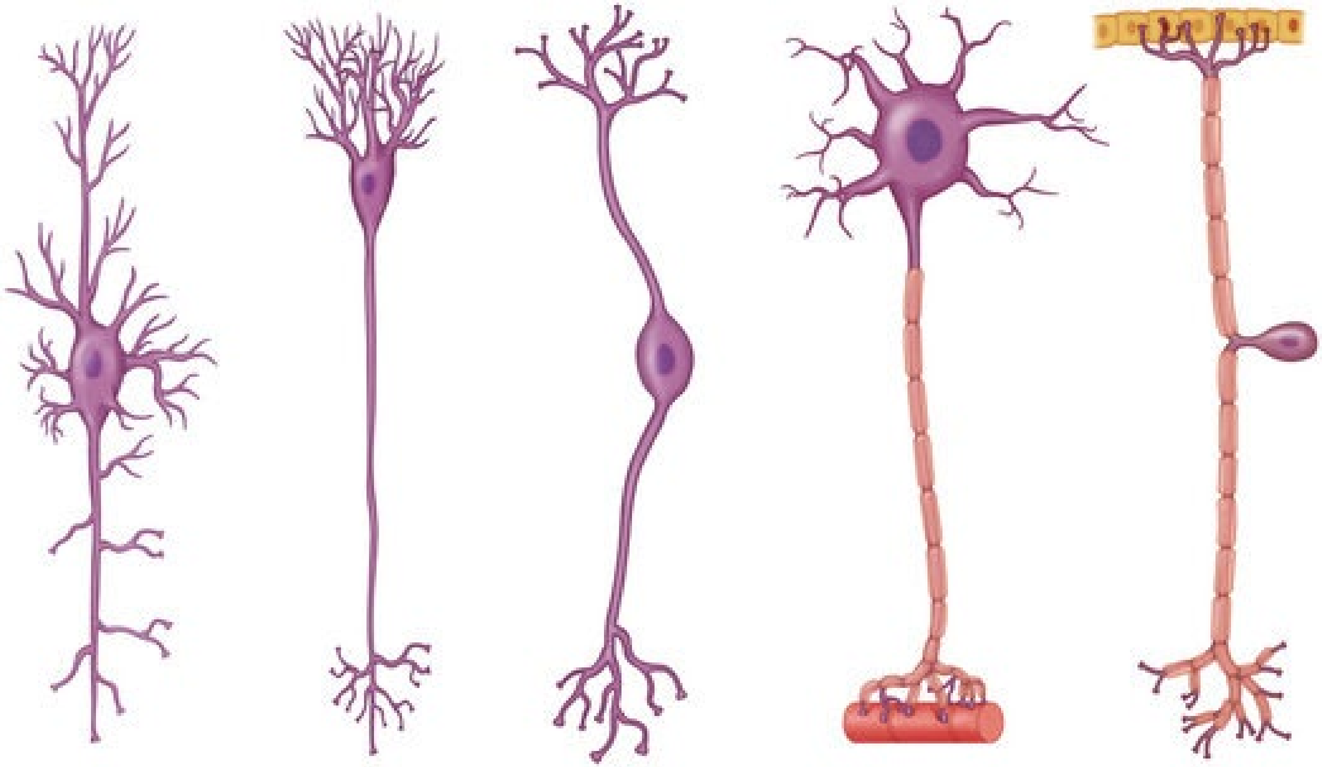
From Math To Mind

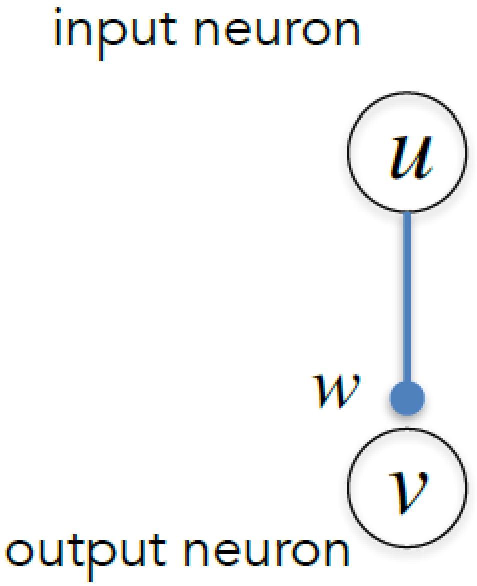
Adibvafa Fallahpour

Neuron

Various Types



Our Model – Linear Neurons



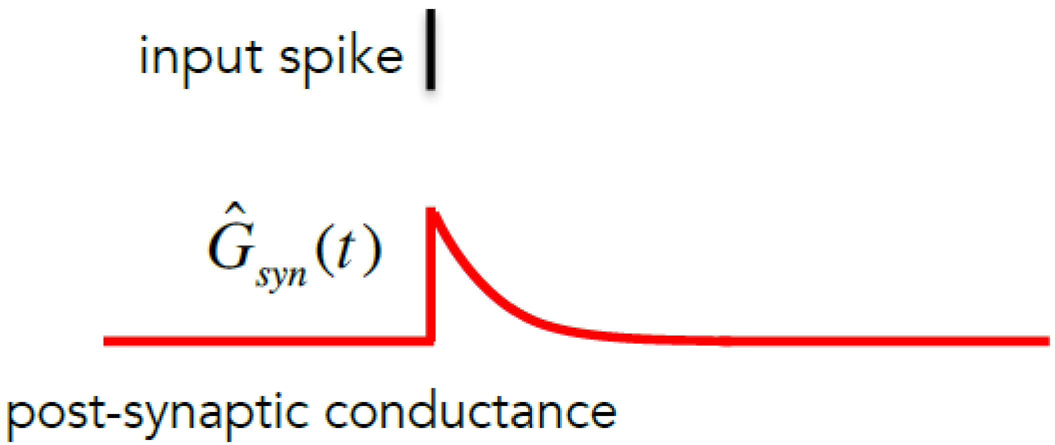
u = firing rate of input neuron

w = synaptic strength (weight)

v = firing rate of output neuron

$$v = wu$$

Input Neuron



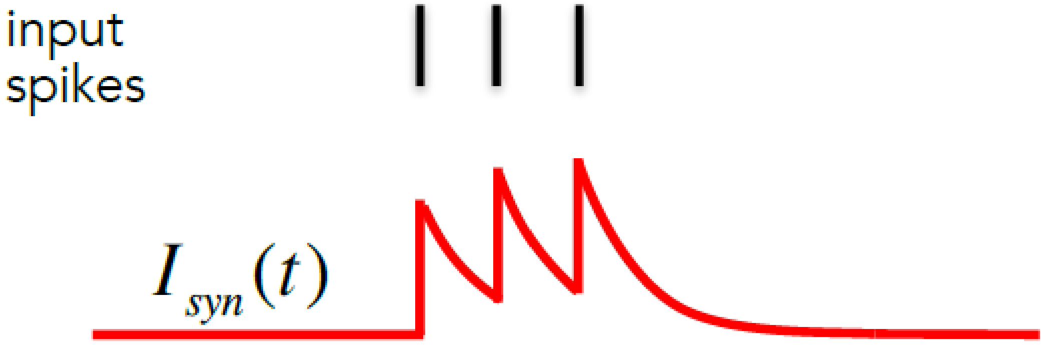
input spike

$\hat{G}_{syn}(t)$

post-synaptic conductance

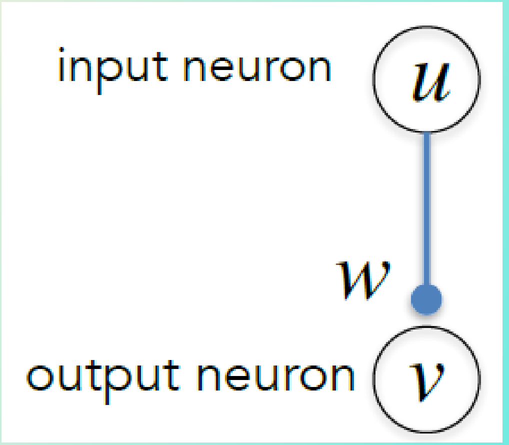
$$\hat{G}_{syn}(t) = G_{\max}\, e^{-t/\tau_s}$$

$$\hat{I}_{syn}(t) = \hat{G}_{syn}(t)$$



input spikes

$I_{syn}(t)$

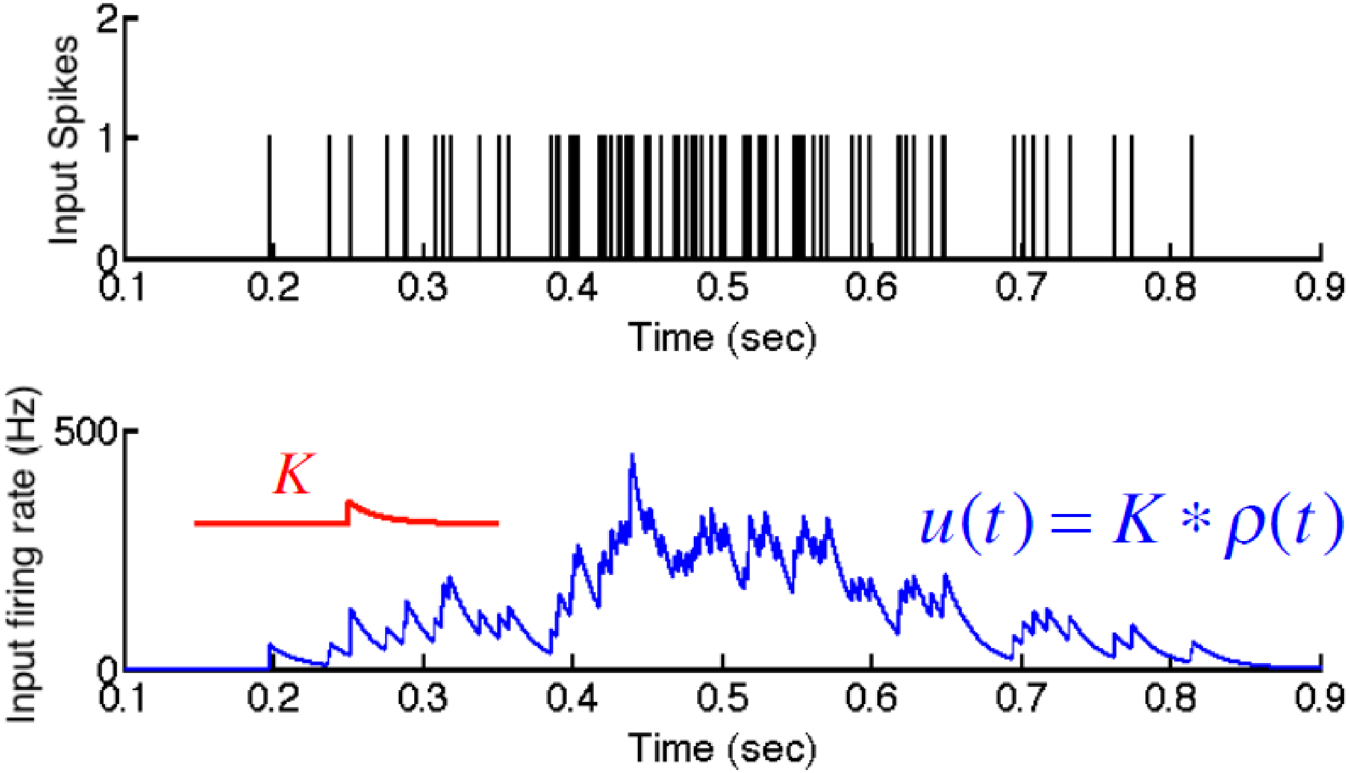


input neuron u

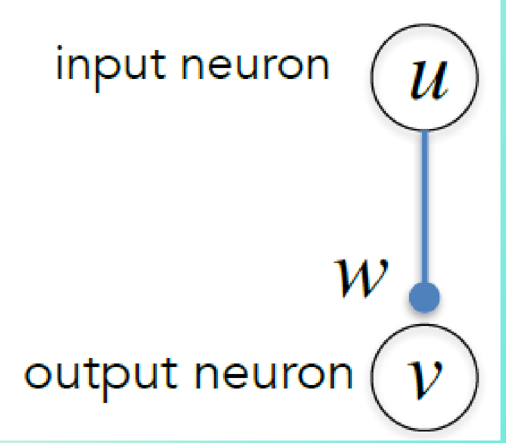
w

output neuron v

Presynaptic Firing Rate



$$I_{syn}(t) = w u(t)$$

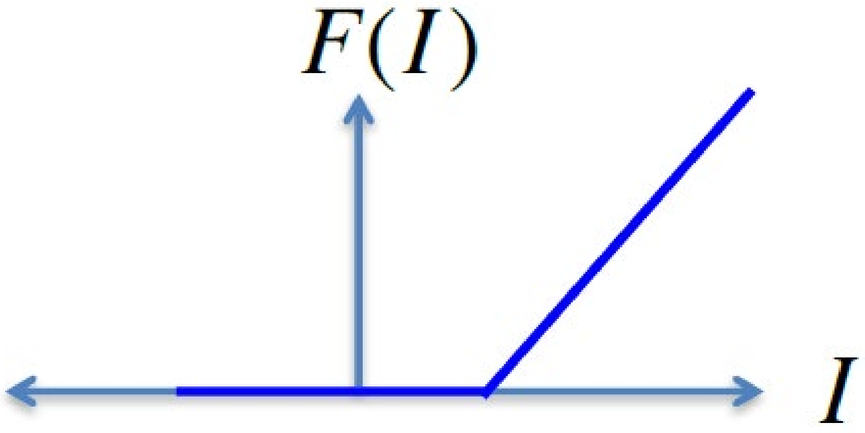


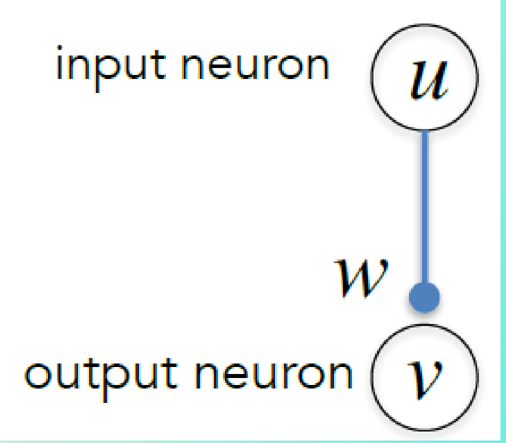
2023-06-20

Output Neuron

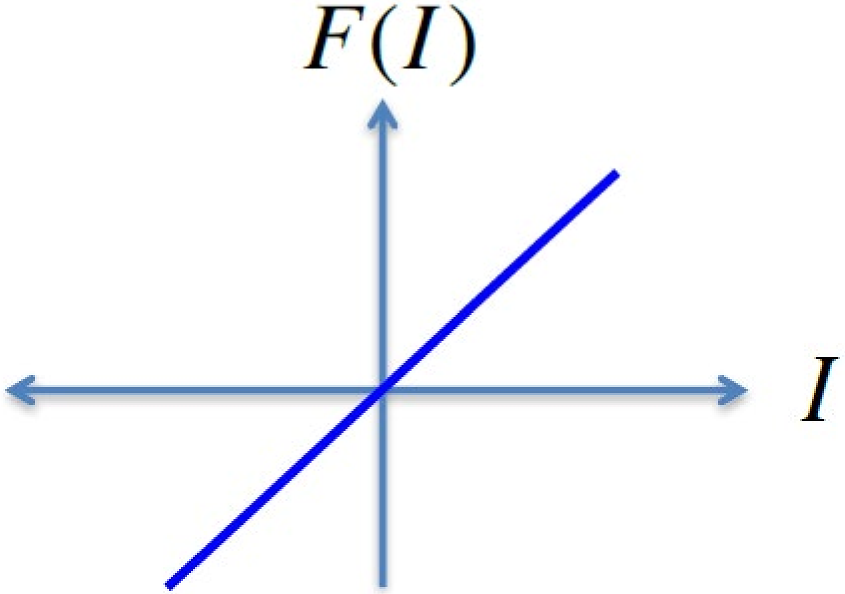
- The output firing rate is some non-linear function of the synaptic input.

