Artificial Intelligence

Fiona Yan Liu

Department of Computing

The Hong Kong Polytechnic University

Learning Outcomes of Lecture 05

- Understand the definition of machine learning and related concepts
- Understand the characters of clustering problem
- Master k-means clustering algorithm
- Understand the characters of classification
- Understand the design of max margin and its usage in support vector machines
- Know the rationale of non-linear problem solving using kernel mapping

Group Project

- Make a micro-movie for the 50th anniversary celebration of department of computing
  - Welcome video before the 50th banquet
  - The audience are VIP guest, alumni, staffs, and students
- Title
  - “1974 - 2024”
  - You are strongly encouraged to use different kinds of generative AI techniques
  - All materials should be collected by yourselves
- Length
  - Up to 2 min
- Submission
  - Each group only needs to submit one video by one group member via blackboard
  - Include your group information either in readme file or in the video
- Deadline
  - Thur. Nov. 7 23:59pm
  - 50% per day for late submission

Grading

- 28 points
  - 5 points: Screenplay
  - 5 points: Visual effects
  - 5 points: Audio effects
  - 3 points: Text and subtitles
  - 10 points: Overall performance
- Group work
  - 1-4 members/group
    - The number of members will not influence the grading
  - The group members are given same score
    - If not, please send the request to Fiona by email yan.liu@polyu.edu.hk with the justification and the signature of all group members
  - At least one group member should attend the demonstration on Nov. 12
- Lab on Nov. 5
  - Office hour for group project

