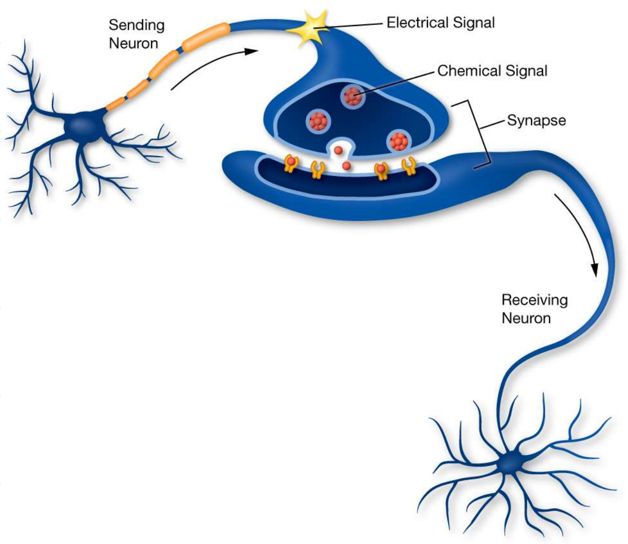
3. Neural networks and Deep learning

2017년 1월 6일 금요일　　오후 3:59

Reference : Michael A. Nielson. (2015), "Neural Networks and Deep Learning",
http://www.neuralnetworksanddeeplearning.com.
Mark F. Bear et. al. (2016). Neuroscience : exploring the brain 4th ed. Wolters Kluwer.

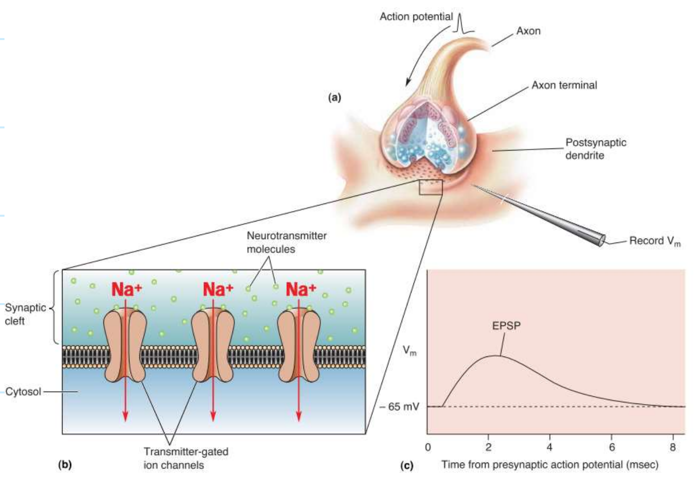
3.1. How do neurons work?

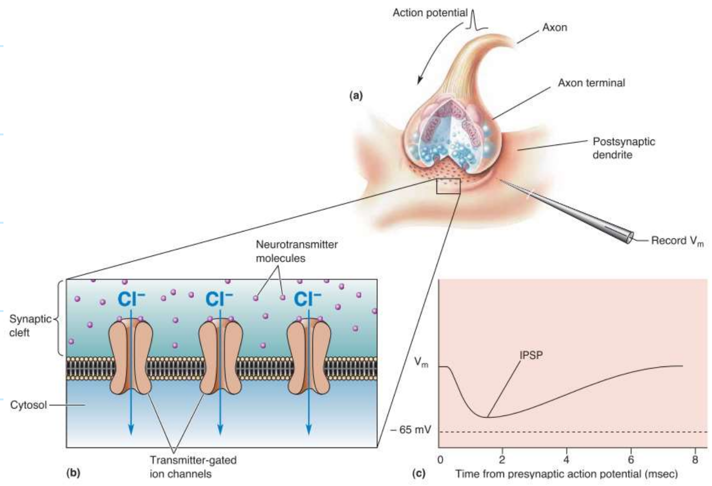
For individual neuron,



electrical signal over threshold.
→ action potential
→ synaptic transmission.
→ transmit signals to the following neurons.