$$v = F[I_s] = F[wu]$$



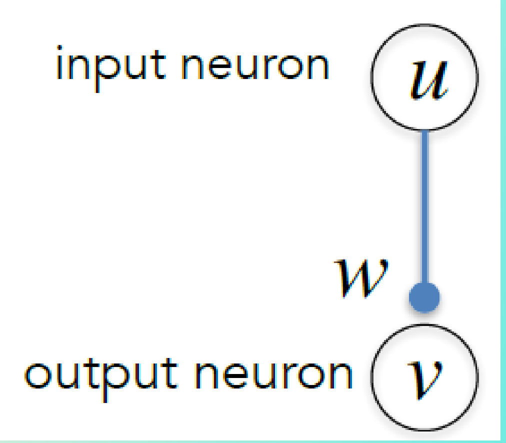


Linear Rate Models



$$F[x] = x$$

$$v = wu$$

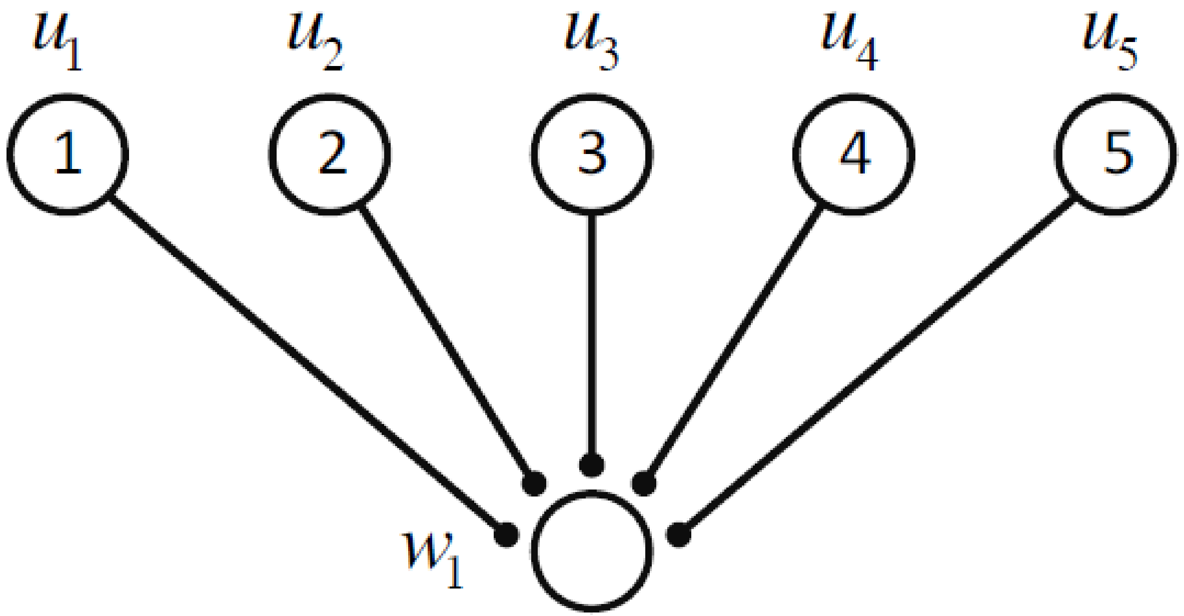


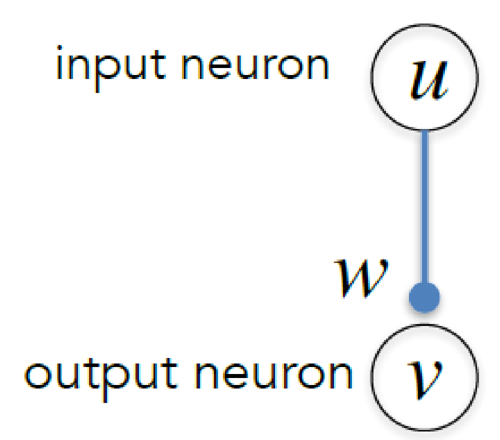
2023-06-20

Multiple Inputs

$$I_{syn} = w_1u_1 + w_2u_2 + w_3u_3 + \ldots$$

$$v = \sum_b w_b u_b$$



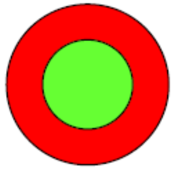


Receptive Field

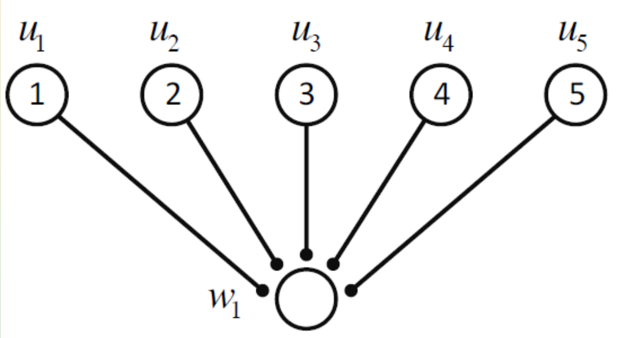
- At the simplest level, we think of the receptive field (RF) as the region of visual space that causes the neuron to spike.

- But a visual neuron doesn't respond to any stimulus within this RF. It responds selectively to certain 'features' in the stimulus.

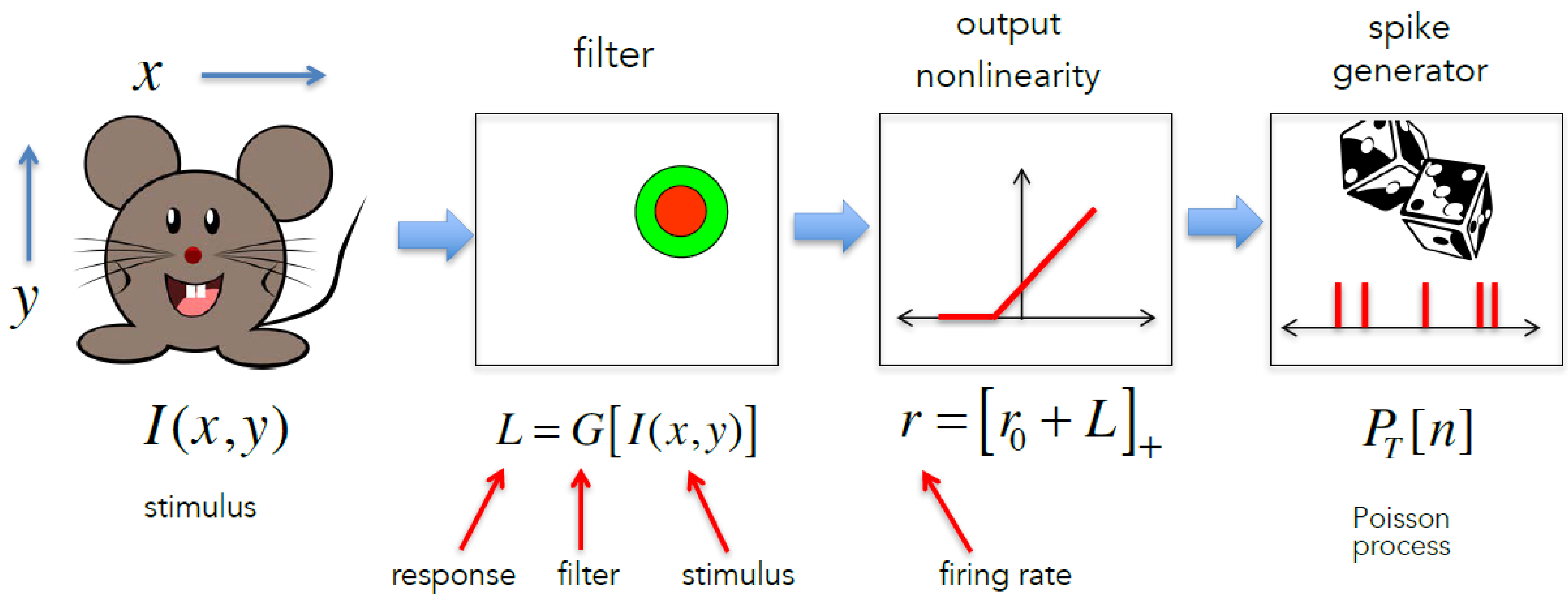
- We can think of a neuron as having a filter (G) that passes certain features in both space and time.



- The better the stimulus 'overlaps' with the filter, the more the neuron will spike.



Our Model



x

y

filter

output nonlinearity

spike generator

$I(x,y)$

stimulus

$L = G[I(x,y)]$

response

filter

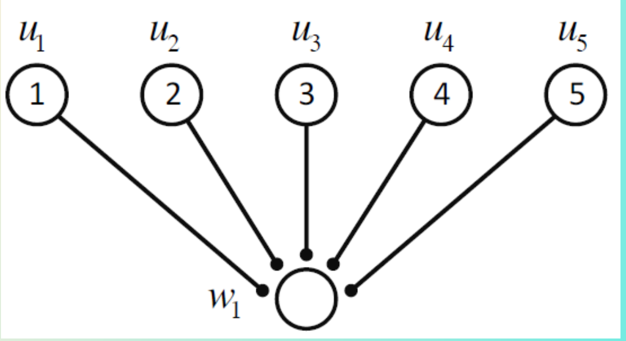
stimulus

$r = [r_0 + L]_+$

firing rate

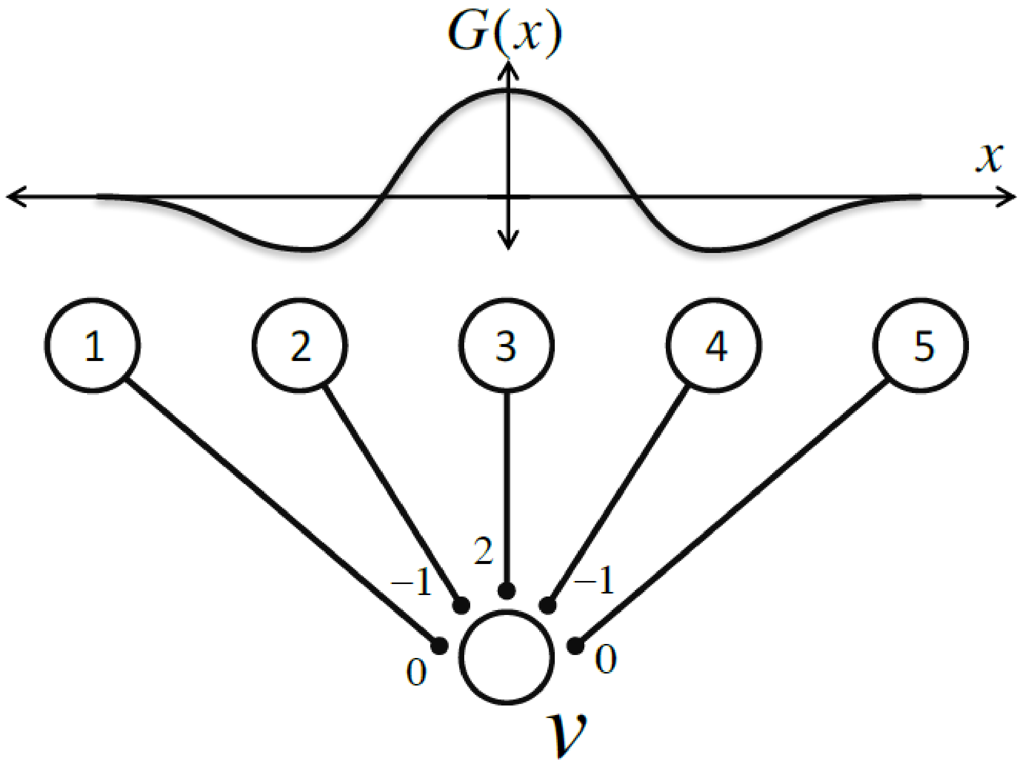
$P_T[n]$

Poisson process

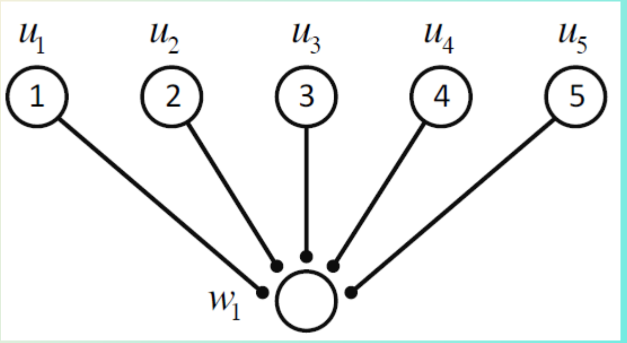


Weights = Receptive Field

$$v = \sum_b w_b u_b$$

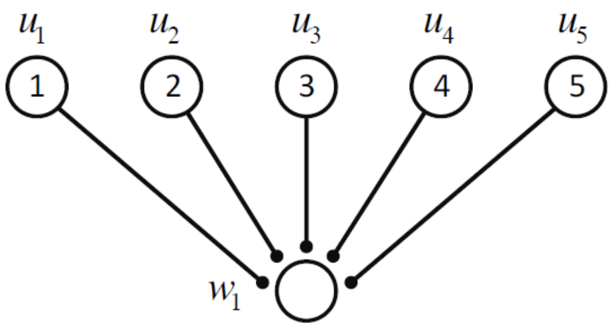


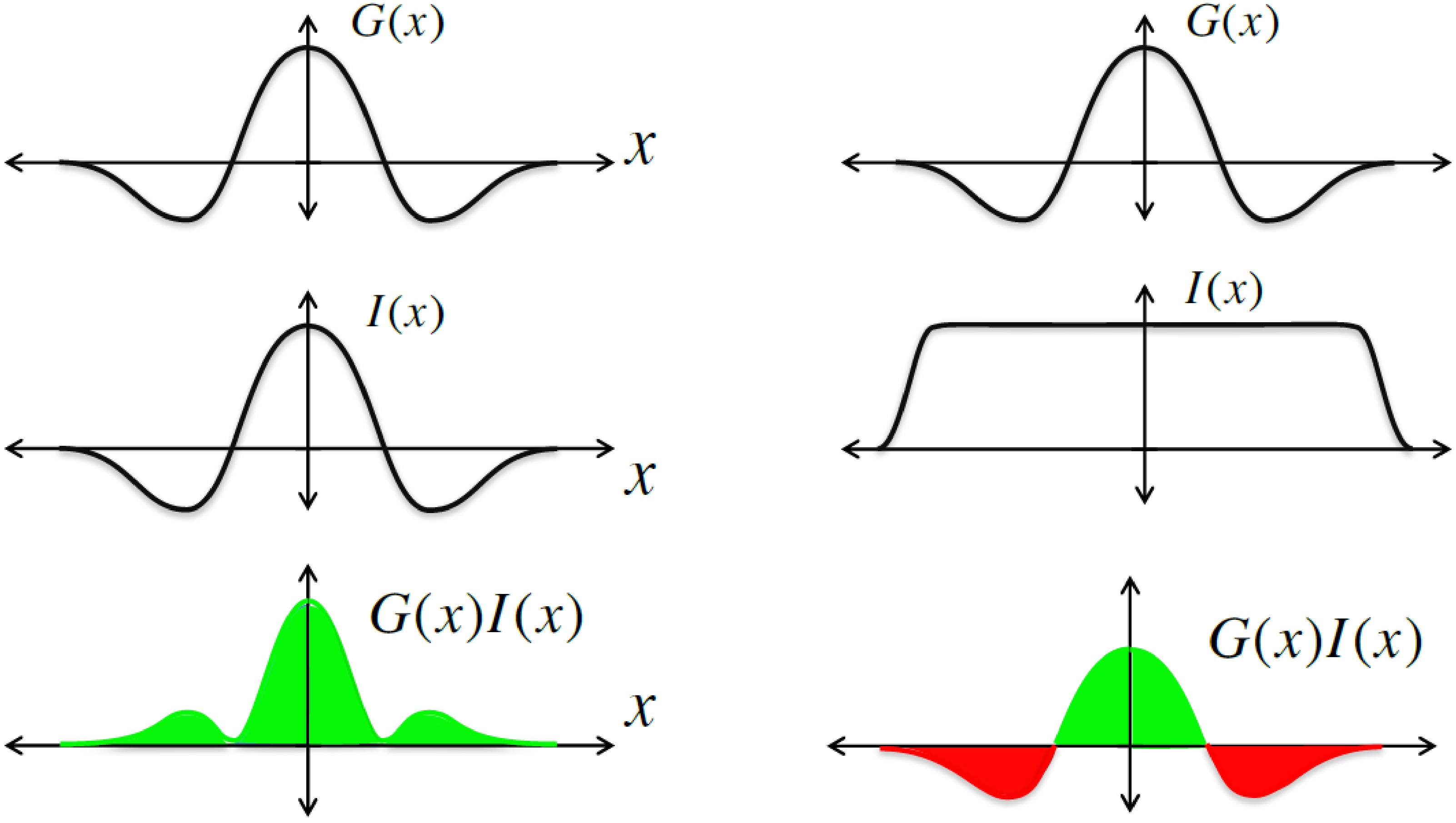
$$\vec{w} = [0\ ,\ -1\ ,\ 2\ ,\ -1\ ,\ 0]$$



2023-06-20

RF Mathematically?





