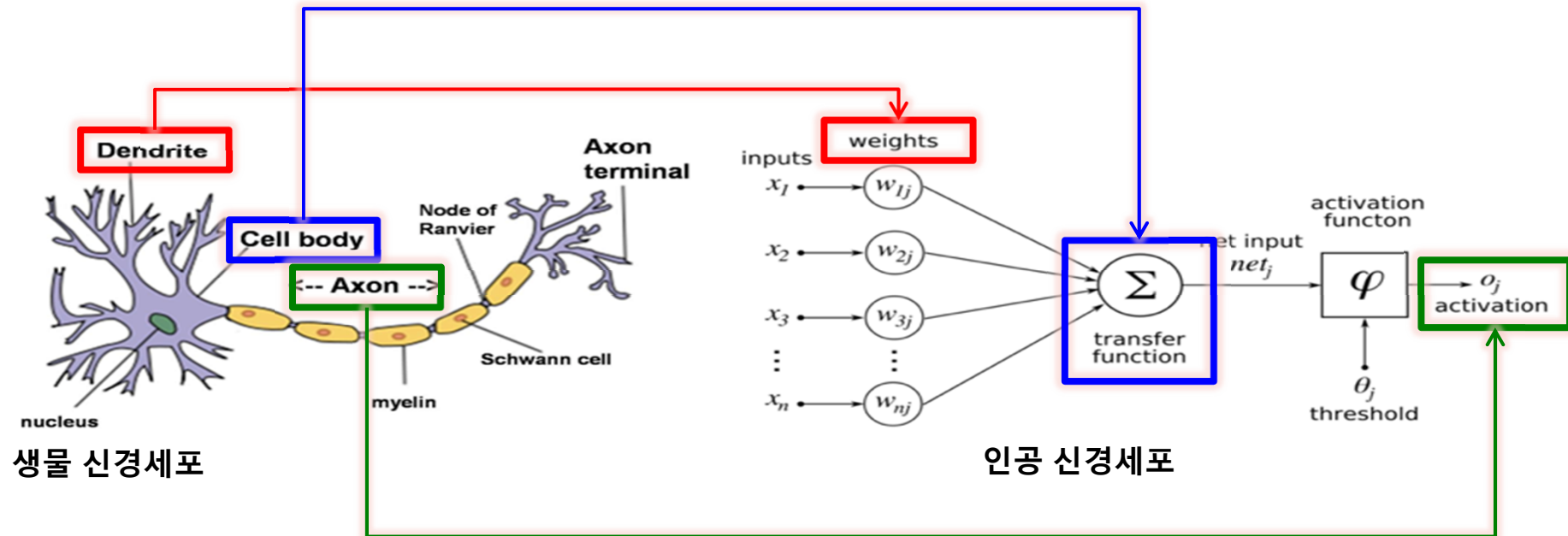


# 신경망이란?

신경망이란?