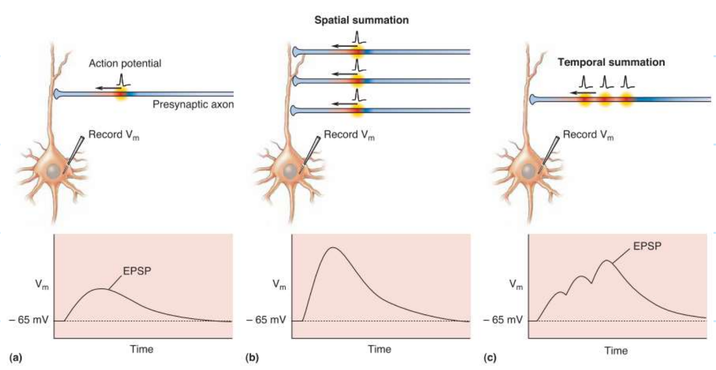
In more detailed and macro view, there are two types of synaptic connections

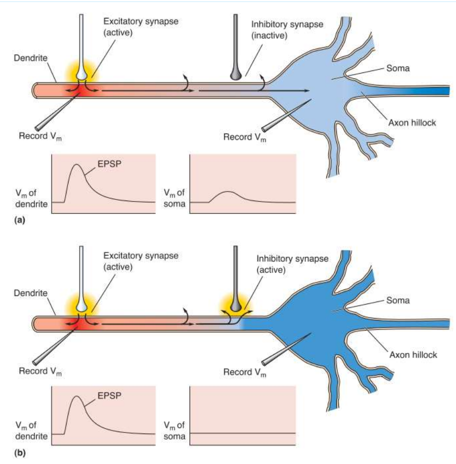




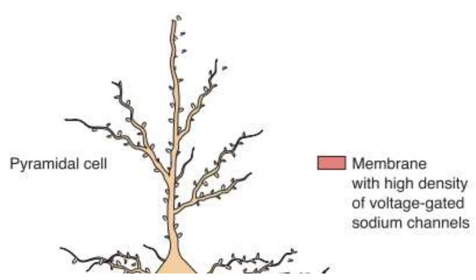
< excitatory post-synaptic potential >　　< inhibitory post-synaptic potential >

↦ Synaptic integration.

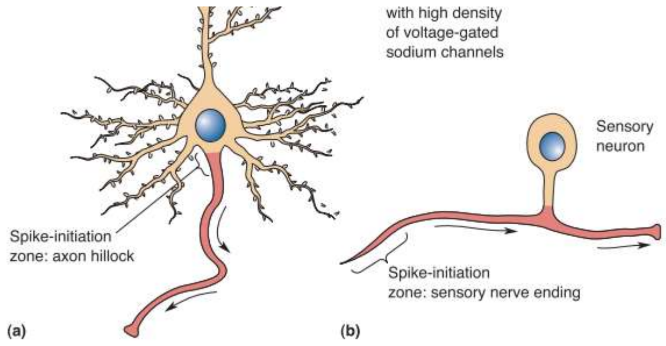




< EPSP summation >　　< Shunting inhibition >

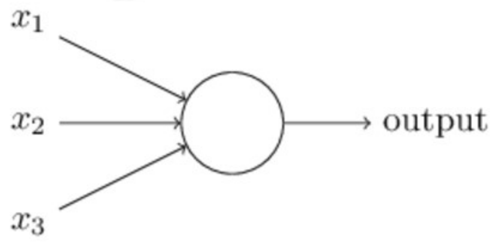


↦ integrated in spike-initiation zone.
(usually in axon hillock).



Conclusion : action potential is an activation from a weighted sum of preceding neural inputs.

3.2. Perceptron : Artificial neural network.



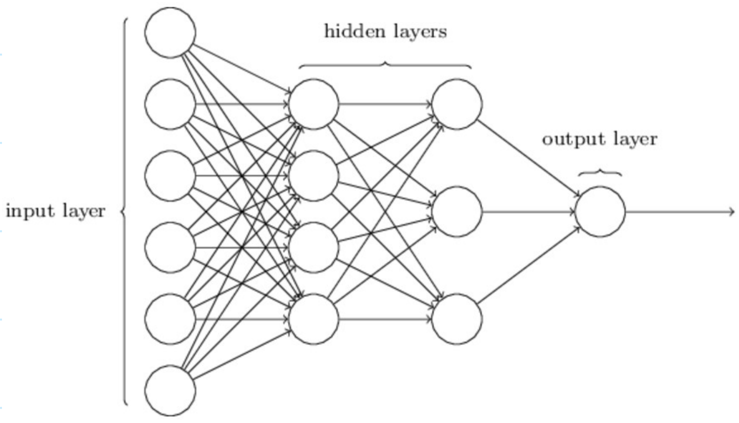
inputs : $x_1, x_2, \cdots, x_n$
weights : $w_1, w_2, \cdots, w_n$
$\mapsto \sum_j w_j x_j$: weighted sum.