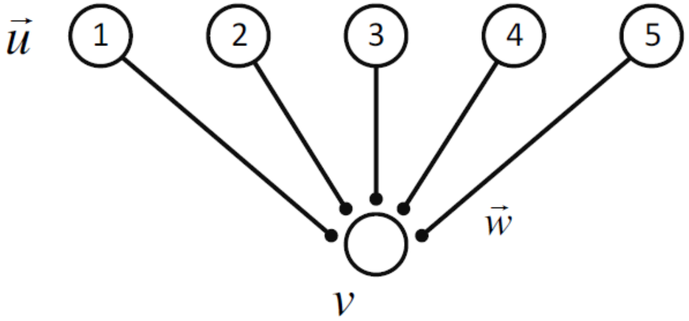
$\int G(x)I(x)dx$ *big*

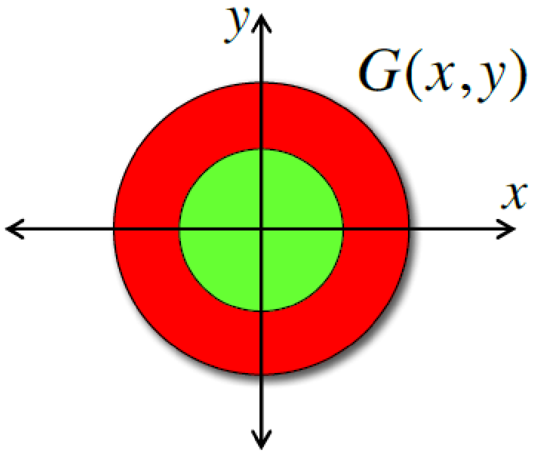
$\int G(x)I(x)dx$ *small*

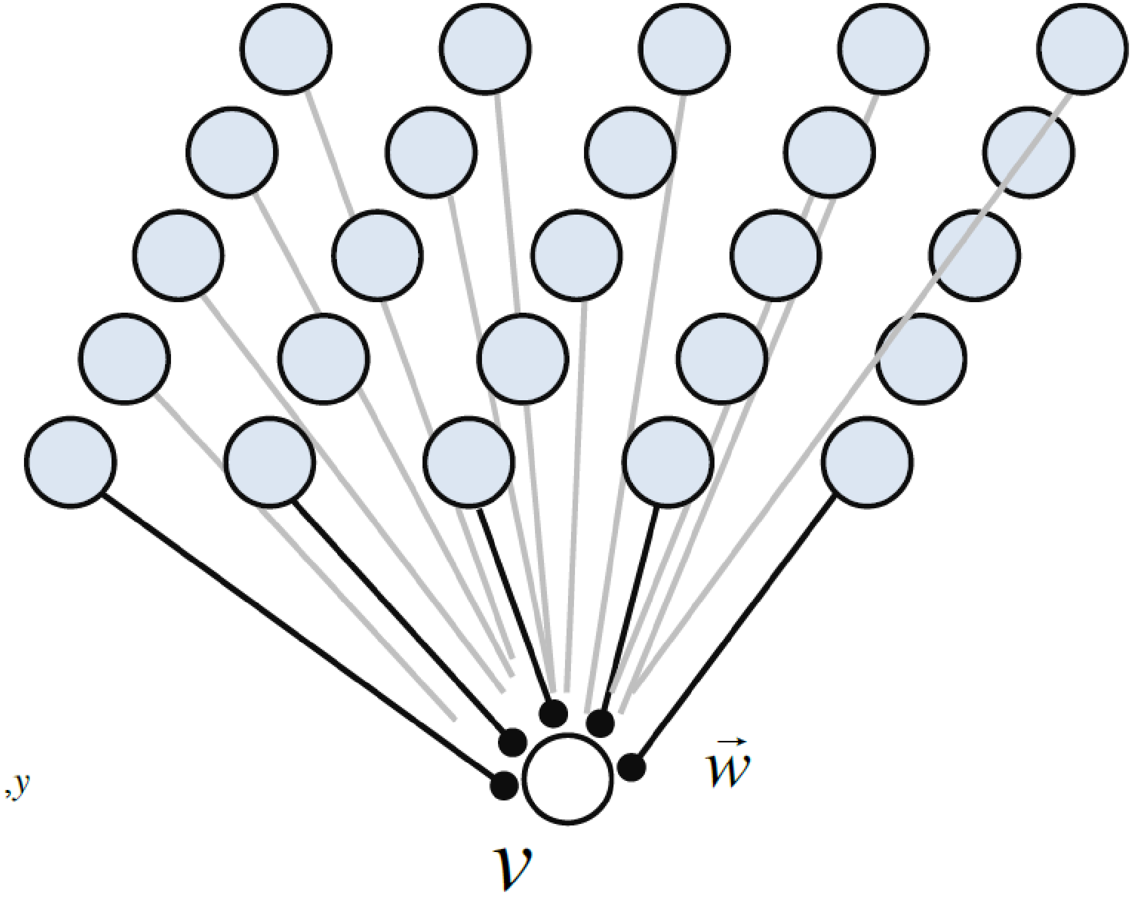
$$r = \int G(x)I(x)$$

2023-06-20

2D RF



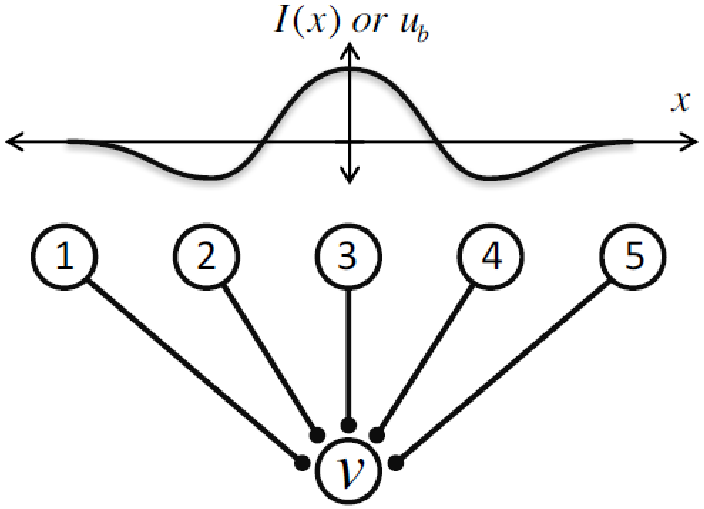




$$v = \sum_{x,y} w_{x,y} u_{x,y}$$

2023-06-20

But that is a Dot Product!



- The response of a neuron is the dot-product of the stimulus vector with the weight vector (receptive field).

- Thus, for a given amount of power in the stimulus , the stimulus that has the best overlap with the receptive field produces the largest neuronal response.

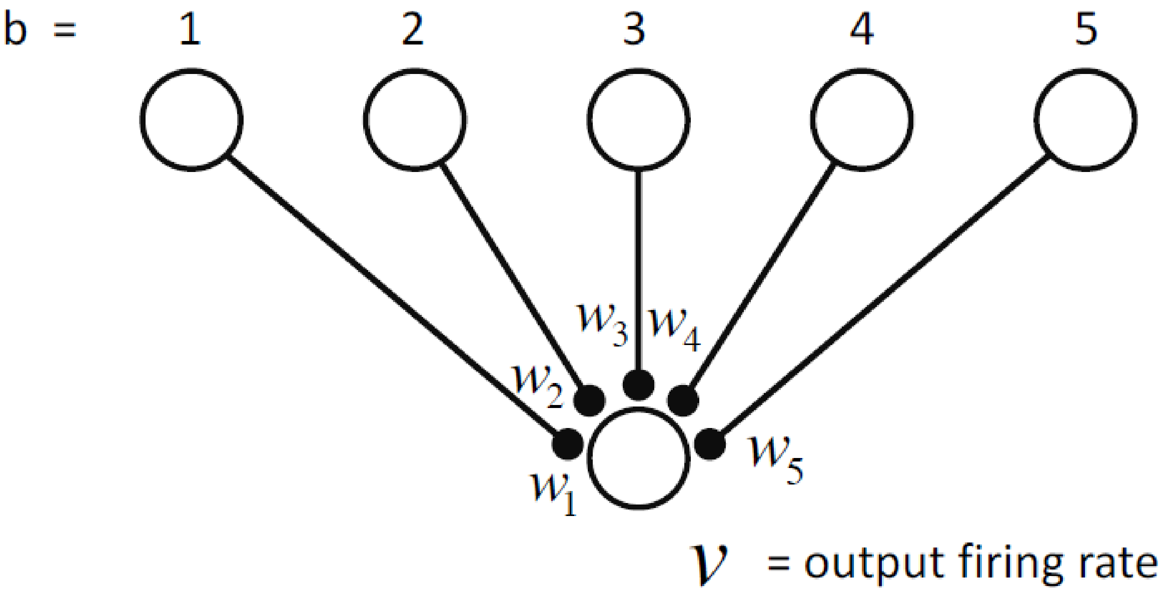
- We now have a definition of the ‘optimal stimulus’

$$v = \sum_b w_b u_b$$

$$v = \vec{w} \cdot \vec{u} \quad = |\vec{w}||\vec{u}| \cos\theta$$

2023-06-20

Summary



Input firing rates

$$\left[u_1\,,\, u_2\,,\, u_3\,,\ldots\; u_{n_b} \right] = \vec{u}$$

Input synaptic weights

$$\left[w_1\,,\, w_2\,,\, w_3\,,\ldots\; w_{n_b} \right] = \vec{w}$$

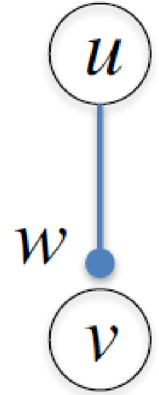
$$I_s = w_1 u_1 + w_2 u_2 + w_3 u_3 + \ldots = \sum_b w_b u_b = \vec{w} \cdot \vec{u}$$

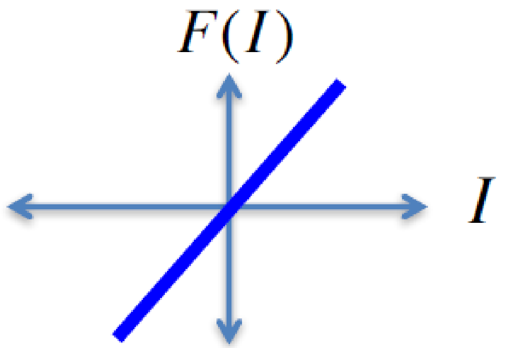
$$\boxed{v = F\left[\vec{w} \cdot \vec{u} \right]}$$

Summary

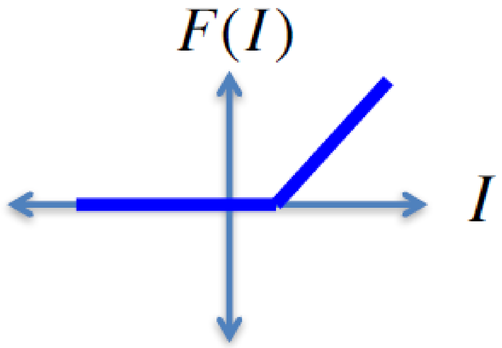
- The output firing rate is some non-linear function of the synaptic input.