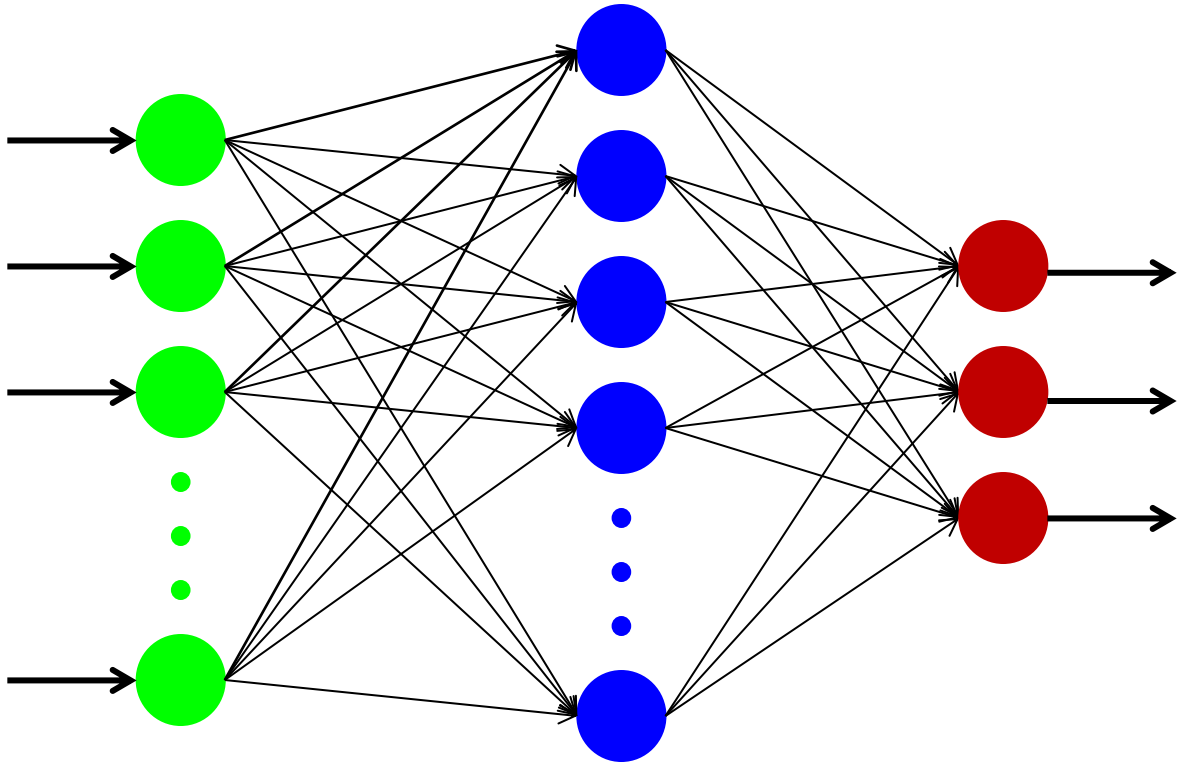
Agenda

- Basic structure
- Artificial neuro
- Perceptron
- Multilayer perceptron

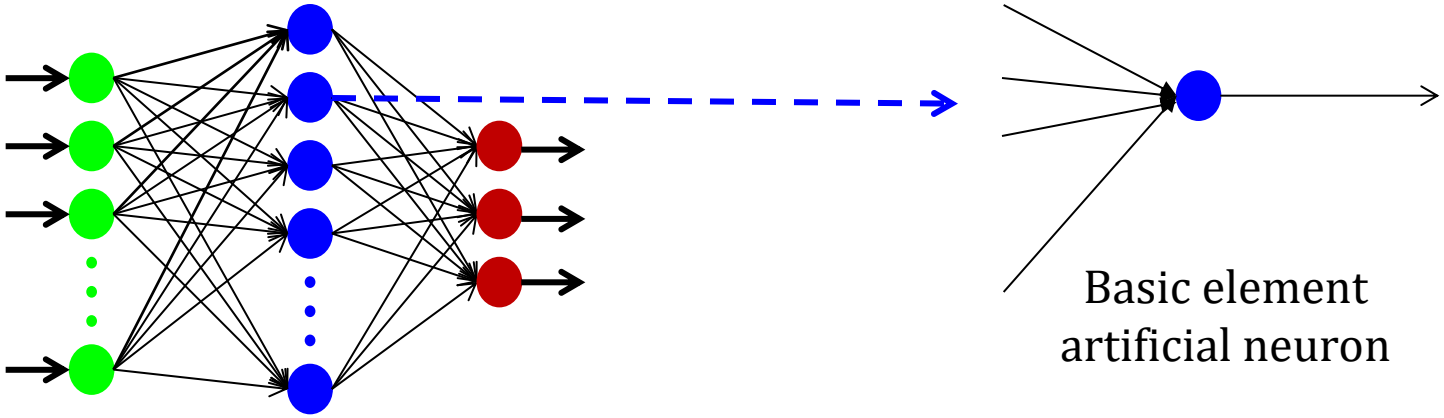
Agenda

- Basic structure
- Artificial neuro
- Perceptron
- Multilayer perceptron

A Typical Artificial Neural Networks



Basic Element

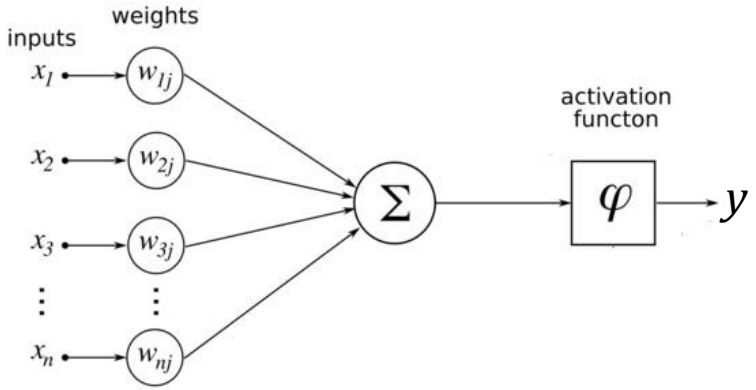


Basic element
artificial neuron

An artificial neuron transforms weighted input nonlinearly:

$$y = \varphi(\mathbf{w}^T \mathbf{x})$$

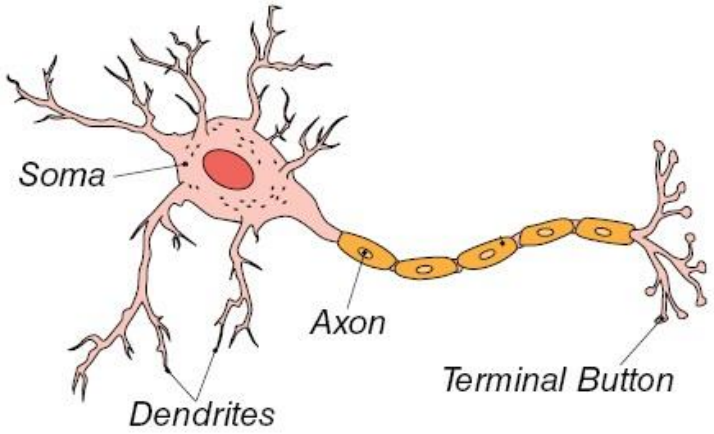
φ is called activation function



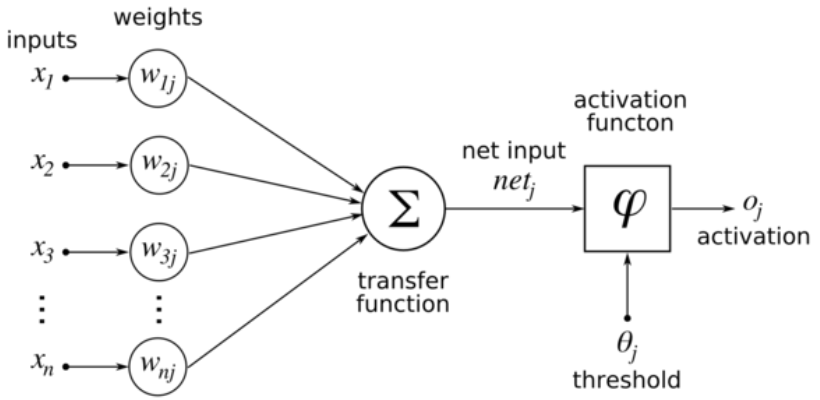
Agenda

- Basic structure
- Artificial neuro
- Perceptron
- Multilayer perceptron

Artificial Neuron



Biological neuron



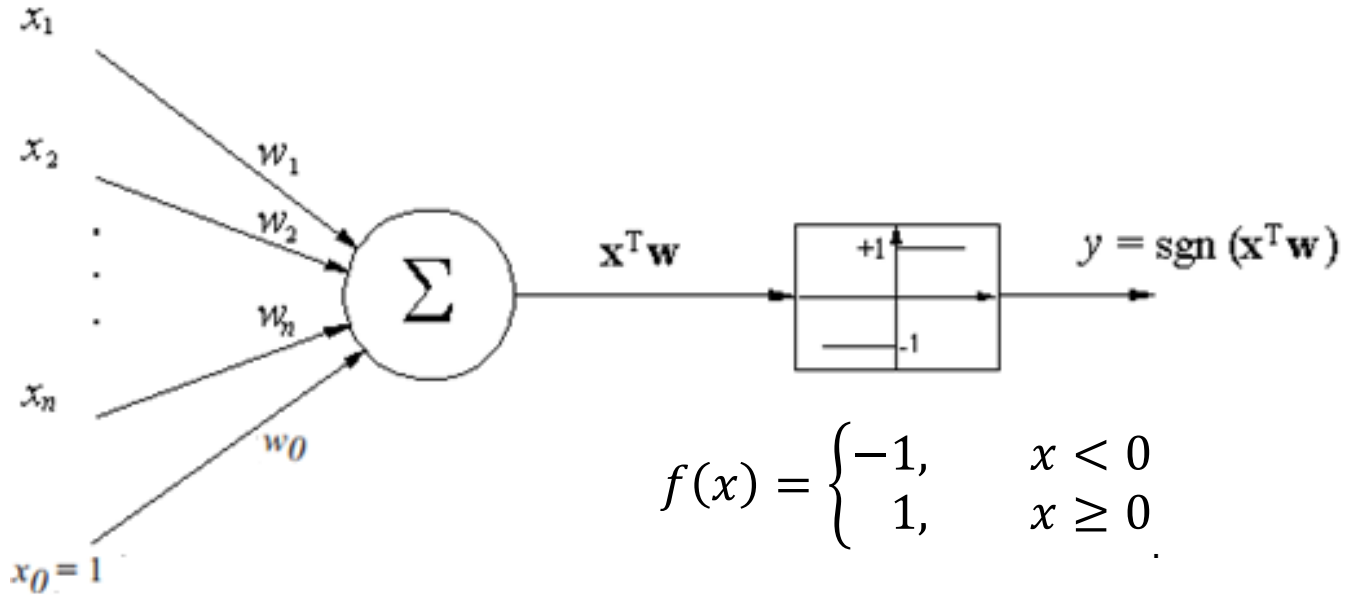
Artificial neuron

- Why input is a vector?
- Why sum function?
- Why outputs the same value to different inputs?
- Why activation function?
- Why weight?

