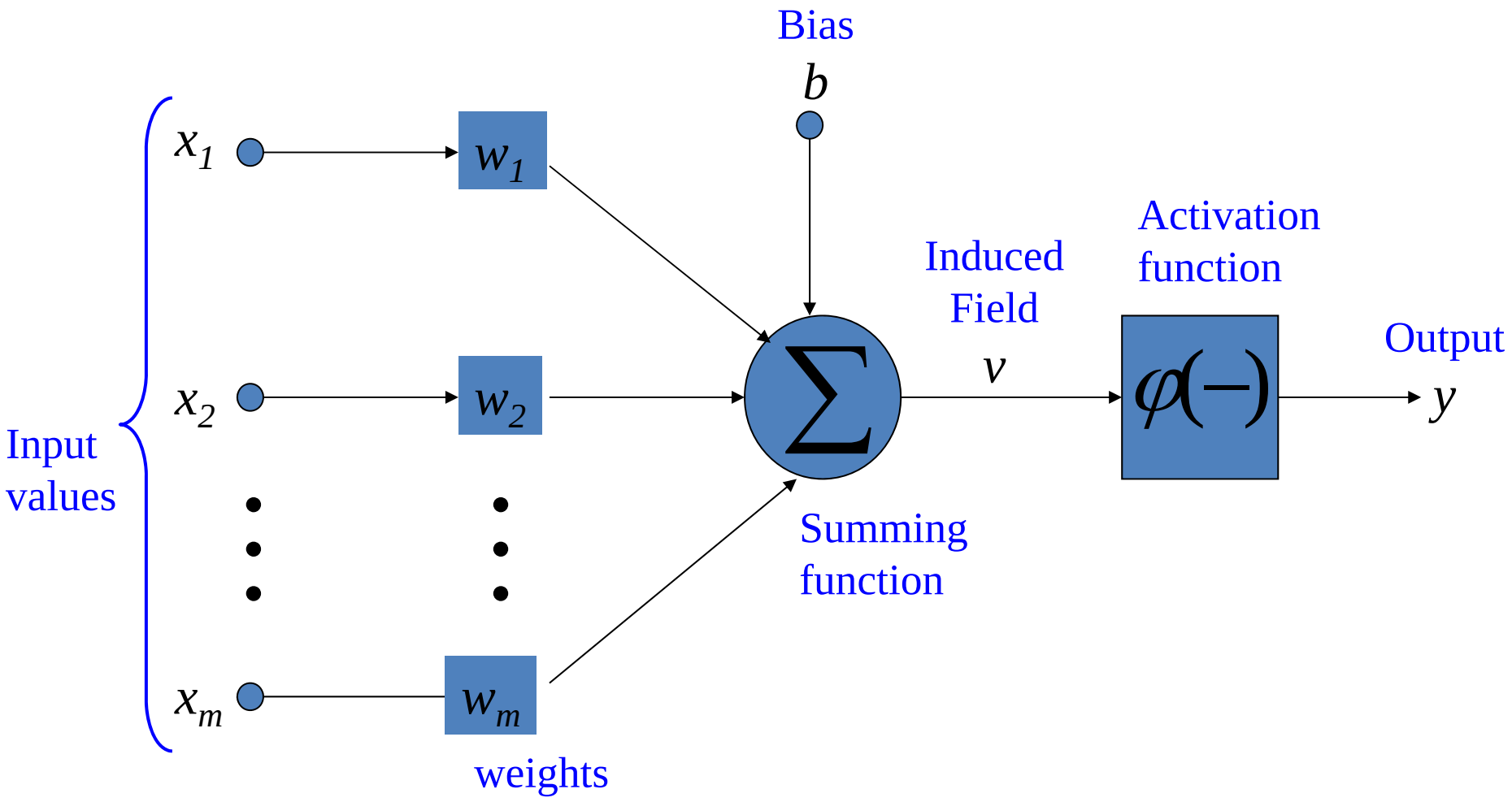
- **Unit of network: artificial neuron**
- **Size(#of units) : much smaller than biological neural network**



Artificial neural network (ANN)

- **Machine learning approach that models human brain**
- **Neuron in ANNs tend to have fewer connections than biological neurons**
  - # of connections of biological neuron : 1,000
- **Knowledge about the learning task is given in the form of training instances**
  - ANN should behave correctly on new instances
- **ANN is specified by**
  - neuron model: *which kind of neuron is used*
  - architecture: *how ANN looks like (how neurons are connected)*
  - learning algorithm: *how to train ANN*

Diagram of typical neuron



- The bias *b* has the effect of applying a transformation to the weighted sum *u*

$$u = \Sigma_{1 \le i \le m} w_i x_i$$

$$v = u + b$$

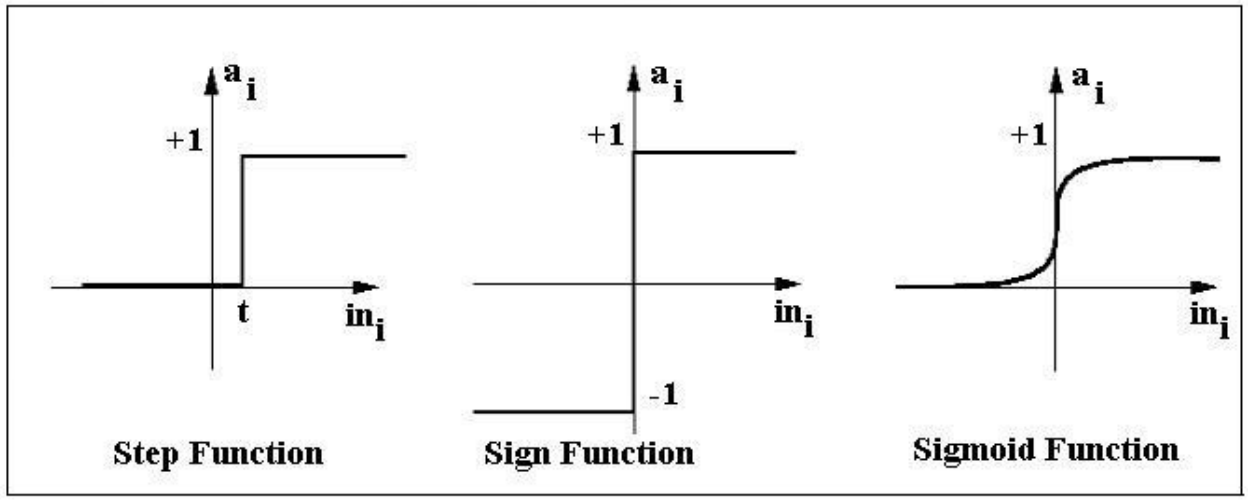
- *v* is called induced field of the neuron

- Activation function φ limits the amplitude of the neuron output

$$y = \varphi(u + b)$$

Neuron model

- **The choice of activation function φ determines the neuron model**
- **Examples of φ**
  - step function : *+1 if $v \geq t$, else 0*
  - sign function : *+1 if $v \geq 0$, else -1*
  - sigmoid function : *$1 / (1+e^{-v})$*



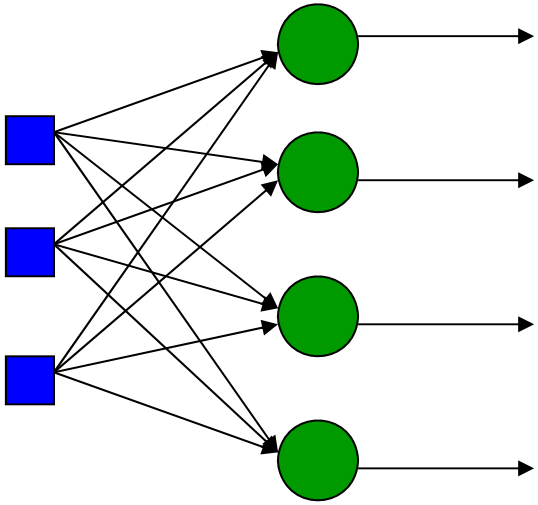
Architectures of ANN

- **Three different classes of network architectures**
  - single-layer feed-forward
  - multi-layer feed-forward
  - recurrent
  - Radial Basis Function
  - Hopfield