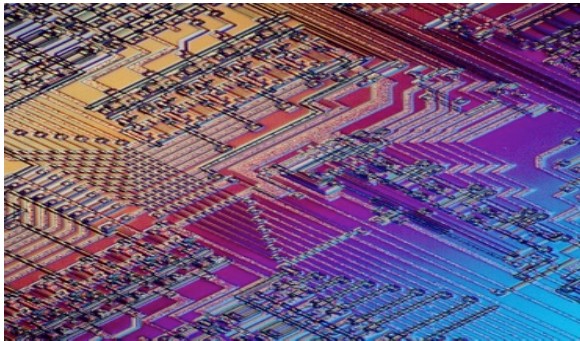
→ 뇌의 신경세포망에서의 정보처리 기작을 모사한 머신러닝 모델



## 인공 신경세포 (Artificial Neuron)

# 인공 신경망과 생물학적 신경망

## 컴퓨터와 뇌의 비교



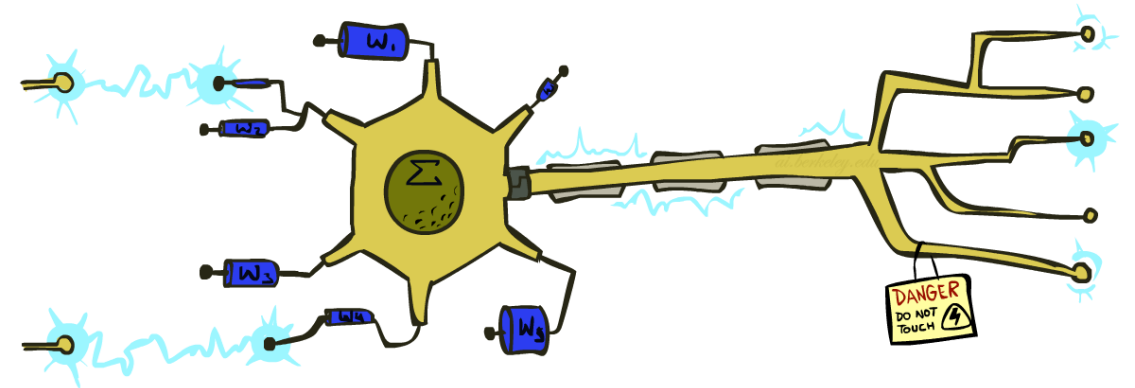
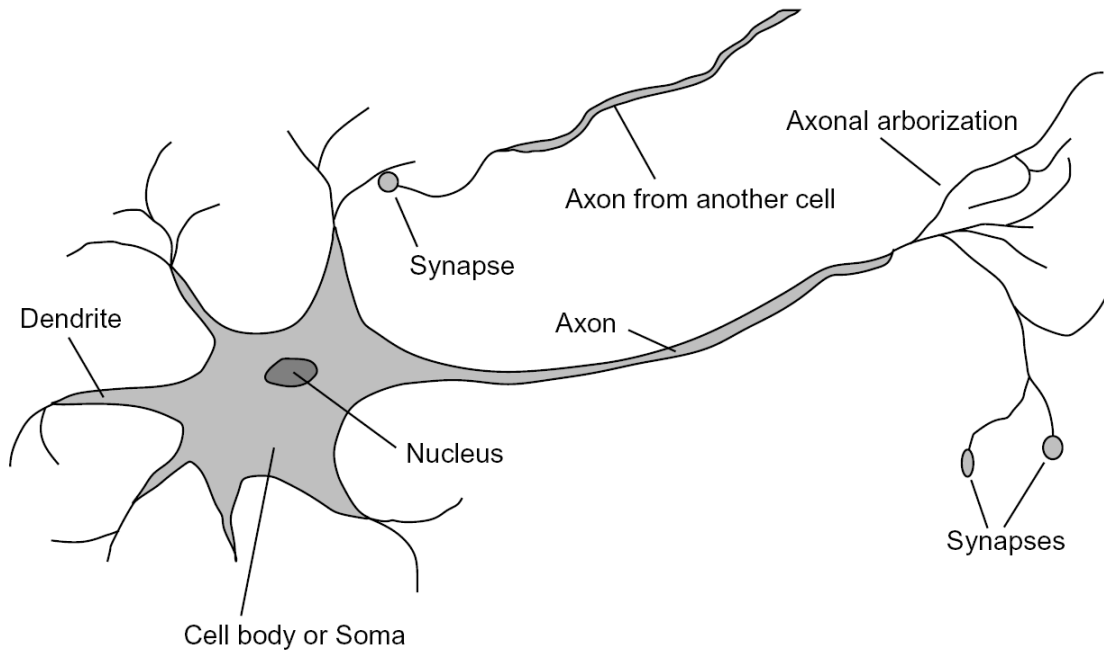
- 정확한 소자 (논리 소자)
- 소자 속도 매우 빠름 ( $10^{-9}$  초)
- 디지털 회로망 (전자 회로망)
- 주소기반 메모리 (국지적/독립적)
- 논리/산술적 연산 (조작적)
- 중앙집중식 순차 처리
- 프로그래밍기반 명시적 지식



- 부정확한 소자 ( $10^{11}$  개 뉴런)
- 소자 속도 느림 ( $10^{-3}$  초)
- 아날로그 회로망 ( $10^{14}$  개 연결선)
- 내용기반 메모리 (전역적/관계적)
- 패턴/연상기반 연산 (연관적)
- 분산적 병렬 처리
- 학습(경험/데이터)기반 암묵적 지식

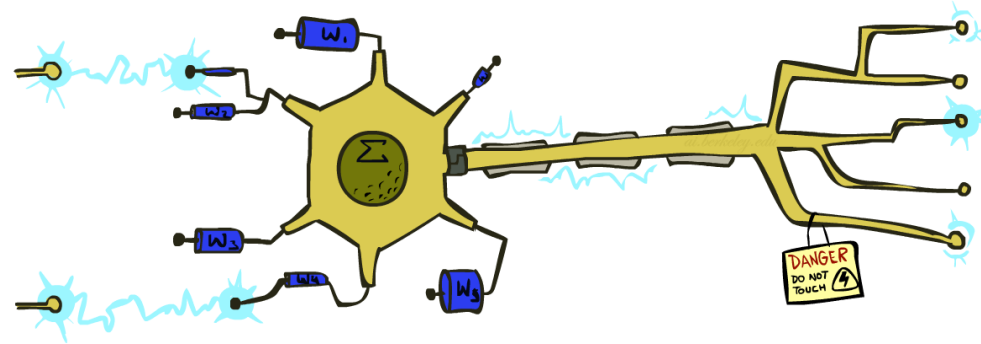
# Some (Simplified) Biology

- Very loose inspiration: human neurons



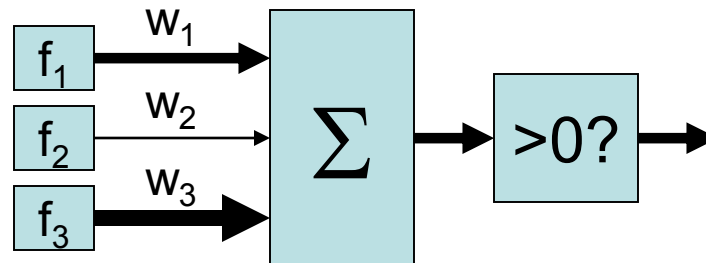
# Linear Classifiers

- Inputs are **feature values**
- Each feature has a **weight**
- Sum is the **activation**