$\mapsto$ output $= \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold.} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold.} \end{cases} = \sigma(w^T x)$

$\hookrightarrow$ activation function.

3.2.1. Multilayer Perceptron. (OR feedforward. network)

- multilayer perceptron



simple decision making $\rightarrow$ complex decision making.
$\mapsto$ engage sophisticated decision making.

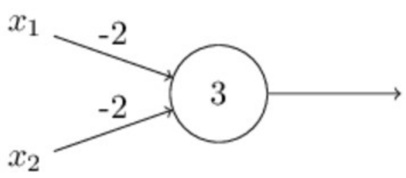
- feedforward vs. recurrent neural network.

feed-forward network : output from one layer is used as an output of the next layer.
(no loops in the network).

recurrent neural network. $\leftarrow$ cascade of neural firing.
$\hookrightarrow$ loops doesn't cause problems.

3.2.2. Computational university of MLP.

MLP learns NAND gate. $\mapsto$ can immitate all hard-coded programs (logical functions)



Furthermore than just mimicking logical functions,

we can devise learning algorithms by tuning the parameter automatically.

$\mapsto$ Obtained by optimization.

3.3. Training feedforward neural networks.

3.3.1. Gradient descent.

cost function : $C(\varphi) = \mathbb{E}_x(C_x(\varphi; y))$ while C_x : individual cost on an example x.

x : input, y : desired output, φ : parameters.

Target : "minimize the cost function by gradient descent"

e.g. error rate, $-$log-likelihood.

For parameters, $\varphi = (\varphi_1, \varphi_2, \cdots, \varphi_n)$

$$\Delta C = \frac{\partial C}{\partial \varphi_1}\Delta\varphi_1 + \frac{\partial C}{\partial \varphi_2}\cdot\Delta\varphi_2 + \cdots + \frac{\partial C}{\partial \varphi_n}\Delta\varphi_n$$

while $\nabla C := \left(\frac{\partial C}{\partial \varphi_1}, \cdots, \frac{\partial C}{\partial \varphi_n}\right)$

$\Delta C = \nabla C \cdot \Delta v$.

$\hookrightarrow$ $\Delta v = -\eta \nabla C \rightarrow$ reduce. ΔC the most.

$v \rightarrow v' = v - \eta\nabla C$: gradient descent algorithm.

For MLP, parameters are weights or biases.

$$w_k \longrightarrow w_k' = w_k - \eta\cdot\frac{\partial C}{\partial w_k}$$

$$b_l \longrightarrow b_l' = b_l - \eta\cdot\frac{\partial C}{\partial b_l}$$

3.3.2. Backpropagation.

(Notation). (Gradient). For a tensor T with dimension n, and cost function C

$$\nabla_T C = \left(\frac{\partial C}{\partial T_{i_1\cdots i_n}}\right) \quad \text{i.e. } (\nabla_T C)_{i_1\cdots i_n} = \frac{\partial C}{\partial T_{i_1\cdots i_n}}$$

(Derivative). For a vector $x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$

$\frac{df}{dx}$ is defined as. $f(x+dx) \approx f(x) + \frac{df}{dx}\cdot dx$.

As a result, if $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $\frac{df}{dx} = (\nabla_x f)^T$

activation of l^{th} layer, a^l is related to the activations in the $(l-1)^{th}$ layer, a^{l-1}

$$a^l = \sigma(w^l a^{l-1} + b^l)$$

When $\frac{dC}{da^l}$ $(:= \nabla_{a^l} C^T)$ is given.

$$\frac{\partial C}{\partial w_{ij}^l} = \frac{dC}{da^l}\cdot\frac{da^l}{dz^l}\frac{\partial z^l}{\partial w_{ij}^l} \quad \text{while } z^l = w^l a^{l-1} + b^l$$

$$= \frac{dC}{da^l}\sigma'(z^l)\, a_i^{l-1} e_j \quad \text{(Note that } \sigma'(z^l) \text{ is a diagonal matrix of } \sigma'(z_i^l)\text{'s)}$$

$$= \frac{dC}{da_j^l}\cdot\sigma'(z_j^l)\, a_i^{l-1}$$

$$\mapsto \nabla_{w^l} C = a^{l-1}\cdot(\sigma'(z^l)\cdot\nabla_{a^l} C)^T$$

$$\frac{dC}{db^l} = \frac{dC}{da^l}\cdot\frac{da^l}{db^l} = \frac{dC}{da^l}\frac{da^l}{dz^l}\frac{dz^l}{db^l} = \frac{dC}{da^l}\cdot\sigma'(z^l).$$