Agenda

- Basic structure
- Artificial neuro
- Perceptron
- Multilayer perceptron

Perceptron with Activation Function – Sign Function

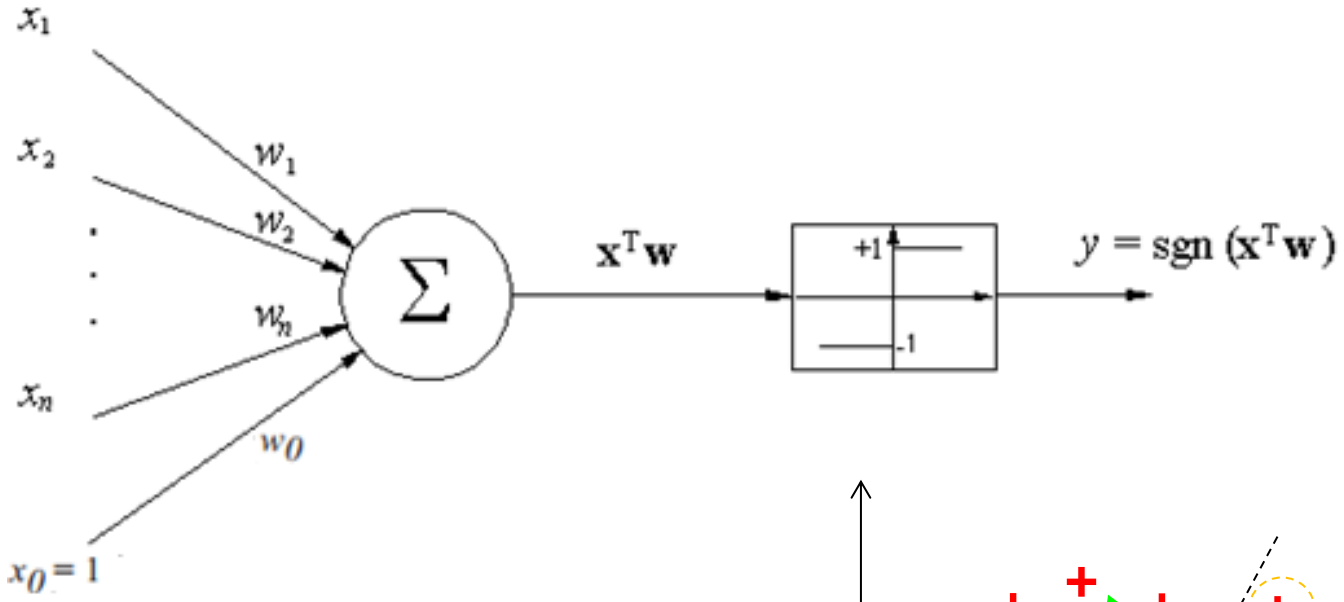


$$f(x) = \begin{cases} -1, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

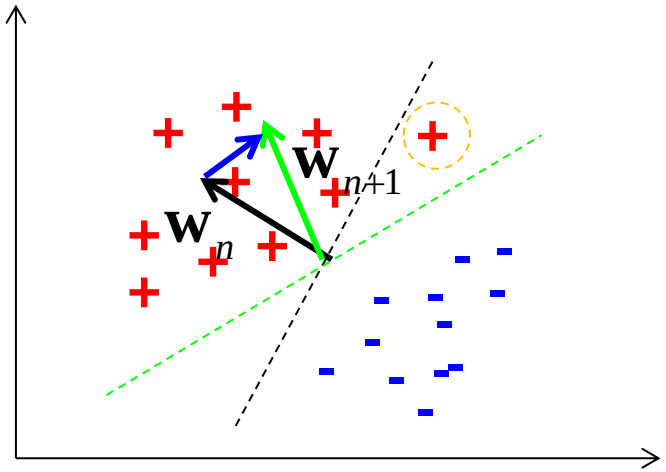
- Input layer and output layer
- Sign function for classification
- Assume $\{\boldsymbol{x}_d\}_{i=1}^N$ are N-dimensional vector and the labels $y_d \in \{-1, +1\}$
- Perceptron aims to find a weight w satisfying

$$y_d = \text{sgn}(\boldsymbol{x}_d^T \boldsymbol{w})$$

Perceptron Learning Algorithm

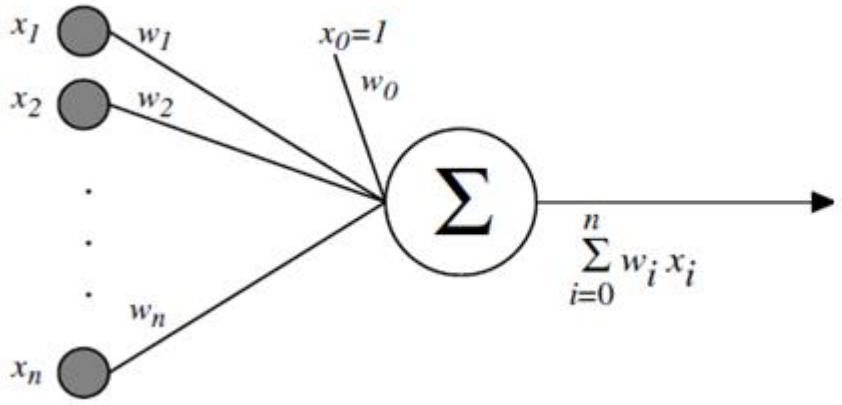


- Initialize random $\boldsymbol{w}_0$;
- For each $(\boldsymbol{x}_d, \mathrm{y}_d)$, η is the learning rate
  - Calculate the estimated label $\widetilde{y_d} = \text{sgn}(\boldsymbol{x}_d^T \boldsymbol{w}_k)$
  - Update the weight $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \eta(y_d - \widetilde{y_d})\boldsymbol{x}_\mathrm{d}$
- The algorithms stops until $\boldsymbol{w}$ converges.



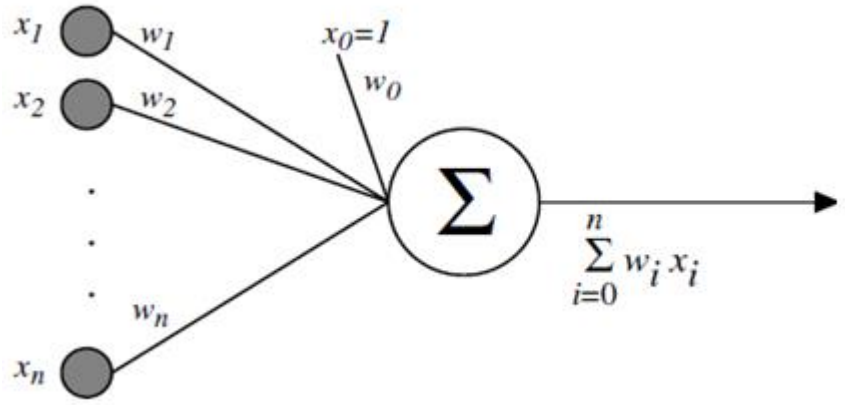
Perceptron without Activation Function

- Linear regression
- With one-dimensional input
  - $o = w_1 x + w_0$
- Loss function
  - $Loss(h_w) = \sum_d (t_d - (w_1 x_{1,d} + w_0))^2 / 2D$