$$v = F[I_s] = F[wu]$$

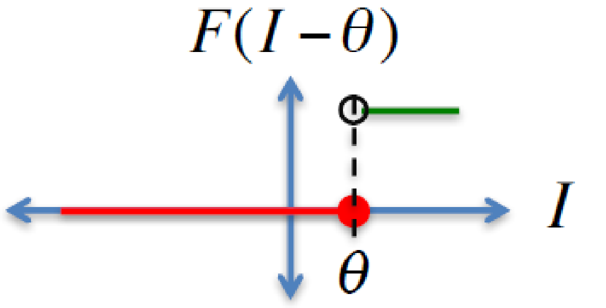




linear neuron

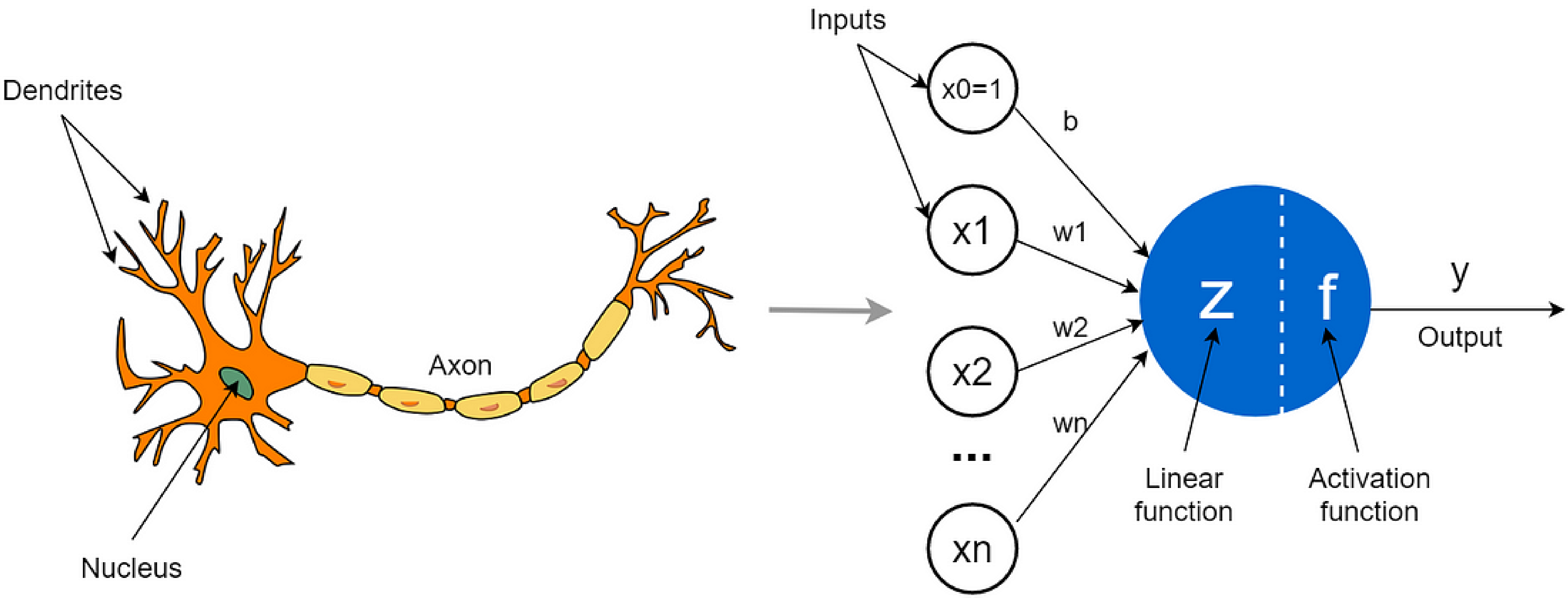


Linear threshold neuron

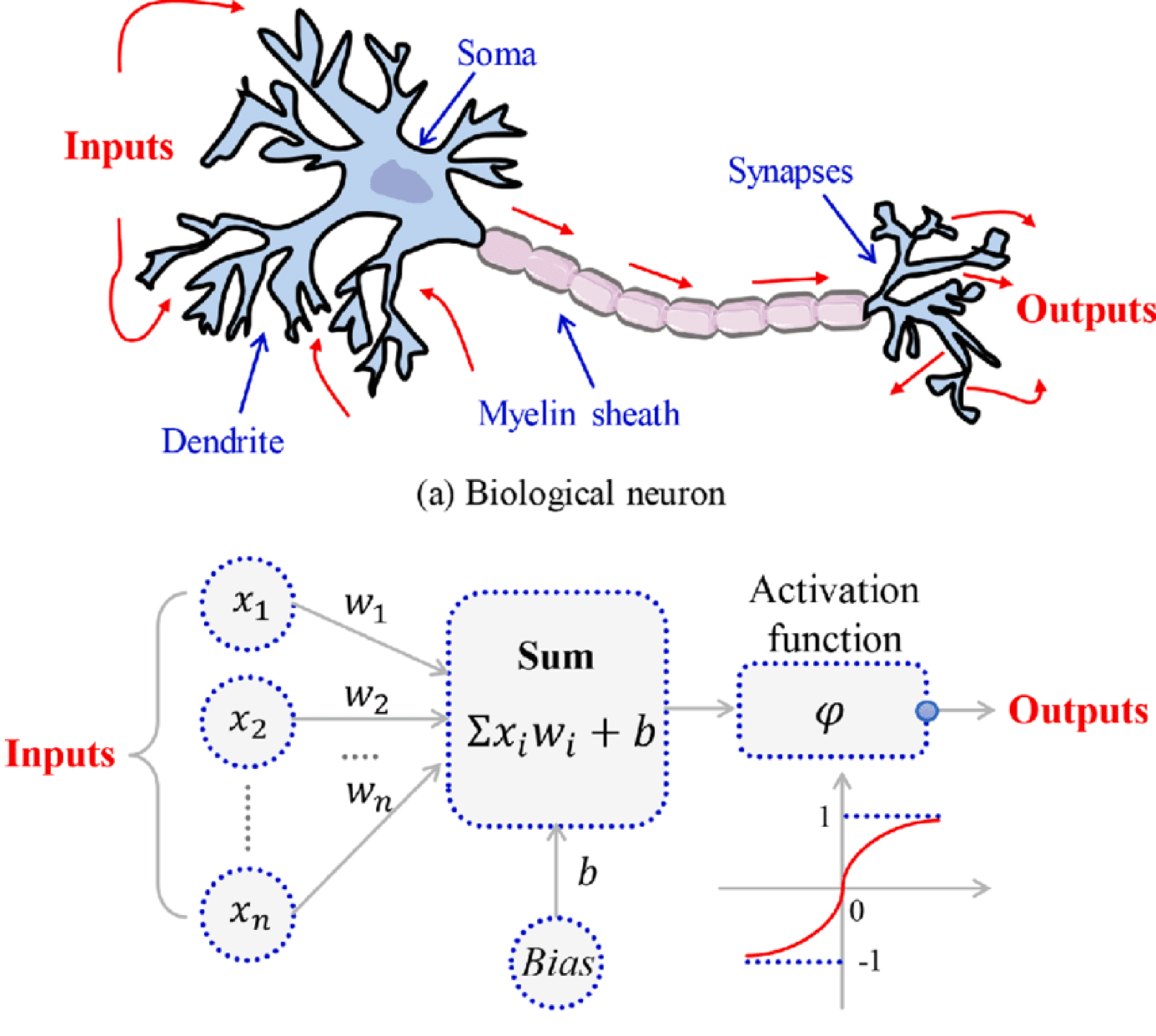


Binary threshold neuron

Artificial Neuron



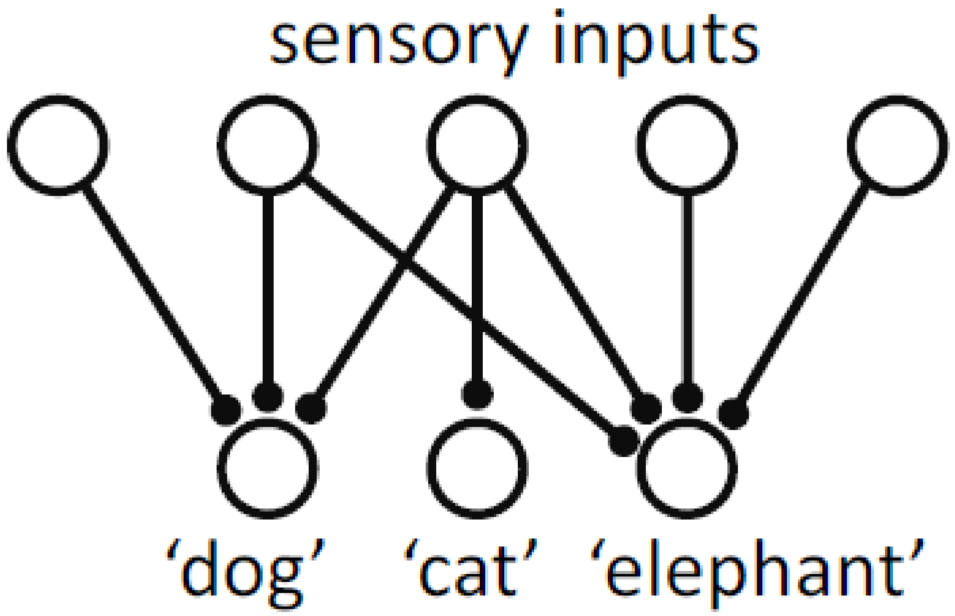
Artificial Neuron



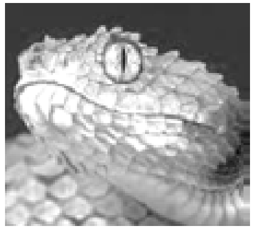
(a) Biological neuron

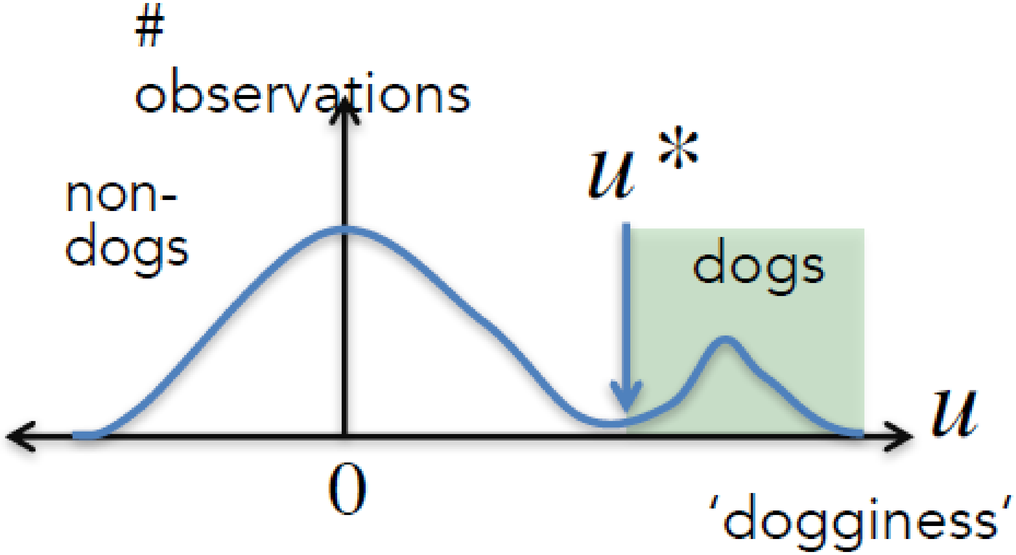
(b) Artificial neuron

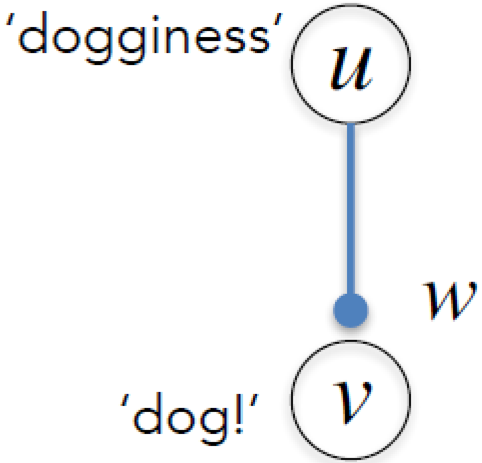
Classification



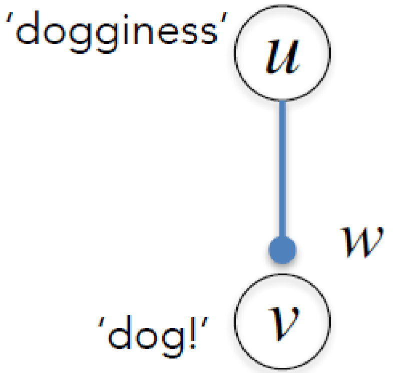
Perceptron (Single Neuron)







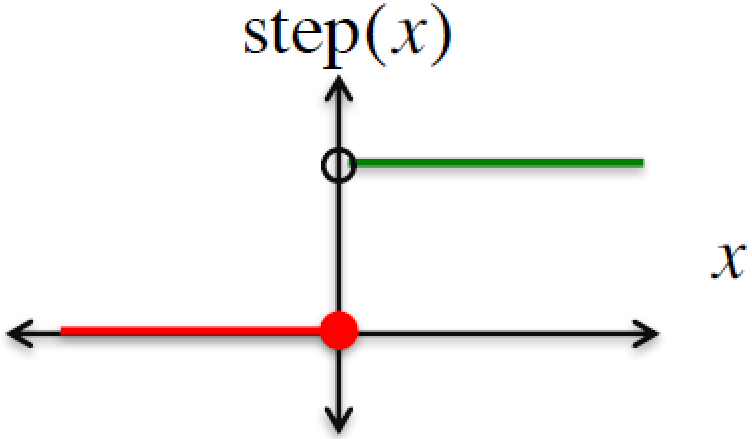
Spike Threshold!

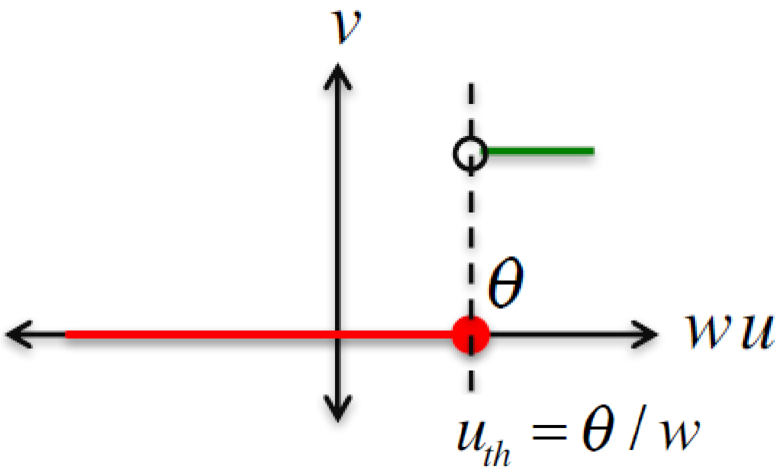


$$F(x) = \text{step}(x)$$

$$v = F(wu - \theta)$$

Theta is the threshold, not an angle.