- **The architecture of a neural network is linked with the learning algorithm used to train**

Single Layer Feed-Forward (SLFF) NN

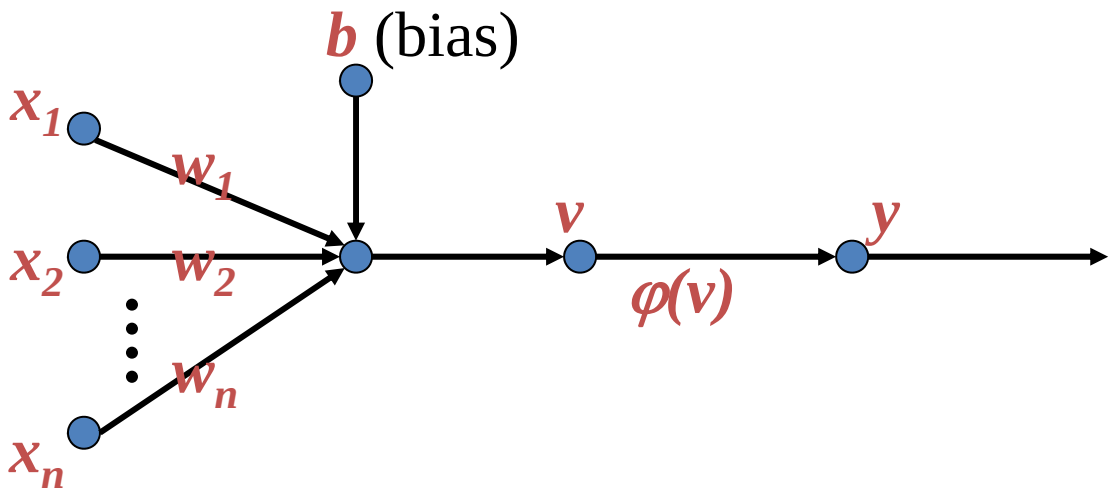
Input layer of source nodes



Output layer of neurons

Perceptron : Special form of SLFF

- The perceptron was first proposed by Rosenblatt (1958) is a simple neuron that is used to classify its input into one of two categories
- A perceptron uses a step function for φ



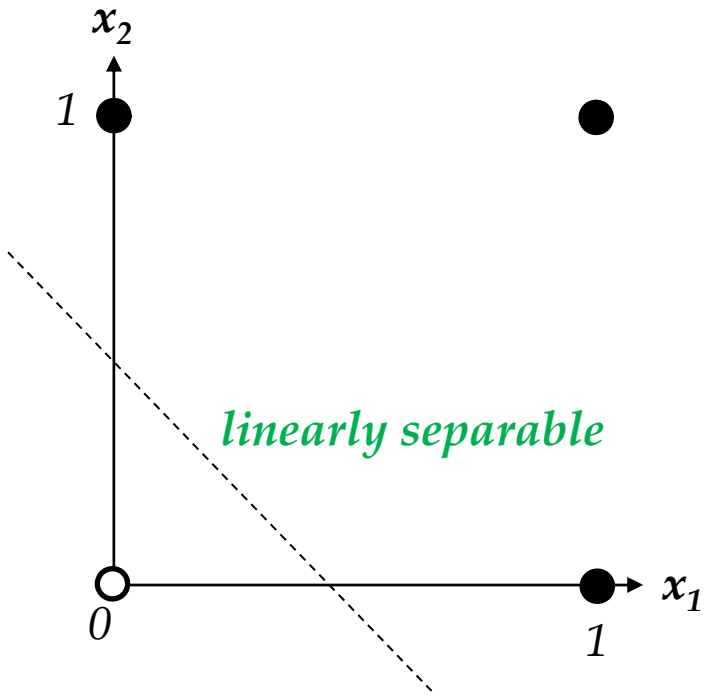
Perceptron for Classification

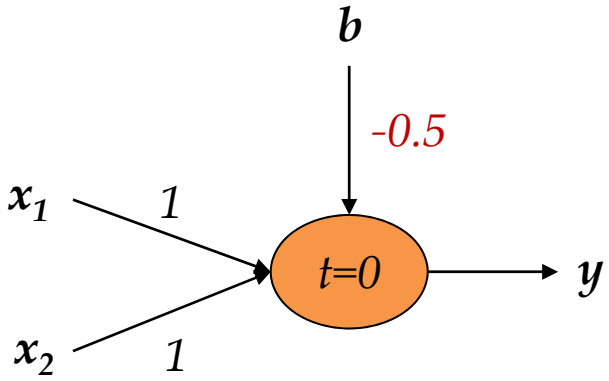
- **The perceptron is used for binary classification**

- **Training a perceptron for a classification task**
  - find suitable weights and bias in such a way that the training examples are correctly classified
  - geometrically try to find a single hyper-plane that separates the examples of the two classes

- **The perceptron can only model linearly separable classes**
  - it cannot model XOR function as it is non linearly separable

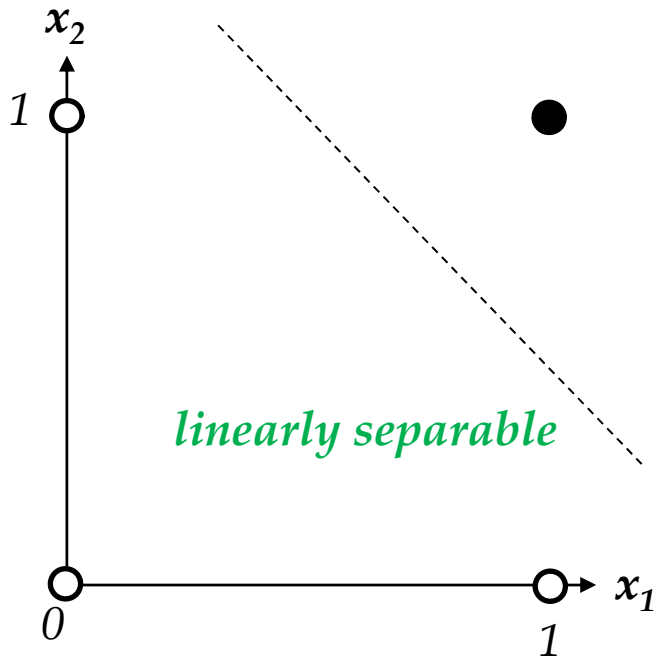
Boolean function OR – Linearly separable