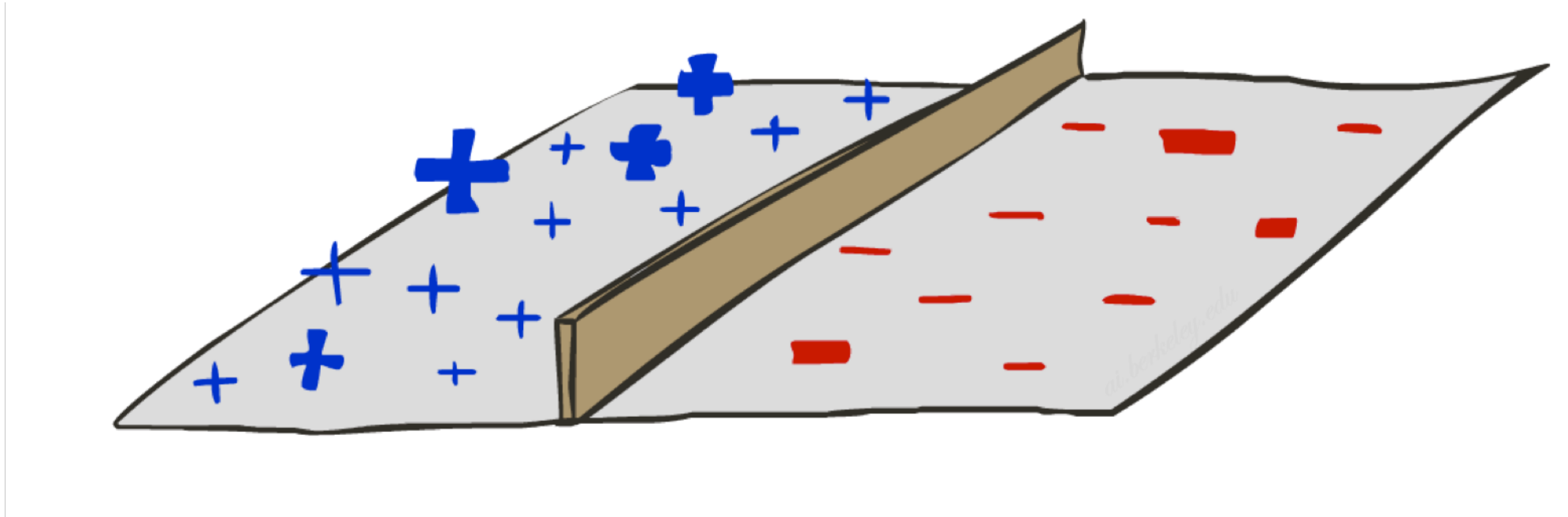
$$\text{activation}_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)$$

- If the activation is:
  - Positive, output +1
  - Negative, output -1



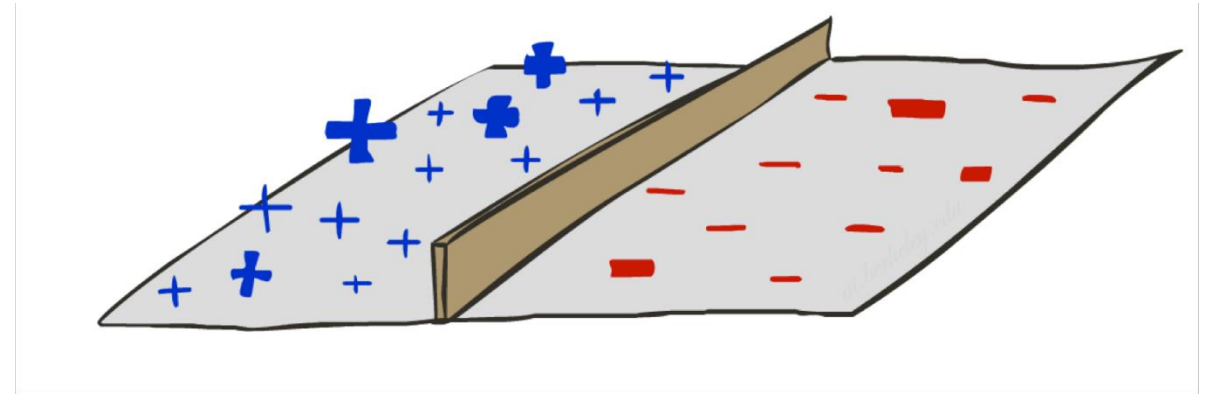
# Decision Rules

---



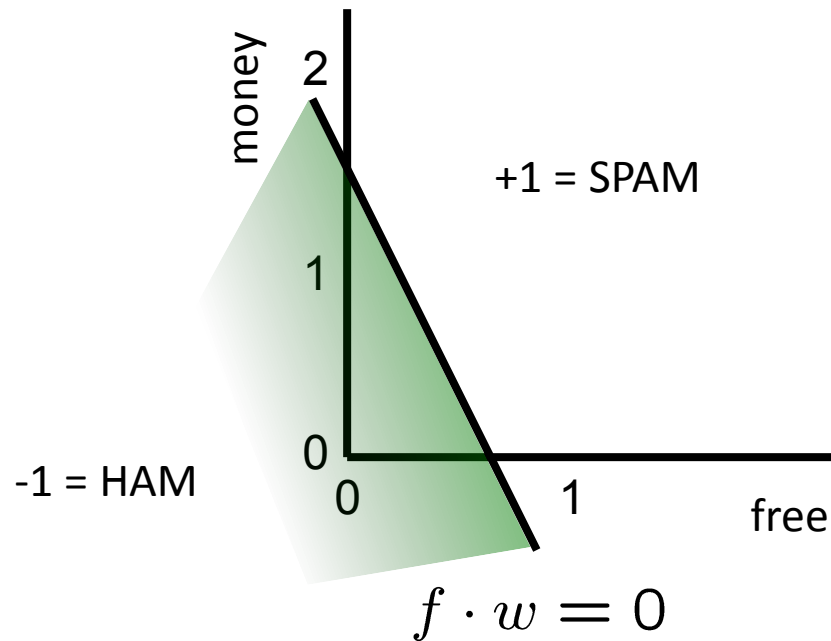
# Binary Decision Rule

- In the space of feature vectors
  - Examples are points
  - Any weight vector is a hyperplane
  - One side corresponds to  $Y=+1$
  - Other corresponds to  $Y=-1$