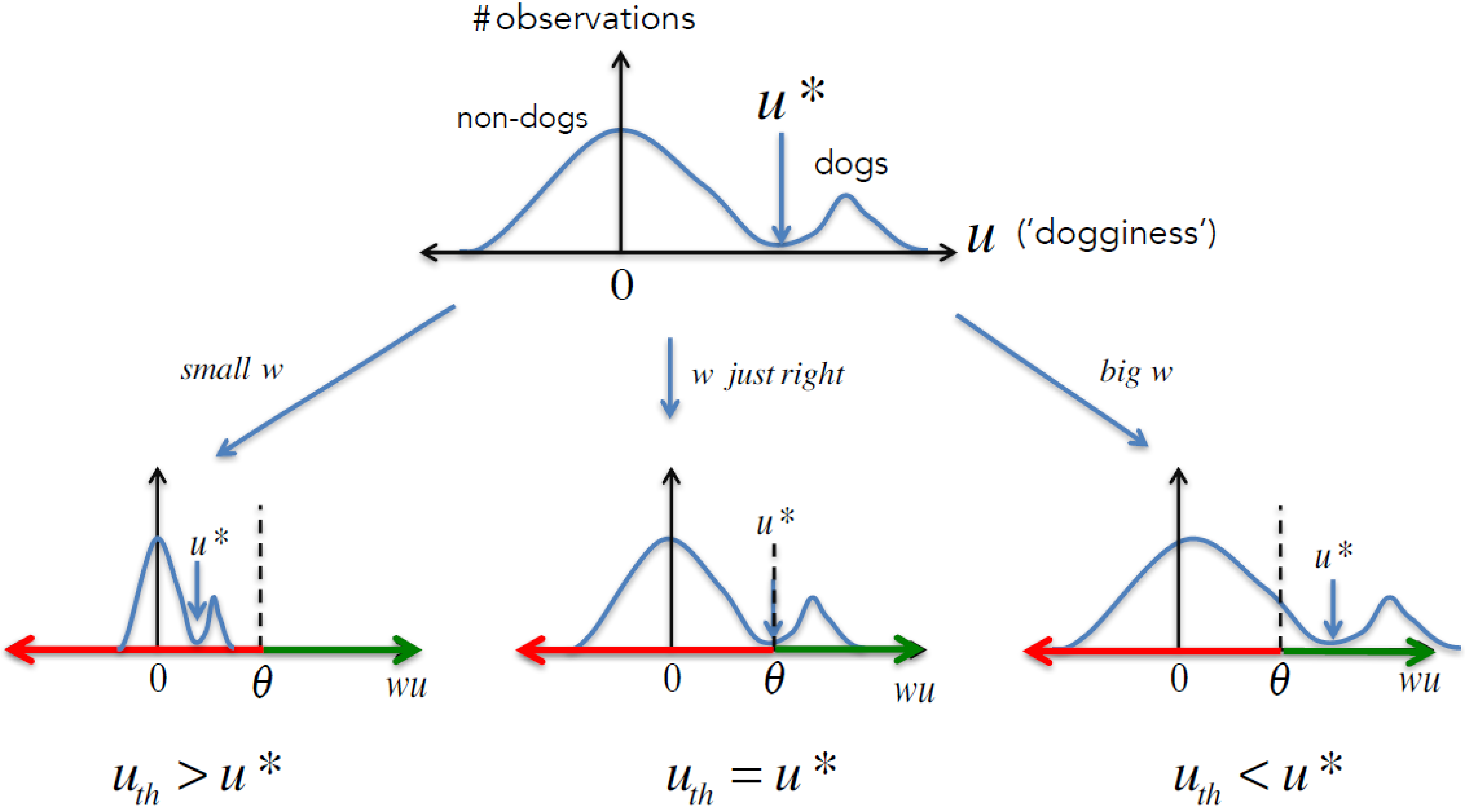




Neuron fires when the input $wu > \theta$

Learn the right Weight

$u_{th} = u^*$



2023-06-20

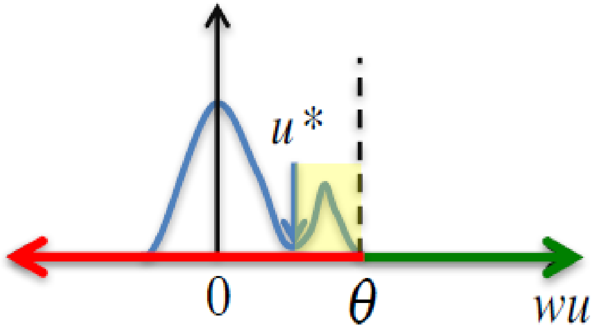
Learn the right Weight

Error: Dogs classified as non-dogs

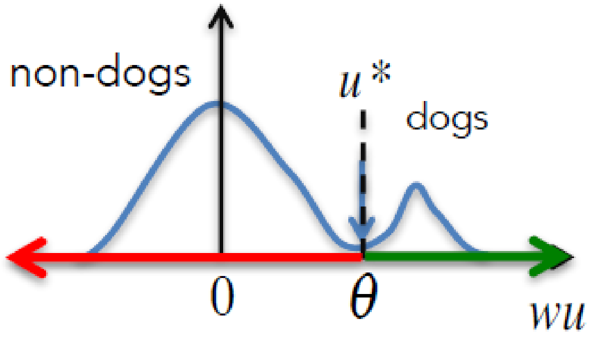
$\Rightarrow$ Make w bigger

Error: Non-dogs classified as dogs

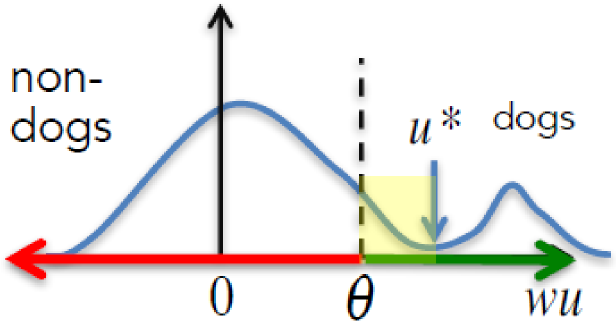
$\Rightarrow$ Make w smaller



$$u_{th} > u*$$

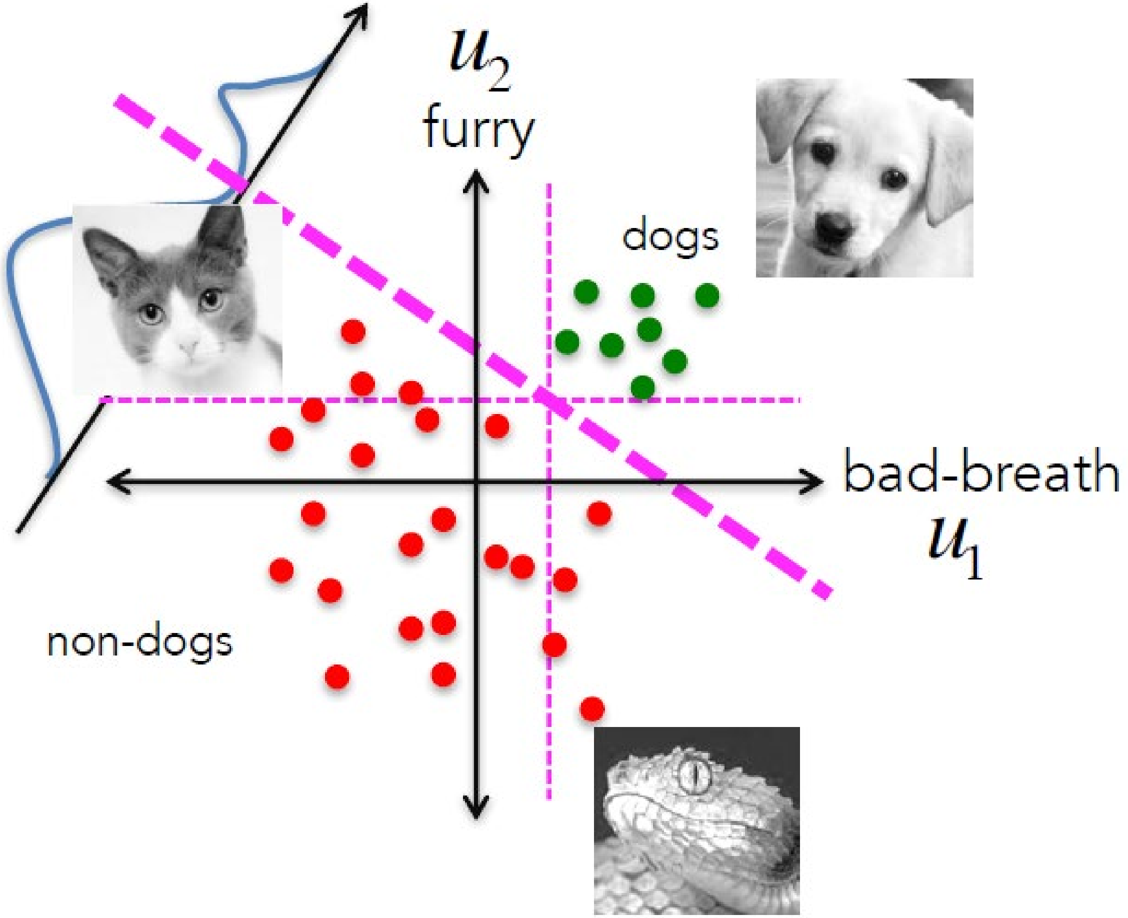


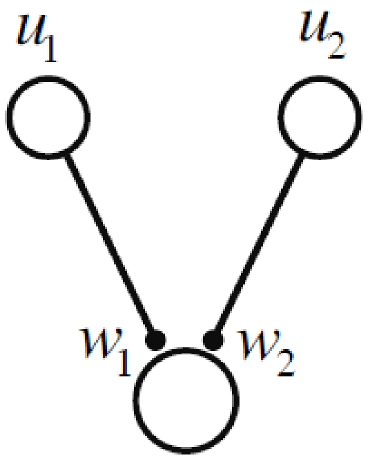
$$u_{th} = u*$$



$$u_{th} < u*$$

Decision boundary in 2D

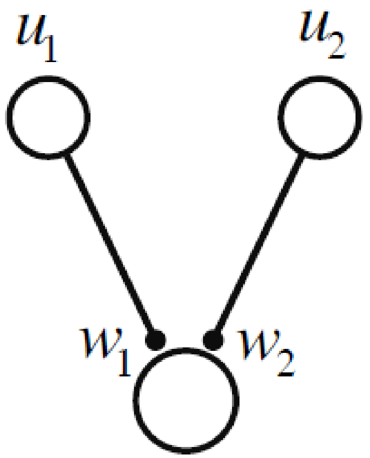




$$v = \vec{w} \cdot \vec{u}$$

2023-06-20

Decision boundary in 2D



$$v = \vec{w}\cdot\vec{u}$$

$$v = F(\vec{w}\cdot\vec{u} - \theta)$$

$$\vec{w}\cdot\vec{u} - \theta = 0$$

$$\boxed{\vec{w}\cdot\vec{u} = \theta}$$

what is this?

$$w_1u_1 + w_2u_2 = \theta$$

Decision boundary in 2D

- Let's start by looking at the case where $\theta = 0$

$$v = F(\vec{w} \cdot \vec{u})$$

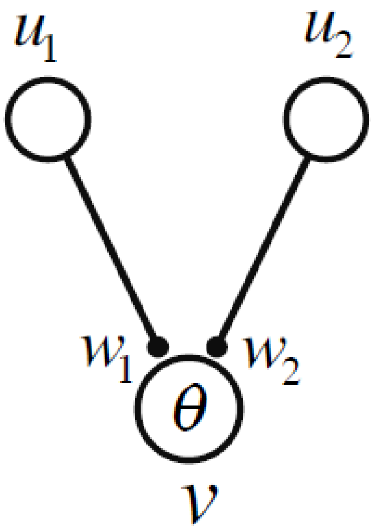
- The neuron now fires when the projection of $\vec{u}$ along $\vec{w}$ is positive $\vec{w} \cdot \vec{u} > 0$

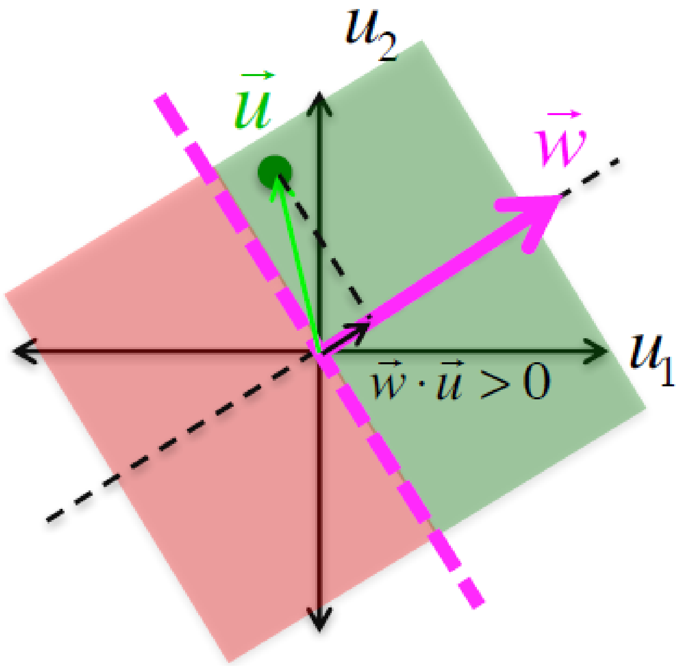
- The decision boundary is given by

$$\vec{w} \cdot \vec{u} = 0$$

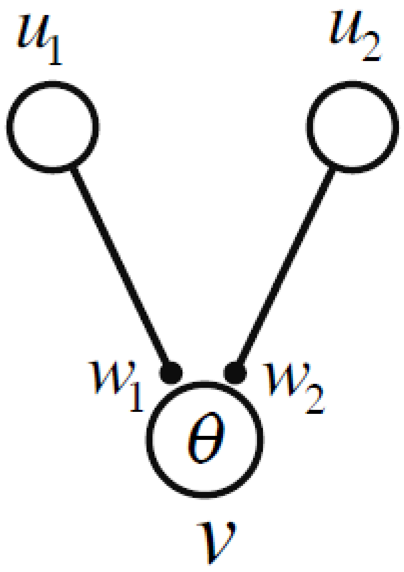
- This is the set of all vectors u that have zero projection along w.

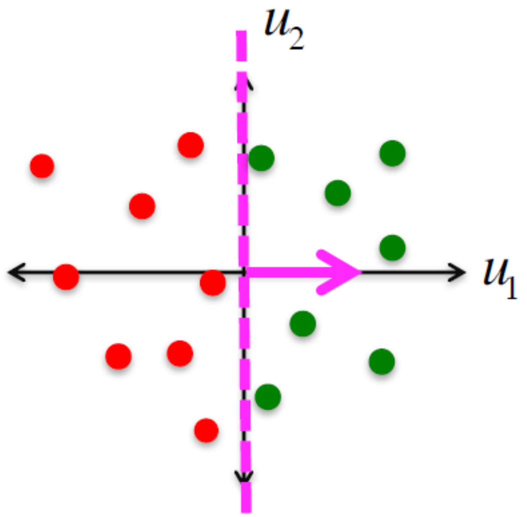
All vectors on a line going through the origin and perpendicular to w !





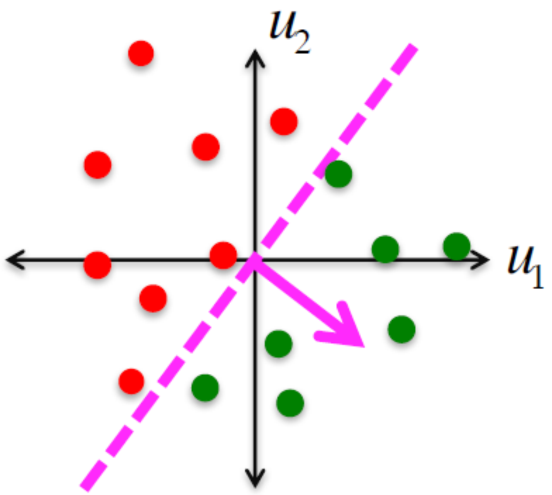
Simple Cases





$$\vec{w} = (1\ ,0)$$

$$\theta = 0$$



$$\vec{w} = (1\ ,-1)$$

$$\theta = 0$$

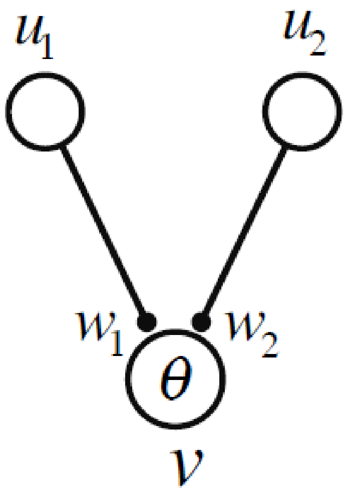
Decision boundary in 2D

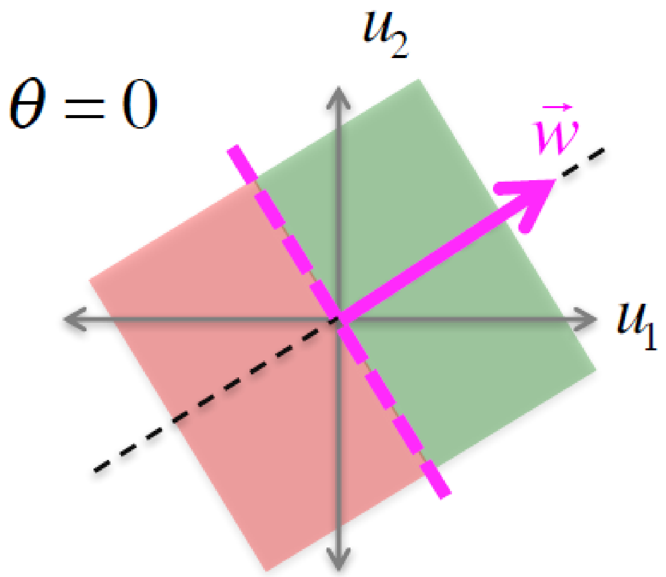
- Now let's look at the case where $\theta \neq 0$

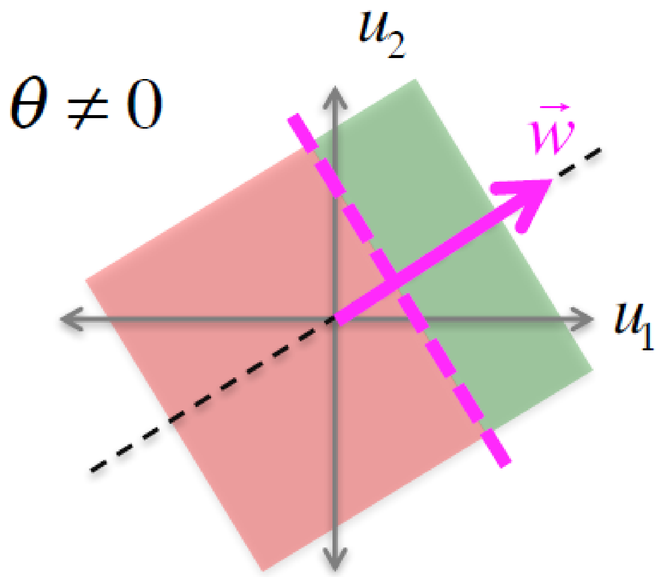
$$v = F(\vec{w} \cdot \vec{u} - \theta)$$

- Now the decision boundary is $\vec{w} \cdot \vec{u} = \theta$

- This is the set of all vectors $\vec{u}$ whose projection along $\vec{w}$ is given by θ .





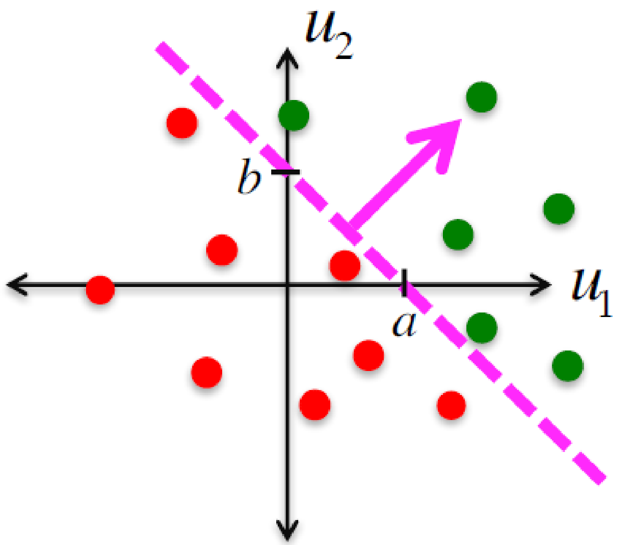


Decision Boundary

$$v = F(\vec{w} \cdot \vec{u} - \theta)$$

The decision boundary is $\vec{w} \cdot \vec{u} = \theta$

- Let's calculate the weight vector $\vec{w} = (w_1 , w_2)$ that gives us the decision boundary shown below. Assume $\theta = 1$.