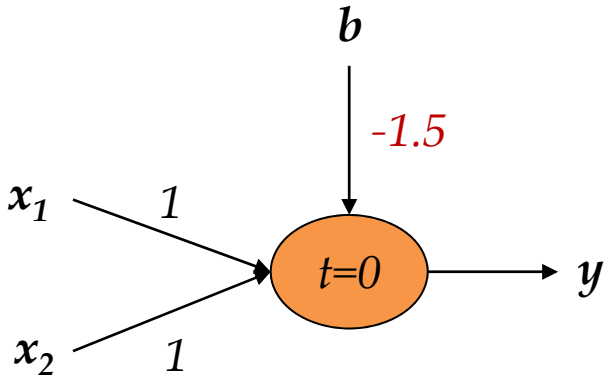




x_1	x_2	v	y
0	0	-0.5	0
1	0	0.5	1
0	1	0.5	1
1	1	1.5	1

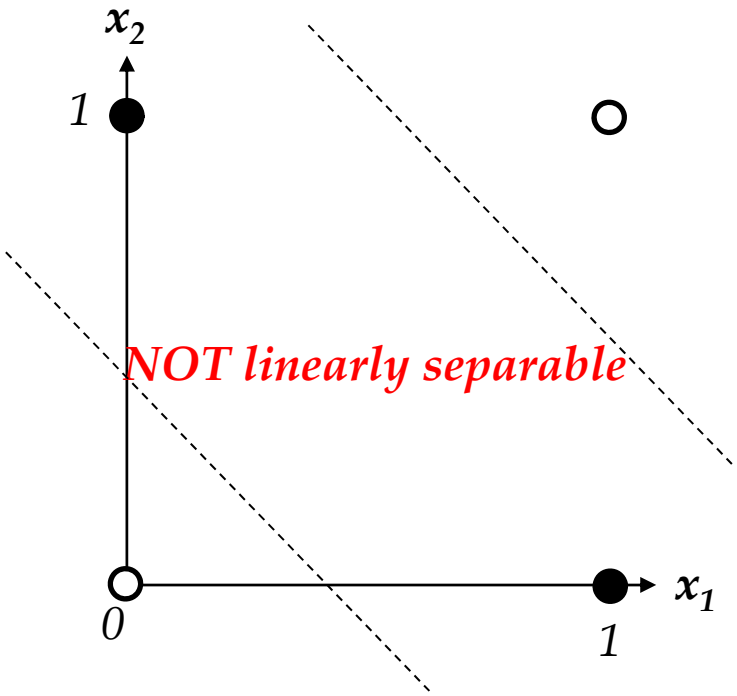
Boolean function AND – Linearly separable





x_1	x_2	v	y
0	0	-1.5	0
1	0	-0.5	0
0	1	-0.5	0
1	1	0.5	1

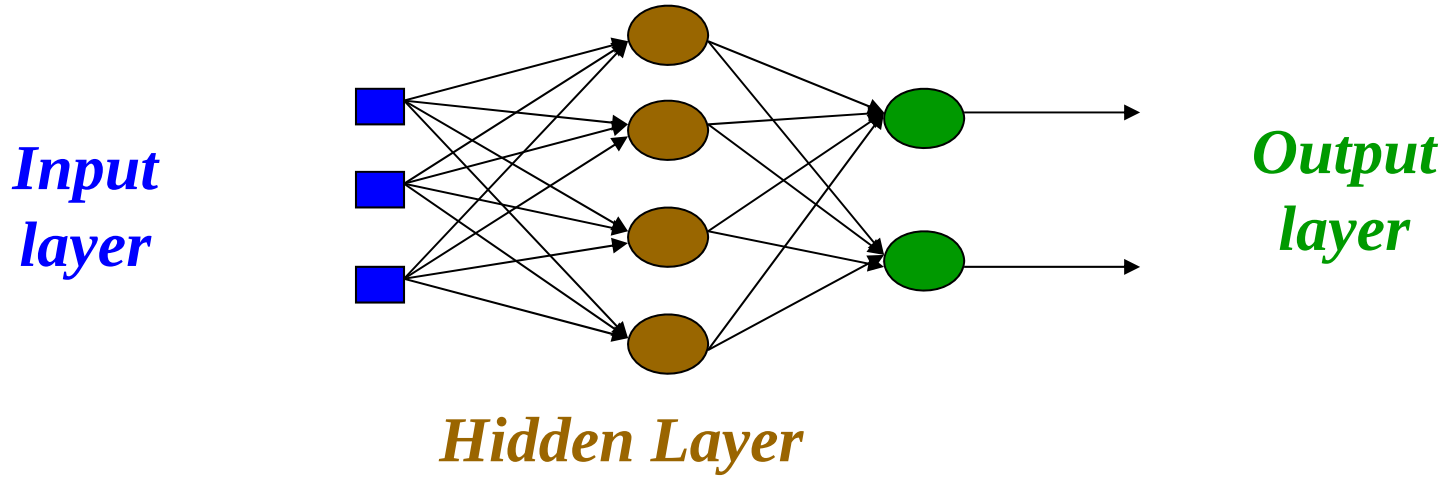
Boolean function XOR – NOT Linearly separable



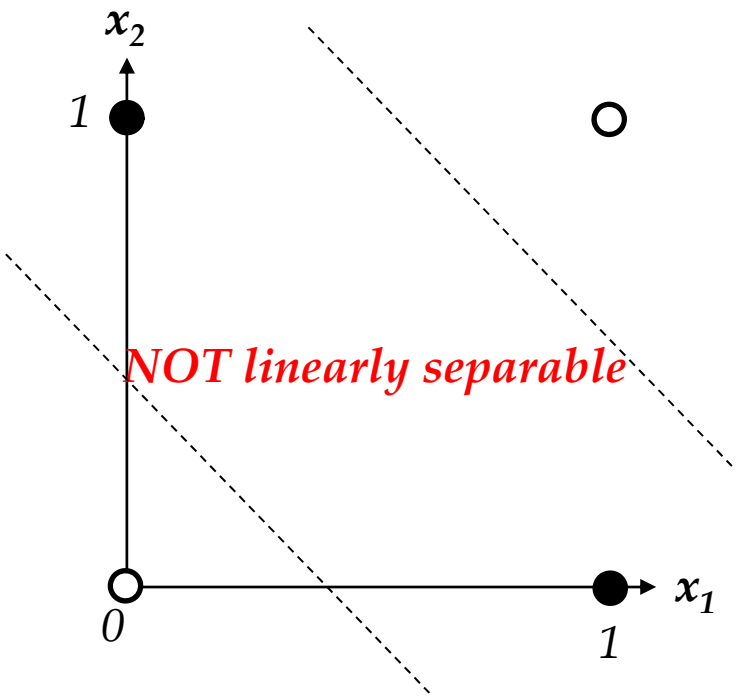
?

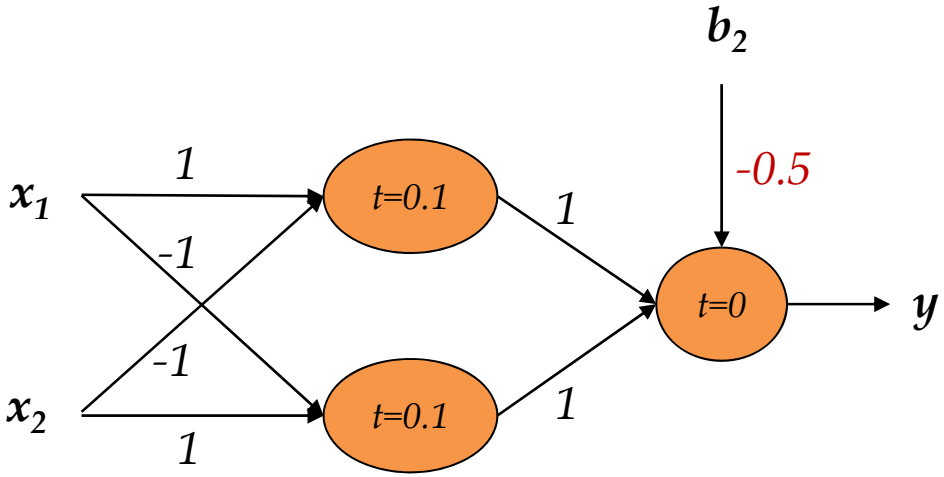
Multi Layer Feed-Forward (MLFF) NN

- MLFF is a more general network architecture, where there are hidden layers between input and output layers
- Hidden nodes do not directly receive inputs nor send outputs to the external environment
- MLFFs overcome the limitation of single-layer NN
- They can handle non-linearly separable learning tasks