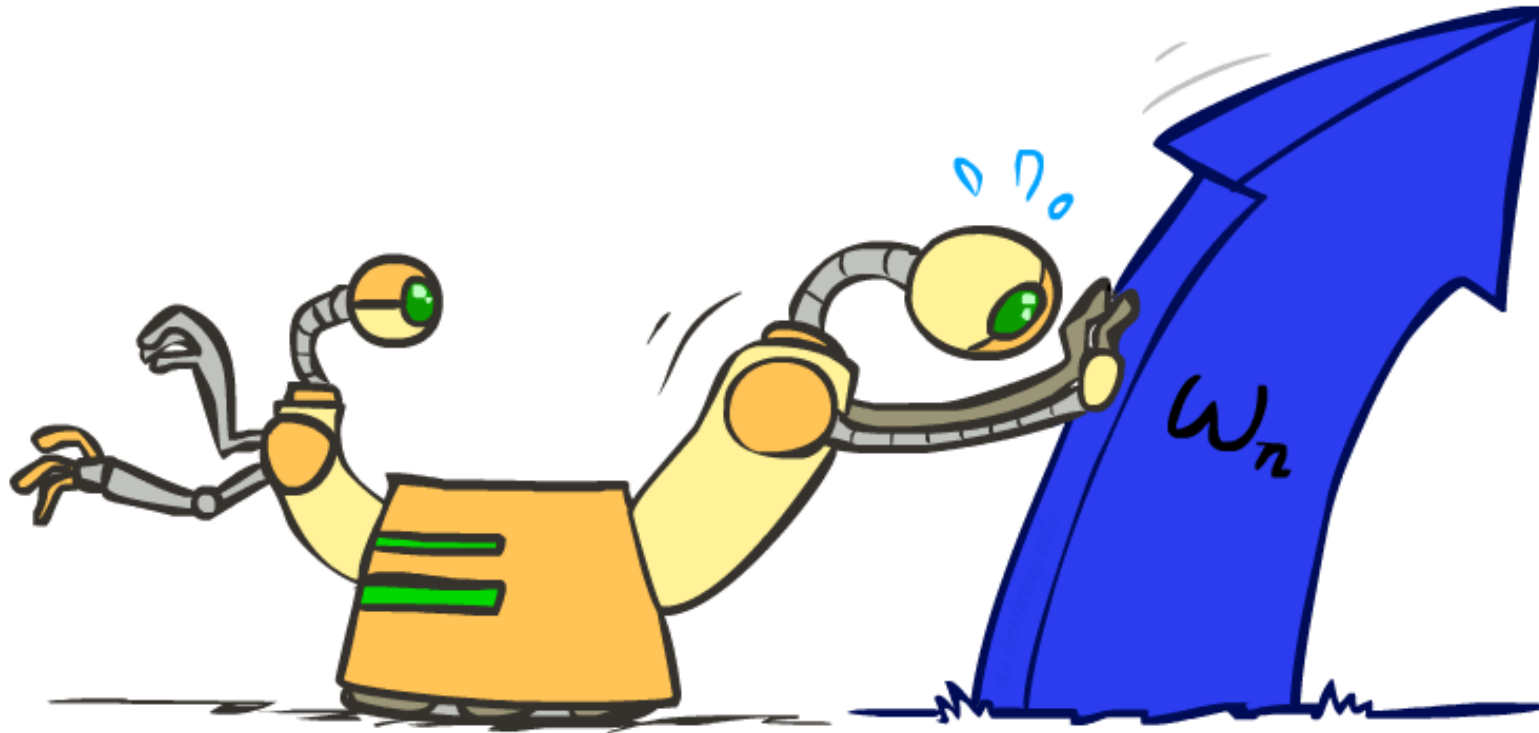
$w$

BIAS	:	-3
free	:	4
money	:	2
...		