We have two points on the decision boundary we know, and two unknowns…

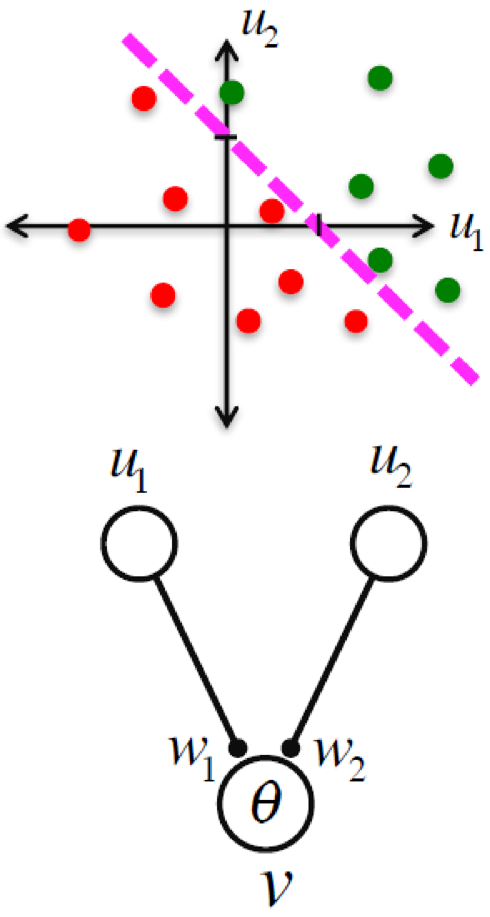
$$\vec{u}_a = (a , 0) \qquad \vec{u}_a \cdot \vec{w} = \theta$$

$$\vec{u}_b = (0 , b) \qquad \vec{u}_b \cdot \vec{w} = \theta$$

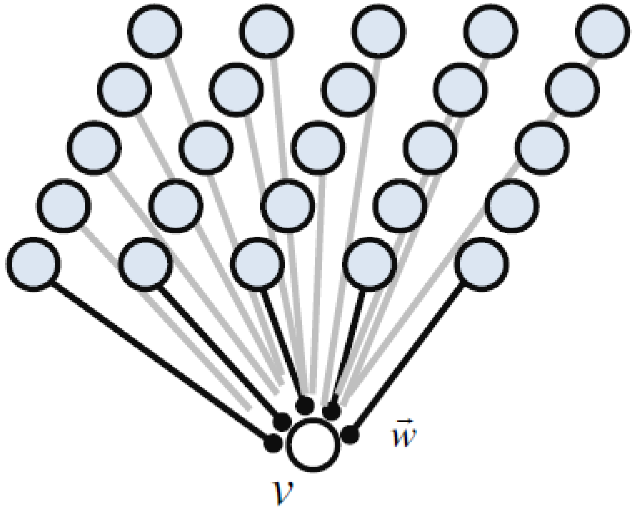
2023-06-20

Higher Dimensions

Low-dimensional

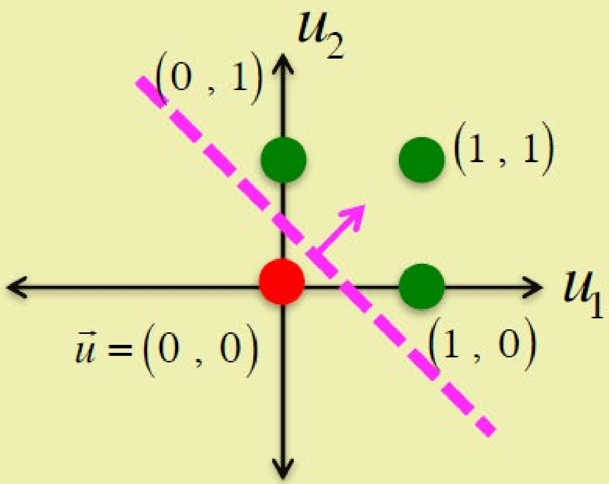


High-dimensional



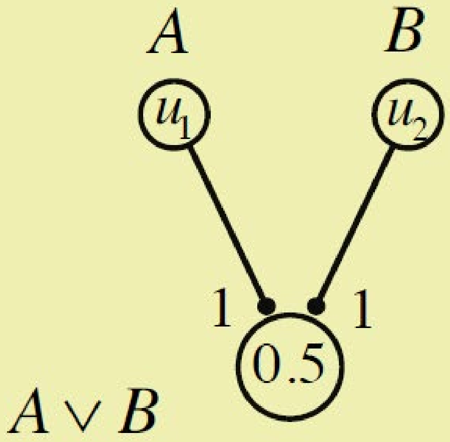
Neuronal Logic

OR

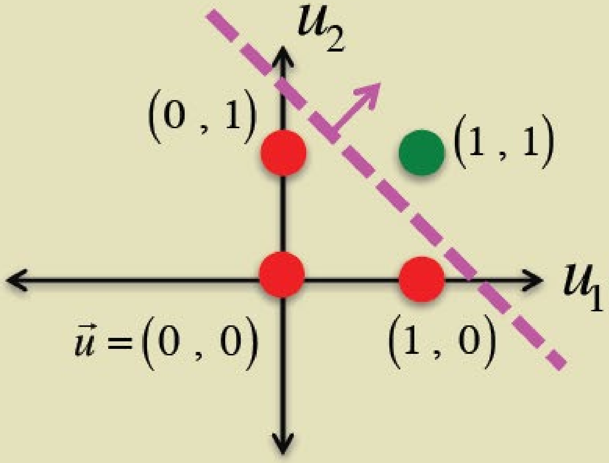


$\vec{w} = (1\ ,\ 1)$

$\theta\ =\ 0.5$

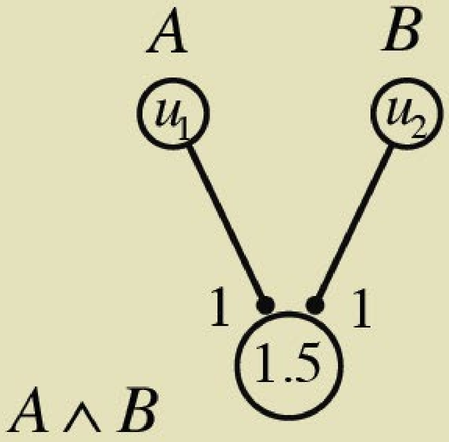


AND



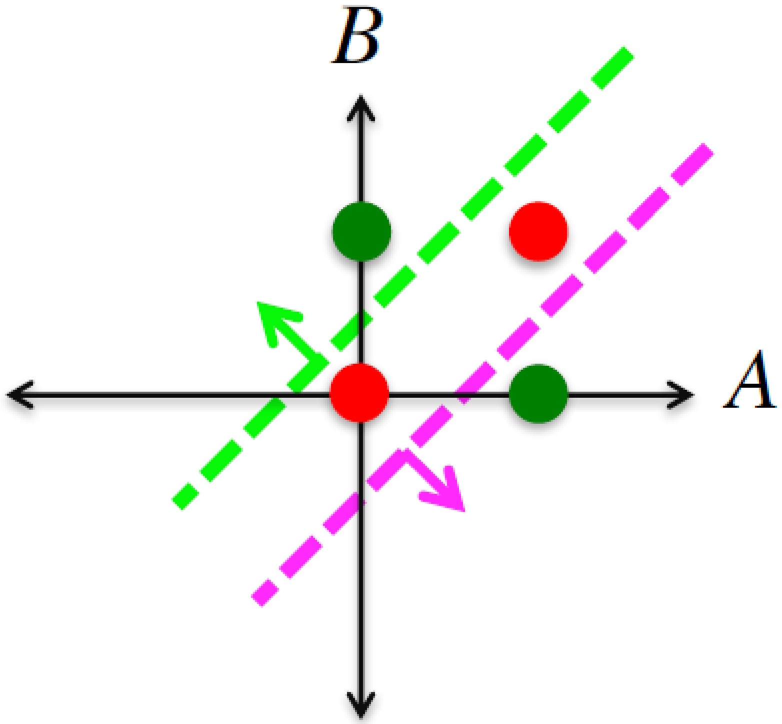
$\vec{w} = (1\ ,\ 1)$

$\theta\ =\ 1.5$

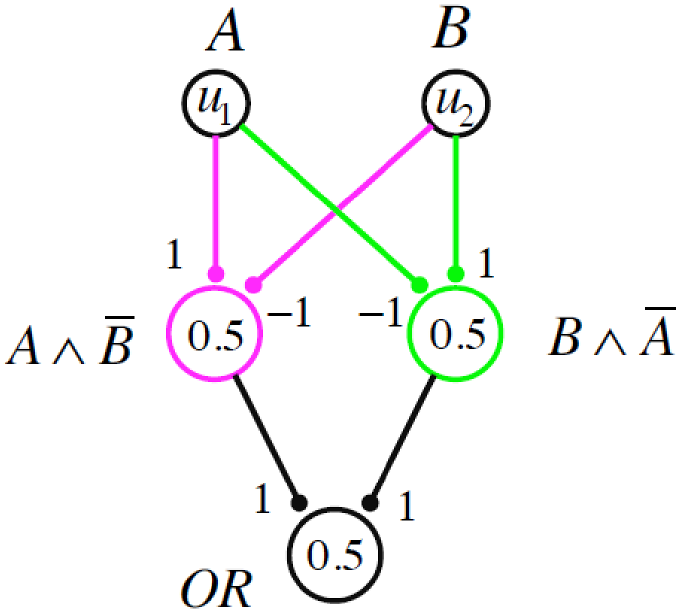


Linear Separability

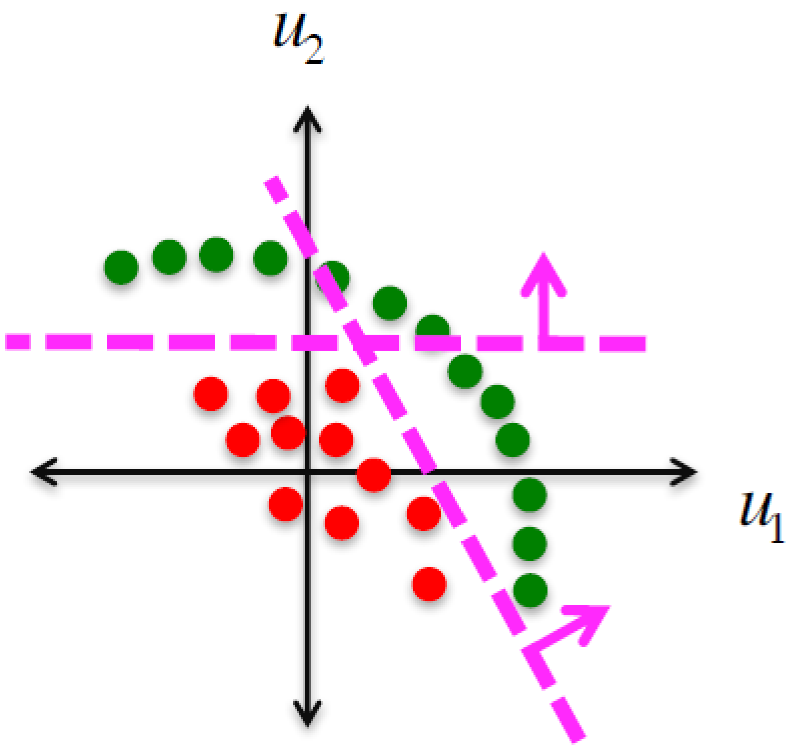
- Exclusive OR (XOR) – A or B but not both



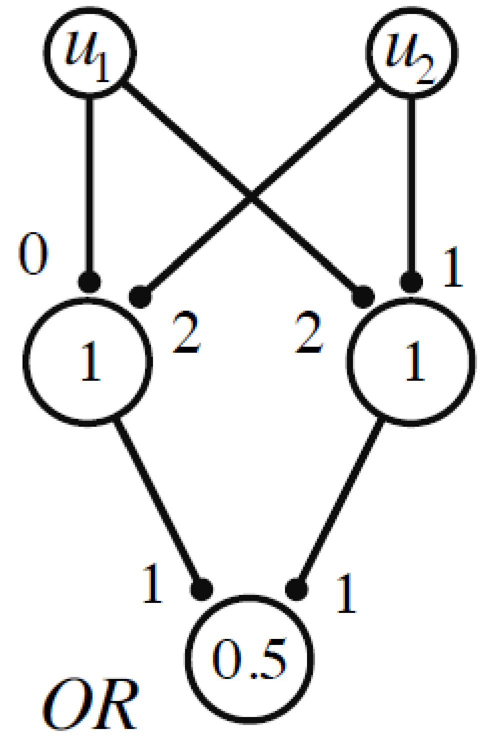
Multi-layer perceptron



Multi-Layer Perceptron

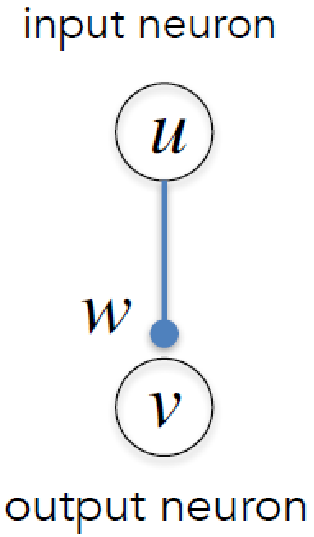


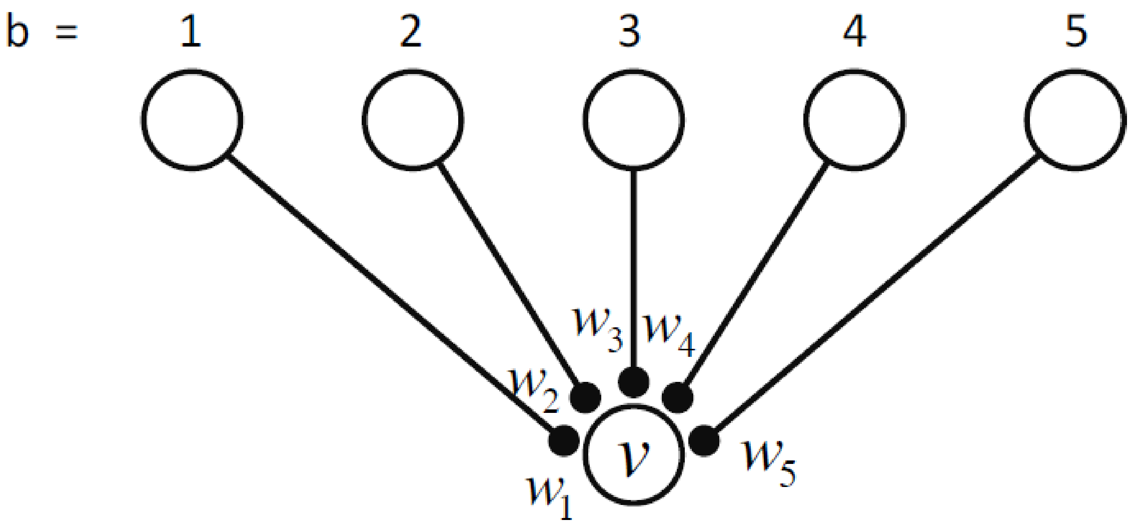
Multi-layer perceptron



So Far...

input firing rates $\left[u_1\,, u_2\,, u_3\,, \ldots\; u_{n_b}\;\right] = \vec{u}$





$$I_s = wu$$

$$v = F\left[wu\right]$$

$$I_s = \sum_b w_b u_b \;\; = \vec{w}\cdot\vec{u}$$

$$v = F\left[\vec{w}\cdot\vec{u}\right]$$

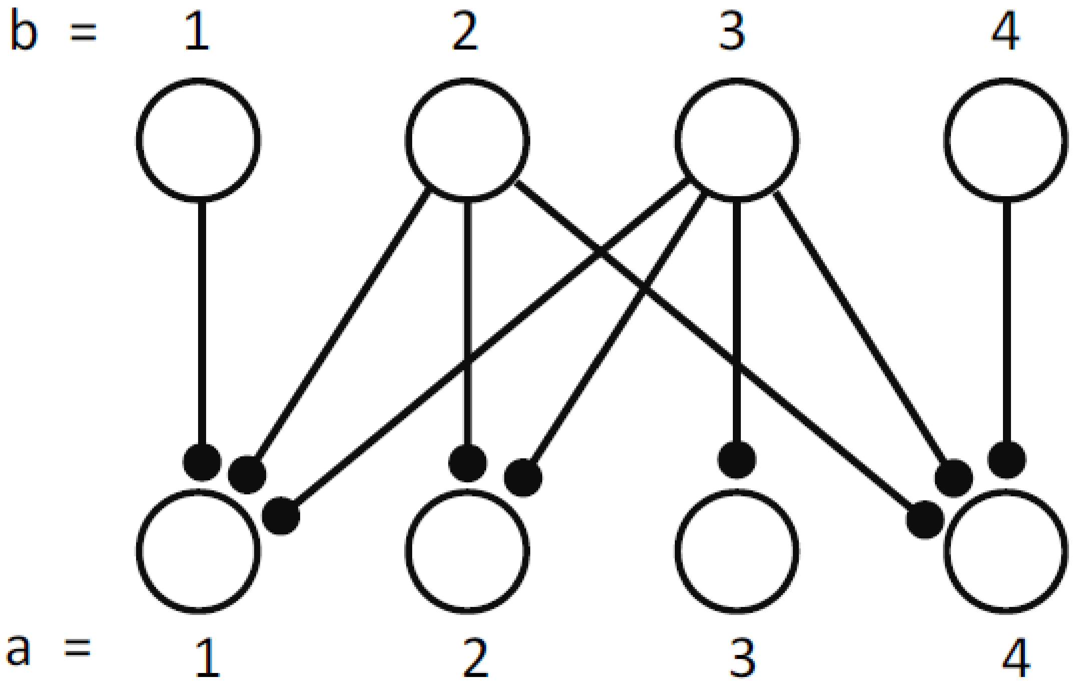
Two-Layer Feed-Forward Neural Network

input firing rates

$$\left[u_1 , u_2 , u_3 , \ldots \; u_{n_b} \right] = \vec{u}$$

output firing rates

$$\left[v_1 , v_2 , v_3 , \ldots \; v_{n_a} \right] = \vec{v}$$



2023-06-20

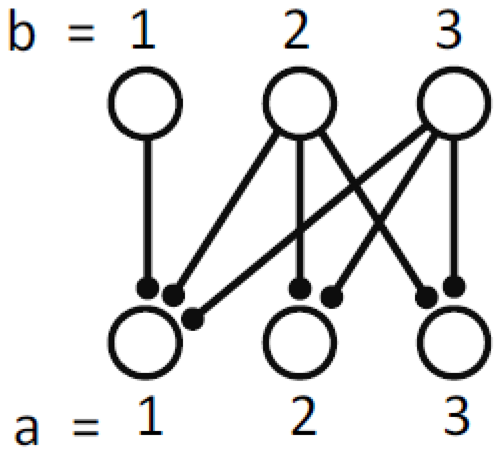
Wait, Matrix Multiplication?