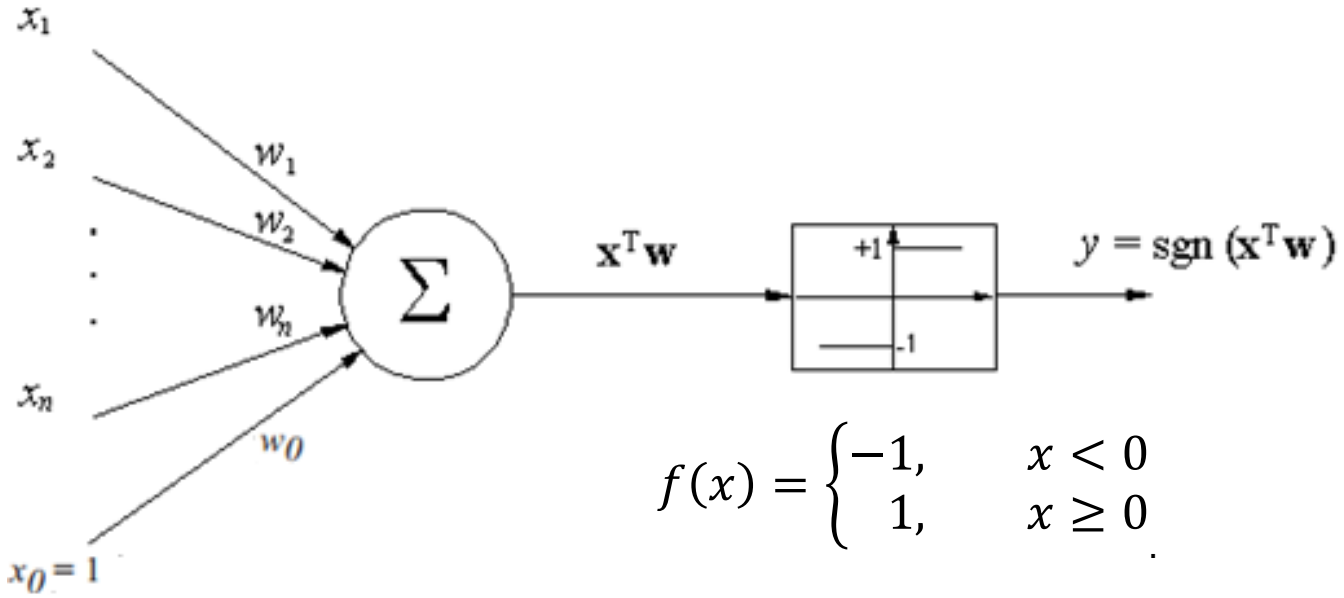


Linear Combination with One-dimensional Input

- The partial derivatives of $Loss(h_w)$ with respect to w_0 and w_1 are zero:
  - $\partial \sum_d (t_d - (w_1 x_{1,d} + w_0))^2 / \partial w_0 = 0$
  - $\partial \sum_d (t_d - (w_1 x_{1,d} + w_0))^2 / \partial w_1 = 0$
- These equations has analytic solutions
  - $w_1 = (D(\sum x_{1,d} t_d) - (\sum x_{1,d})(\sum t_d)) / (D(\sum x_{1,d}^2) - (\sum x_{1,d})^2)$
  - $w_0 = (\sum t_d - w_1 (\sum x_{1,d})) / D$



Perceptron with Sign Function



$$f(x) = \begin{cases} -1, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

- Nonlinear activation function

$$y_d = \text{sgn}(\boldsymbol{x}_d^T \boldsymbol{w})$$

- Only labels $y_d \in \{-1, +1\}$ are known

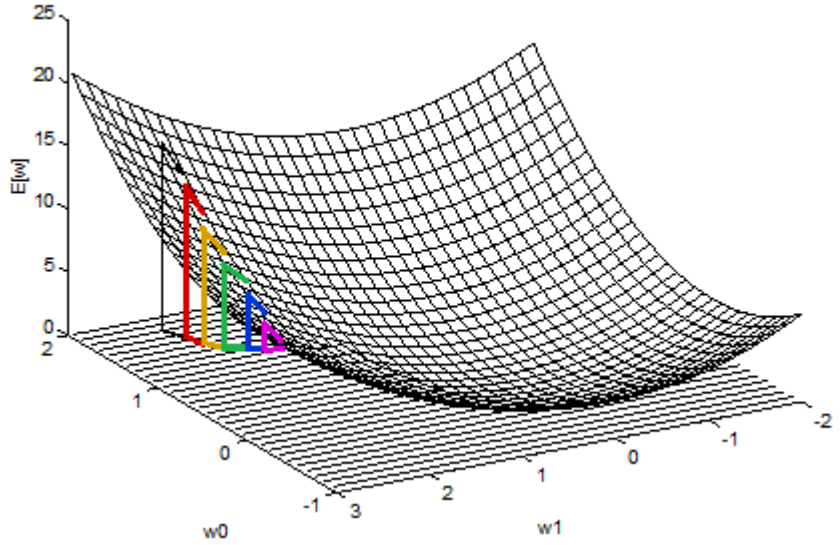
$$\boldsymbol{arg\,min}_{\boldsymbol{w}} \frac{1}{2D} \sum ||\mathbf{y}_d - f(\mathbf{w}, \mathbf{x}_d)||^2$$

Gradient Descent

- Objective function

$$\boldsymbol{arg\,min}_{\boldsymbol{w}} \frac{1}{2D} \sum ||t_d - (wo + w_1 x_d)||^2$$

- Linear combination with one-dimensional Input
  - $o = w_1 x + w_0$



- Gradient
  - $\nabla E(\boldsymbol{w}) = [\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots \dots \frac{\partial E}{\partial w_n}]$
- Gradient decent
  - $\Delta \boldsymbol{w} = -\eta \nabla E(\boldsymbol{w})$
- Training of perceptron with Sgn function
  - $\Delta \boldsymbol{w} = -\eta [\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots \dots \frac{\partial E}{\partial w_n}]$

$$= -\eta [\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots \dots \frac{\partial E}{\partial w_n}]$$

$$= -\eta \frac{1}{2D} \sum_{d=1}^{D} ||t_d - (wo + w_1 x_d)||^2$$

$$= -\eta \frac{1}{2D} \sum_{d=1}^{D} 2 * (t_d - od) * \frac{\partial (td - o_d)}{\partial w_i}$$