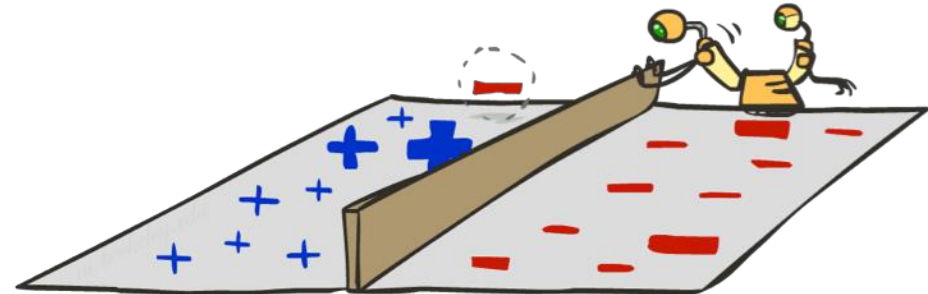
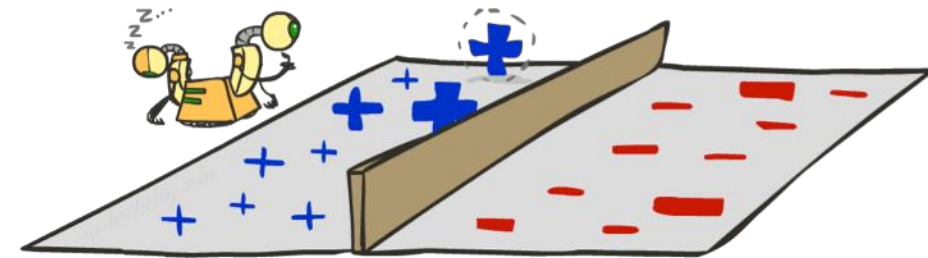
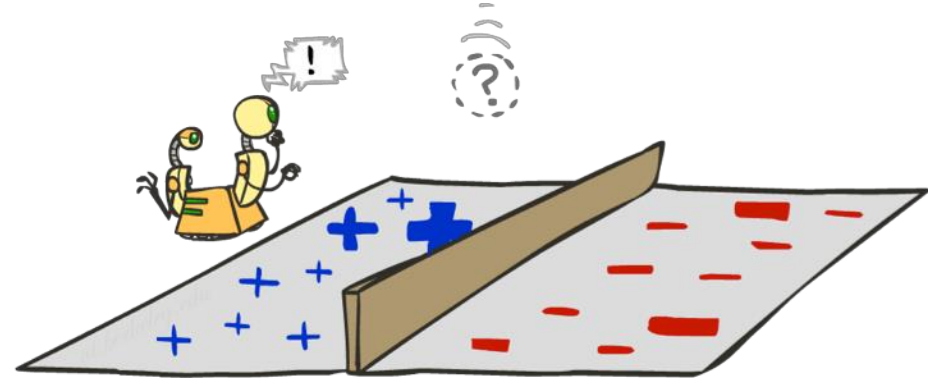
# Weight Updates

---



# Learning: Binary Perceptron

- Start with weights = 0
- For each training instance:
  - Classify with current weights
- If correct (i.e.,  $y=y^*$ ), no change!
- If wrong: adjust the weight vector

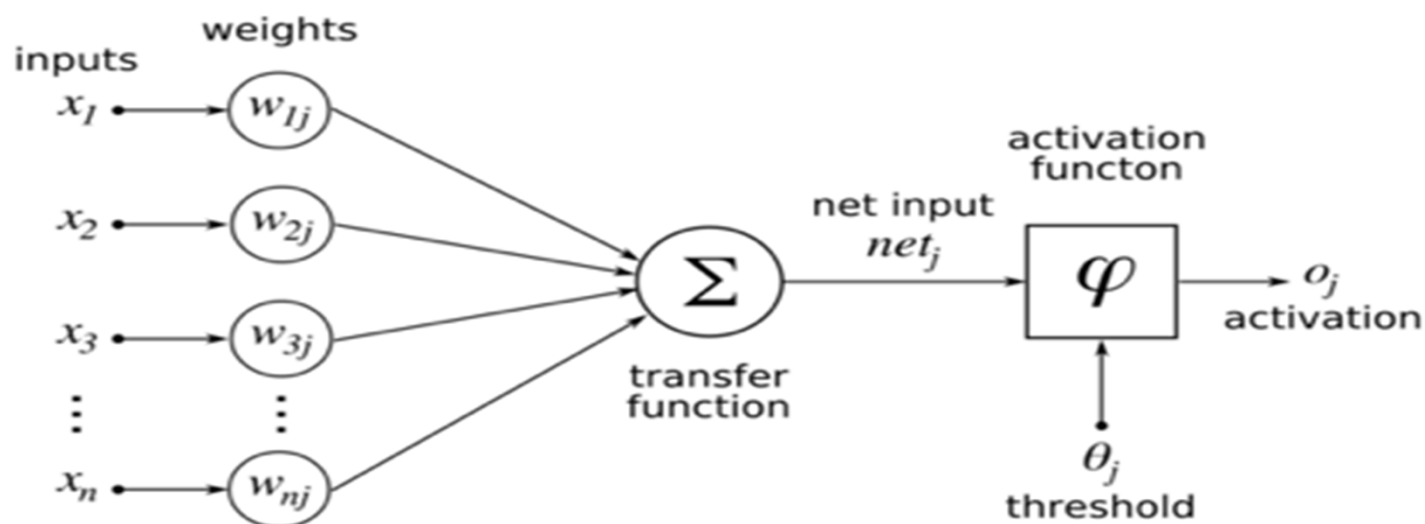




# 퍼셉트론 (Perceptron)

## ■ 퍼셉트론

- 생물학적 신경세포의 작동 방식을 수학적으로 모델링한 인공 뉴런
- 뇌의 구조를 모사한 정보처리 방식
  - 1958년 Rosenblatt가 학습이 가능함을 증명