$$v_1 = \vec{w}_{a=1} \cdot \vec{u} \qquad v_1 = \sum_b W_{1b} u_b$$

$$v_2 = \vec{w}_{a=2} \cdot \vec{u} \qquad v_2 = \sum_b W_{2b} u_b$$

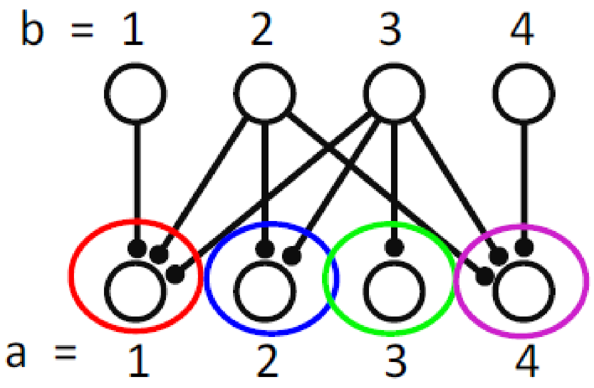
$$v_3 = \vec{w}_{a=3} \cdot \vec{u} \qquad v_3 = \sum_b W_{3b} u_b$$

$$v_a = \vec{w}_a \cdot \vec{u} \qquad v_a = \sum_b W_{ab} u_b$$



$$\vec{v} = W\,\vec{u}$$

Linear Algebra Comes to the Rescue



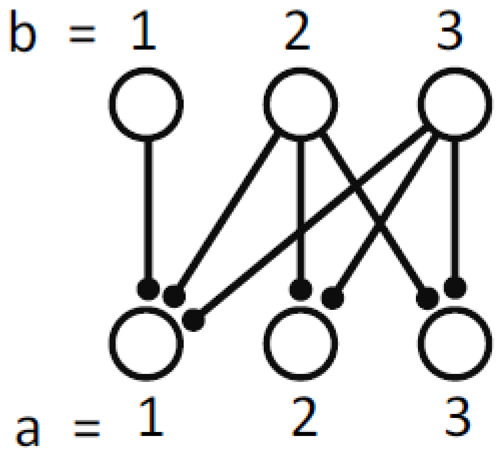
weight matrix

$$W_{ab} = \begin{array}{c|cccc} & b=1 & 2 & 3 & 4 \\ \hline a=1 & w_{11} & w_{12} & w_{13} & w_{14} \\ 2 & w_{21} & w_{22} & w_{23} & w_{24} \\ 3 & w_{31} & w_{32} & w_{33} & w_{34} \\ 4 & w_{41} & w_{42} & w_{43} & w_{44} \end{array} = \begin{bmatrix} \vec{w}_{a=1} \\ \vec{w}_{a=2} \\ \vec{w}_{a=3} \\ \vec{w}_{a=4} \end{bmatrix}$$

2023-06-20

Linear Algebra Comes to the Rescue

$$\vec{v} = W\,\vec{u} \qquad v_a = \sum_b W_{ab} u_b$$



$$\vec{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} \vec{w}_{a=1} \cdot \vec{u} \\ \vec{w}_{a=2} \cdot \vec{u} \\ \vec{w}_{a=3} \cdot \vec{u} \end{bmatrix}$$

2023-06-20

In a different perspective

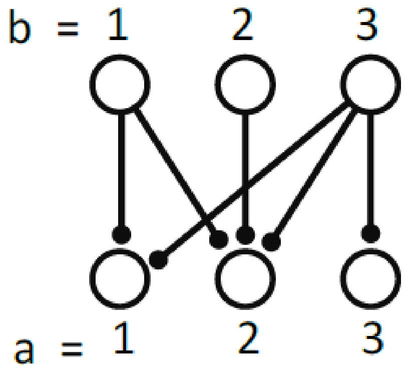
$$\vec{v} = W\vec{u} = \begin{array}{c} \begin{array}{ccc} b = 1 & 2 & 3 \end{array} \\ \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \end{array} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

$$\left[\vec{w}^{(1)} \;\middle|\; \vec{w}^{(2)} \;\middle|\; \vec{w}^{(3)} \right]$$

vector of weights from input neuron 1

vector of weights from input neuron 2

vector of weights from input neuron 3



$$W = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

No Way!

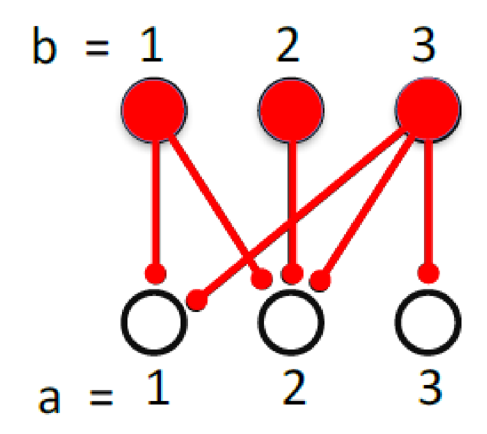
The output pattern is a linear combination of contributions from each of the input neurons!

$$\vec{v} = W\,\vec{u} = \begin{array}{c} \begin{array}{ccc} b=1 & 2 & 3 \end{array} \\ \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \end{array} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

$$\left[\, \vec{w}^{(1)} \;\middle|\; \vec{w}^{(2)} \;\middle|\; \vec{w}^{(3)} \,\right]$$

$$\vec{v} = u_1 \begin{bmatrix} w_{11} \\ w_{21} \\ w_{31} \end{bmatrix} + u_2 \begin{bmatrix} w_{12} \\ w_{22} \\ w_{32} \end{bmatrix} + u_3 \begin{bmatrix} w_{13} \\ w_{23} \\ w_{33} \end{bmatrix}$$

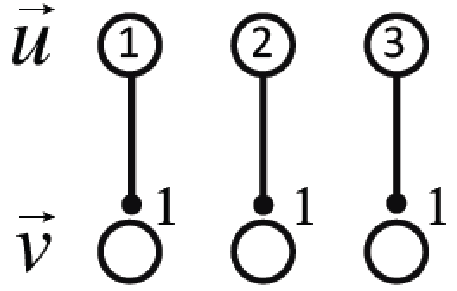
$$\vec{v} = u_1\, \vec{w}^{(1)} + u_2\, \vec{w}^{(2)} + u_3\, \vec{w}^{(3)}$$



2023-06-20

Identity!

$$W = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad W = I$$



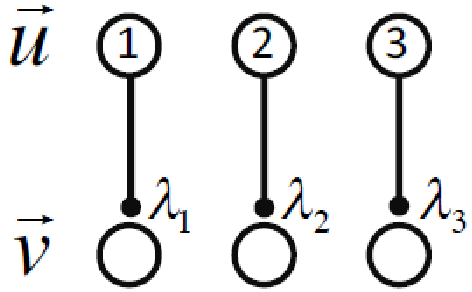
$$\vec{v} = W\vec{u} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

$$\vec{v} = \vec{u}$$

2023-06-20

Scaling!

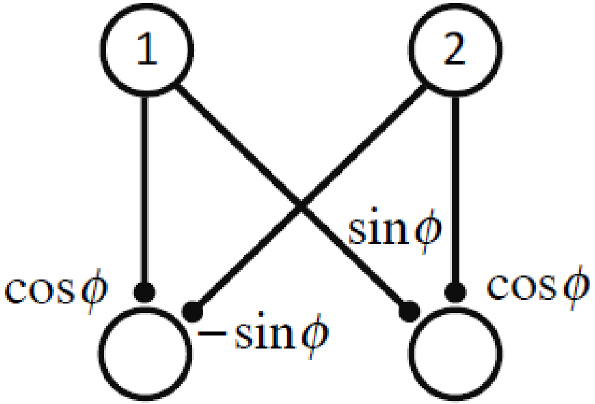
$$W = \Lambda \qquad \Lambda = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix}$$



$$\vec{v} = \Lambda \vec{u} = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} \lambda_1 u_1 \\ \lambda_2 u_2 \\ \lambda_3 u_3 \end{bmatrix}$$

Rotation!

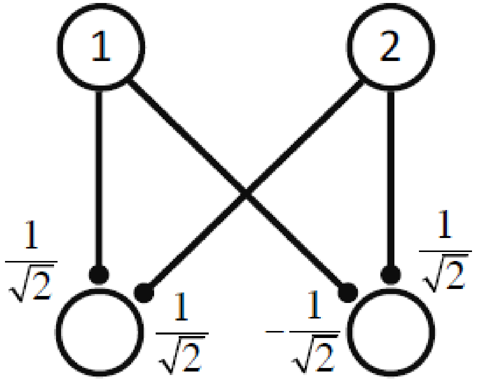
$$W = \Phi \qquad \Phi = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$$



$$\vec{v} = \Phi \cdot \vec{u} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} u_1\cos\phi - u_2\sin\phi \\ u_1\sin\phi + u_2\cos\phi \end{bmatrix}$$

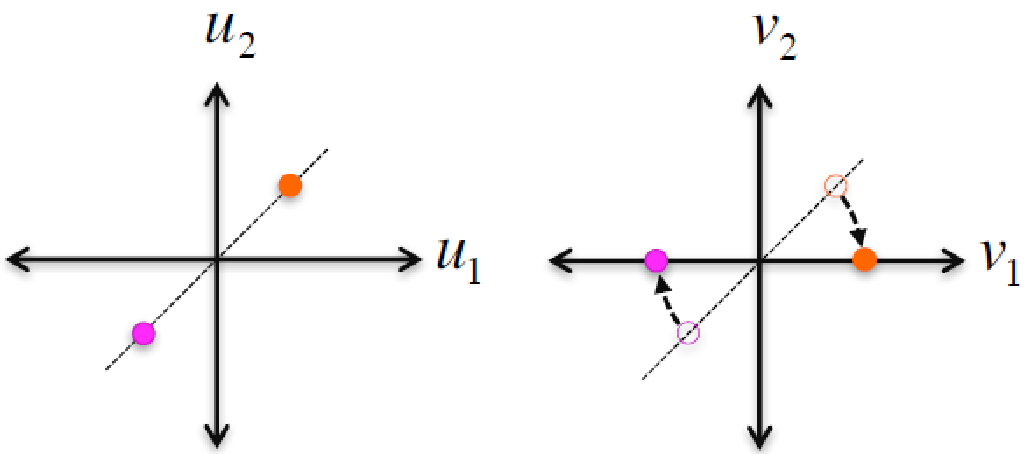
Rotation - Example

$$\Phi(-45^0) = \begin{bmatrix} \cos(-\frac{\pi}{4}) & -\sin(-\frac{\pi}{4}) \\ \sin(-\frac{\pi}{4}) & \cos(-\frac{\pi}{4}) \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$



$$\vec{v} = \Phi \cdot \vec{u} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

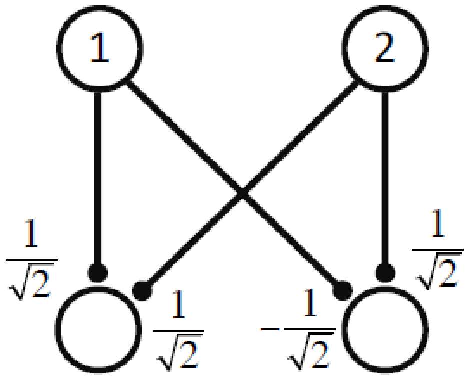
$$\vec{v} = \frac{1}{\sqrt{2}} \begin{bmatrix} u_2 + u_1 \\ u_2 - u_1 \end{bmatrix}$$

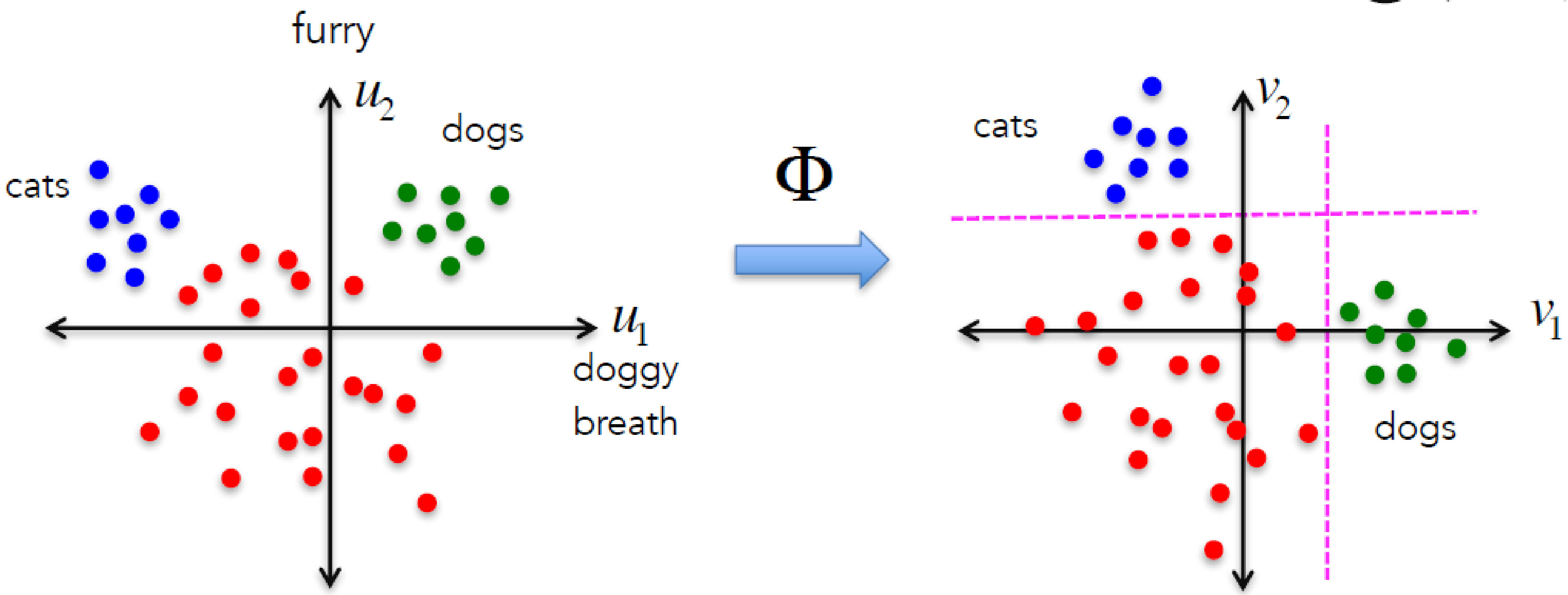


2023-06-20

Rotation in Perceptron

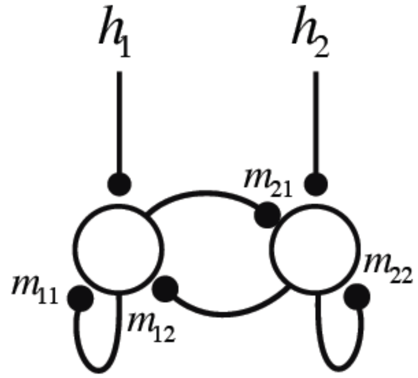
$$\Phi(-45^0) = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$