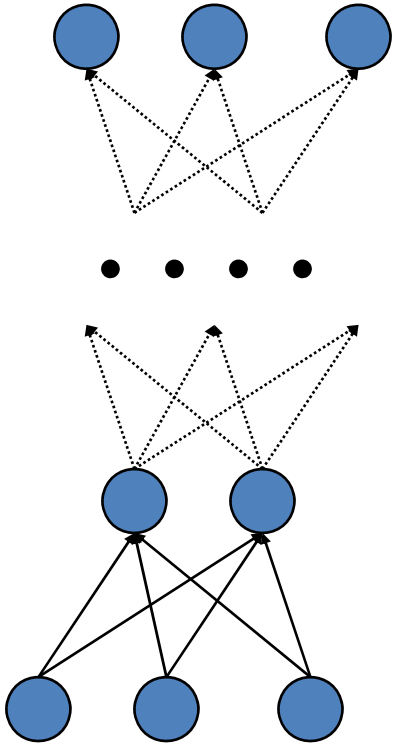
Boolean function XOR – NOT Linearly separable





x_1	x_2	H_1	H_2	v	y
0	0	0	0	-0.5	0
1	0	1	0	0.5	1
0	1	0	1	0.5	1
1	1	0	0	-0.5	0

Multilayer Perceptron



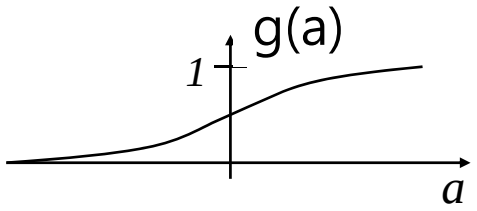
Output neurons

One or more layers of hidden units (hidden layers)

Input nodes

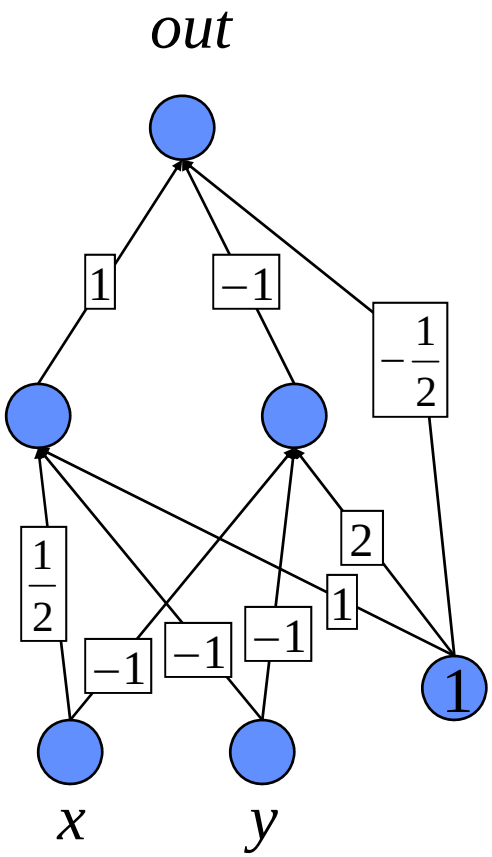
The most common output function (Sigmoid):

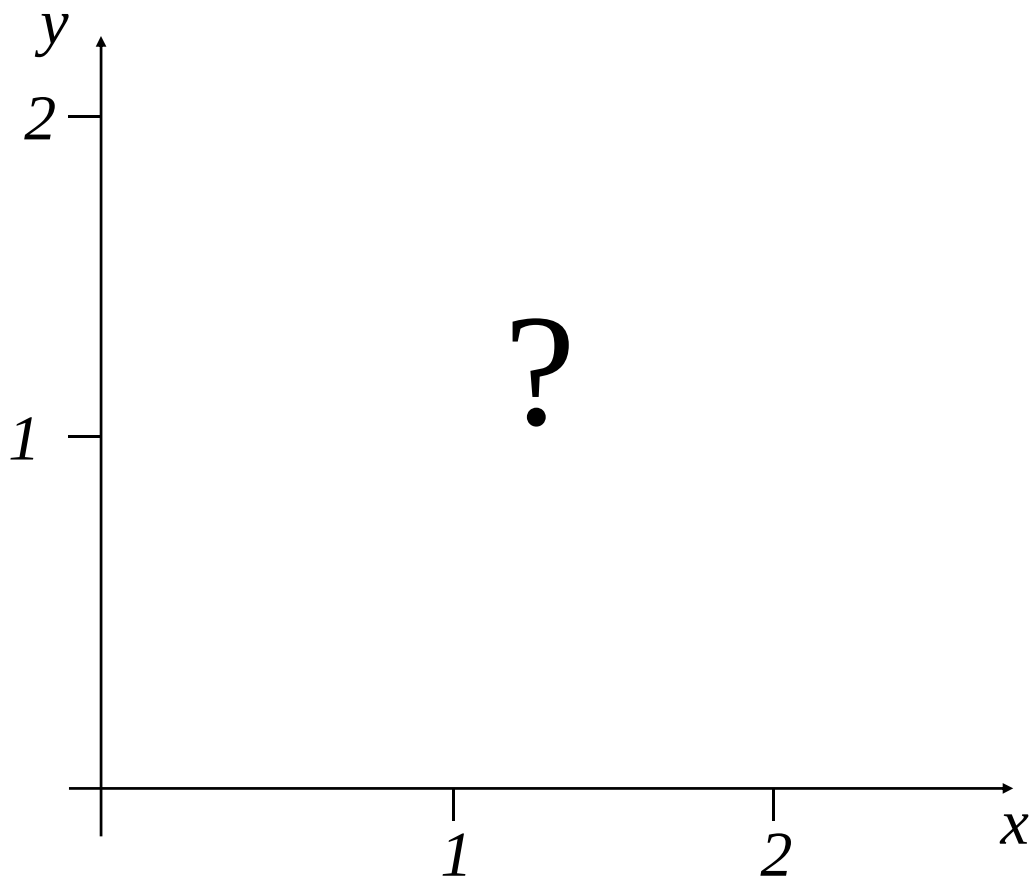
$$g(a) = \frac{1}{1+e^{-\beta a}}$$



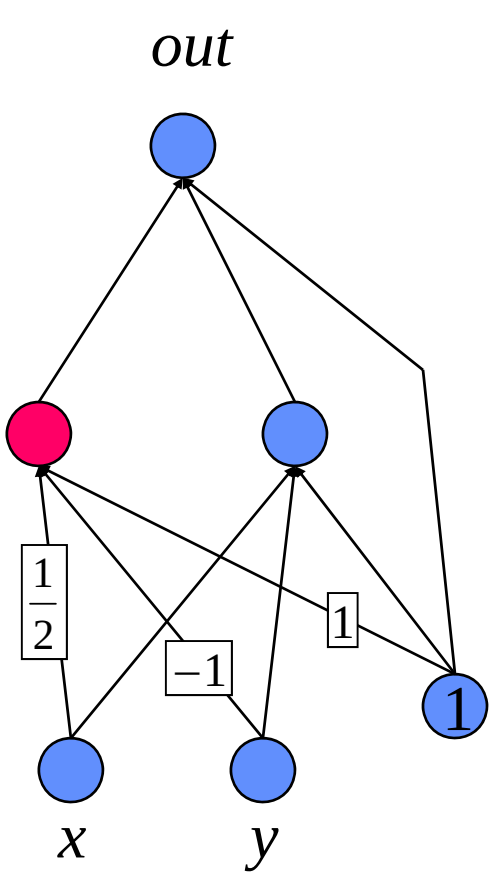
(non-linear squashing function)