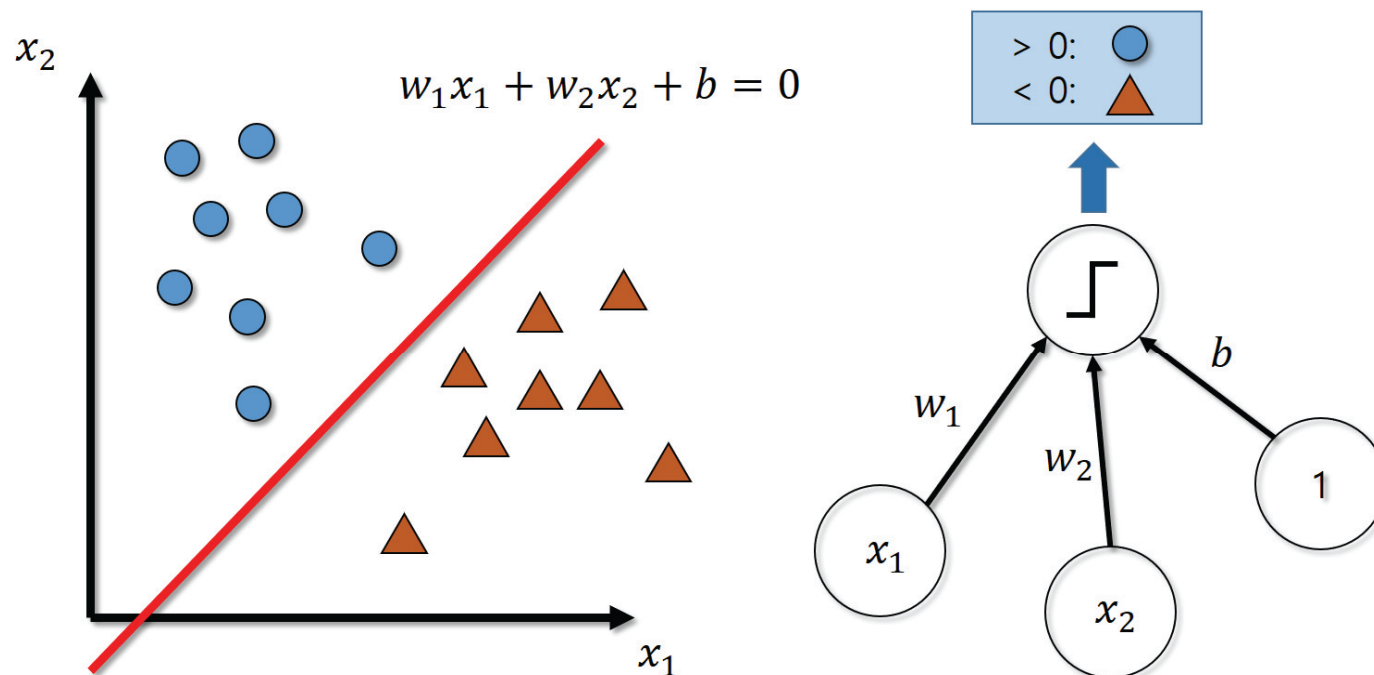
## 인공 신경세포 (Artificial Neuron)

# 퍼셉트론 (Perceptron)

## ■ 퍼셉트론

- 목표: 주어진 데이터를 기준으로 두 개의 클래스로 분류하는 **경계면 찾기**
- 인접한 두 뉴런의 연결부분인 **시냅스**에서 학습이 일어남
- 출력노드의 값

$$s = \sum_{i=1}^n w_i x_i + w_0 = \sum_{i=0}^n w_i x_i = \mathbf{w} \cdot \mathbf{x}$$



# 퍼셉트론 (Perceptron)

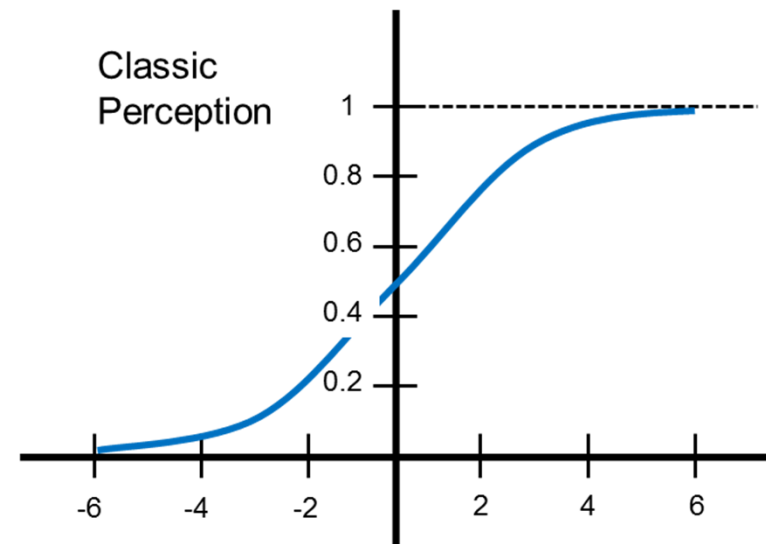
## ■ 활성화 함수 (Activation function) $s = \mathbf{w} \cdot \mathbf{x}$