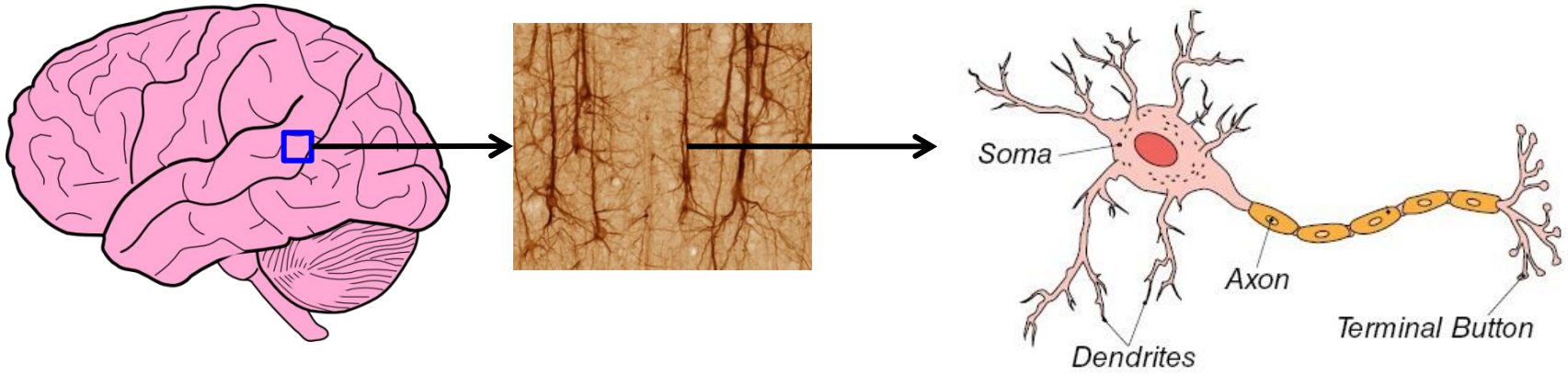
$$= -\eta \frac{1}{D} \sum_{d=1}^{D} \big((t_d - od)\big) * (0 - \frac{\partial (o_d)}{\partial w_i})$$

$$= \eta \frac{1}{D} \sum_{d=1}^{D} (t_d - od) * \boldsymbol{x}_d$$

Agenda

- Basic structure
- Artificial neuro
- Perceptron
- Multilayer perceptron

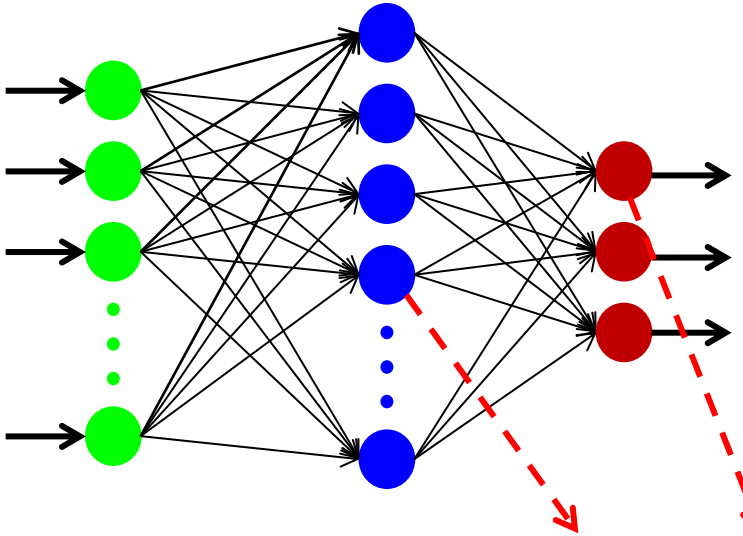
Difference of Computation Ability



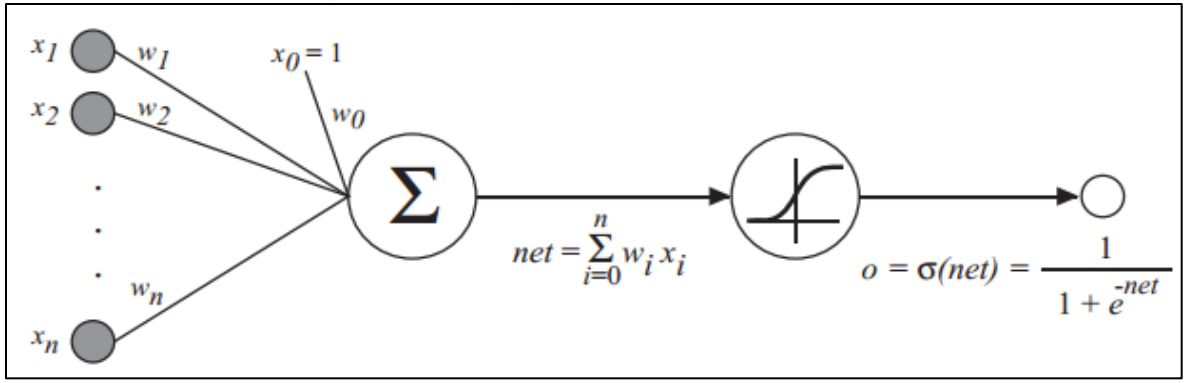
Brain Cerebral cortex Neuron

- Each neuron is a cell that uses biochemical reactions to receive, process and transmit information
- Cerebral cortex has the multiple layer structure
- The brain is a collection of about 10 billion interconnected neurons, with approximately 60 trillion connections