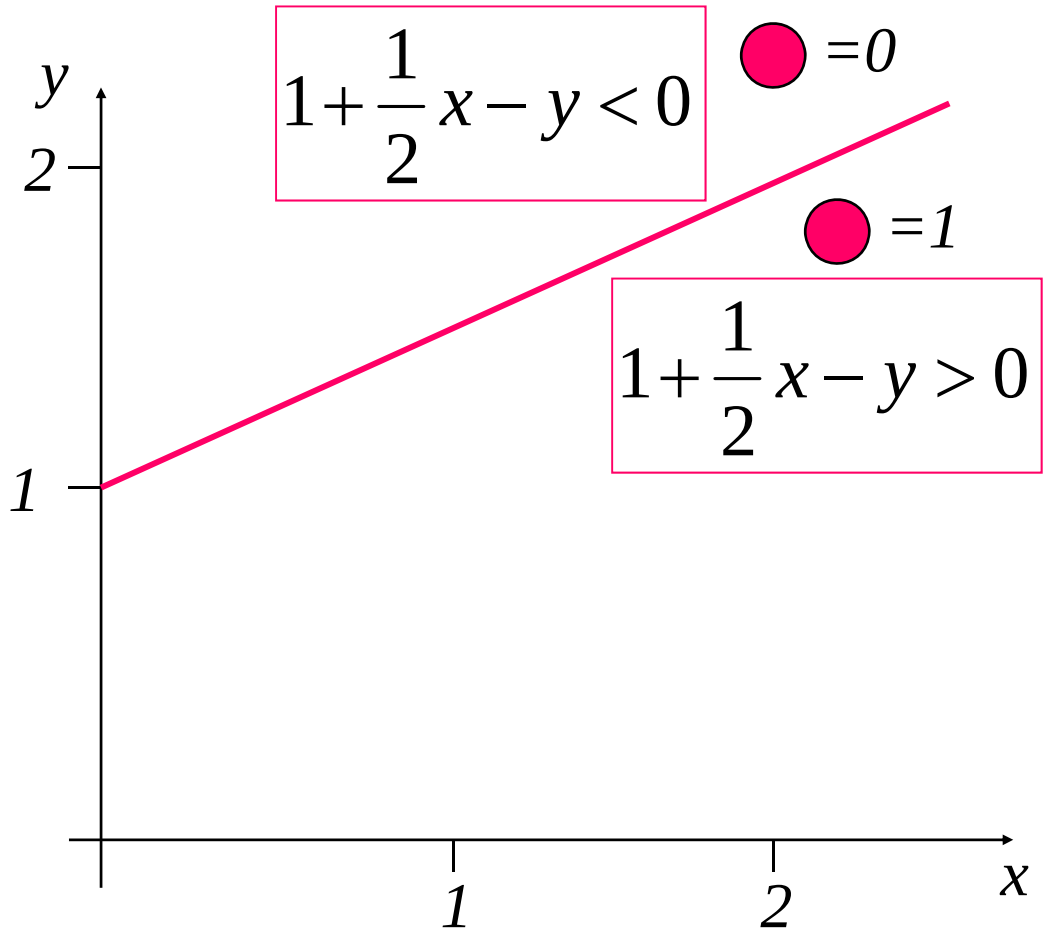
Example: Perceptrons as Constraint Satisfaction Networks



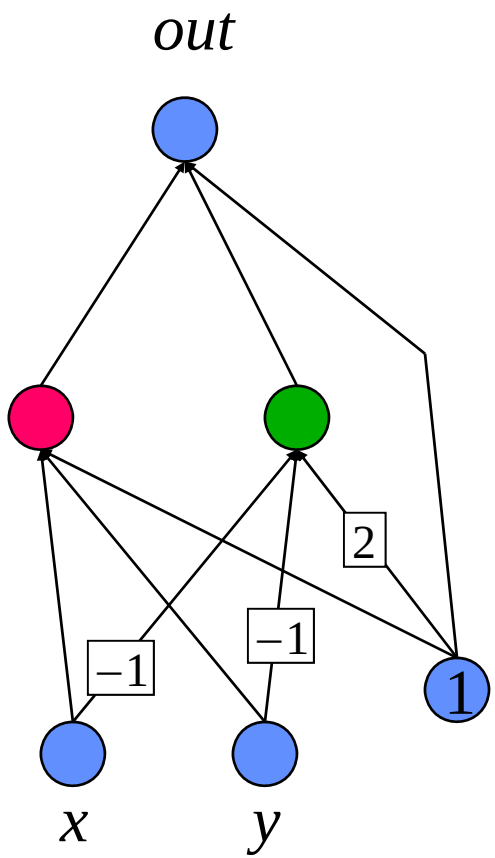


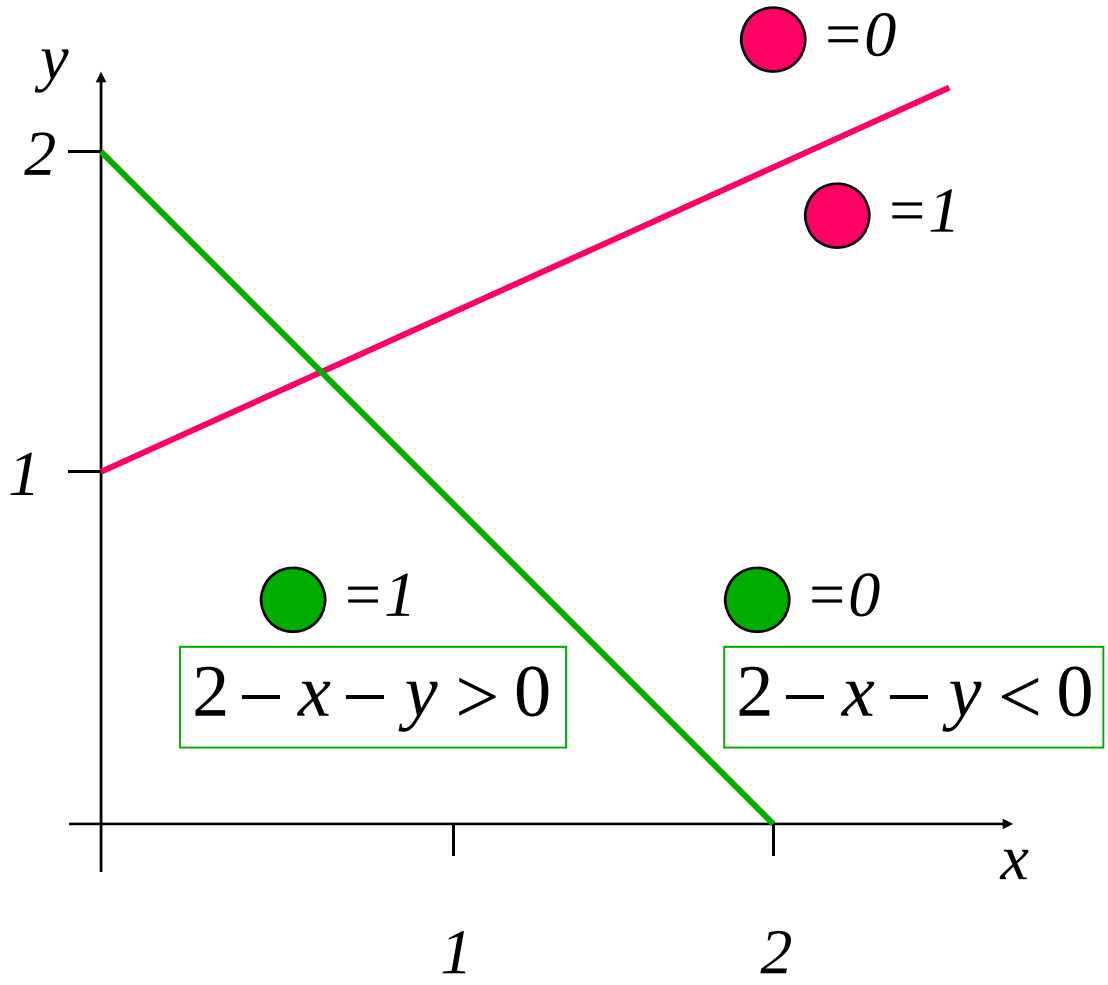
Example: Perceptrons as Constraint Satisfaction Networks