- 경계면의 부근의 모양을 결정
- 선형 활성화 함수 (Linear Activation function)

$$y = s$$

- 시그모이드 함수 (sigmoid function)

$$y = \sigma(s) = \frac{1}{1 + e^{-s}}$$

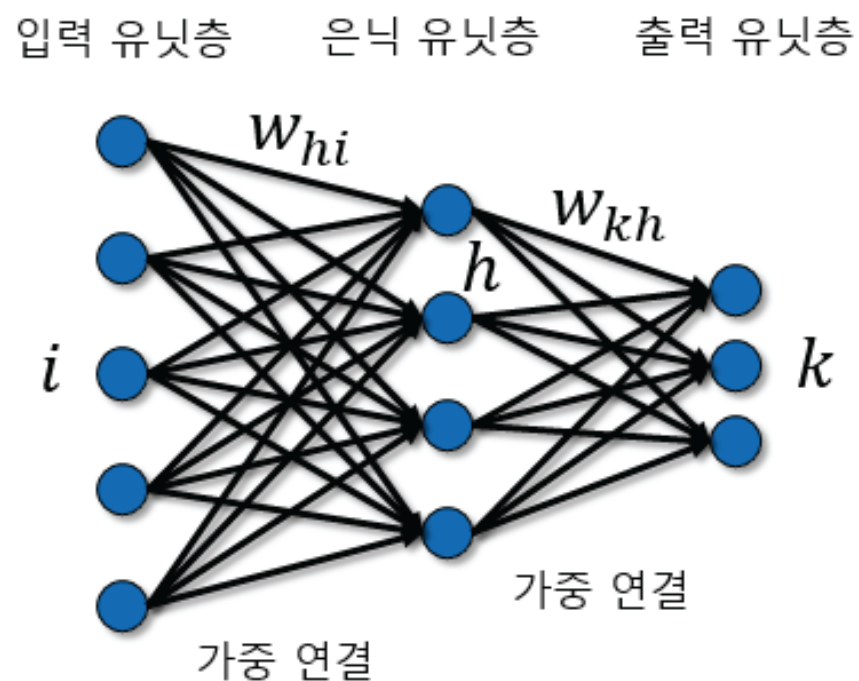


# 여러 퍼셉트론들

- 단순 퍼셉트론(Simple Perceptron)
  - 활성화 함수로 임계논리 활성화 함수를 가지는 퍼셉트론
  - 임계논리 활성화 함수(threshold logic unit)
    - 입력의 총합이 임계치를 넘으면 +1, 그렇지 않으면 -1을 출력

# 다층퍼셉트론

## ■ 다층퍼셉트론 신경망(MLP)



# 다층퍼셉트론

## ■ 다층 뉴런의 필요성

- 복잡한 패턴 분류를 위해서는 입출력 간의 복잡한 변환 구조가 필요
- 단일 뉴런으로는 **선형분리**가 가능한 결정 경계선만을 생성 가능
- 사용하는 뉴런의 수를 늘리고 층을 추가하여 복잡한 구조의 의사결정 경계(decision boundary)를 생성할 수 있음


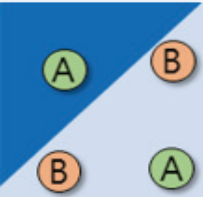
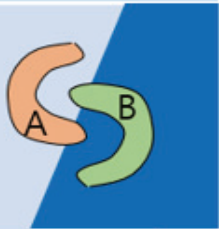
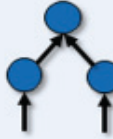
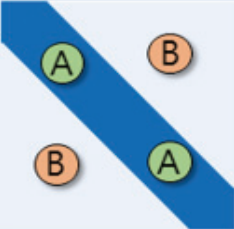
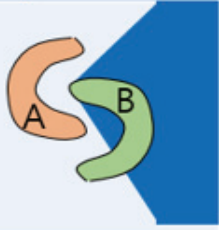


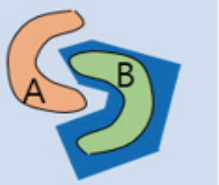
## ■ 다층퍼셉트론 (Multilayer Perceptron, MLP)

- 여러 개의 퍼셉트론 뉴런을 여러 층으로 쌓은 다층신경망 구조
- 각 층 안에서는 뉴런간 연결이 없으며 **인접한 두 층의 뉴런 간에는 완전 연결됨**
- 층별 활성화값
  - $j$ : 시냅스후 뉴런,  $i$ : 시냅스전 뉴런

$$s_j = \sum_i w_{ji} x_i \quad f_j = \sigma(s_j) = \frac{1}{1 + \exp(-s_j)}$$

# 다층퍼셉트론

## ■ 다수 / 다층 뉴런의 필요성

Structure	Regions	XOR	Meshed Regions
Single layer 	Halfplane bounded by hyperplane		
Two layers 	Convex Open or closed regions		
Three layers 	Arbitrary (limited by # of nodes)		