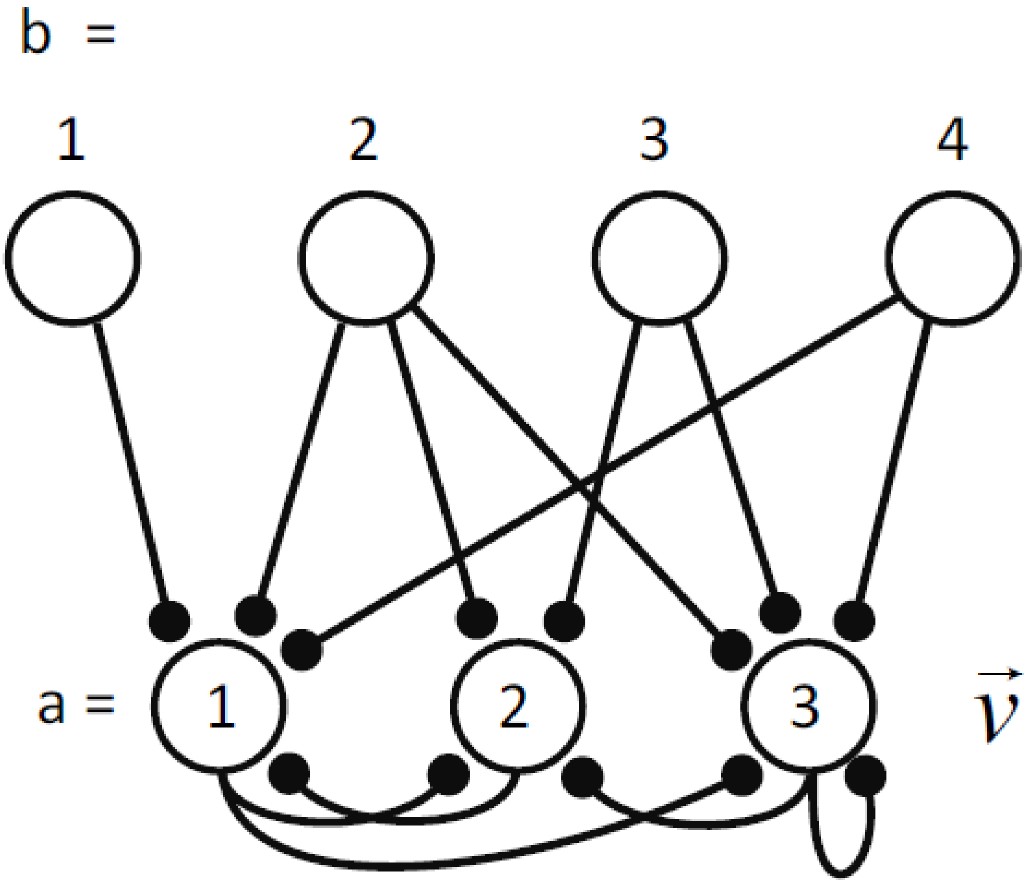


2023-06-20

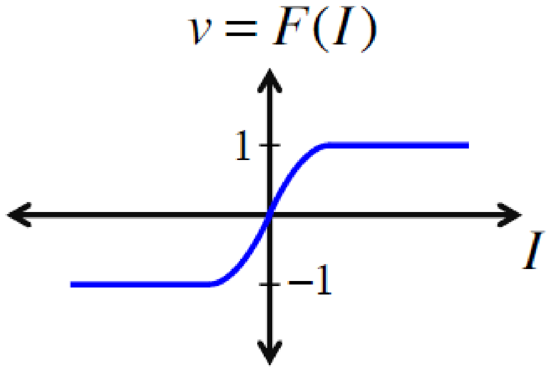
Recurrent Networks

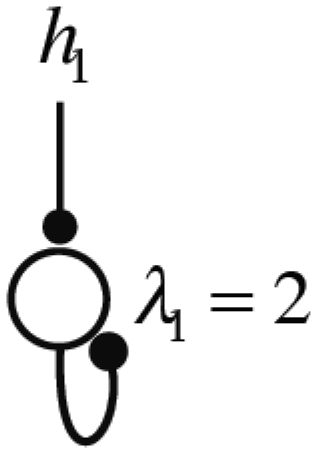


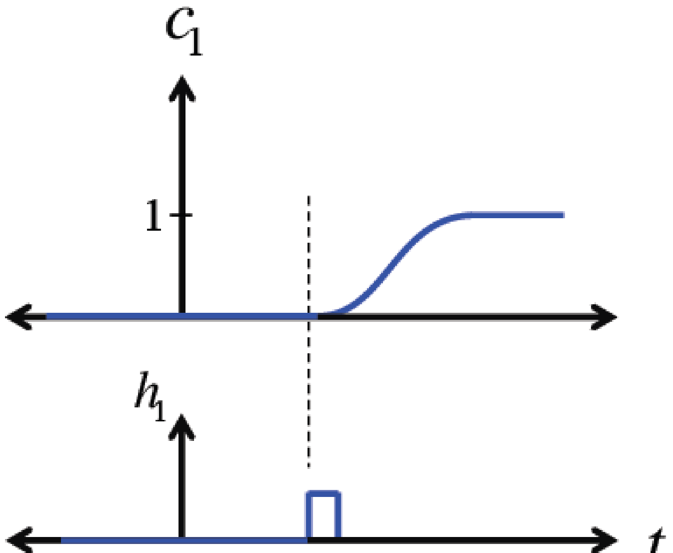
$$M = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix}$$

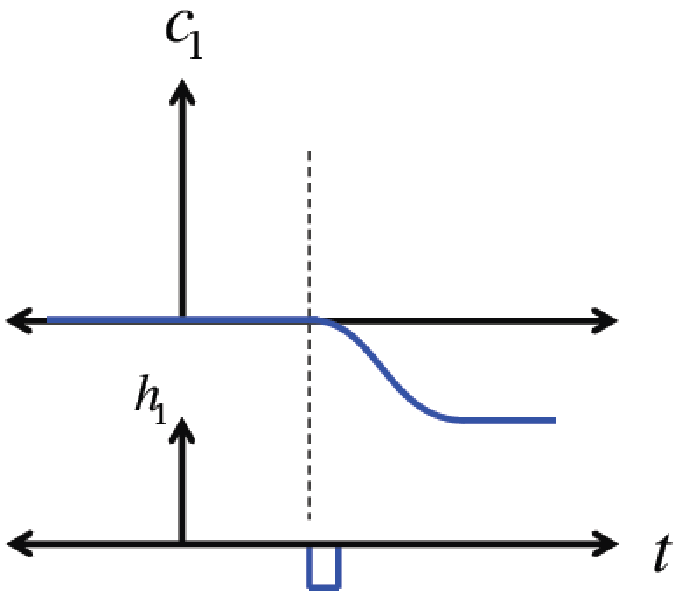


Saturating Activation Function



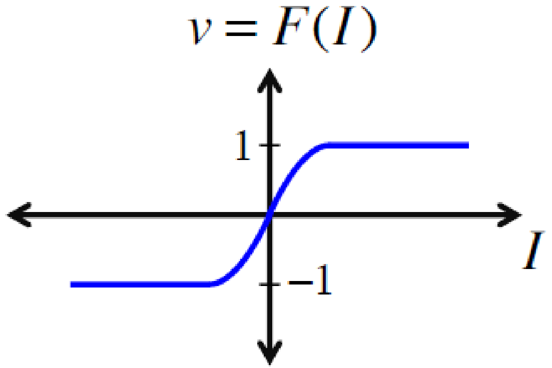






2023-06-20

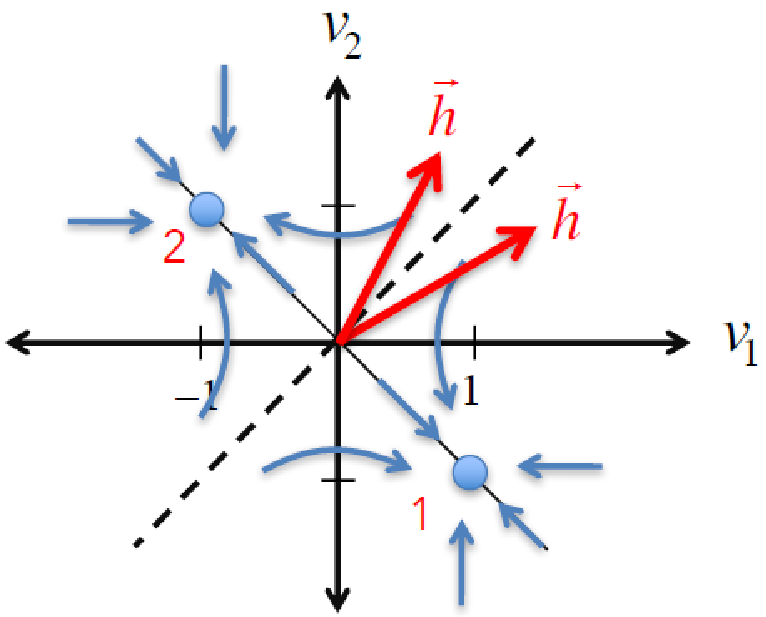
Winner-Take-All Network

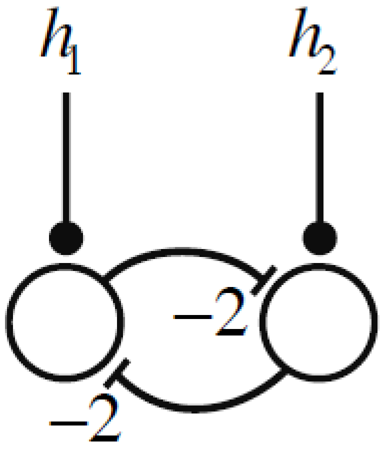


- Implements decision making

Network will remain in attractor 1 if $h_1 > h_2$

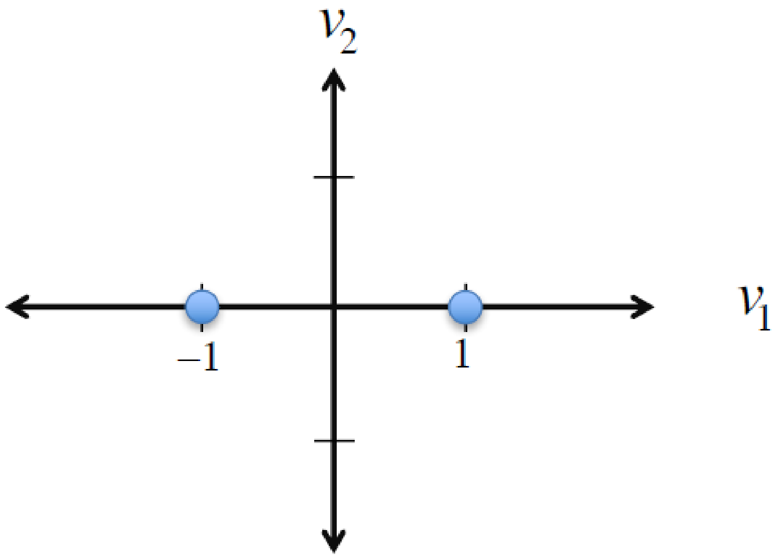
Network will remain in attractor 2 if $h_2 > h_1$

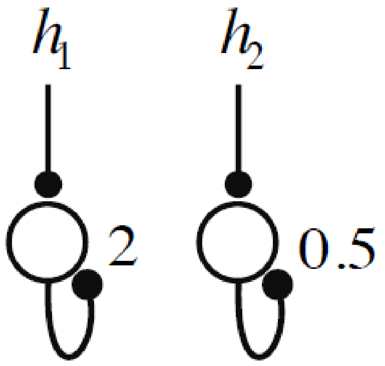


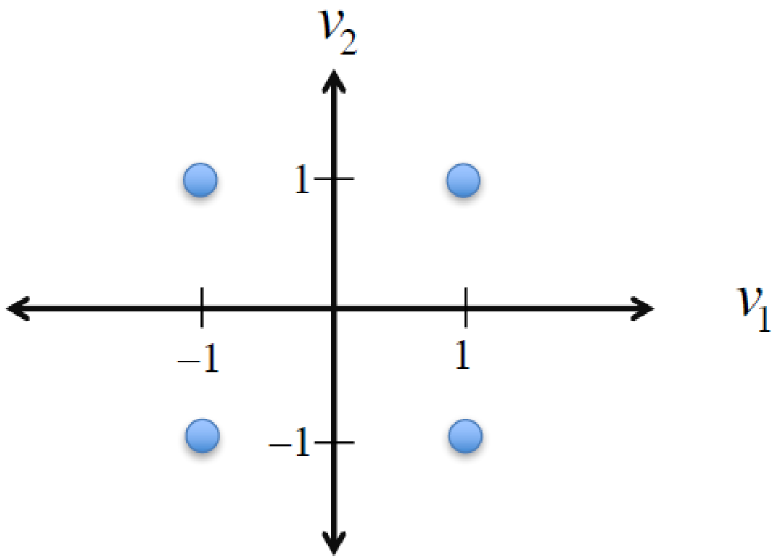


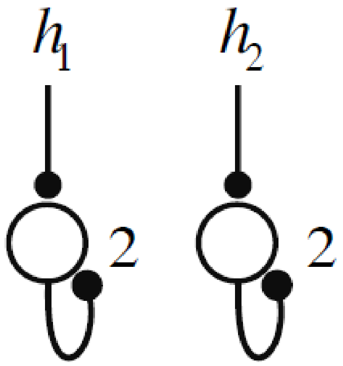
2023-06-20

Memory!

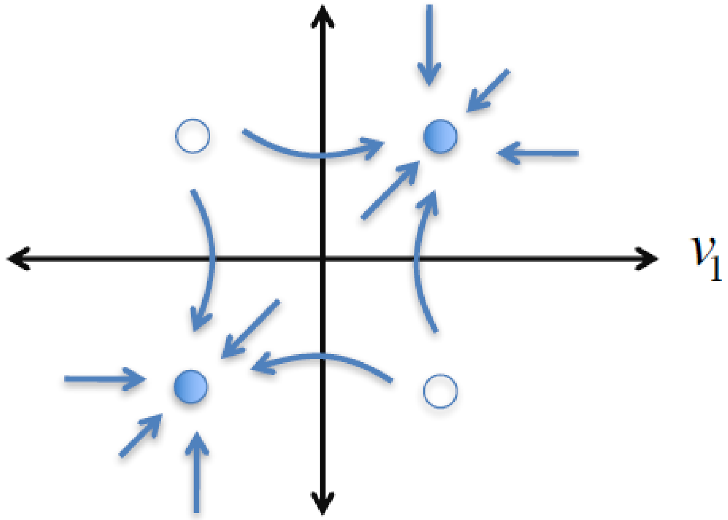


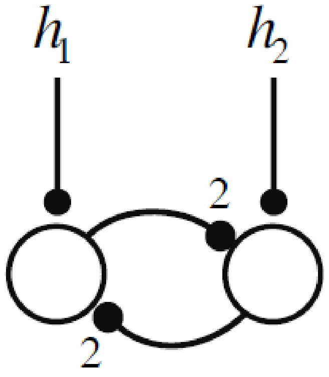


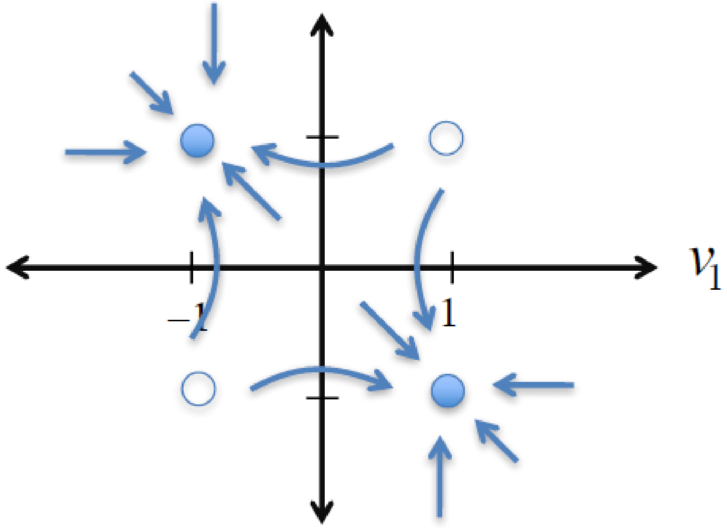


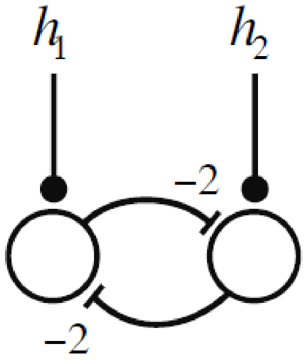


Hopfield Networks









THANK YOU!

Adibvafa Fallahpour