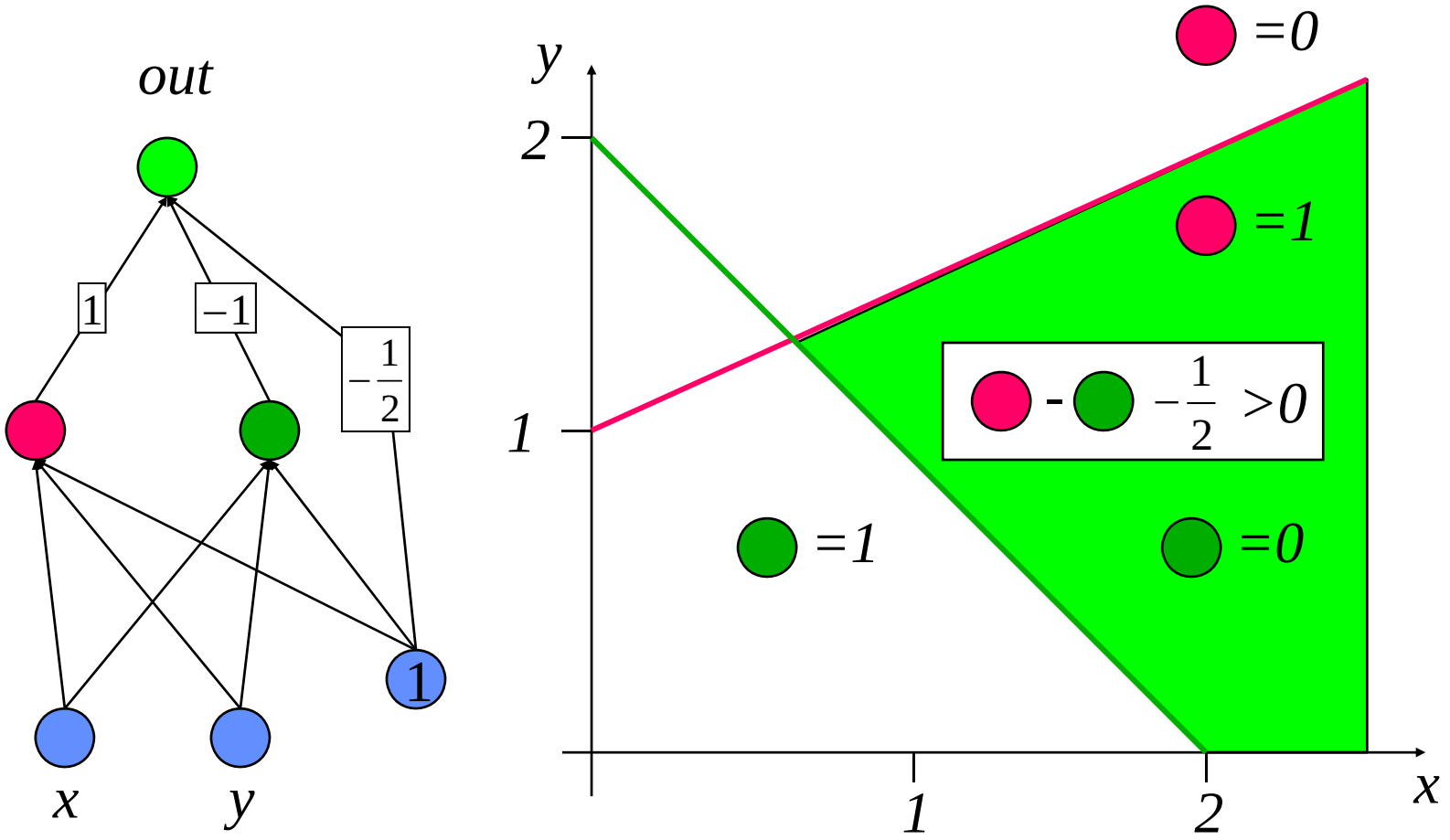


Example: Perceptrons as Constraint Satisfaction Networks

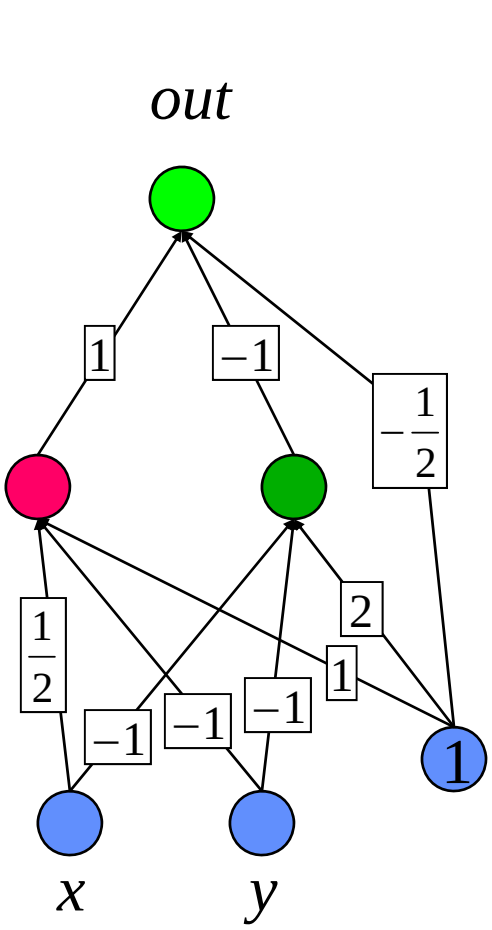


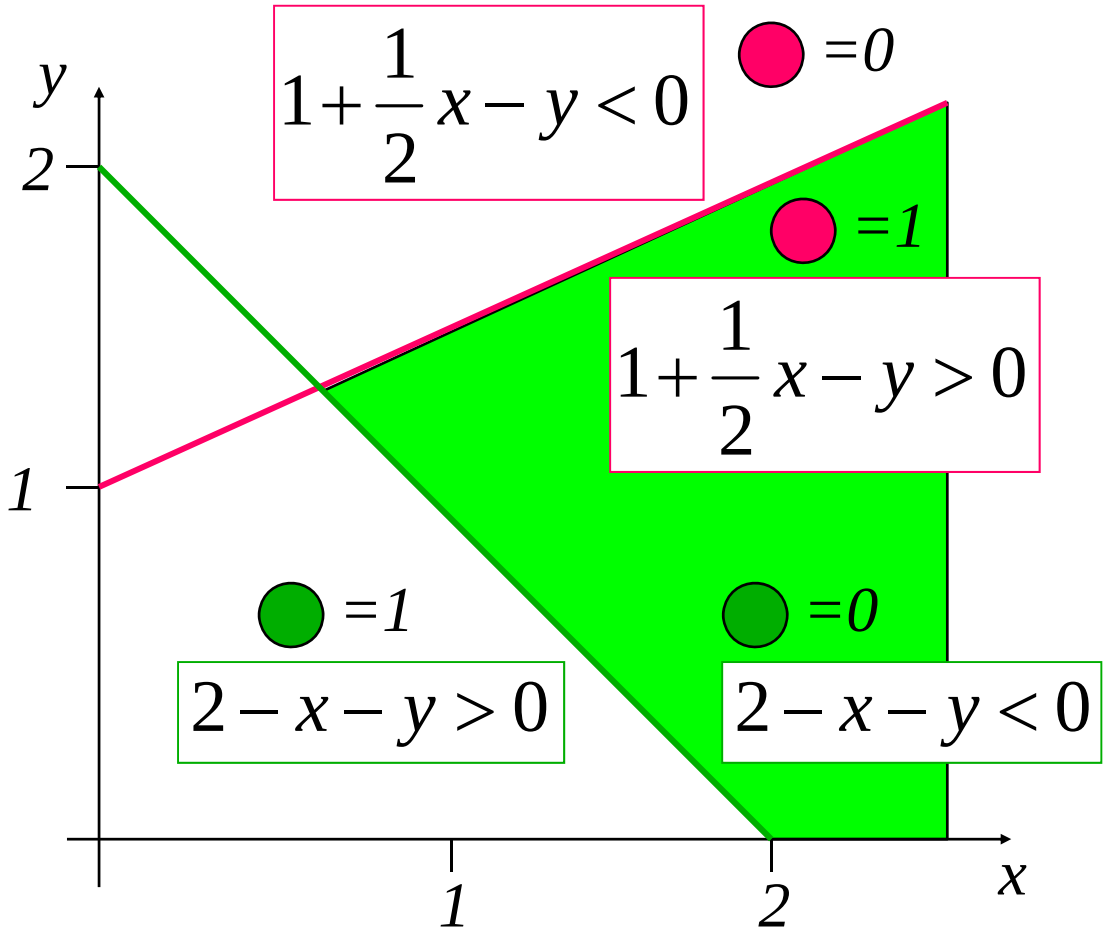


Example: Perceptrons as Constraint Satisfaction Networks



Perceptrons as Constraint Satisfaction Networks



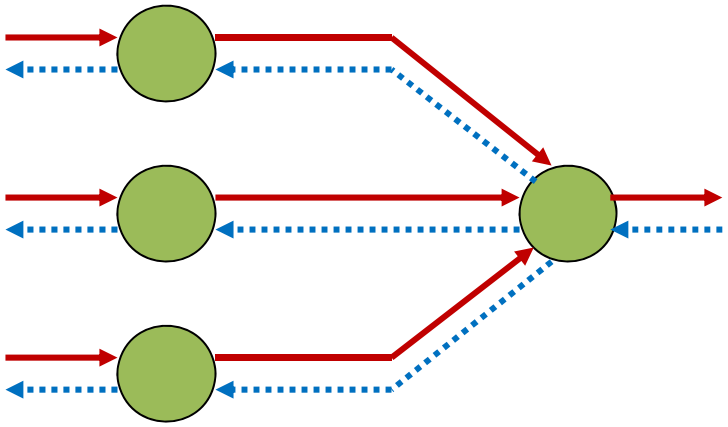


Training Algorithm: Backpropagation

- **Backpropagation searches for weight values that minimize the total error of the network over the set of training instances**
- **Backpropagation consists of the repeated application of the following two passes:**
  - Forward pass
    - NN is activated on one training instance
    - Error of (each neuron of) the output layer is computed
  - Backward pass
    - Error is used for updating the weights
    - Error is propagated backwards from the output layer through the network layer-by-layer
    - This is done by recursively computing the local gradient of each neuron

Backpropagation

- **Backpropagation adjusts the weights of the NN in order to minimize the network total mean squared error**



Network activation
Forward Step

Error propagation
Backward Step

Total Mean Squared Error

- **Error of output neuron *k* after the activation of the network on the *n*-th training instance (x(n), d(n)) is:**

$$e_k(n) = d_k(n) - y_k(n)$$

- **Network error is the sum of the squared errors of the output neurons:**

$$E(n) = \sum e_k^2(n)$$

- **Total mean squared error is the average of the network errors of the training instance**