Multiple boundaries needed  
(e.g. XOR problem)  
→ **Multiple units**

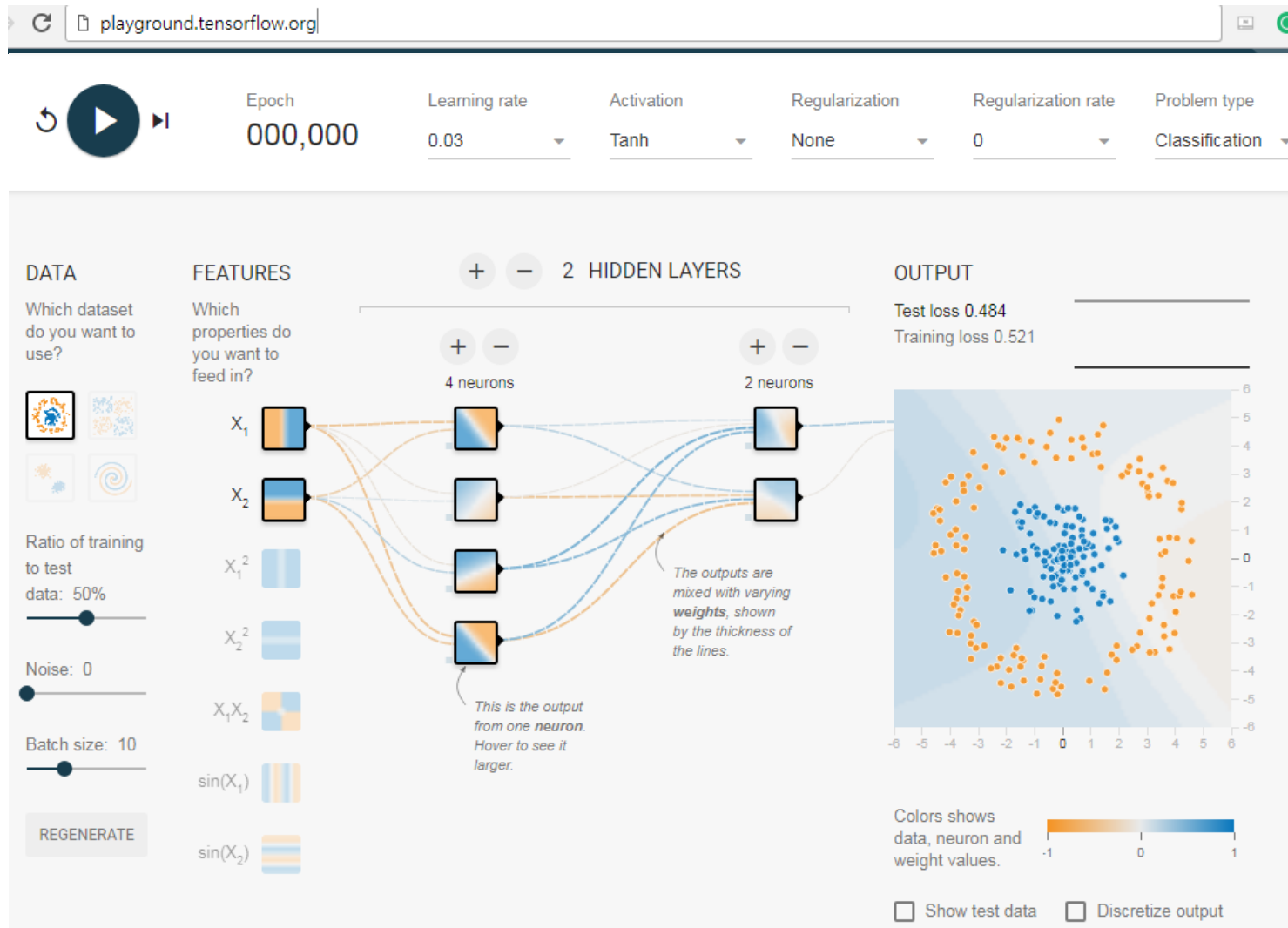
More complex regions needed  
(e.g. Polygons)  
→ **Multiple layers**

# 다층퍼셉트론


- 다층퍼셉트론의 혁신
  - 시그모이드 뉴런의 사용
    - 비선형 모델로 확장 가능
    - 연속함수이므로 미분 가능
    - 은닉 뉴런층의 학습을 가능하게 하는 오류역전파 알고리즘의 개발로 이어짐
  - 은닉 뉴런층 추가
  - 은닉층 학습방법 도입



# Playground (playground.tensorflow.org)



[→](#) [↺](#) [🔒](#) GitHub, Inc. [US] | <https://github.com/tensorflow/playground> [★](#) [📺](#)

 This repository Search Pull requests Issues Marketplace Explore [🔔](#) [+](#) [👤](#)


tensorflow / playground [👁 Watch](#) 400 [★ Star](#) 6,373 [🍴 Fork](#) 1,207

[↔ Code](#) [🔔 Issues](#) 43 [🔗 Pull requests](#) 5 [📁 Projects](#) 0 [📖 Wiki](#) [📊 Insights](#)

Play with neural networks! <http://playground.tensorflow.org>

[📌 216 commits](#) [🌿 4 branches](#) [📦 0 releases](#) [👥 12 contributors](#) [📄 Apache-2.0](#)

[Branch: master](#) [New pull request](#) [Create new file](#) [Upload files](#) [Find file](#) [Clone or download](#)

 dsmilkov committed on 18 Jan Check for good noise values Latest commit 91149e1 on 18 Jan

[src](#) Update playground.ts 2 months ago

[.gitignore](#) ignore dist folder from git 2 years ago

[📖 README.md](#)

# Deep playground

Deep playground is an interactive visualization of neural networks, written in TypeScript using d3.js. We use GitHub issues for tracking new requests and bugs. Your feedback is highly appreciated!

If you'd like to contribute, be sure to review the [contribution guidelines](#).

## Development

To run the visualization locally you just need a server to serve all the files from the `dist` directory. You can run `npm install`