$\Rightarrow \nabla_{b^l} C = \sigma'(z^l)\cdot\nabla_{a^l} C.$

$$\frac{dC}{da^{l-1}} = \frac{dC}{da^l}\frac{da^l}{dz^l}\frac{dz^l}{da^{l-1}} = \frac{dC}{da^l}\sigma'(z^l)\cdot w^l.$$

$\Rightarrow \nabla_{a^{l-1}} C = {w^l}^T\cdot\sigma'(z^l)\cdot\nabla_{a^l} C.$

Recursively, we can calculate all $\nabla_{w^l} C$ and $\nabla_{b^l} C$.

3.4. Modification on gradient descent algorithm.

3.4.1. Difficulties on training gradient algorithms.

① undifferentiability of activation function.

② undifferentiability and under-representation of individual loss.

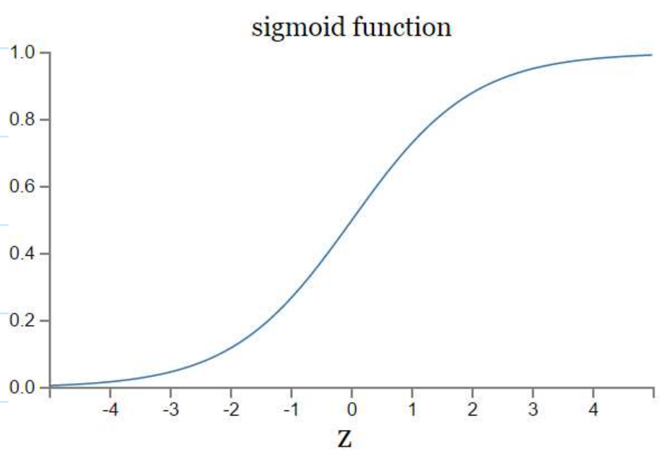
③ unavailable loss function

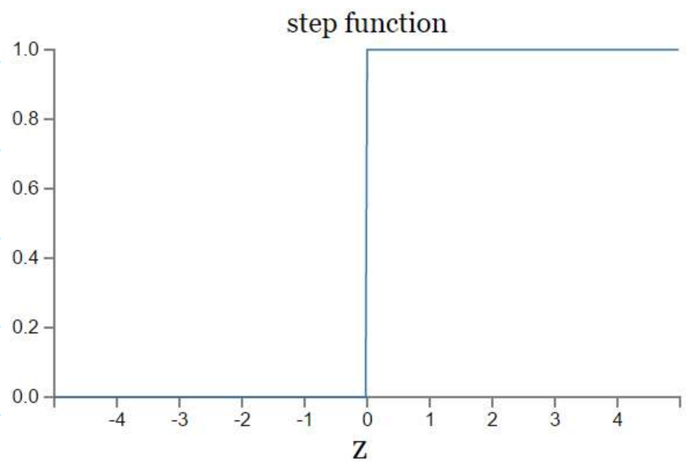
⇒ Solutions : sigmoid neuron, proxy loss function, stochastic gradient descent.

3.4.2. Sigmoid neuron.

sigmoid function : $\sigma(z) = \frac{1}{1+e^{-z}}$

↳ sigmoid neuron. output $= \sigma(\sum_j w_j x_j + b)$





$wx + b \rightarrow$ linear function.

$\sigma \rightarrow$ activation function (nonlinearity).

⇒ sigmoid neuron is much easier to get trained.

3.4.3. Proxy loss function

$C_x(\phi; y) := \|y - \hat{y}(x;\phi)\|^2$

where $\hat{y}$ is the output of the neural network. with parameter ϕ.

⇒ ① differentiable with respect to parameters.

② motivates the learning further than direct loss function as accuracy.

3.4.4. Stochastic gradient descent.

: estimate the gradient ∇C from a small sample.

$$\frac{1}{m}\sum_{j=1}^{m}\nabla C_{X_j} \approx E[\nabla C_x] = \nabla C.$$

$$w_k \rightarrow w_k - \frac{\eta}{m}\sum_j \frac{\partial C_{X_j}}{\partial w_k}$$

$$w_k \rightarrow w_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_k}$$

$$b_l \rightarrow b_l - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b_l}$$