$$E_{avg} = \left(\sum_{1 \le n \le N} E(n)\right) / N$$

Weight Update Rule

- **Backpropagation weight update rule is based on the gradient descent method**
  - It takes a step in the direction yielding the maximum decrease of the network error *E(n)*
  - This direction is the opposite of the gradient of *E(n)*
- **Iteration of the Backpropagation algorithm is usually terminated when**
  - sum of squares of errors of the output values for all training data in an epoch is less than some threshold (e.g., 0.01)

$$w_{ij} = w_{ij} + \Delta w_{ij}$$

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$$

learning rate

Backprop learning algorithm

n=1;

initialize **weights** randomly;

while (stopping criterion not satisfied or n < *max_iterations*)

one epoch

for each example (**x**,*d*)

- run the network with input x and compute the output y

- update the weights in backward order starting from those of the output layer: $w_{ij} = w_{ij} + \Delta w_{ij}$

with Δw_{ij} computed using the (generalized) Delta rule

end-for

n = n+1;

end-while;

Choice of learning rate

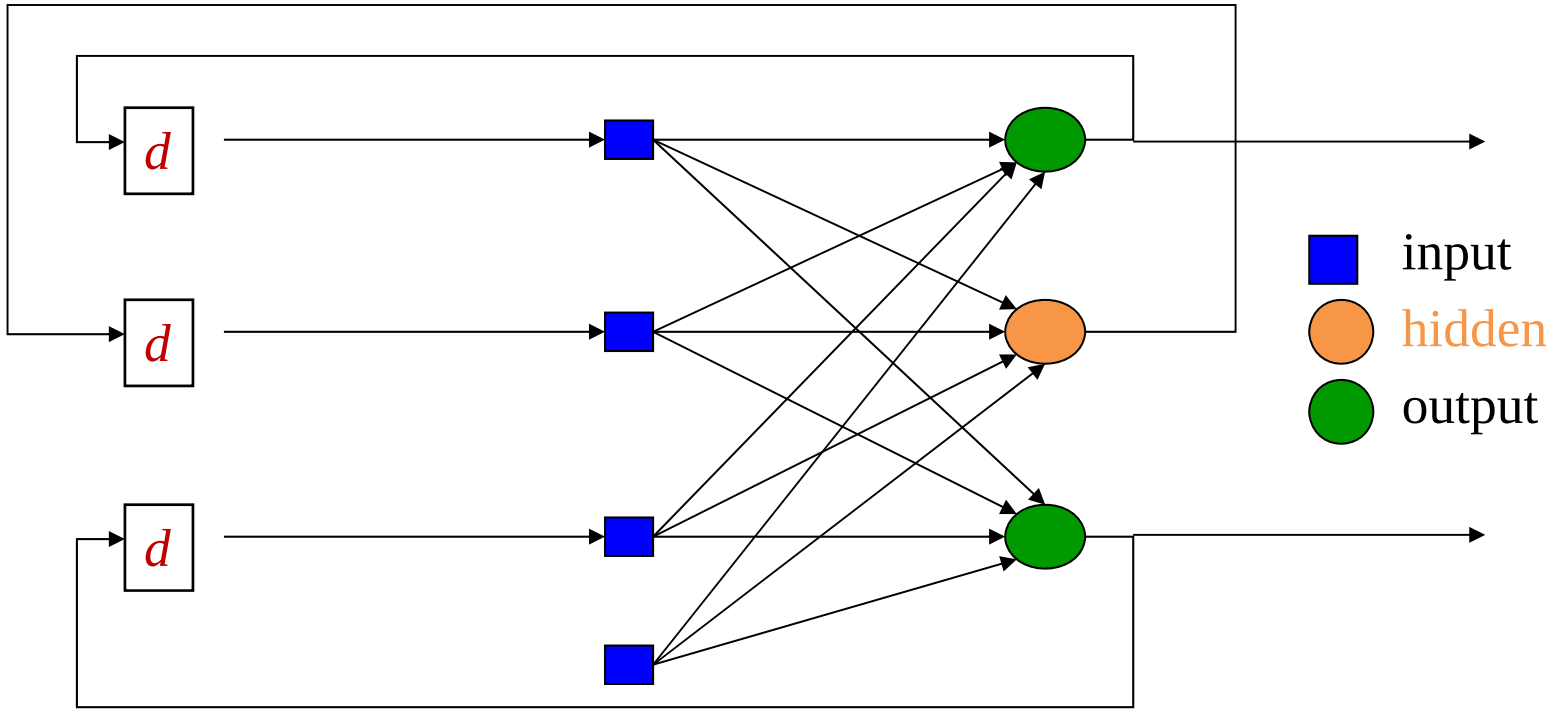
- **The right value of η depends on the application**
- **Values between 0.1 and 0.9 have been used in many applications**
  - Backpropagation is considered to be converged when the learning rate η is sufficiently small (e.g., in the range [0.1, 0.01])
- **Other heuristics is that adapt η during the training**