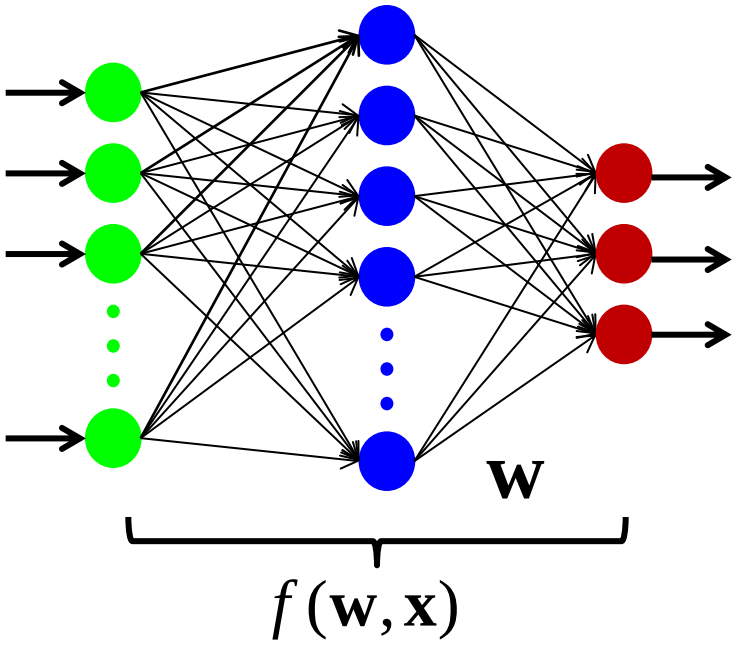
Multilayer Perceptron (MLP)



- One or several hidden layers
- Continuous activation function, sigmoid
- Learning with training data
- Powerful ability for classification and function approximation



MLP Learning: Back Propagation

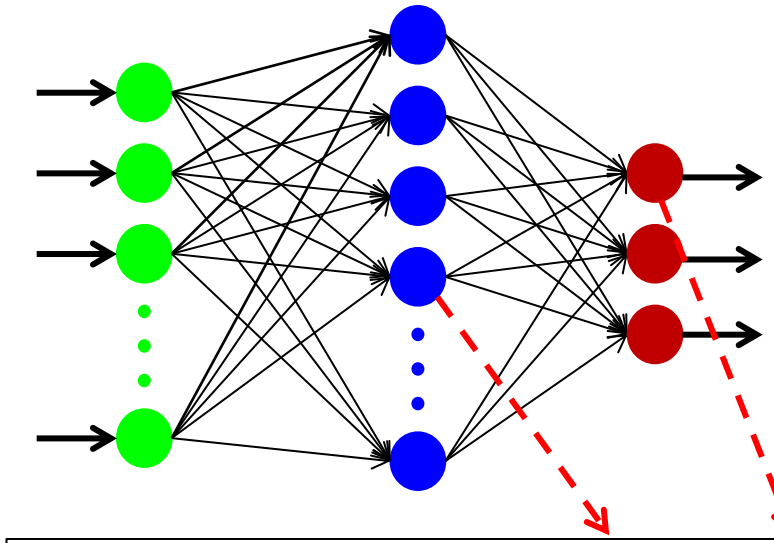


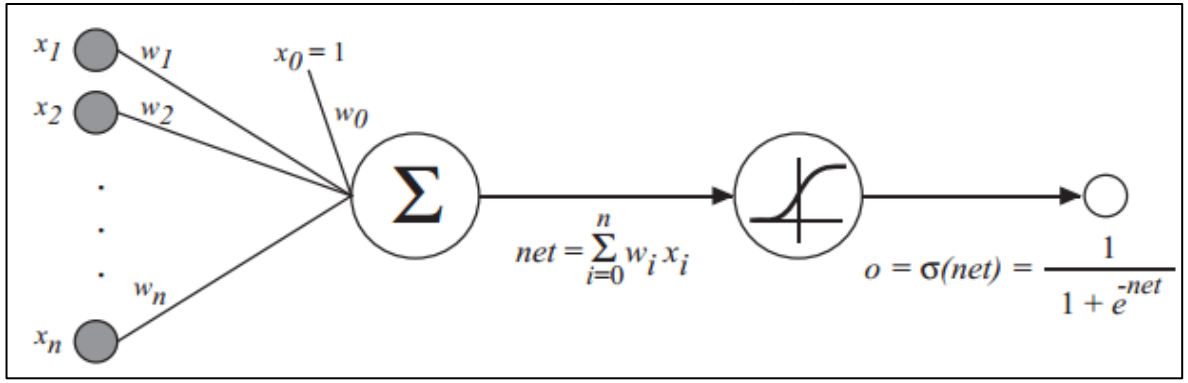
- Back propagation searches weights **w**
  - to minimize the total error of the network over the set of training examples
  - Gradient descent based method
- Backpropagation consists of two repeated process
  - forward process
  - error propagation process

$\{(\mathbf{x}_d, \mathbf{y}_d)\}_{d=1}^{D}$ is the training set, back propagation tries to minimize

$$\boldsymbol{arg\,min}_{w} \frac{1}{2D} \sum ||\mathbf{y}_d - f(\mathbf{w}, \mathbf{x}_d)||^2$$

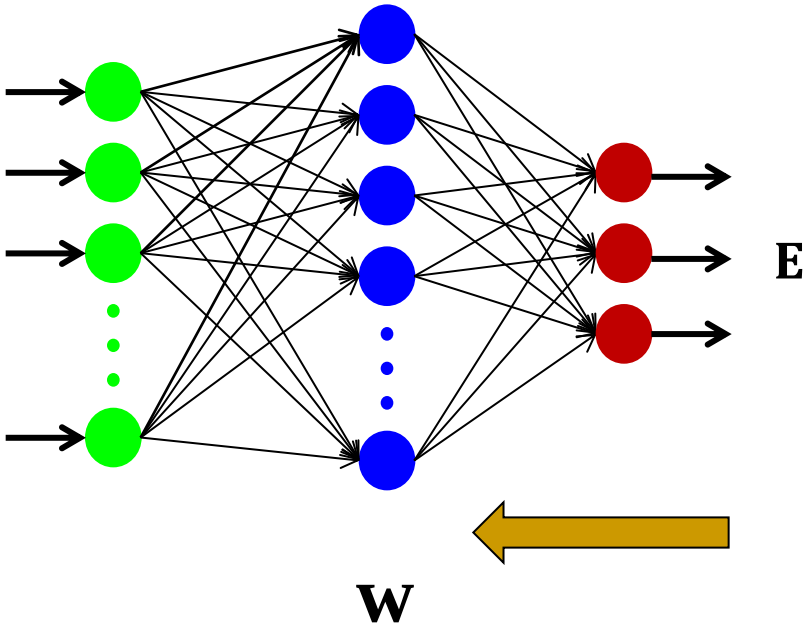
Forward Processing





- Initialization
  - Assign the weight for each level by random number
- Forward processing
  - Calculate the outputs according to the activation function for each layer
  - $o_j = \frac{1}{1+e^{-net}} = \frac{1}{1+e^{-\sum_{i=0}^{n} w_{ij} x_i}}$
  - $\widetilde{y_p} = \frac{1}{1+e^{-net}} = \frac{1}{1+e^{-\sum_{j=0}^{M} w_{jp} o_j}}$

Error Propagation



- For each output neuro p

$$w_{jp}(k+1) = w_{jp}(k) + \eta\ \delta_p o_j$$

$$Def.\ \delta_p = \left(y_p - \widetilde{y_p}\right)\widetilde{y_p} * (1 - \widetilde{y_p})$$

- For each hidden neuro j

$$w_{ij}(k+1) = w_{ij}(k) + \eta\ \delta_j x_i$$

$$Def.\ \delta_j = \left(\sum_p w_{jp}\ \delta_p\right)\widetilde{o_j} * (1 - \widetilde{o_j})$$

$$w_{jp}(k+1) = w_{jp}(k) + \Delta w_{jp}$$

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}$$

Gradient Descent for Sigmoid

- For each output neuro p

$$\widetilde{y_p} = \frac{1}{1+e^{-net}}$$

$$= \frac{1}{1 + e^{-\sum_{j=0}^{M} w_{jp} o_j}}$$

- Gradient decent

$$w_{jp}(k+1) = w_{jp}(k) + \Delta w_{jp}$$

$$\Delta w_{jp} = -\eta \, \nabla E(w_{jp})$$

- Update the weights for one training data

$$\Delta w_{jp} = \eta \, (y_p - \widetilde{y_p})\widetilde{y_p} * (1 - \widetilde{y_p}) * o_j$$

$$Def. \, \delta_p = (y_p - \widetilde{y_p})\widetilde{y_p} * (1 - \widetilde{y_p})$$

$$\Delta w_{jp} = \eta \, \delta_p o_j$$

$$\frac{\partial E}{\partial w_{jp}} = \frac{\partial}{\partial w_{jp}} \frac{1}{2D} \sum_{d \in D} (y_{p,d} - \widetilde{y_{p,d}})^2$$

$$= \frac{1}{2D} \sum_{d \in D} \frac{\partial}{\partial w_{jp}} (y_{p,d} - \widetilde{y_{p,d}})^2$$

$$= \frac{1}{2D} \sum_{d \in D} 2 * (y_{p,d} - \widetilde{y_{p,d}}) \frac{\partial}{\partial w_{jp}} (y_{p,d} - \widetilde{y_{p,d}})$$

$$= \frac{1}{D} \sum_{d \in D} (y_{p,d} - \widetilde{y_{p,d}}) \frac{\partial}{\partial w_{jp}} (-\widetilde{y_{p,d}})$$

$$= -\frac{1}{D} \sum_{d \in D} (y_{p,d} - \widetilde{y_{p,d}}) \frac{\partial \widetilde{y_{p,d}}}{\partial net_d} * \frac{\partial net_d}{\partial w_{jp}}$$

$$= -\frac{1}{D} \sum_{d \in D} (y_{p,d} - \widetilde{y_{p,d}}) \widetilde{y_{p,d}} * (1 - \widetilde{y_{p,d}}) * o_{j,d}$$