Recurrent Network

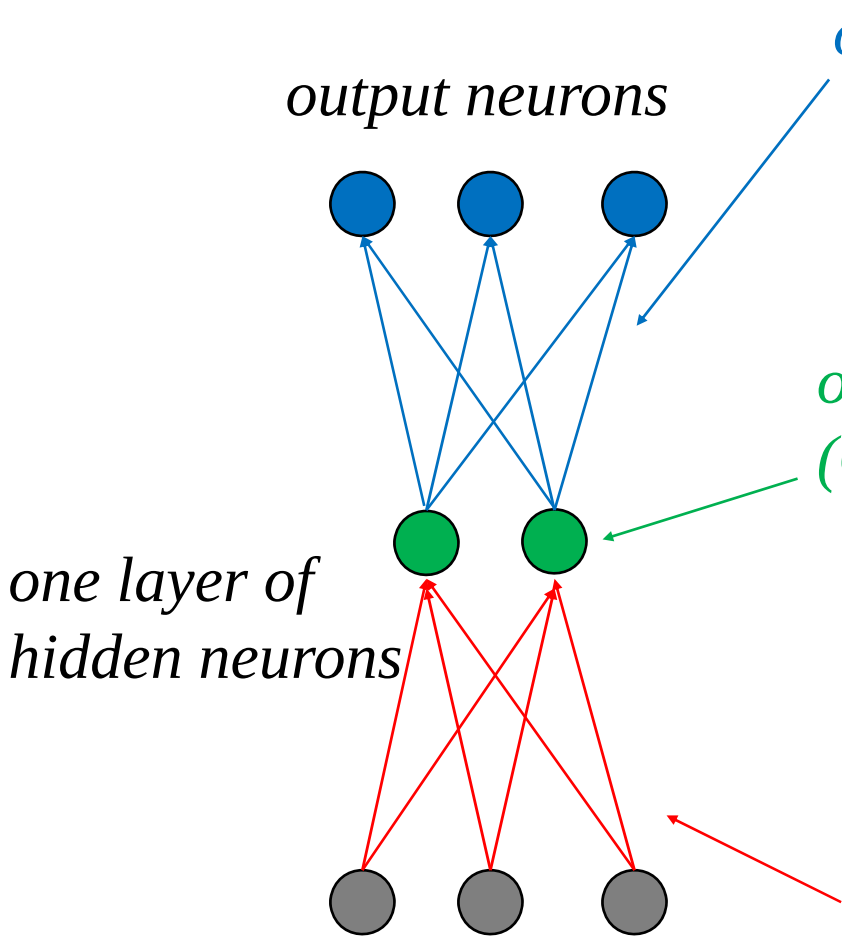
- **FFNN is acyclic where data passes from input to the output nodes and not vice versa**
  - once the FFNN is trained, its state is fixed and does not alter as new data is presented to it (*NOT having memory*)
- **Recurrent network can have connections that go backward from output to input nodes and models dynamic systems**
  - recurrent network's internal state can be altered as sets of input data are presented (*having memory*)
  - It is useful in solving problems where the solution depends not just on the current inputs but on all previous inputs
- **Applications**
  - predict stock market price
  - weather forecast

Recurrent Network Architecture

- **Recurrent Network with hidden neuron: unit *delay operator d* is used to model a dynamic system**



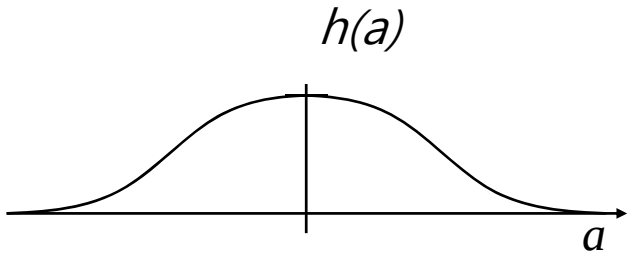
Radial Basis Function Networks



output of network:

$$\text{out}_j = \sum_i w_{i,j} h_i$$

output function:
(Gauss' bell-shaped function)



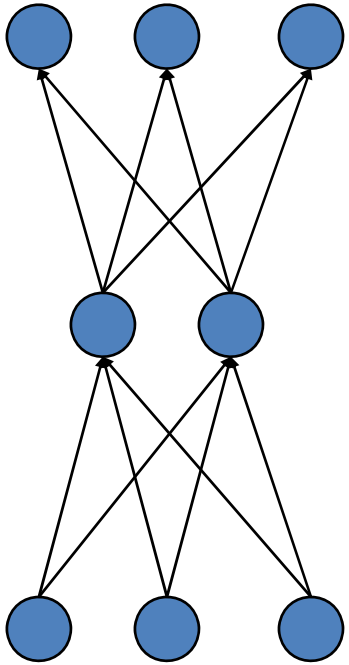
$$h(a) = e^{-\frac{a^2}{2\sigma^2}}$$

propagation function:

$$a_j = \sqrt{\sum_{i=1}^{n} (x_i - \mu_{i,j})^2}$$

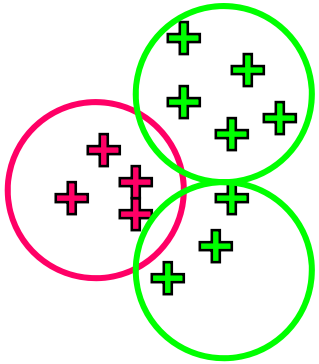
RBF Networks and Multilayer Perceptrons

output neurons

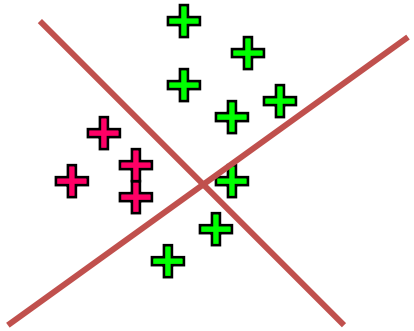


input nodes

RBF:

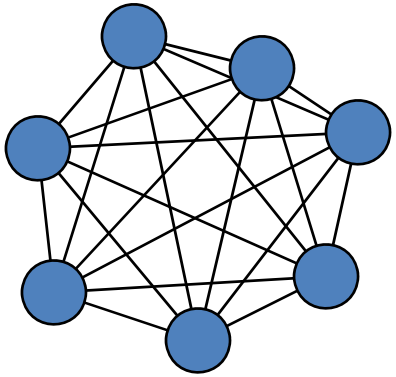


MLP:



Hopfield networks

- **Act as "autoassociative" memories to store patterns**
  - McCulloch-Pitts neurons with outputs -1 or 1, and threshold Θ
  - All neurons connected to each other
    - Symmetric weights ($w_{ij} = w_{ji}$) and $w_{ii} = 0$
  - Asynchronous updating of outputs
    - Let s_i be the state of unit i
    - At each time step, pick a random unit
    - Set s_i to 1 if $\Sigma_j w_{ij} s_j \geq \Theta$; otherwise, set s_i to -1



completely connected

Examples and Applications of ANN

- **NETalk (1987)**
  - Mapping character strings into phonemes so they can be pronounced by a computer
- **Neurogammon (Tesauro & Sejnowski, 1989)**
  - Backgammon learning program
- **Speech Recognition (Waibel, 1989)**
- **Character Recognition (LeCun et al., 1989)**
- **Face Recognition (Mitchell)**

ALVINN

- **Steer a van down the road**
  - 2-layer feedforward (using backpropagation for learning)
  - Raw input is 480 x 512 pixel image 15x per sec
  - Color image preprocessed into 960 input units
  - 4 hidden units
  - 30 output units, each is a steering direction

InfoLab

Thank you!

DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology