Introduction of Machine Learning

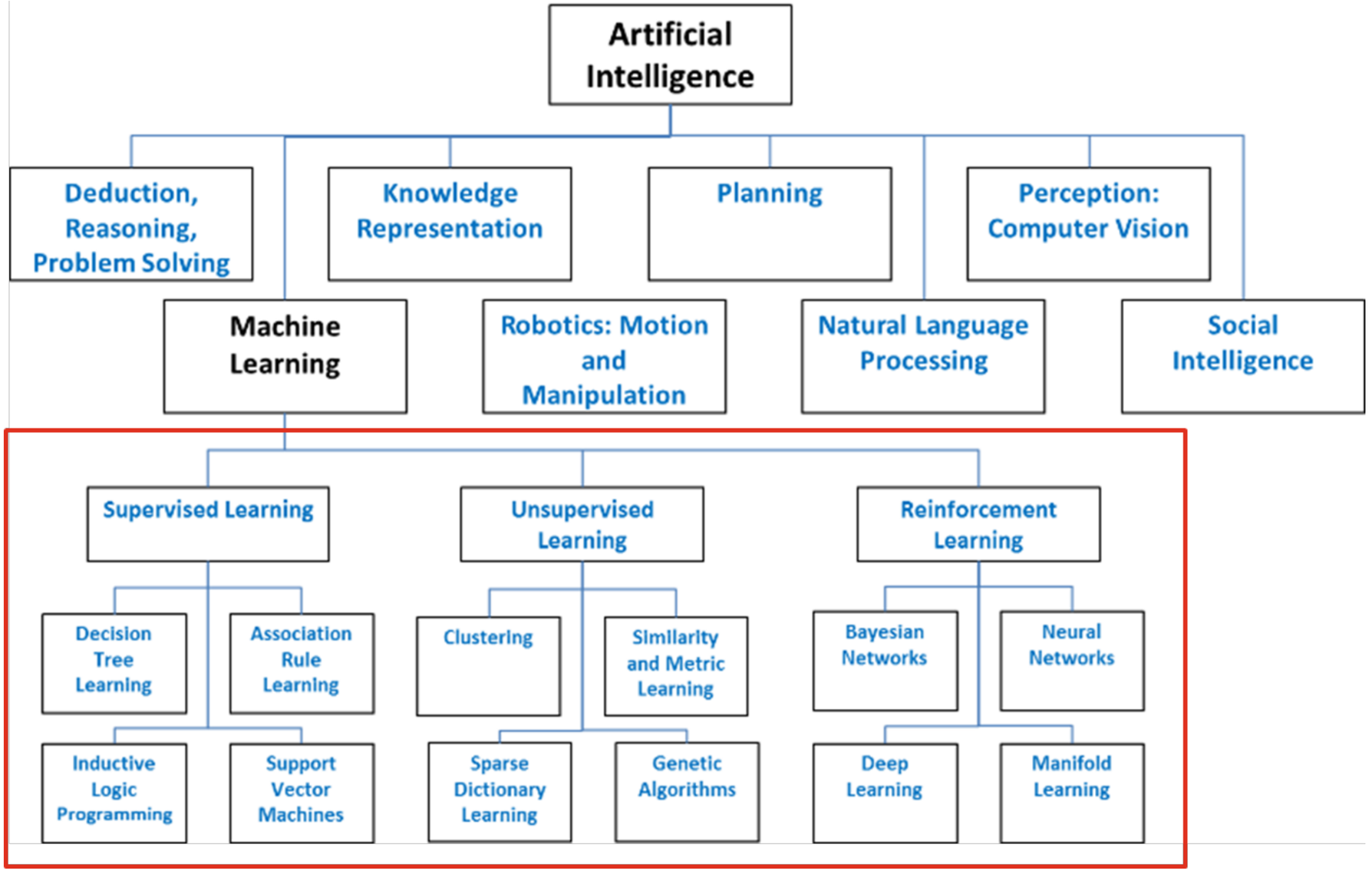
Concept of ML

Artificial Intelligence

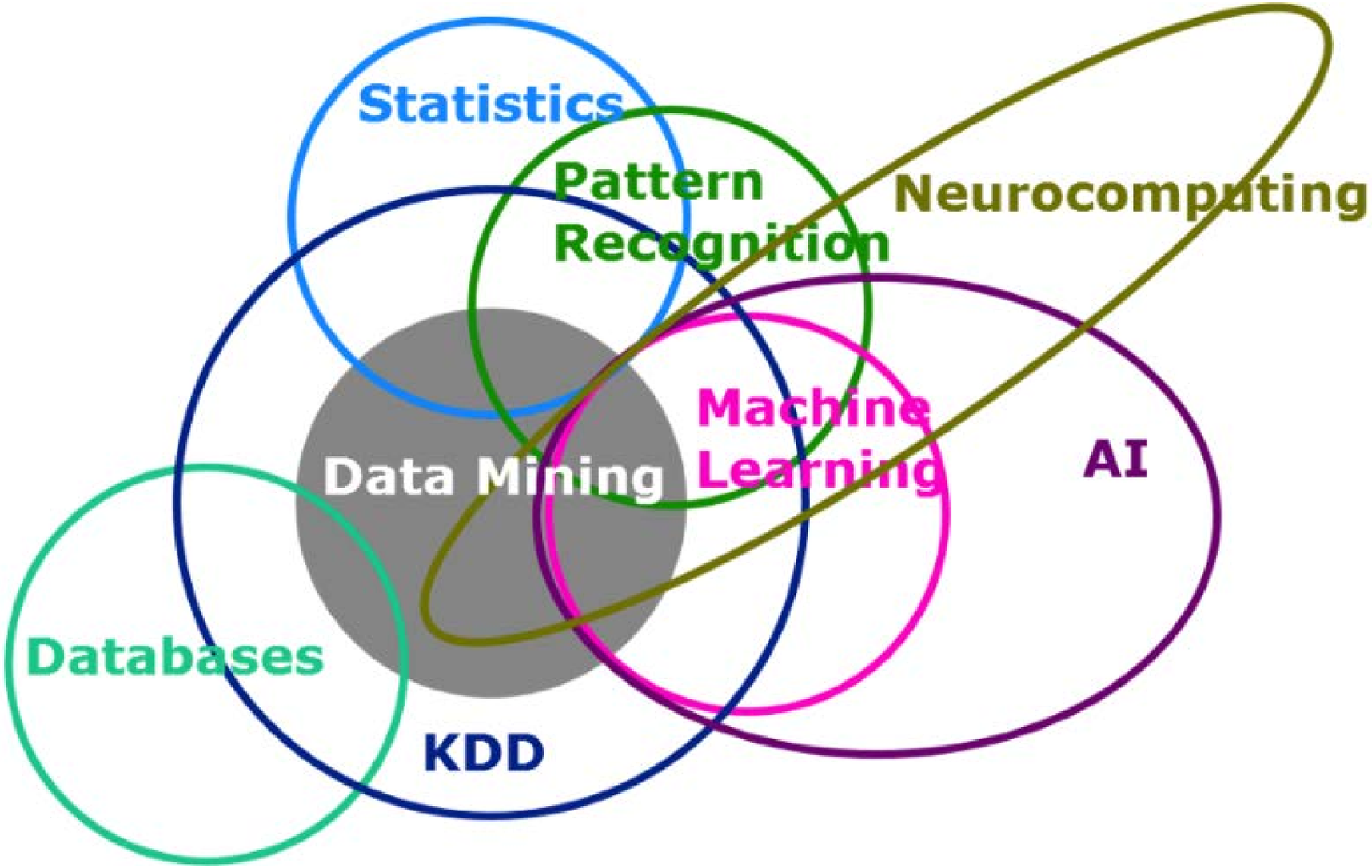
- **지능이란?**
  → 보다 추상적인 정보를 이해하는 능력

- **인공 지능이란?**
  → 이러한 지능 현상을 인공적으로 구현하려는 연구

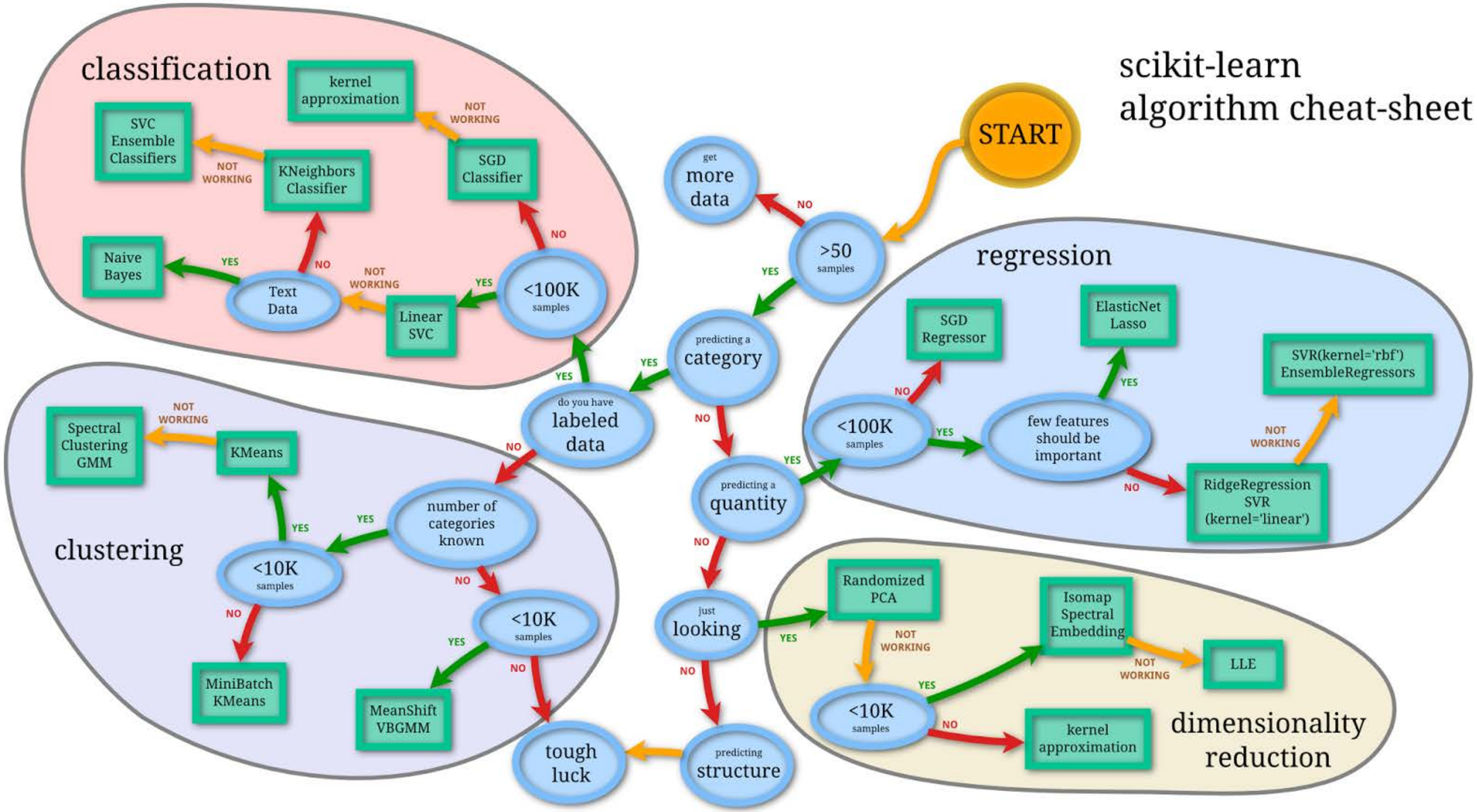
What is AI? ML?


Artificial Intelligence
Deduction, Reasoning, Problem Solving
Knowledge Representation
Planning
Perception: Computer Vision
Machine Learning
Robotics: Motion and Manipulation
Natural Language Processing
Social Intelligence
Supervised Learning
Unsupervised Learning
Reinforcement Learning
Decision Tree Learning
Association Rule Learning
Clustering
Similarity and Metric Learning
Bayesian Networks
Neural Networks
Inductive Logic Programming
Support Vector Machines
Sparse Dictionary Learning
Genetic Algorithms
Deep Learning
Manifold Learning

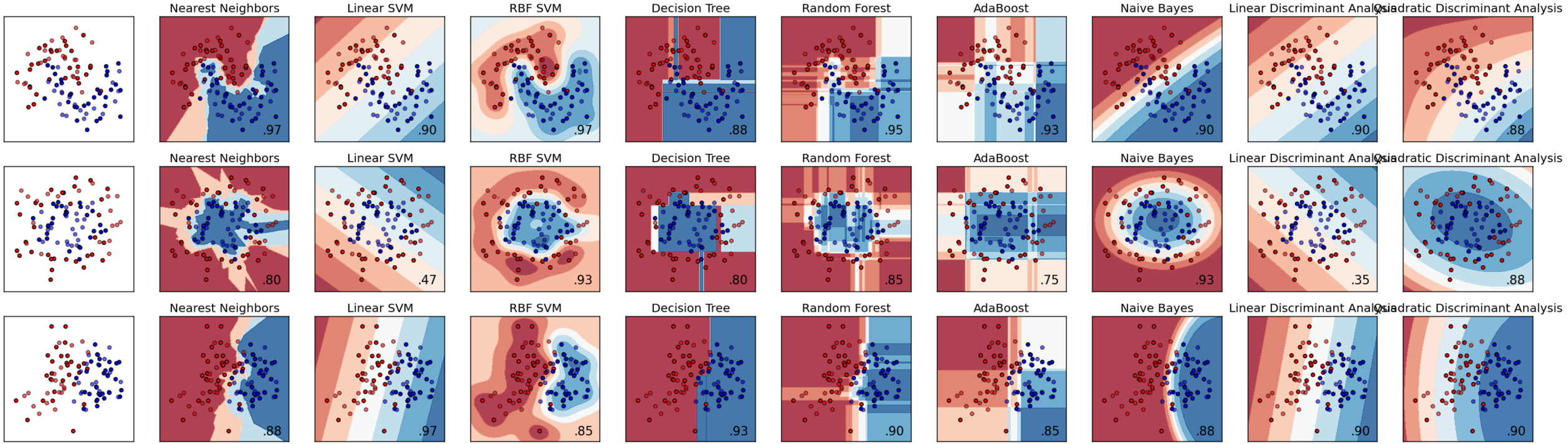
Various Field in ML



Various Task in ML

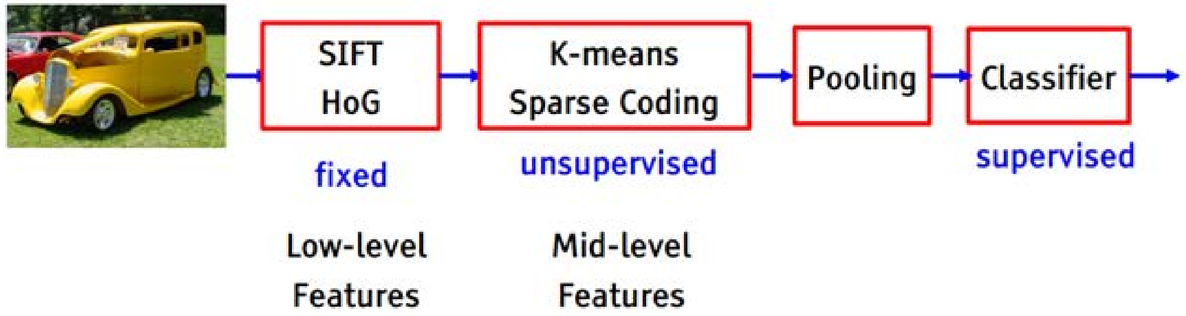


Various Algorithm in ML

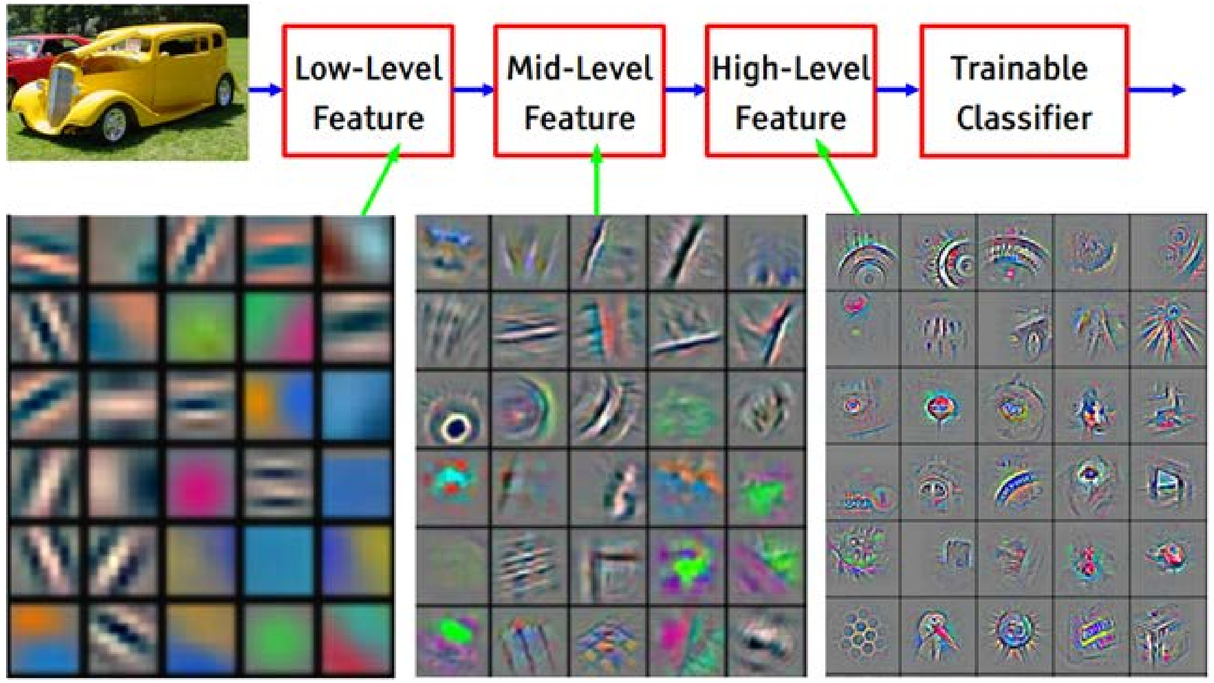


Conventional AI vs ML

Object recognition 2006-2012



State of the art object recognition using CNNs



Machine Learning

- Supervised Learning :

  $y = f(x)$

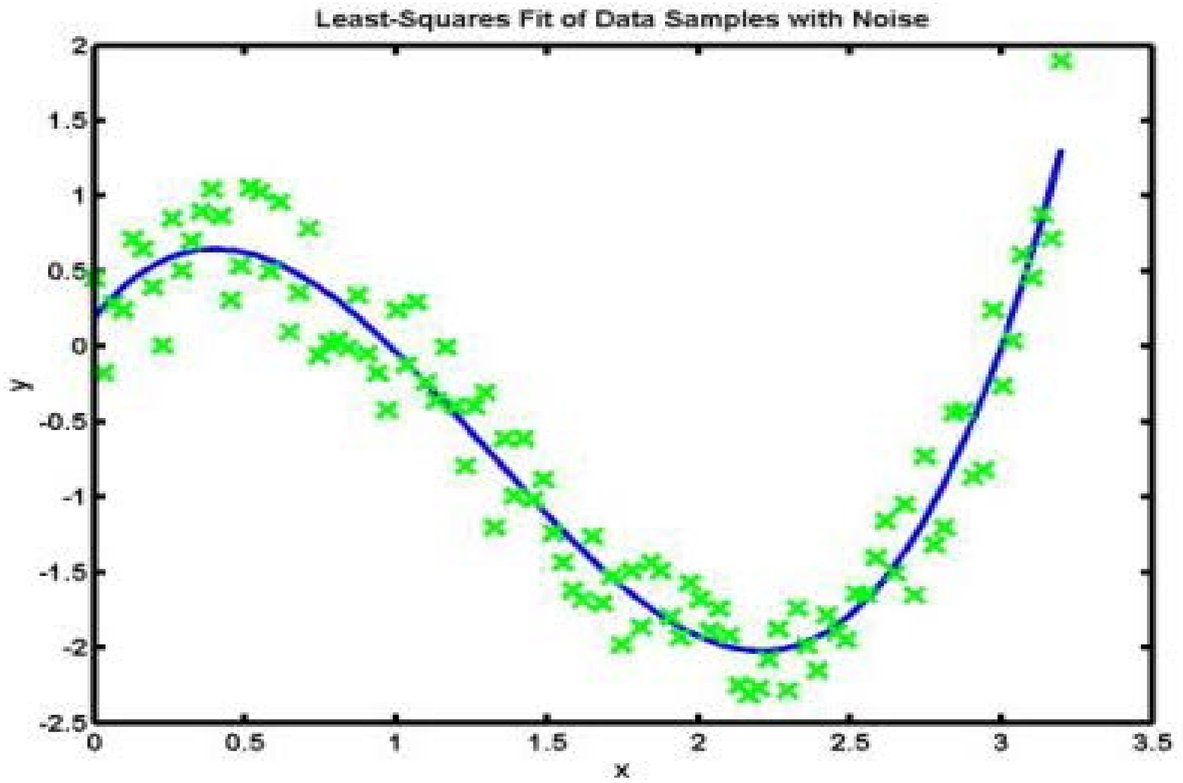
- Unsupervised Learning :

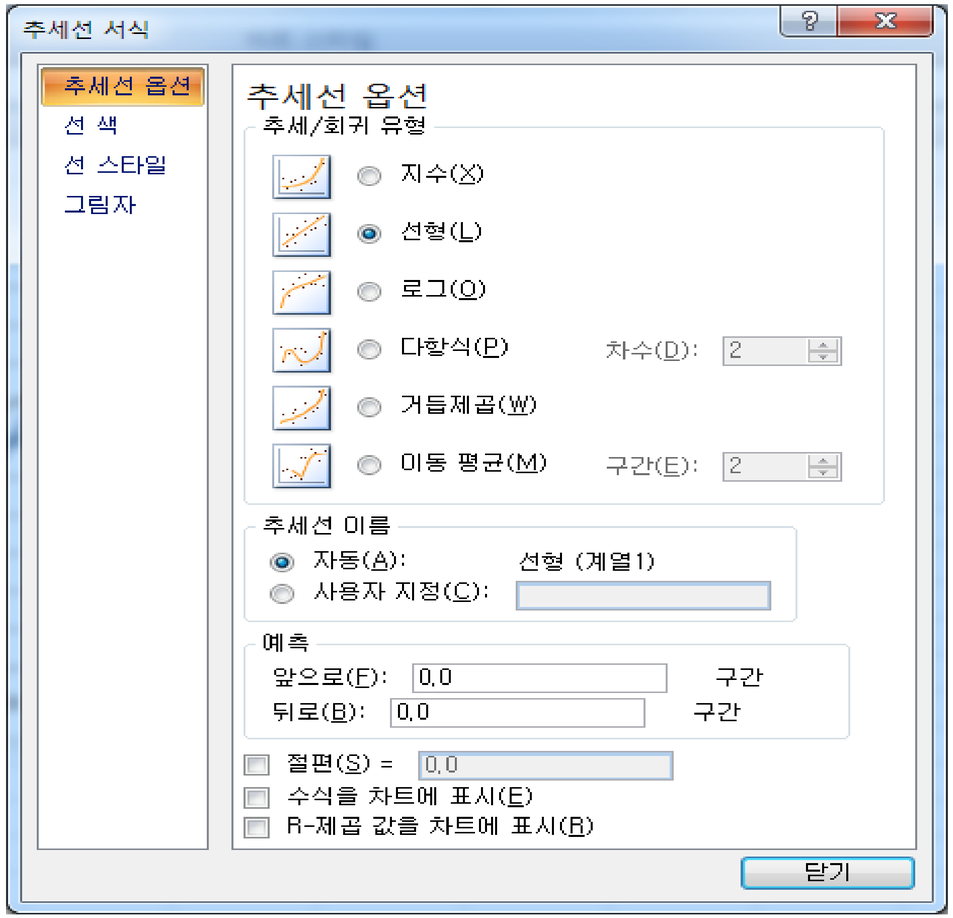
  $x \sim p(x)$ or $x = f(x)$

- Reinforcement Learning :

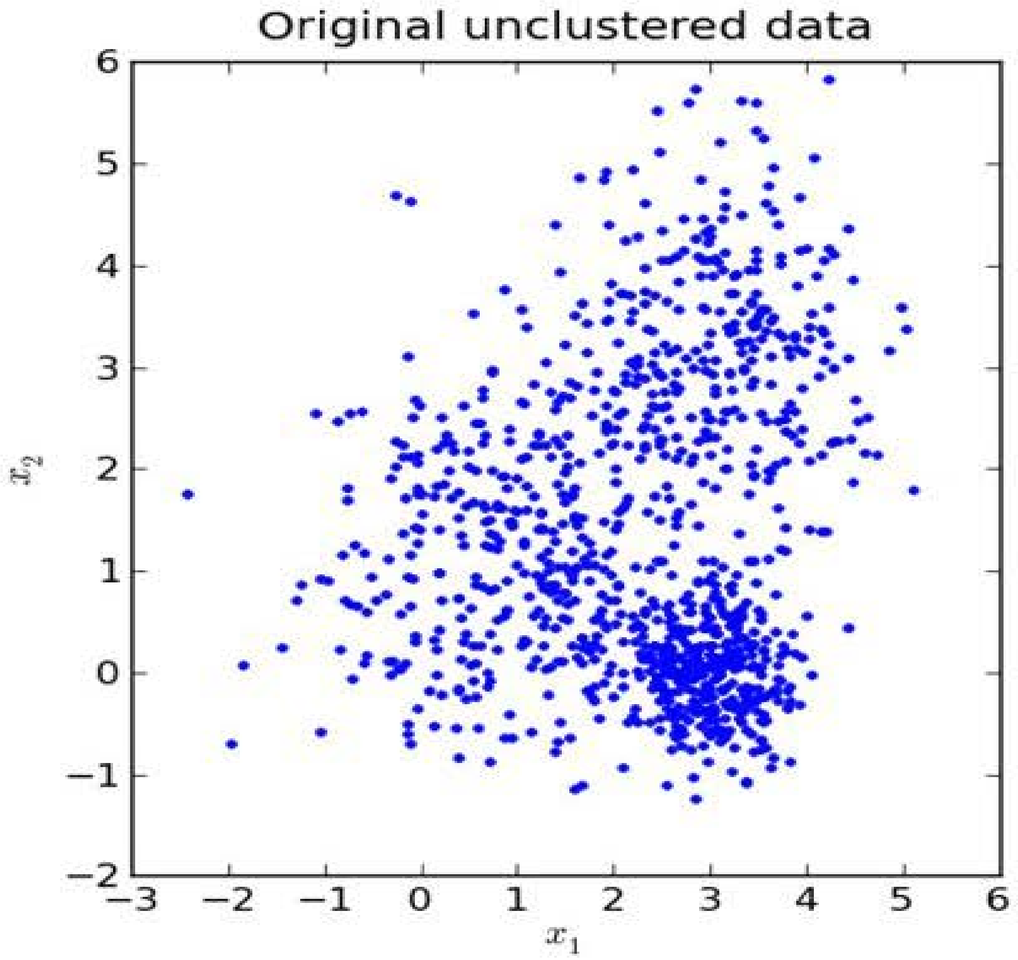
  Find a policy, $p(a|s)$ which maximizes the sum of reward

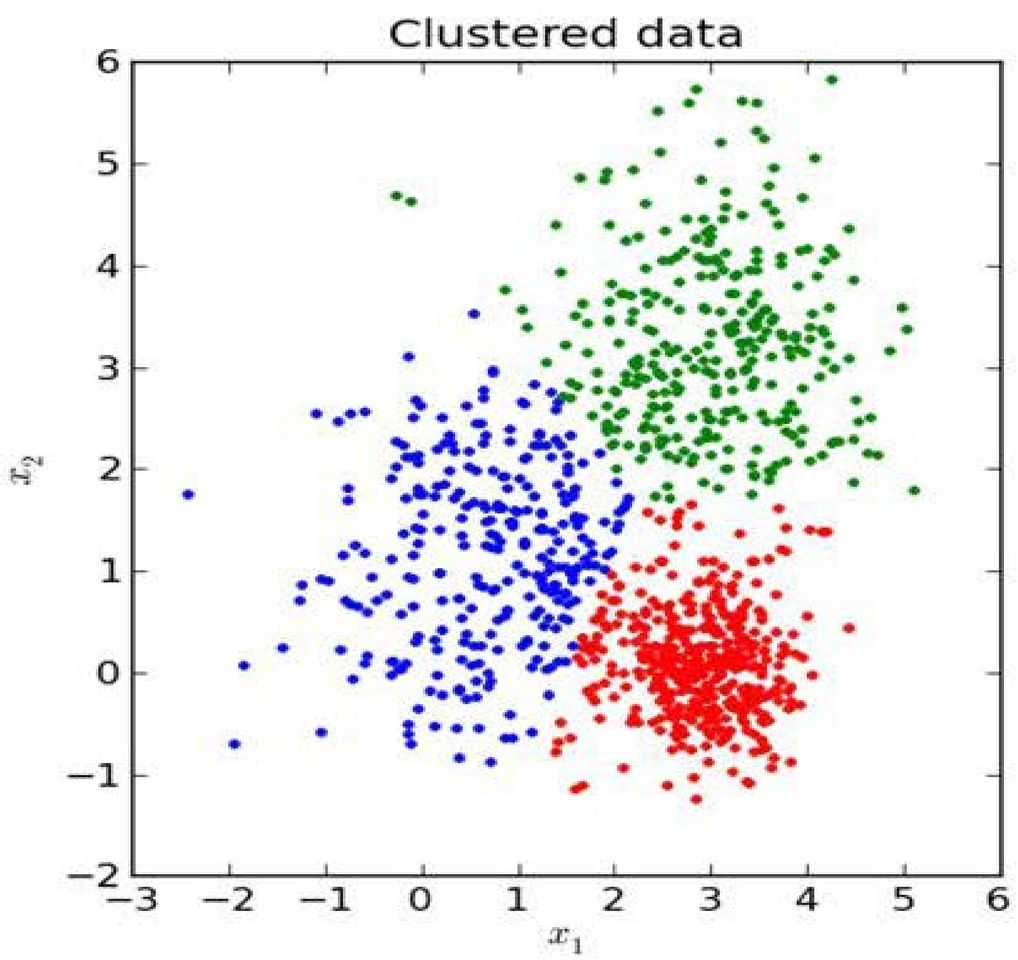
Example of Supervised Learning :
Polynomial Curve Fitting



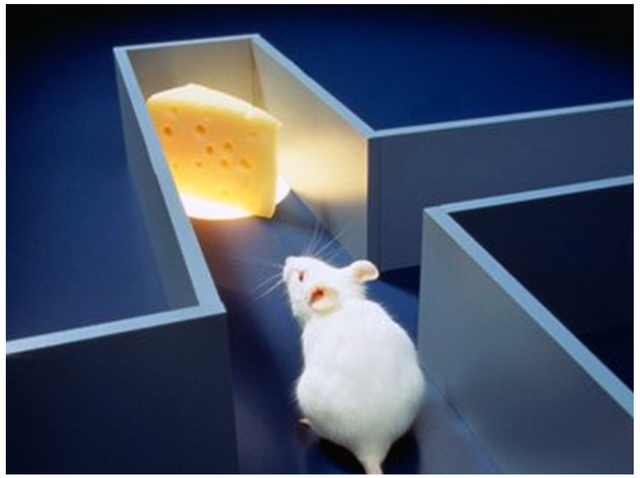


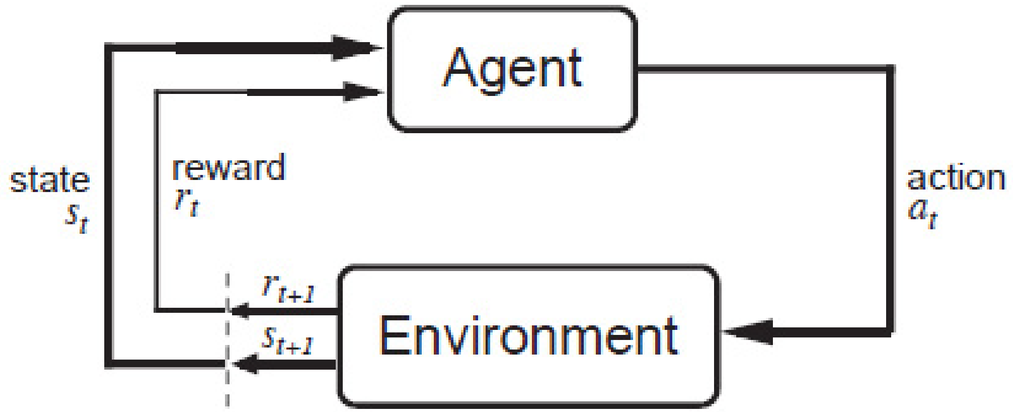
Example of Unsupervised Learning : **Clustering**



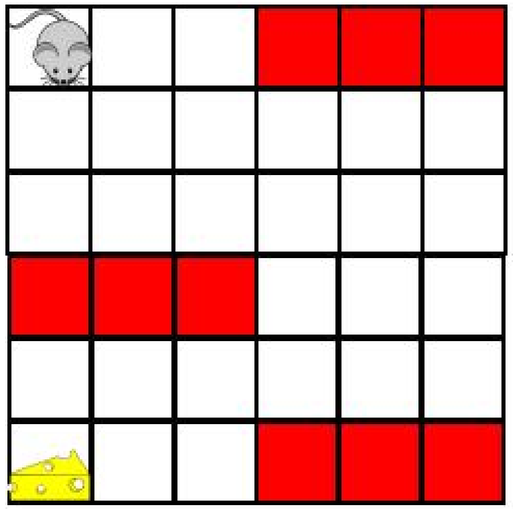


Example of Reinforcement Learning :
Optimal Control Problem

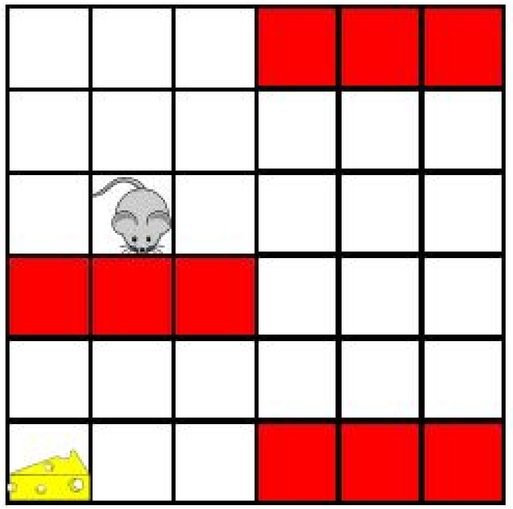




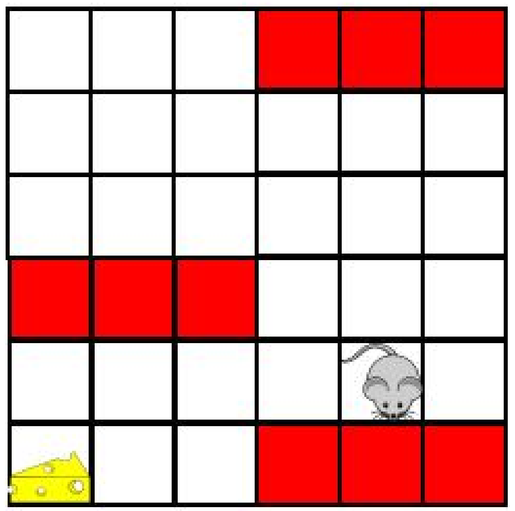
State 1



State 2

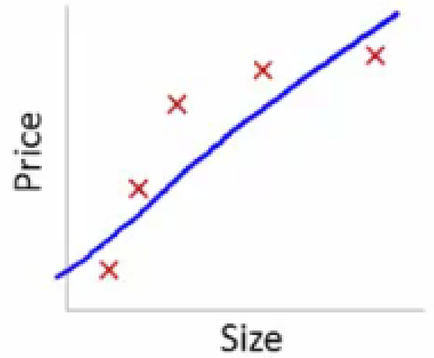


State 3



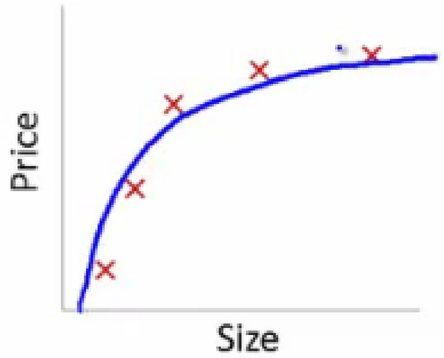
Basic Theory

Overfitting / Underfitting



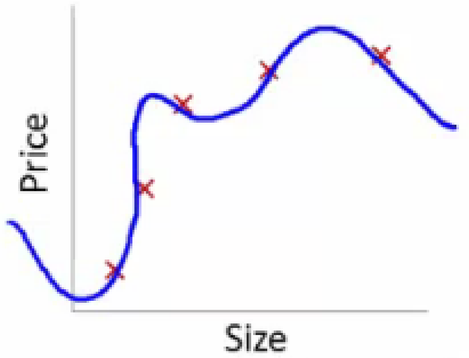
$\theta_0 + \theta_1 x$

High bias
(underfit)



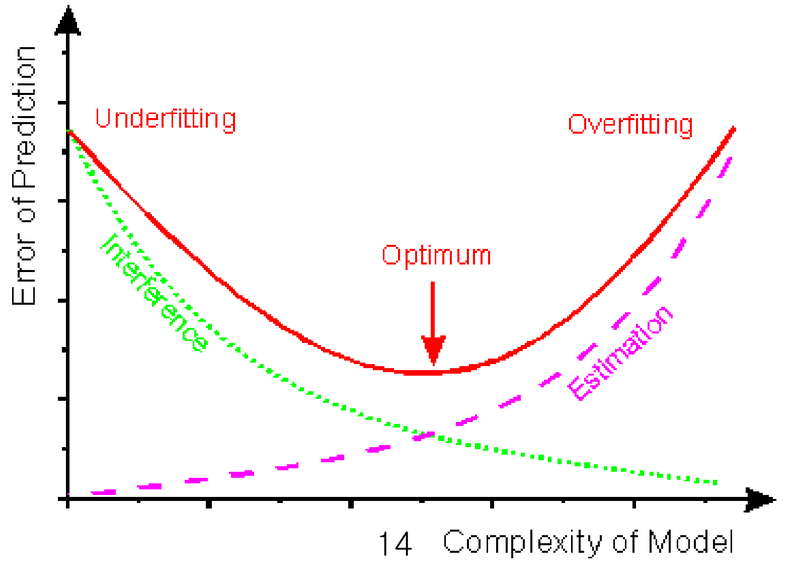
$\theta_0 + \theta_1 x + \theta_2 x^2$

"Just right"



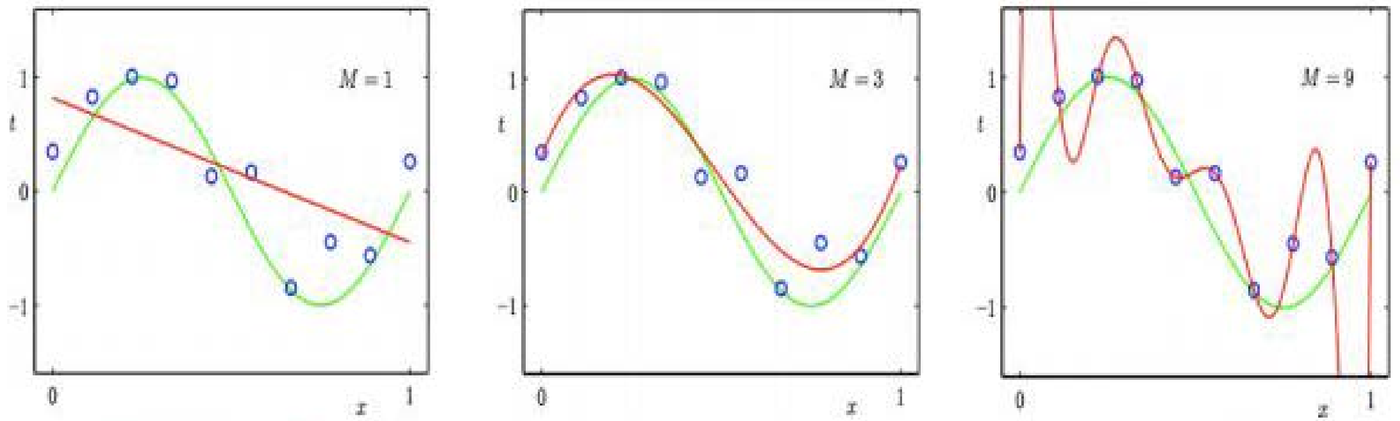
$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

High variance
(overfit)



Overfitting / Underfitting

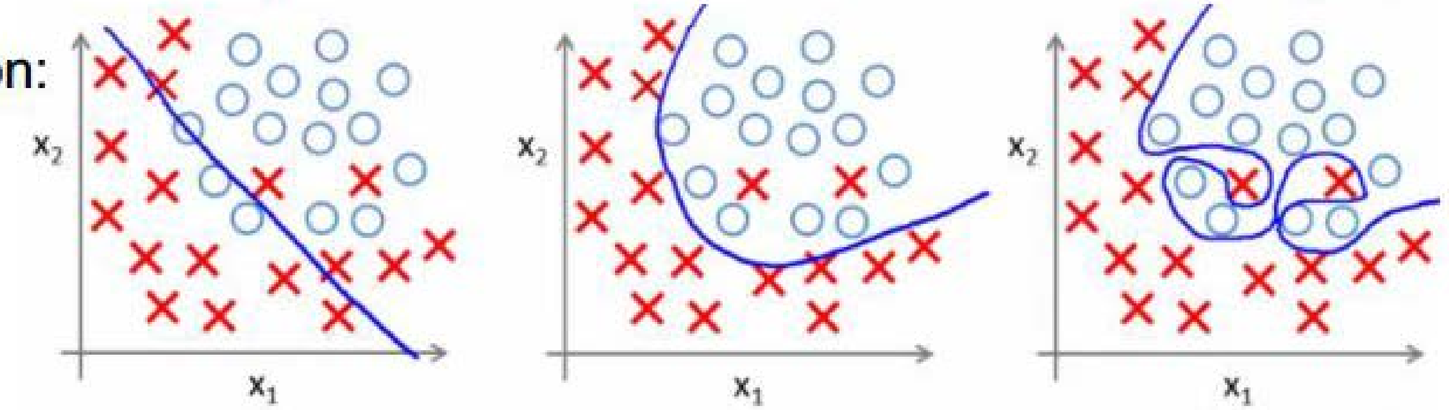
Regression:



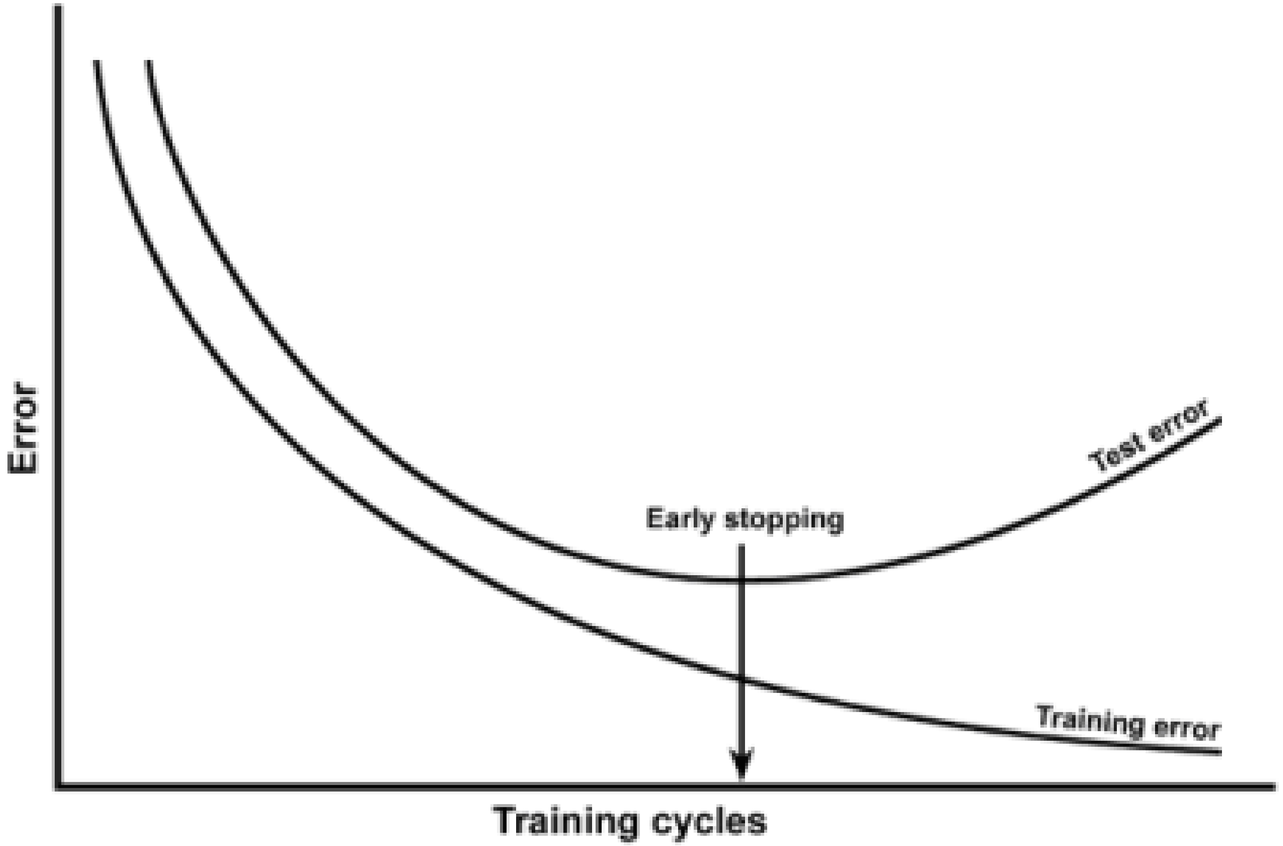
predictor too inflexible:
cannot capture pattern

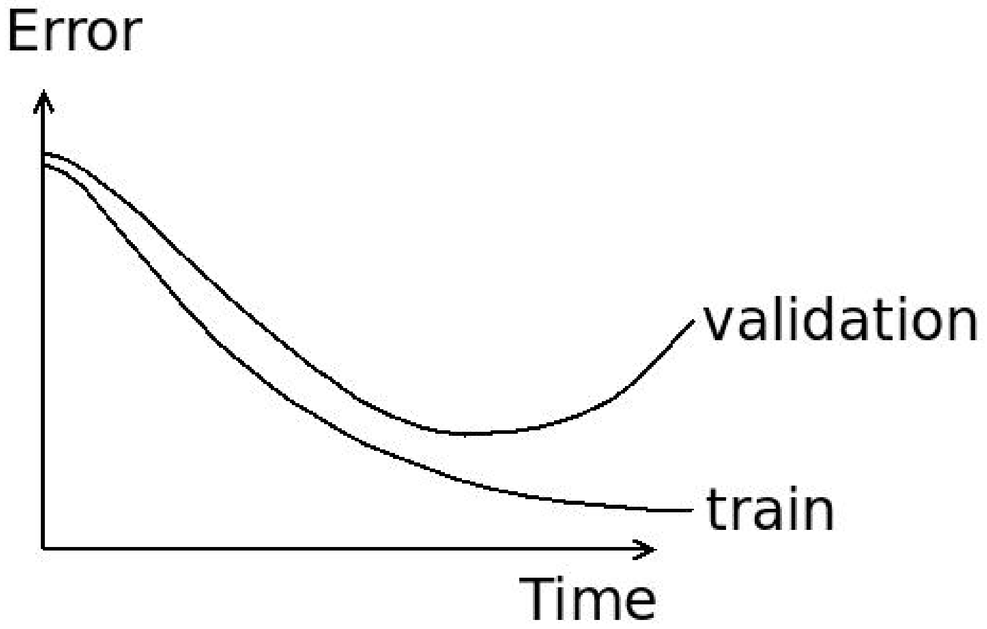
predictor too flexible:
fits noise in the data

Classification:

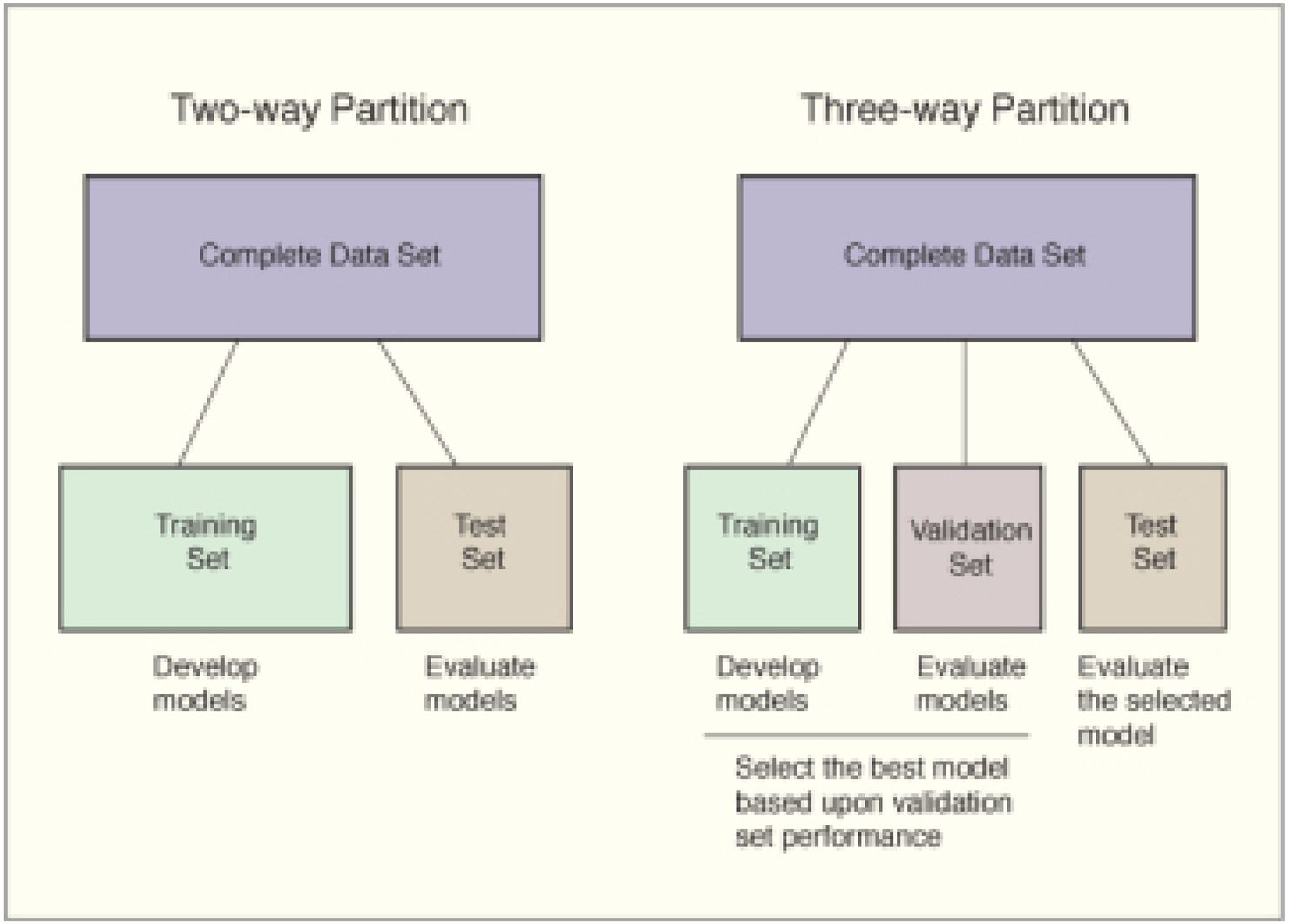


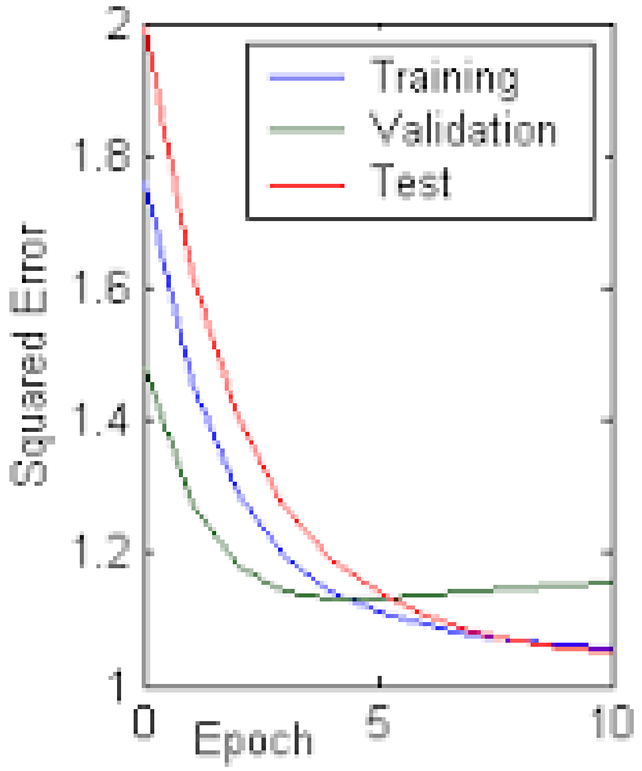
Training / Test / Validation



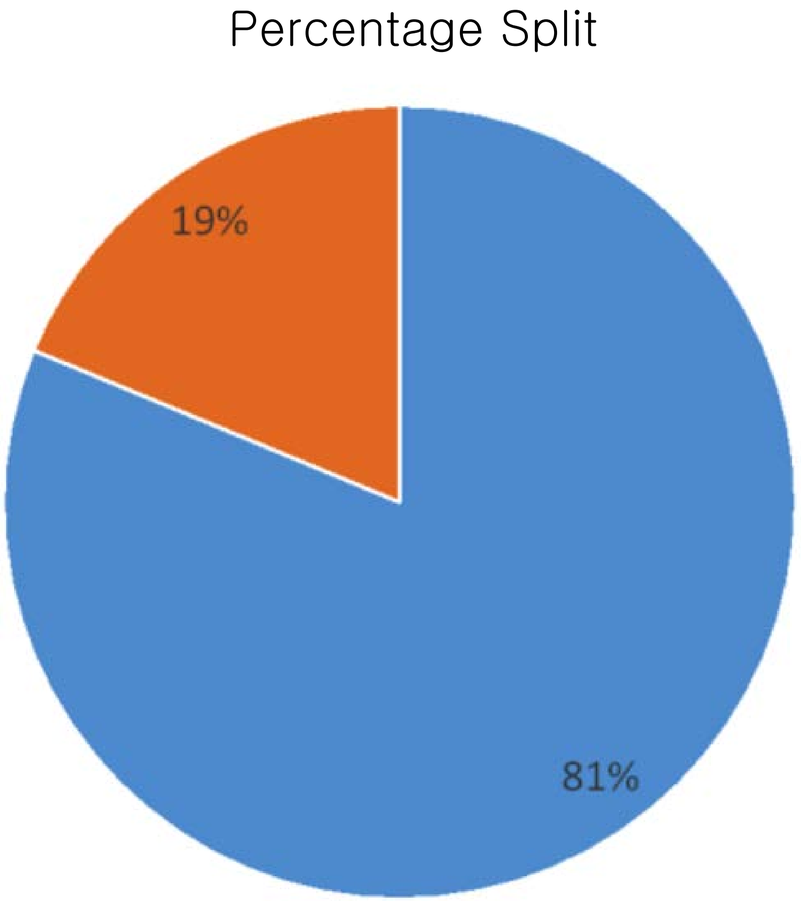


Machine Learning





Validation Dataset

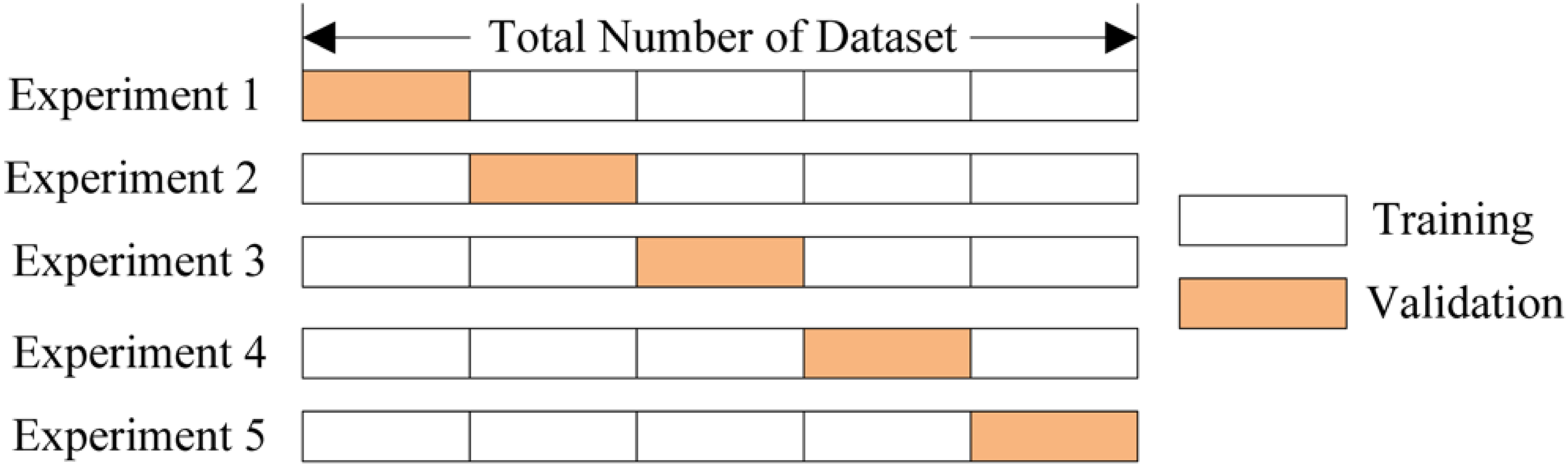


- Problems

1) 모든 데이터를 학습에 사용하지 못함.

2) 해당 split에만 운좋게 잘할 수 있음.(Overfitting)

Cross-validation



Linear Regression

- **Linear combination** of those vectors with those scalars as coefficients

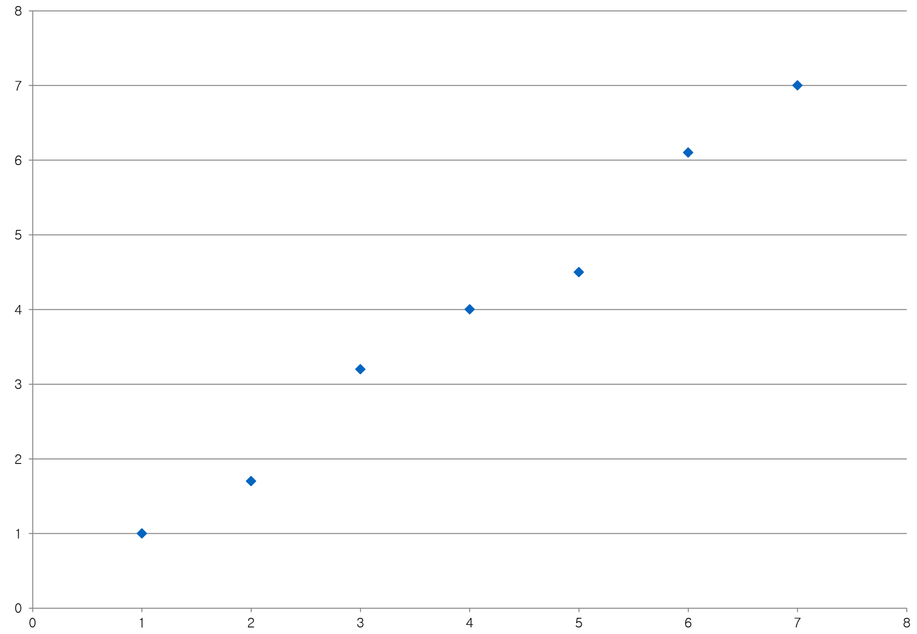
$$a_1\vec{v}_1 + a_2\vec{v}_2 + a_3\vec{v}_3 + \cdots + a_n\vec{v}_n$$

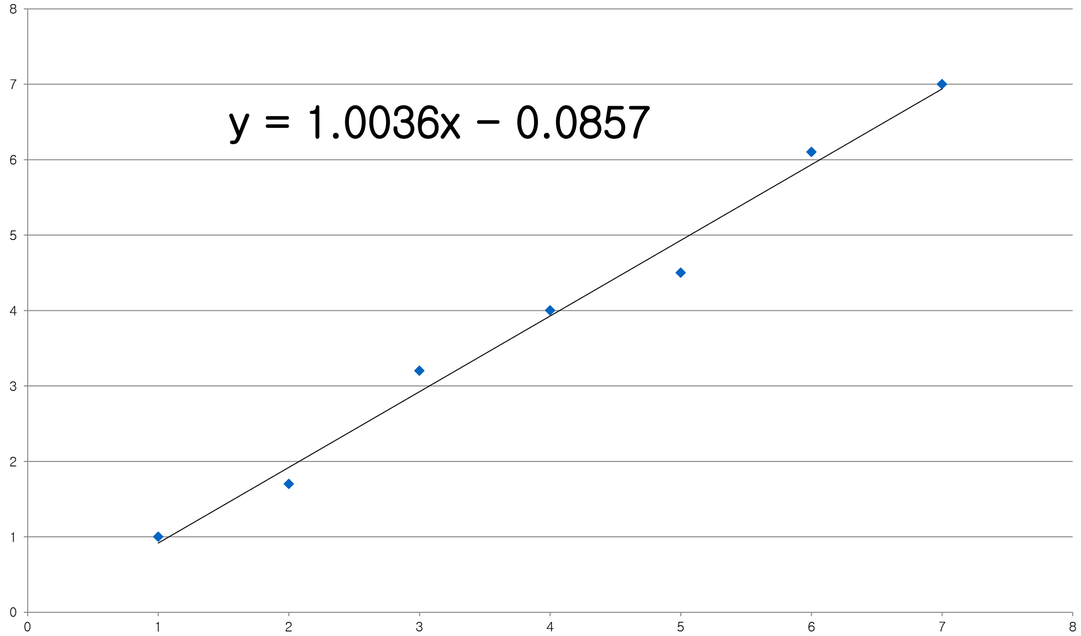
- **Linear Regression :** x에 w scalars를 계수로 곱해서 y를 regression 한 것

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \ldots + w_D x_D$$

Linear Regression

X	Y
1	1
2	1.7
3	3.2
4	4
5	4.5
6	6.1
7	7

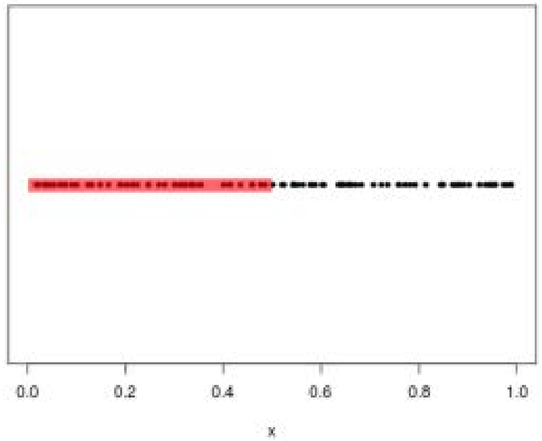




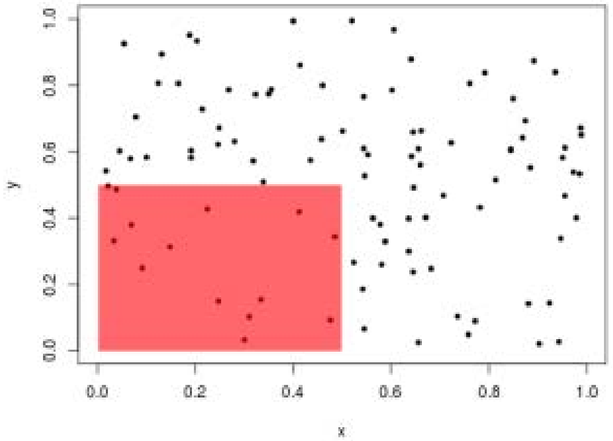
$$y = ax + b$$

Curse of Dimensionality

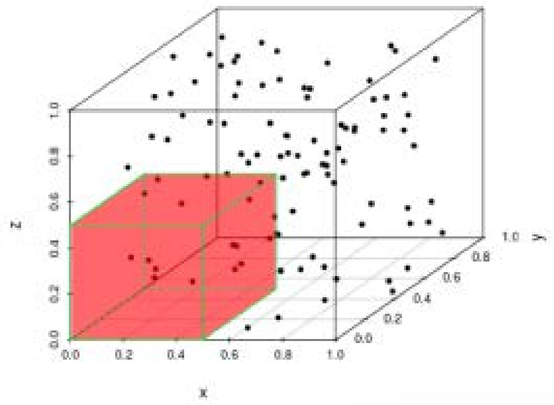
1-D: 42% of data captured.



2-D: 14% of data captured.

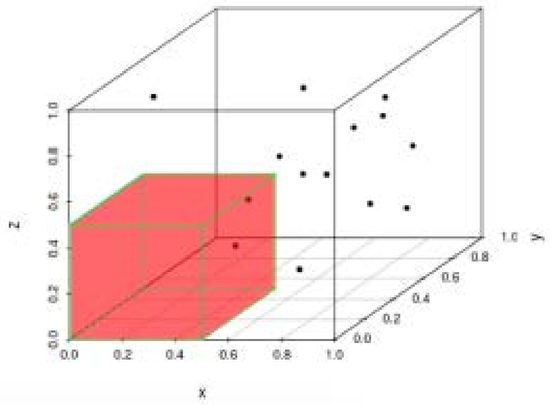


3-D: 7% of data captured.



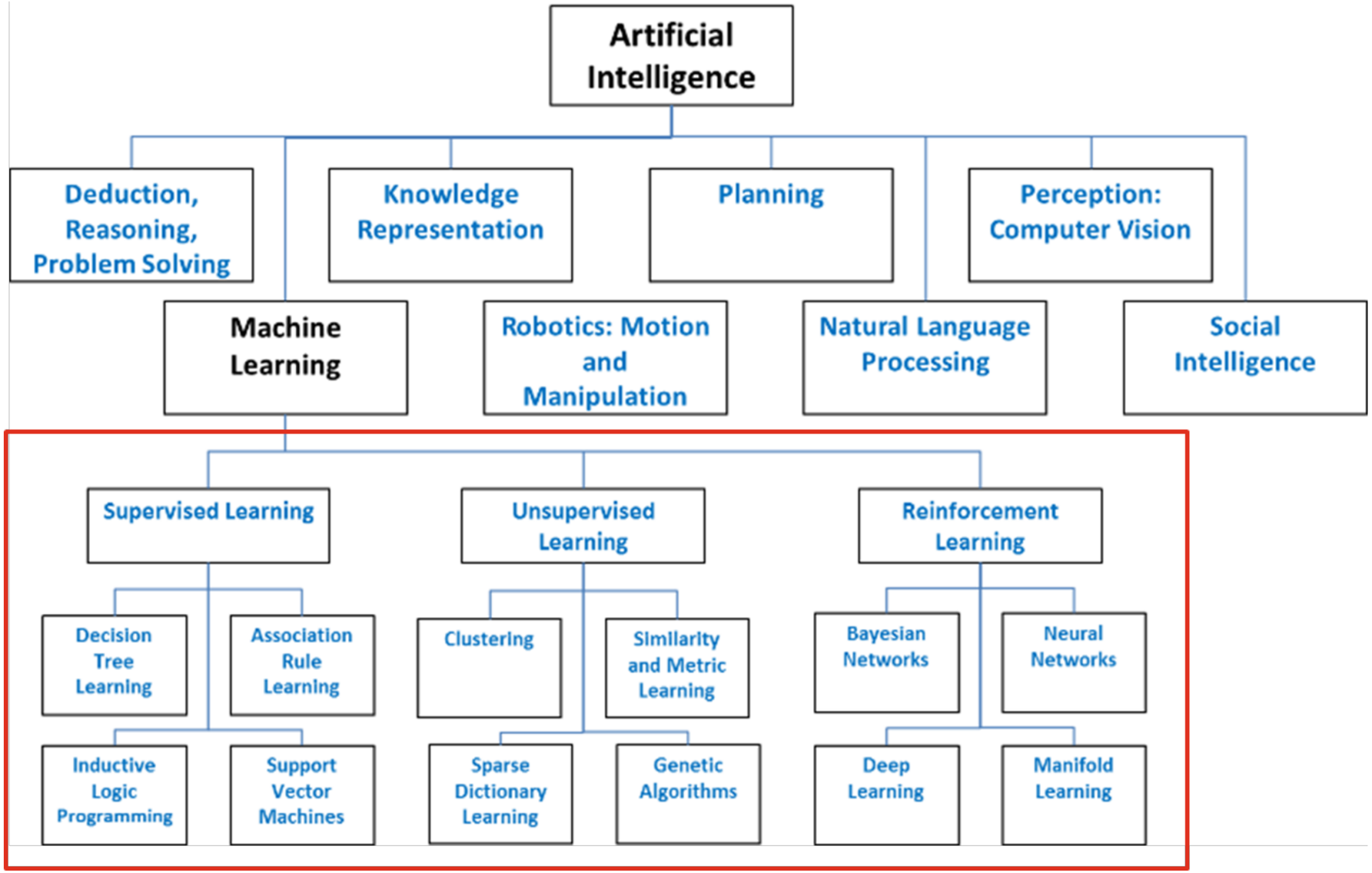
4-D: 3% of data captured.

t = 0

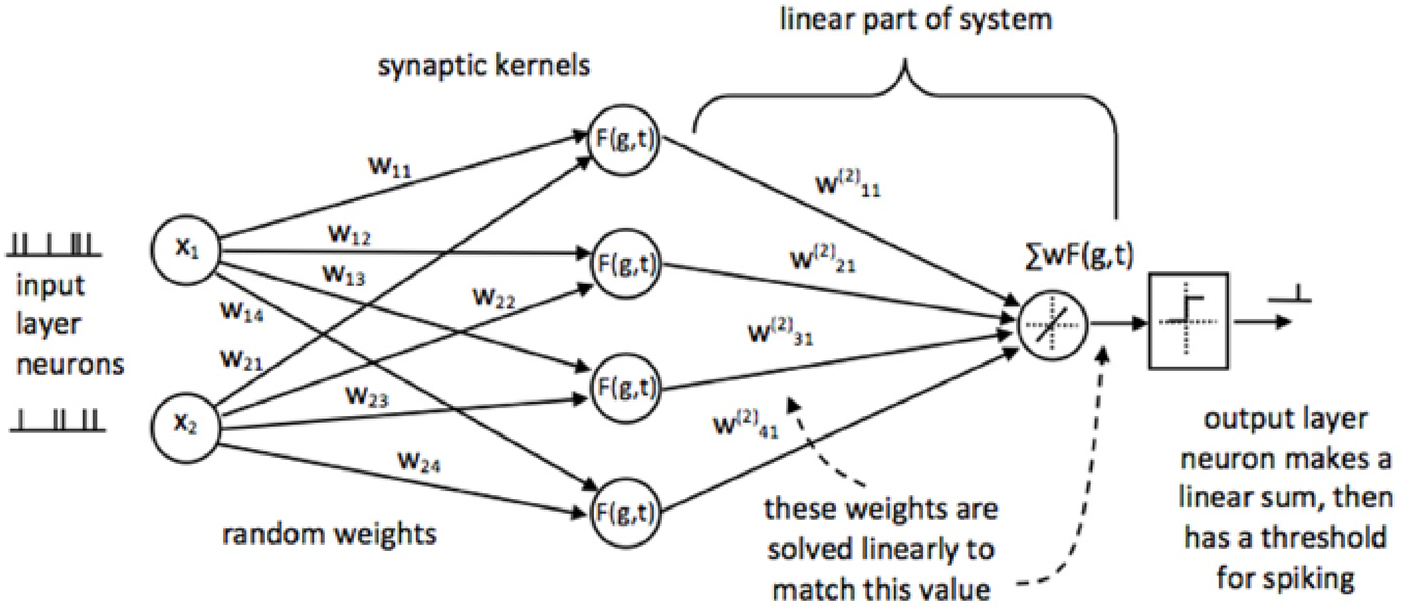


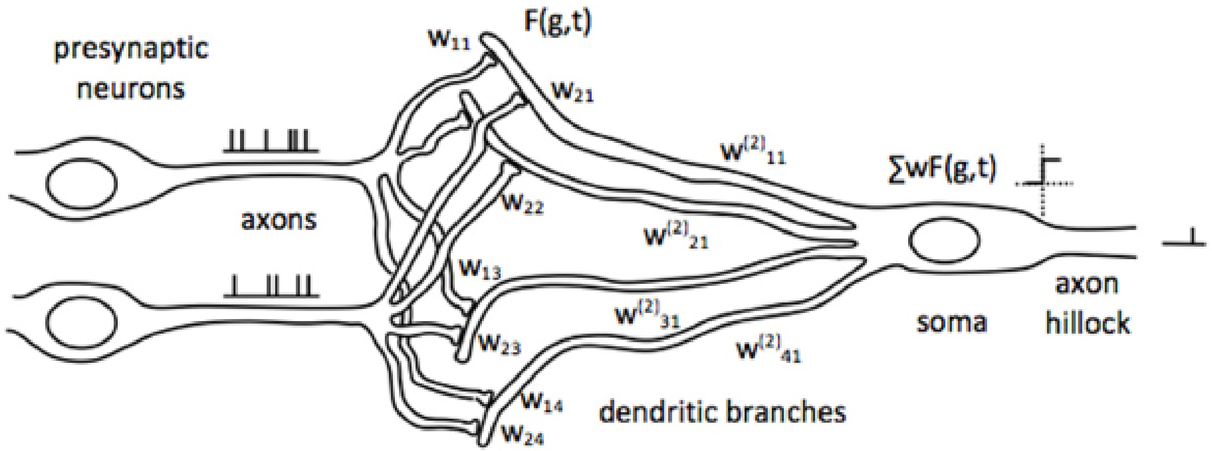
Introduction of Deep Learning

Deep Learning

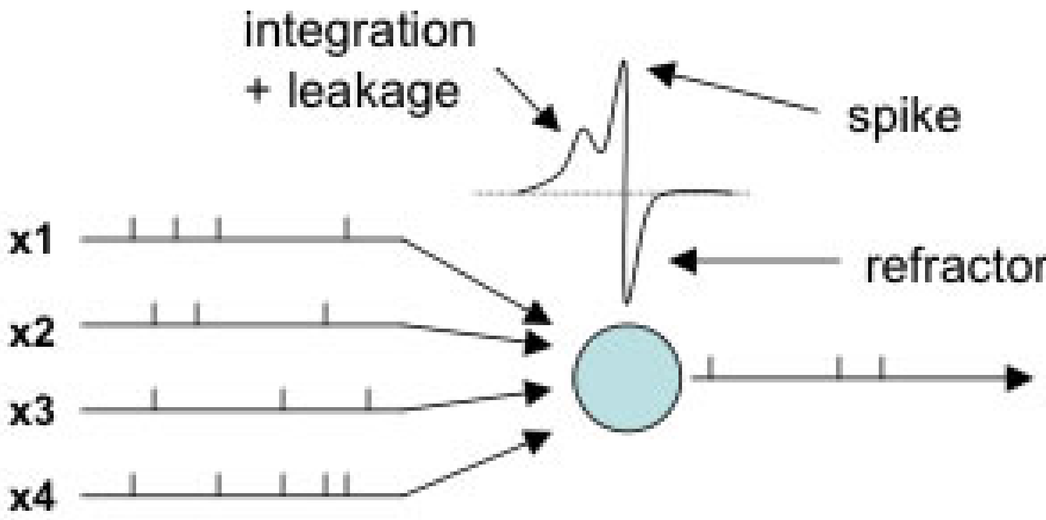


Artificial Neural Networks – Spiking Neuron





Artificial Neural Networks
– Rate Coding Neuron



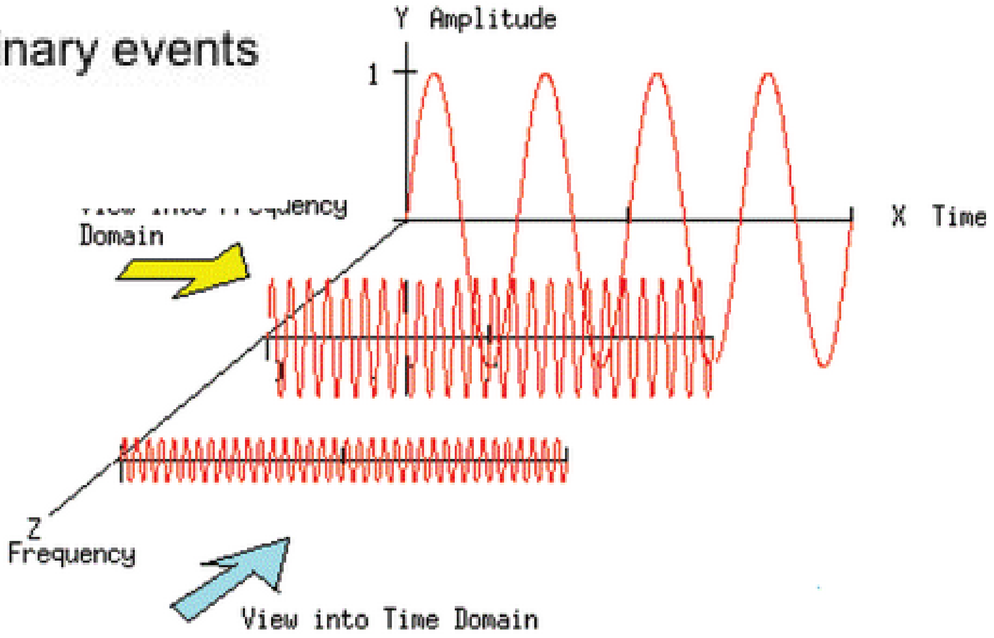
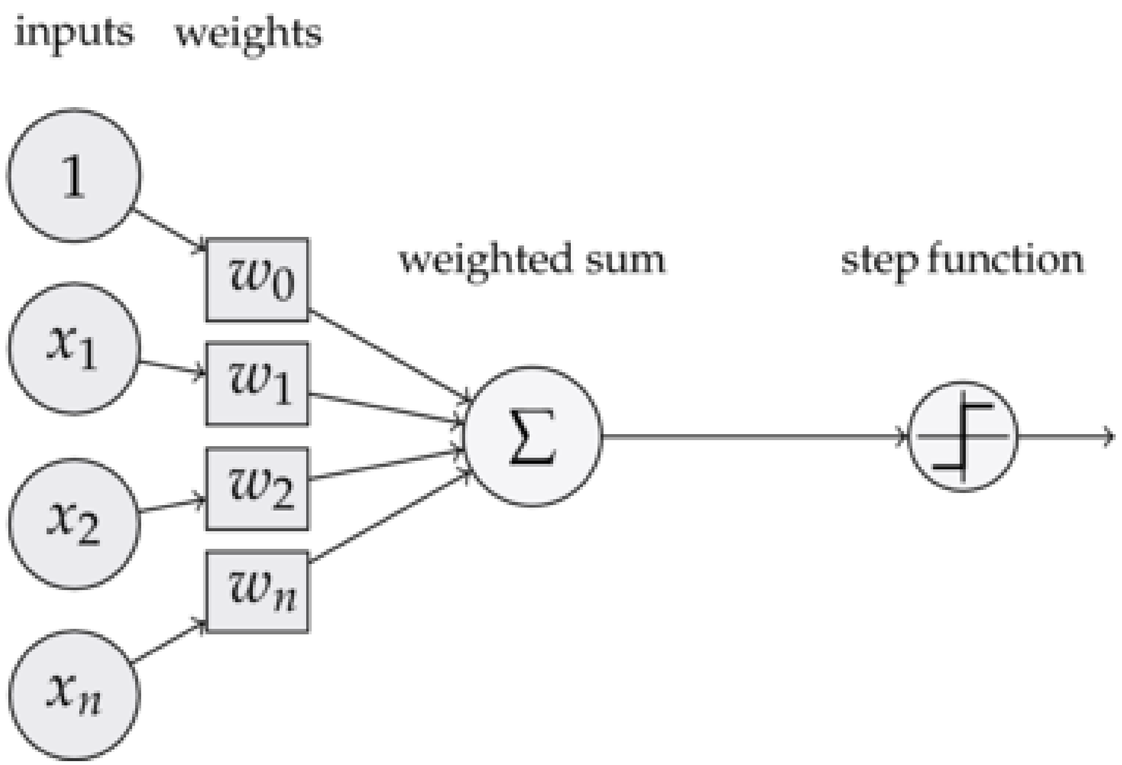


그림 (1) : www.complextoreal.com의 Fourier Analysis Made Easy 인용

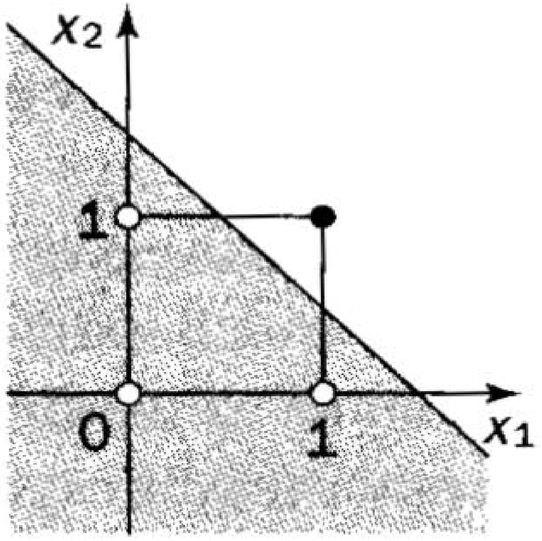
Perceptron

$$s = \sum_{i=0}^{n} w_i \cdot x_i \qquad f(s) = \begin{cases} 1 & \text{if } s \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

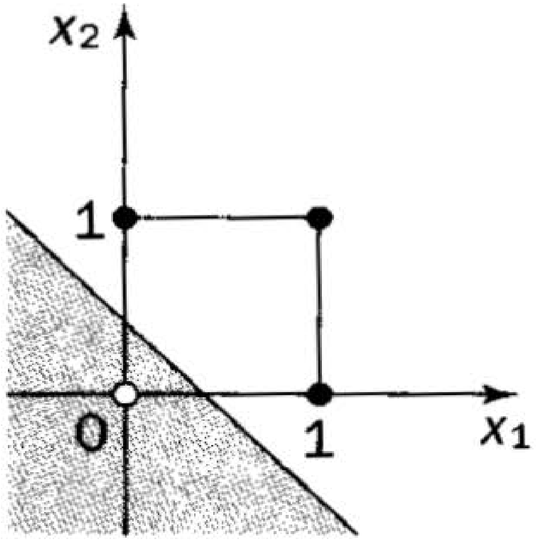


A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

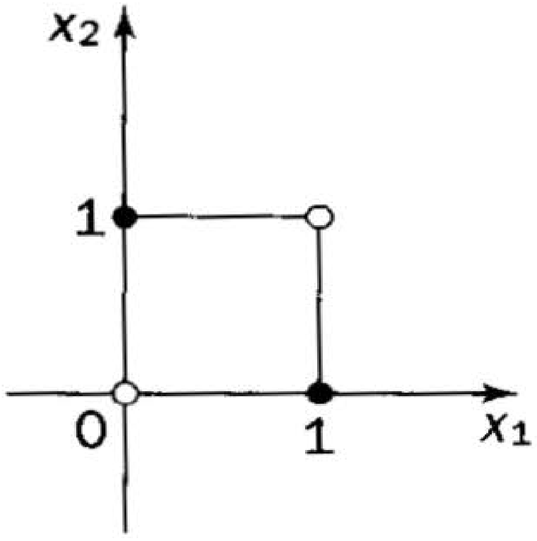
Perceptron – XOR



(a) AND ($x_1 \cap x_2$)

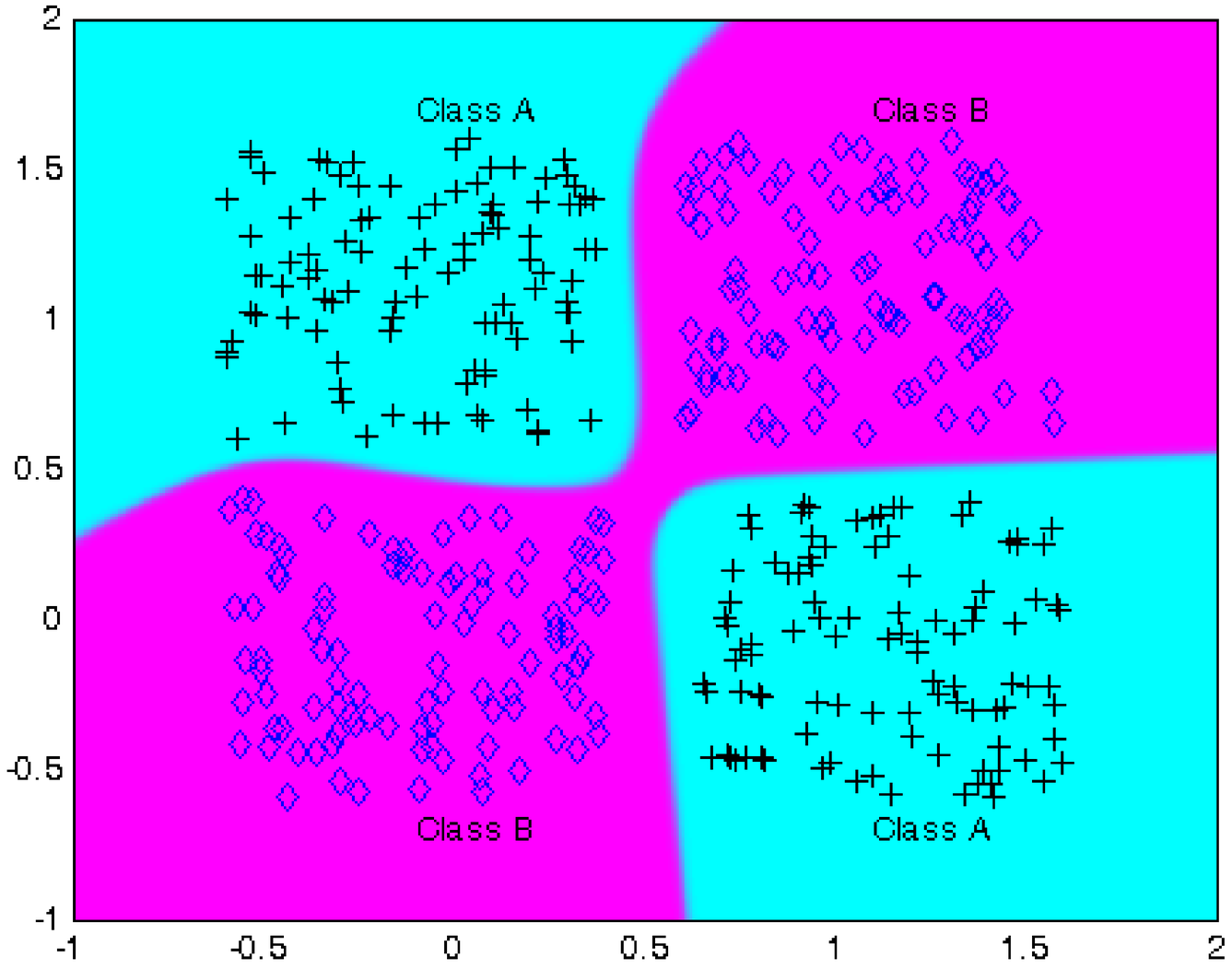


(b) OR ($x_1 \cup x_2$)

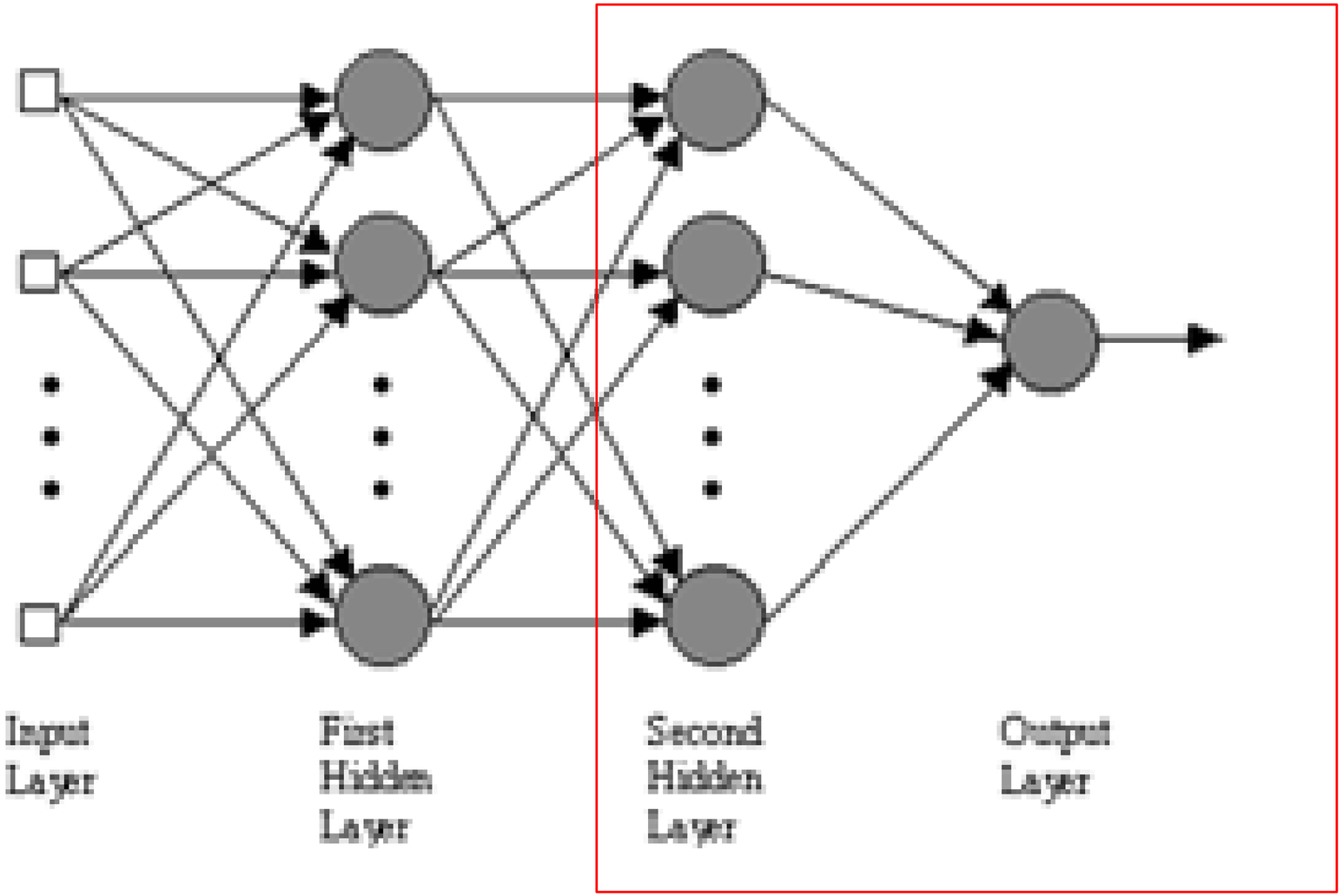


(c) Exclusive-OR
($x_1 \oplus x_2$)

Linearly Non-Separable

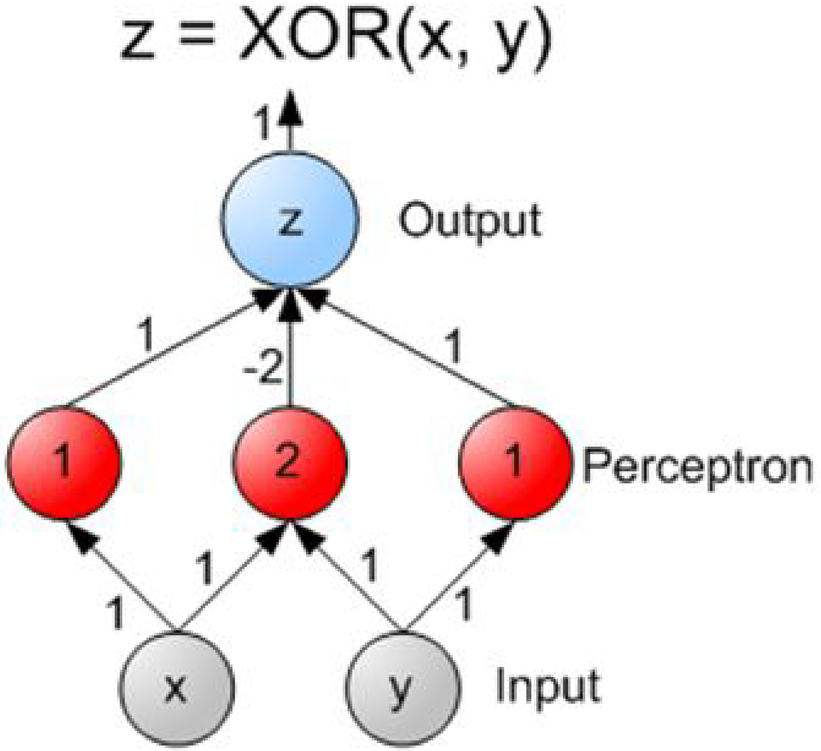


Multi-Layer Perceptron



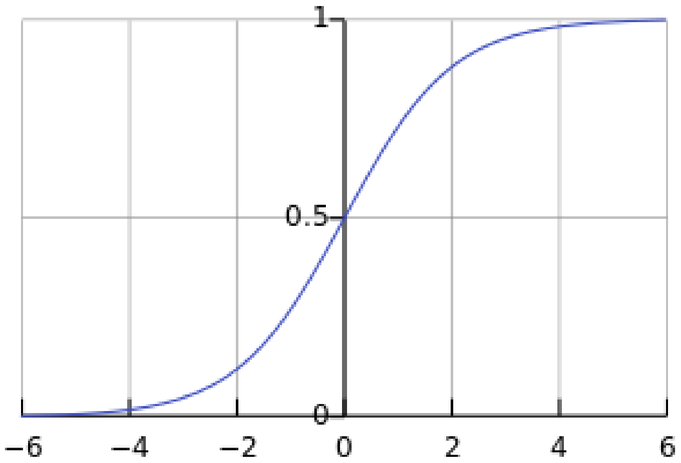
Universal Function Approximation

The *universal approximation theorem* for neural networks states that every continuous function that maps intervals of real numbers to some output interval of real numbers can be approximated arbitrarily closely by a multi-layer perceptron with just one hidden layer. This result holds only for restricted classes of activation functions, e.g. for the sigmoidal functions. (Wikipedia.org)



Sigmoid Function

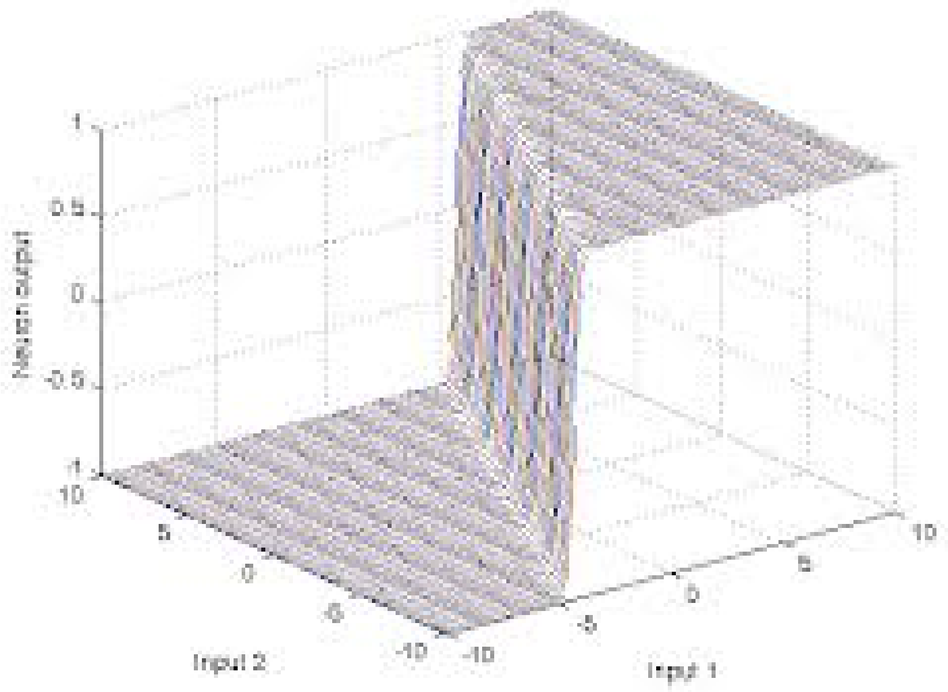
$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

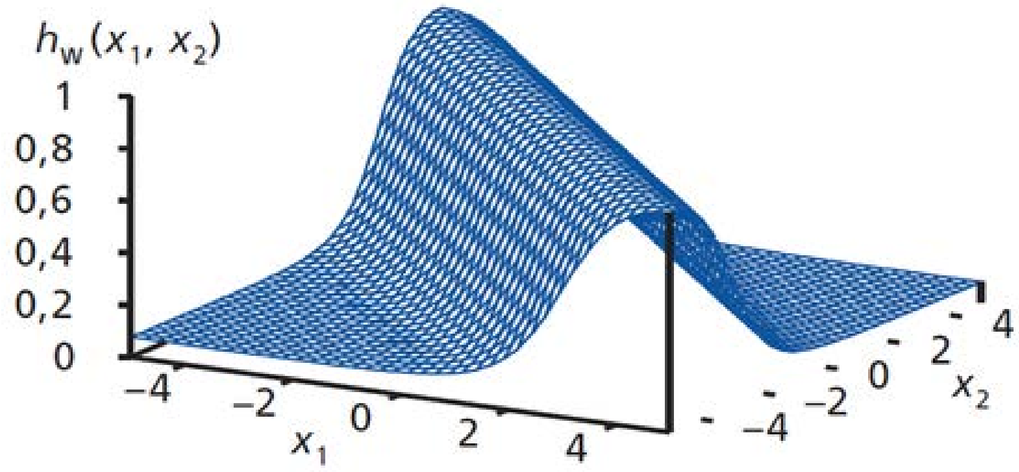


$$\sigma(x)' = \frac{\delta\{1+e^{-x}\}^{-1}}{\delta x} = -(1+e^{-x})^{-2}-e^{-x} = \frac{e^{-x}}{(1+e^{-x})^2}$$

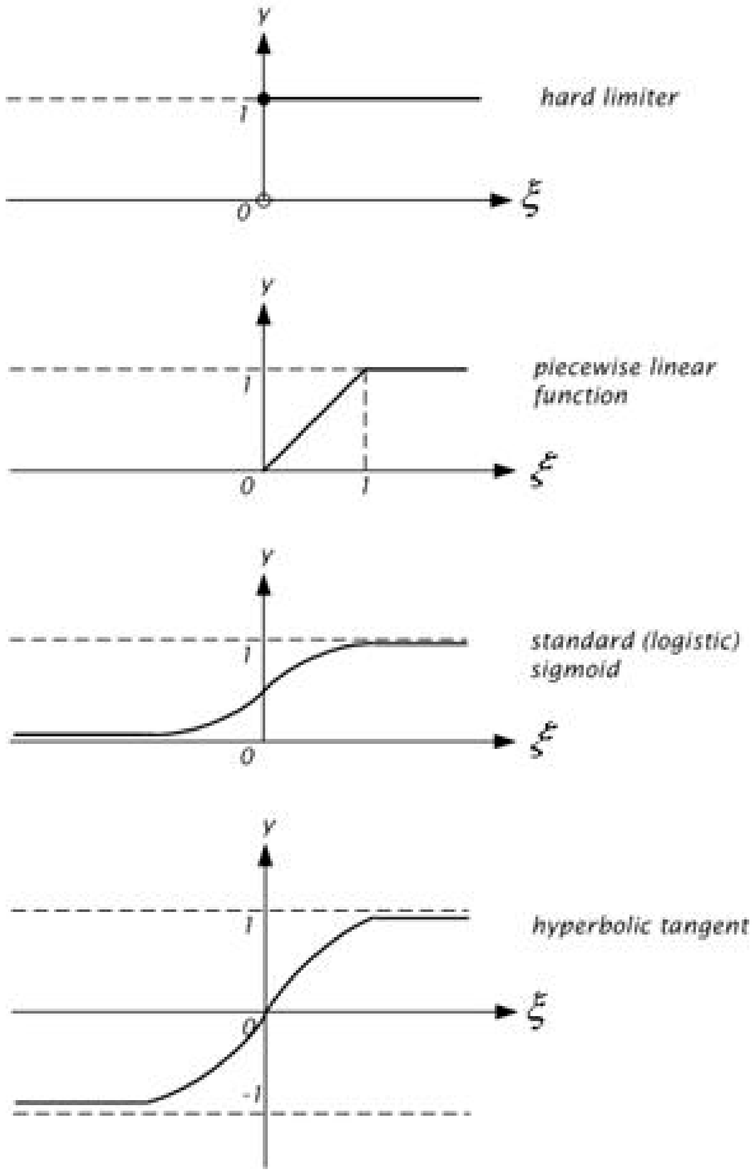
$$\sigma(x)(1-\sigma(x)) = \frac{1}{1+e^{-x}}\left(1-\frac{1}{1+e^{-x}}\right) = \frac{1}{1+e^{-x}}\left(\frac{e^{-x}}{1+e^{-x}}\right) = \frac{e^{-x}}{(1+e^{-x})^2}$$

Sigmoid Function



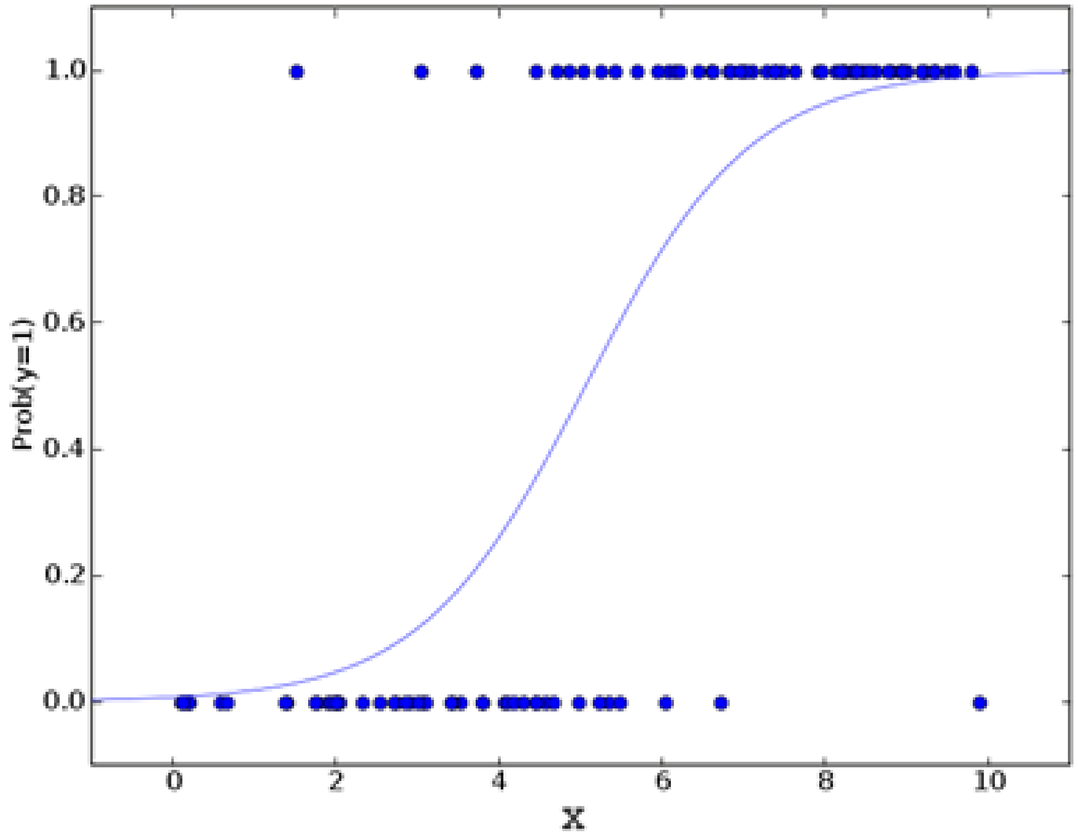


Activation Function



Activation Function

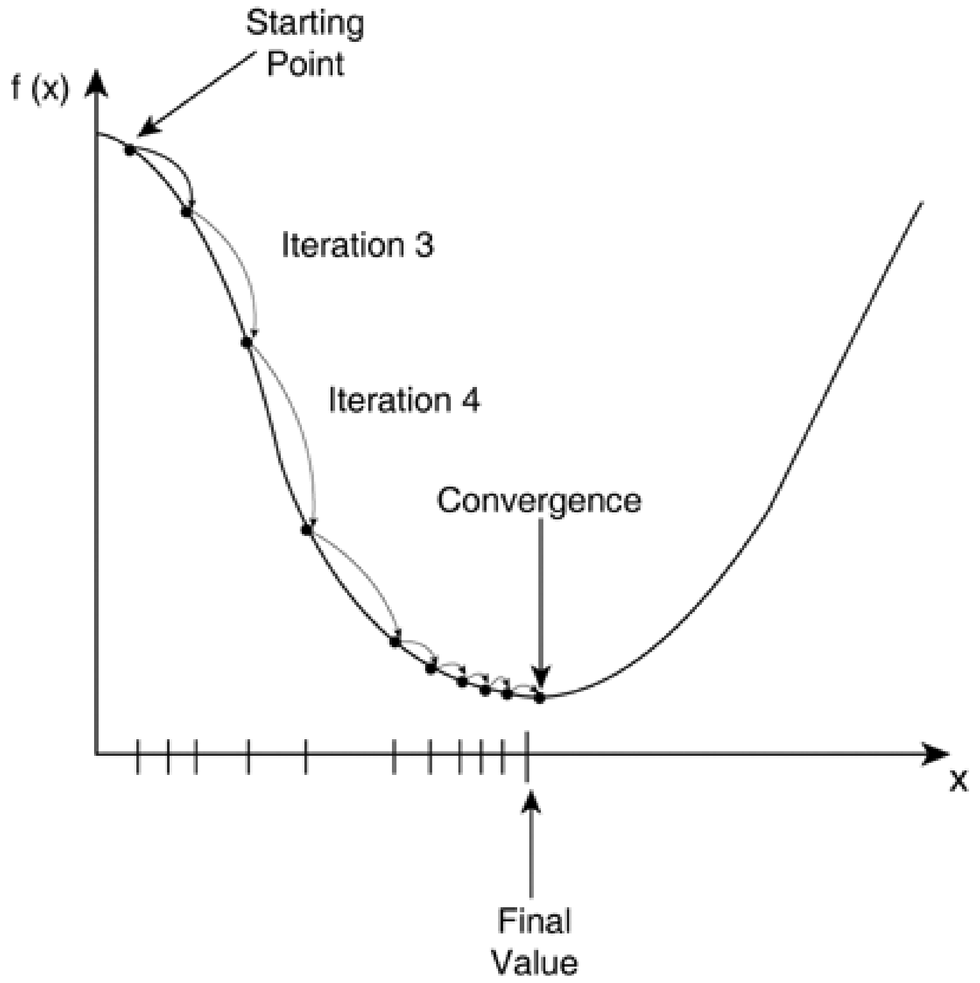
- Hidden Layer에 Activation Function이 없으면?
  → 항상 2층짜리인 Neural Network와 같음



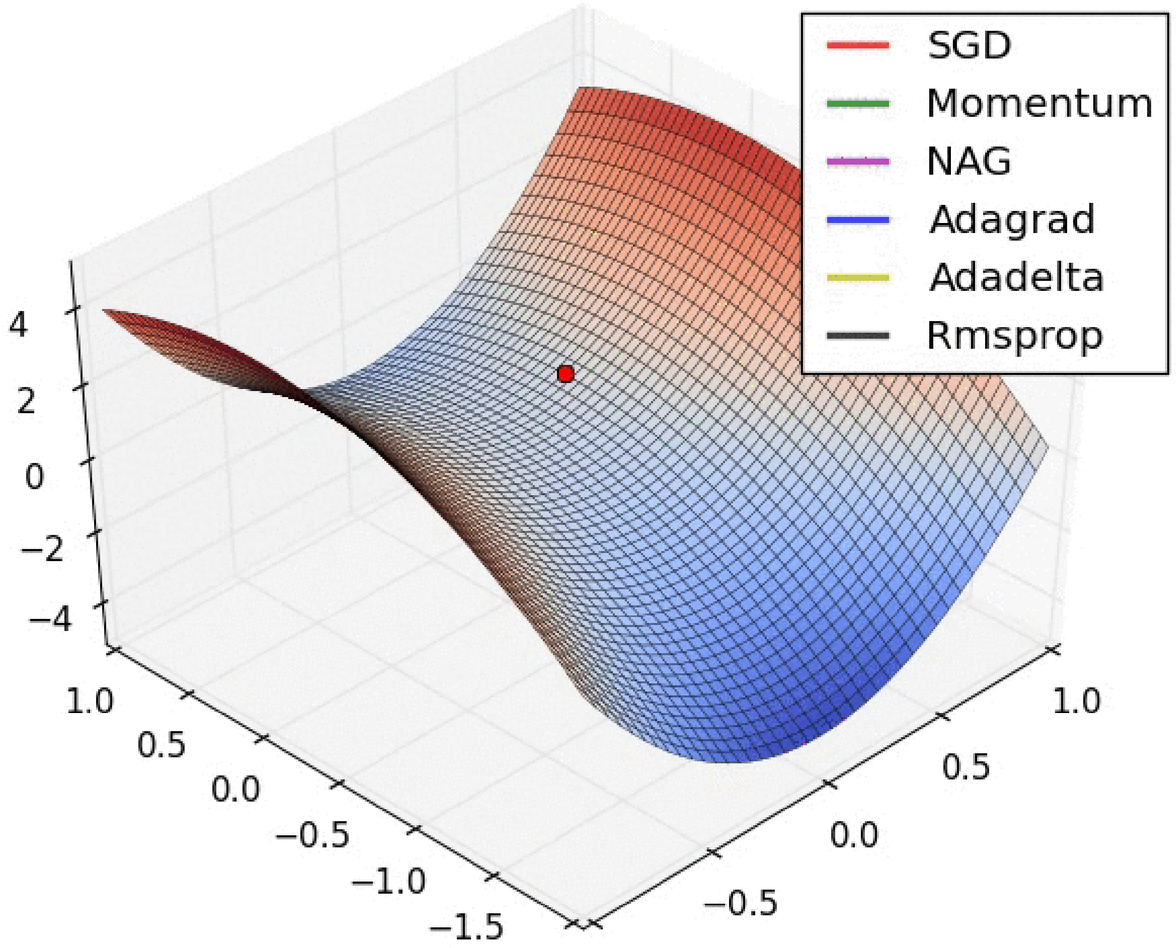
Gradient Descent

1) Loss Function 을 W(parameter)로
   편미분해서 W에 대한 Gradient를 구한다.

2) Gradient를 이용해서 W를 업데이트 한다.
   $W^* = W - \lambda \, Loss'(W)$



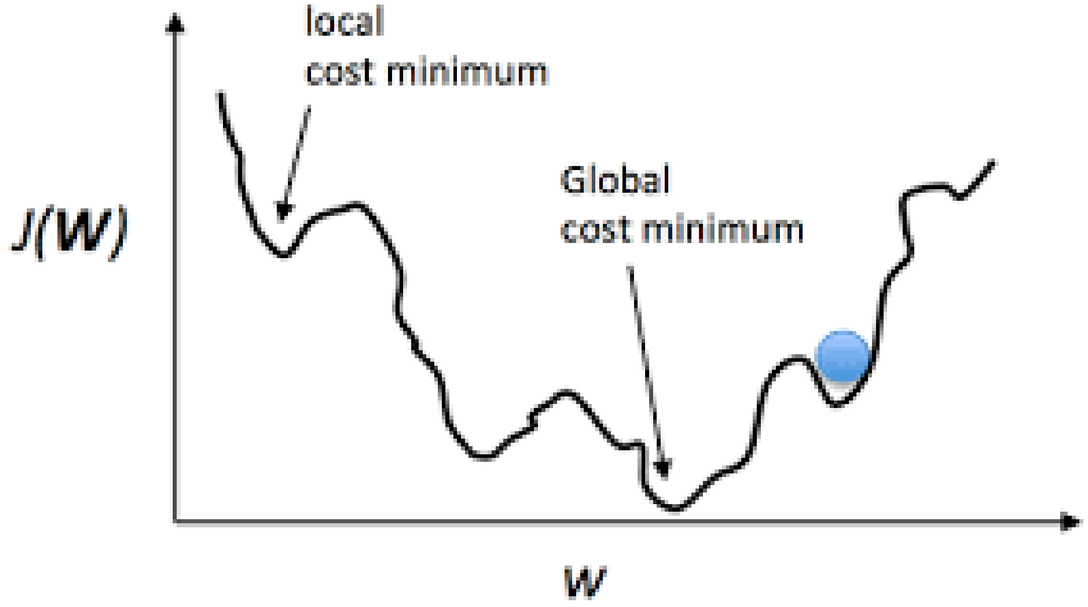
Gradient Descent

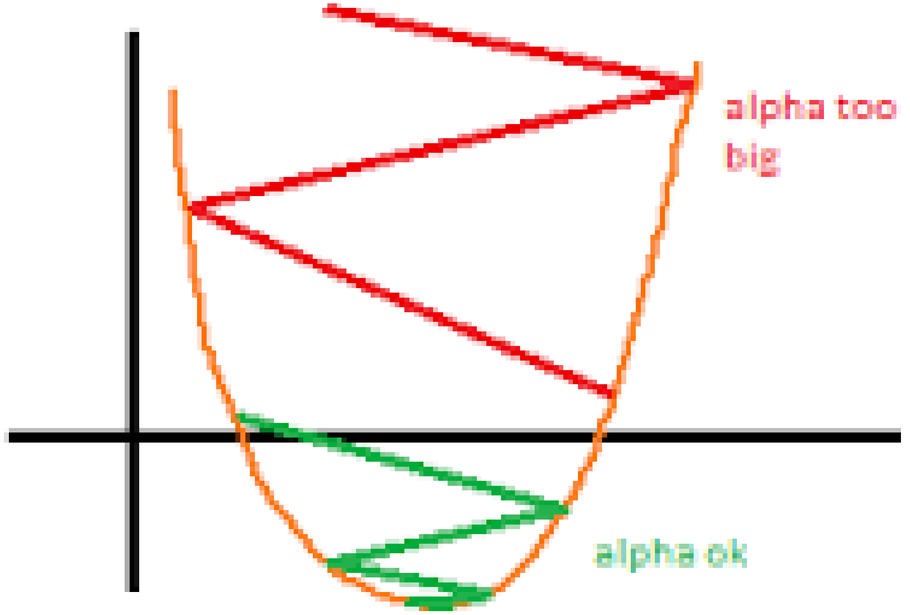


Gradient Descent의 문제점

1) Local Optima
   → Momentum
   (사실은 per-dimension learning rate)
2) Divergence
   → Gradient Decaying

$$V = \mu V' + \lambda \, Loss'(W)$$
$$W* = W - V$$





Gradient Descent의 문제점

3) 느림

→ Stochastic Gradient Descent

(또한 랜덤한 요소의 작용으로 더 수렴이 좋음.
그러나 subset이 전체의 분포를 충분히 반영해야함)

$$\text{for n} = 1 : \text{N}$$

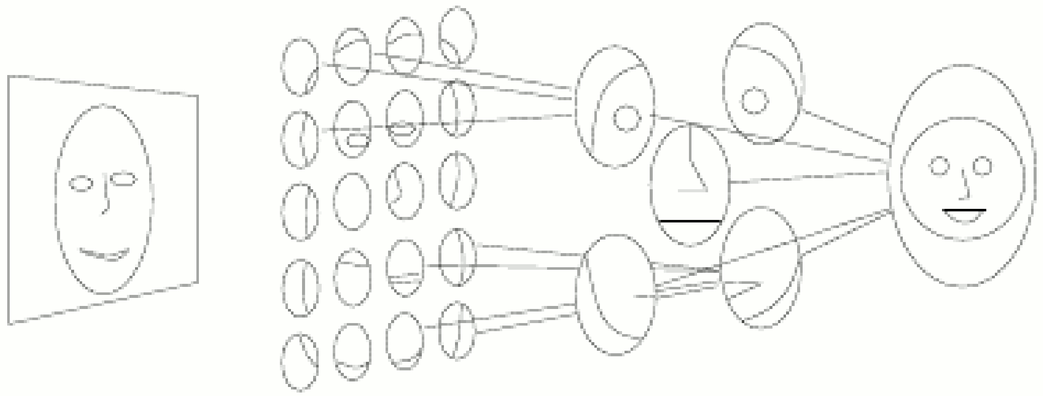
$$w_i^n := w_i^{n-1} - \eta \frac{\partial E_n(\text{w})}{\partial w_i}$$

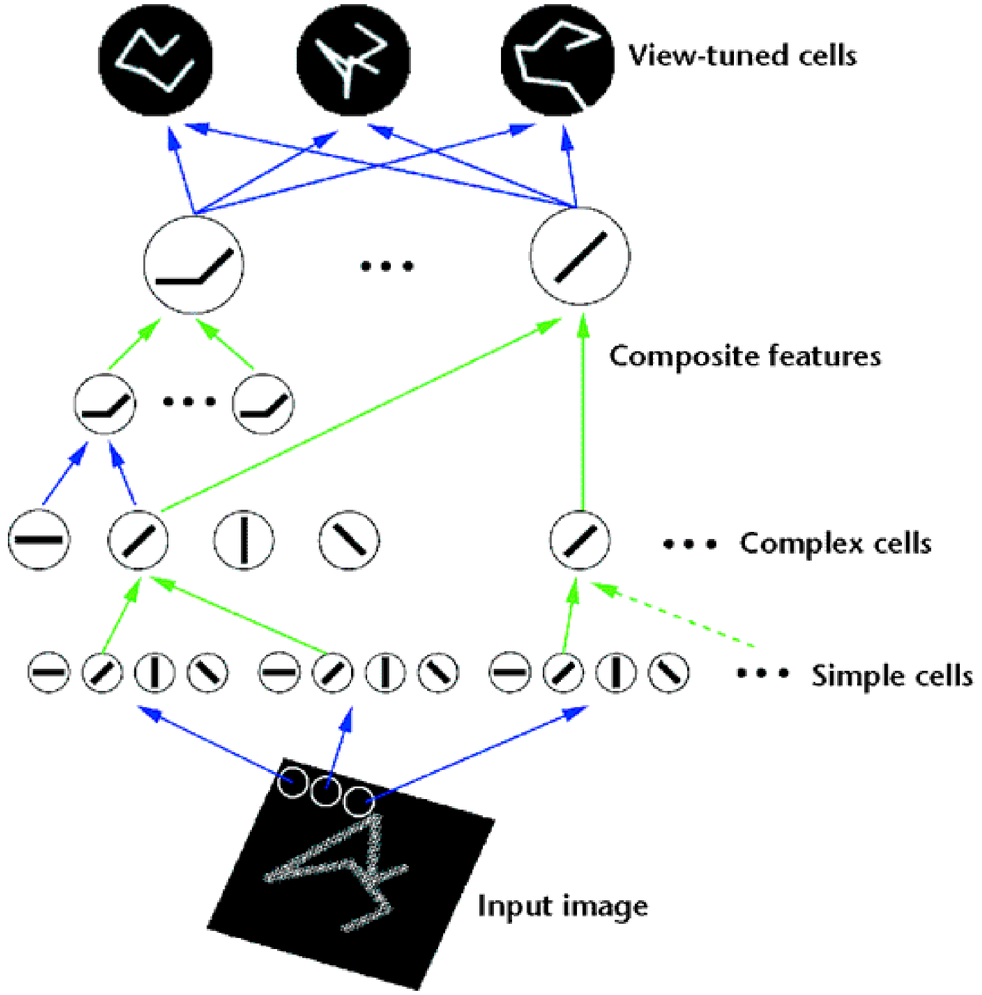
Error Back-propagation

Chain Rule

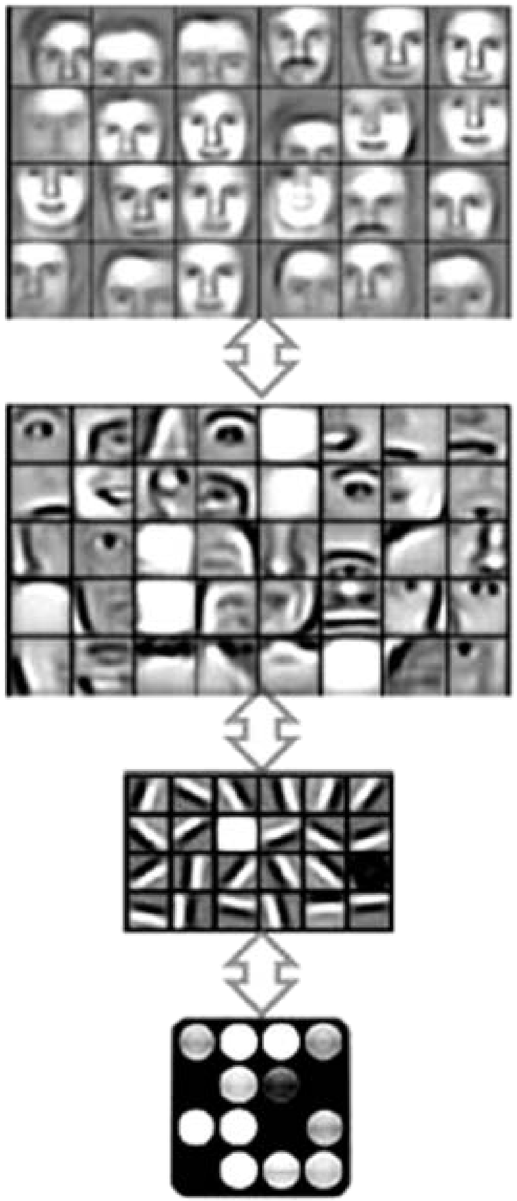
$$
\begin{aligned}
y' &= \lim_{t \to 0} \frac{f(g(x+t)) - f(g(x))}{t} \\
&= \lim_{t \to 0} \left[\frac{f(g(x+t)) - f(g(x))}{g(x+t) - g(x)} \times \frac{g(x+t) - g(x)}{t} \right] \\
&= f'(g(x)) \times g'(x)
\end{aligned}
$$

Deep Layer




View-tuned cells
Composite features
Complex cells
Simple cells
Input image

Deep Learning



Discrete Choices

⋮

Layer 2 Features

Layer 1 Features

Original Data

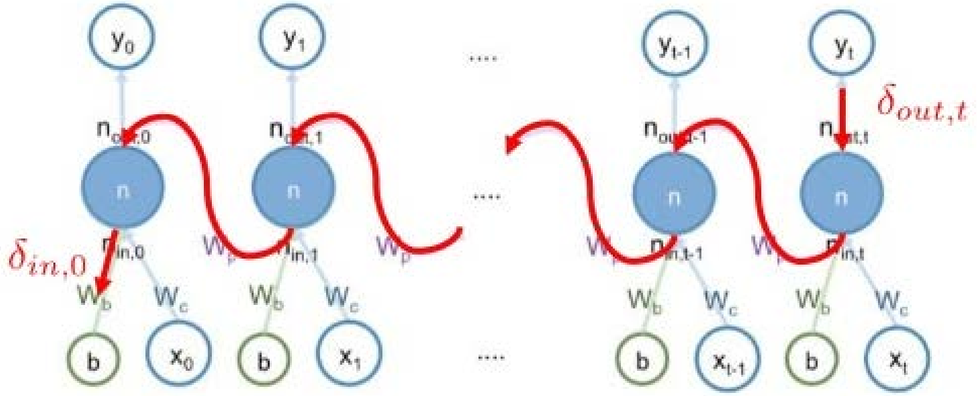
최신 ML에서 딥러닝은 하나의 module로써만 사용된다.

Deep Learning의 강점 :

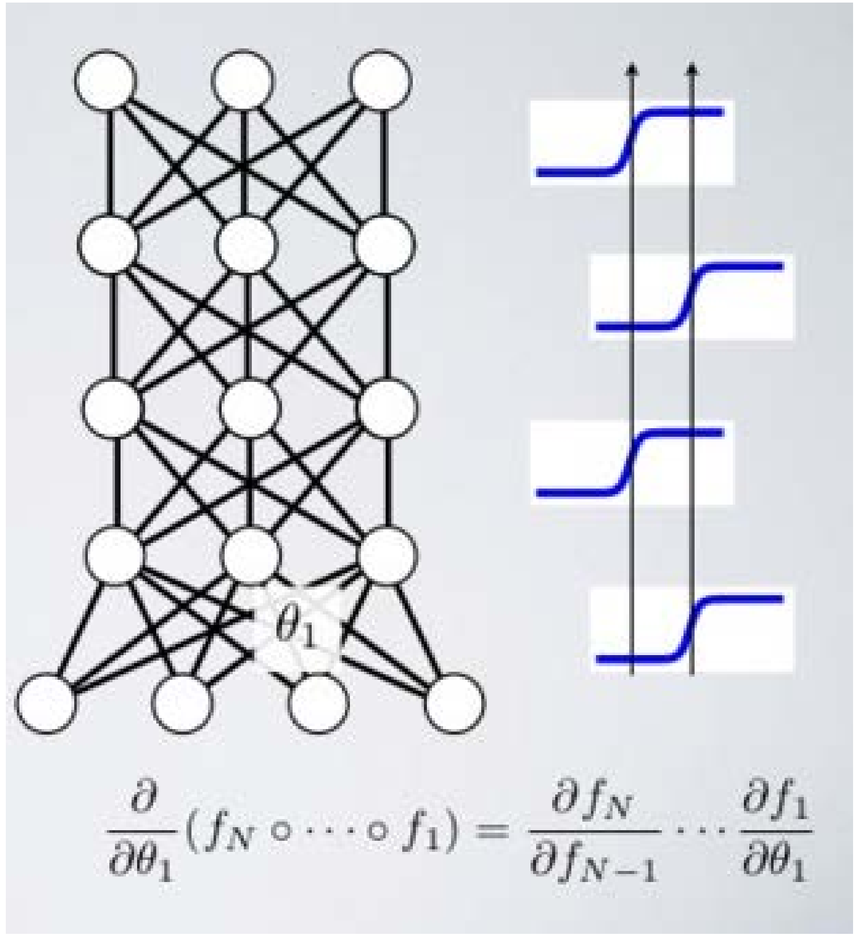
1. 층을 쌓을 수록 더욱 Abstract Feature Learning이 가능한 유일한 알고리즘.
2. Universal Function Approximator.

Gradient Vanishing

Vanishing Gradient Problem

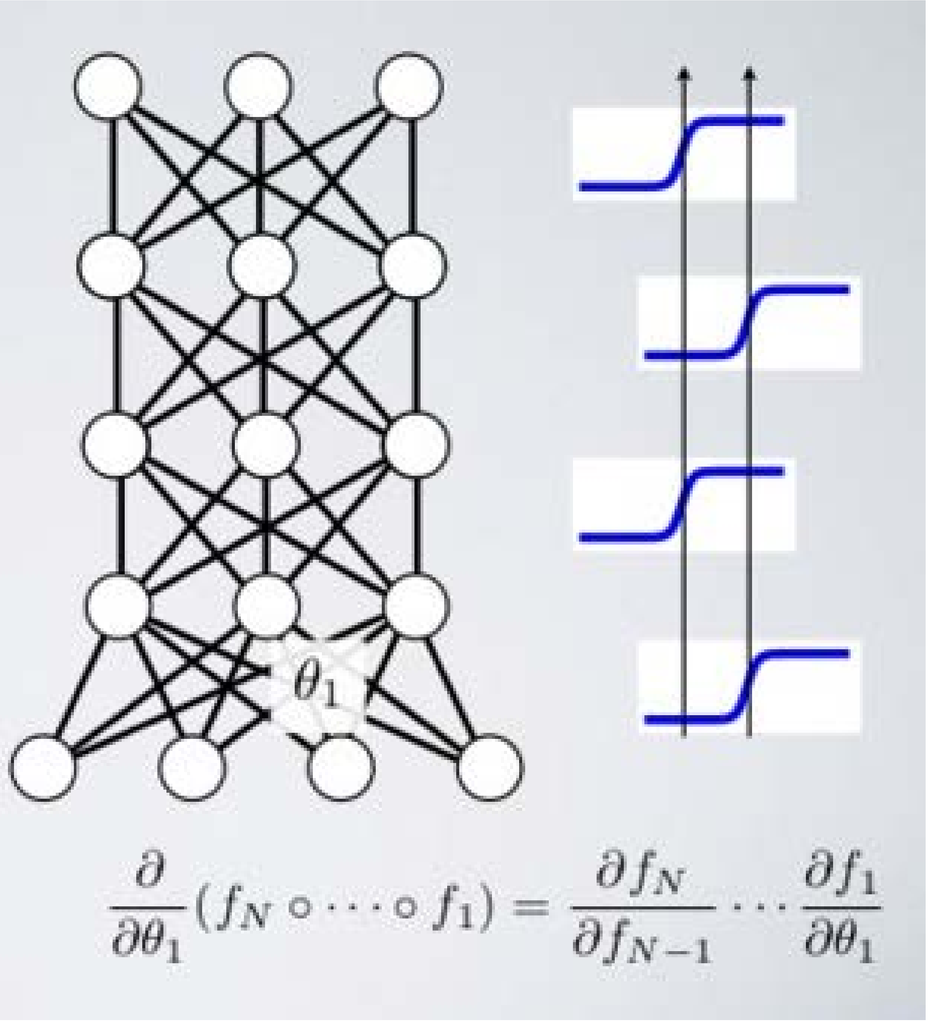


$$\delta_{in,0} = \delta_{out,t} \frac{\partial n_{out,t}}{\partial n_{in,t}} \frac{\partial n_{in,t}}{\partial n_{out,t-1}} \cdots \frac{\partial n_{in,1}}{\partial n_{out,0}} \frac{\partial n_{out,0}}{\partial n_{in,0}}$$

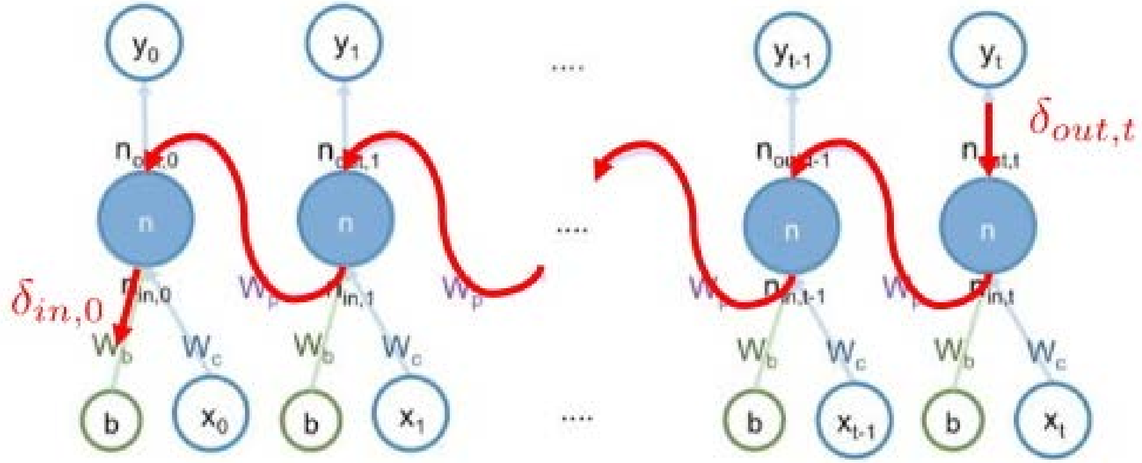


$$\frac{\partial}{\partial \theta_1}(f_N \circ \cdots \circ f_1) = \frac{\partial f_N}{\partial f_{N-1}} \cdots \frac{\partial f_1}{\partial \theta_1}$$

Gradient Vanishing



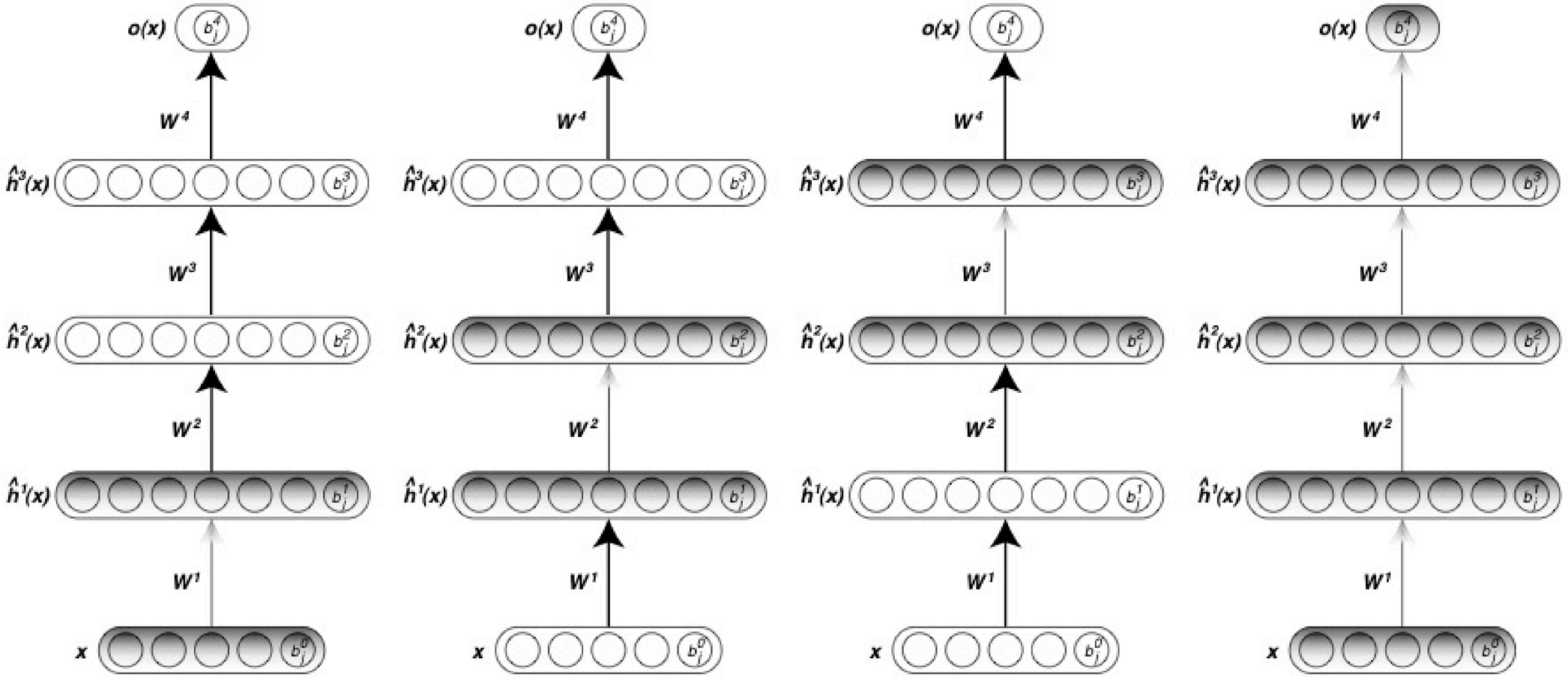
$$\frac{\partial}{\partial \theta_1}(f_N \circ \cdots \circ f_1) = \frac{\partial f_N}{\partial f_{N-1}} \cdots \frac{\partial f_1}{\partial \theta_1}$$



$$\delta_{in,0} = \delta_{out,t} \frac{\partial n_{out,t}}{\partial n_{in,t}} \frac{\partial n_{in,t}}{\partial n_{out,t-1}} \cdots \frac{\partial n_{in,1}}{\partial n_{out,0}} \frac{\partial n_{out,0}}{\partial n_{in,0}}$$

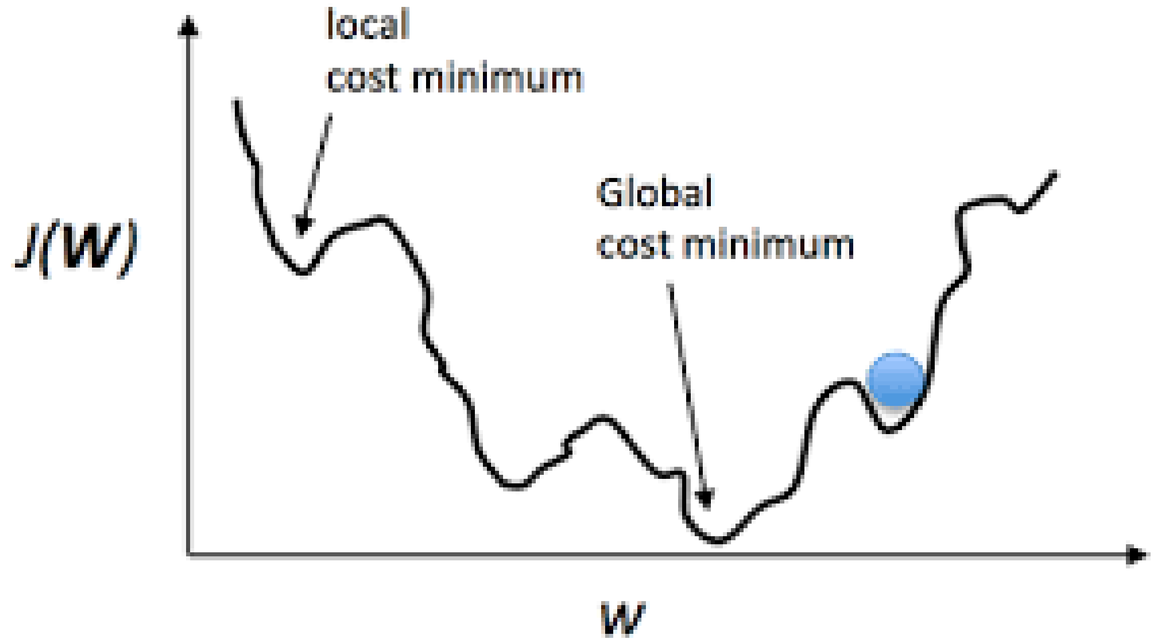
Gradient Vanishing 해결

1) Layer-wise Pretraining



Gradient Vanishing 해결

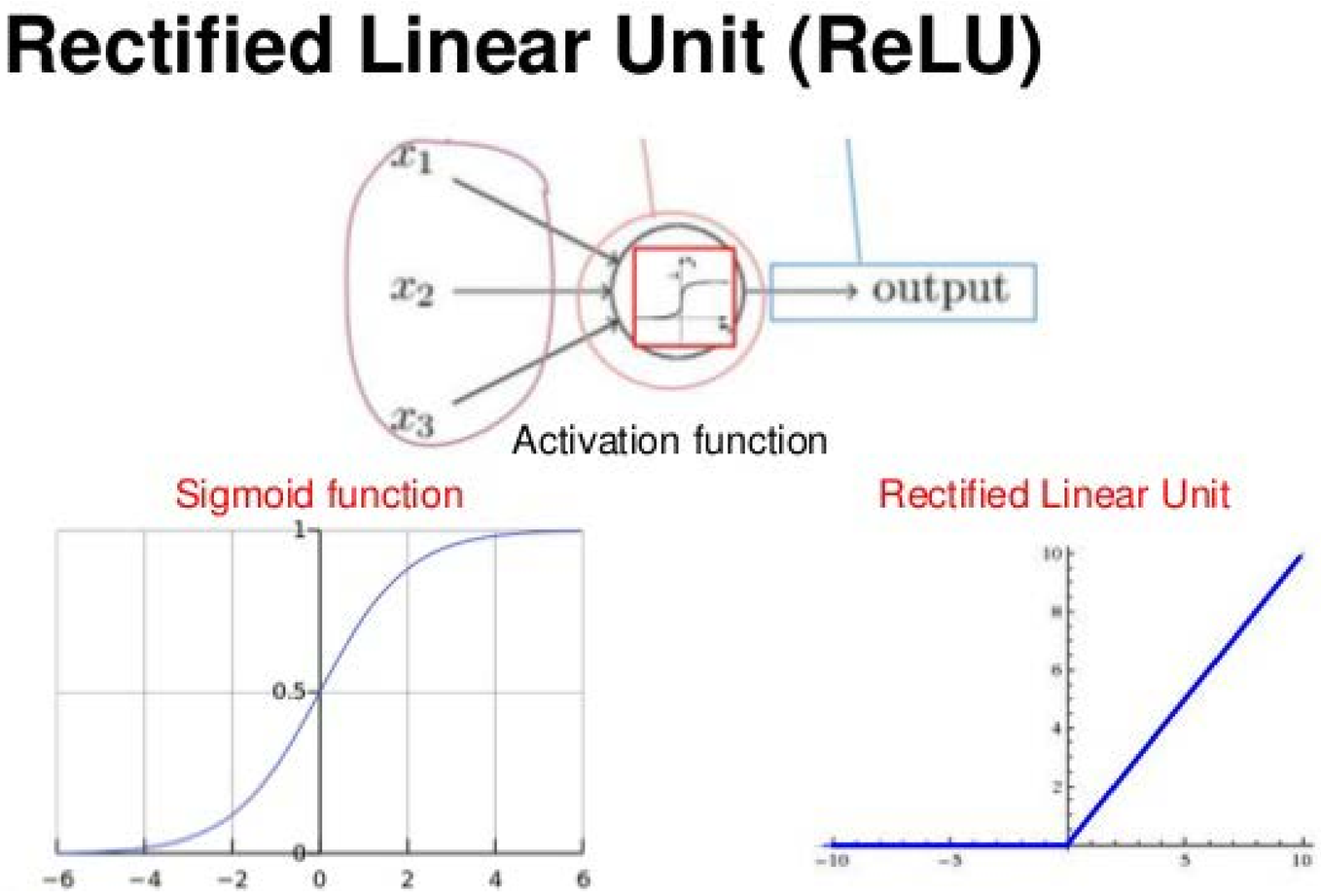
의미 1) Global optima와 더 가까운 Initial Weight 제공



의미 2) Layer간의 긴밀함이 증가해서 gradient가 더 잘 전파됨

Gradient Vanishing 해결

2) ReLU



Output Node

1) Regression → Weighted Sum(No activation) + Least Mean Square Loss

2) Classification → Softmax + Cross-Entropy Loss

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

$$H(p, q) = -\sum_{x} p(x) \log q(x).$$

Applications

(Credit: Sungjoon Choi)

https://github.com/sjchoi86/Tensorflow-101

Scene Recognition (CNN)

Predictions:

- **Type of environment:** outdoor
- **Semantic categories:** rock_arch:0.63, arch:0.30,
- **SUN scene attributes:** rugged, natural light, dry, climbing, far-away horizon, touring, rocky, open area, warm, sand

B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. “Learning Deep Features for Scene Recognition using Places Database.” Advances in Neural Information Processing Systems 27 (NIPS), 2014.

Visual Style Recognition (CNN)

HDR

Macro

Baroque

Roccoco

Vintage

Noir

Northern Renaissance

Cubism

Minimal

Hazy

Impressionism

Post-Impressionism

Long Exposure

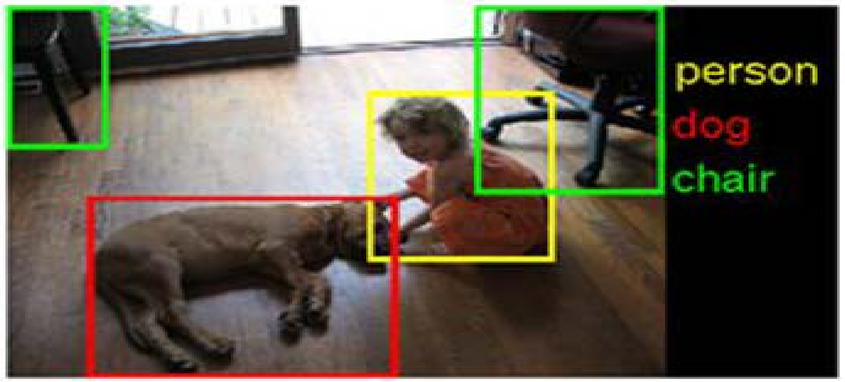
Romantic

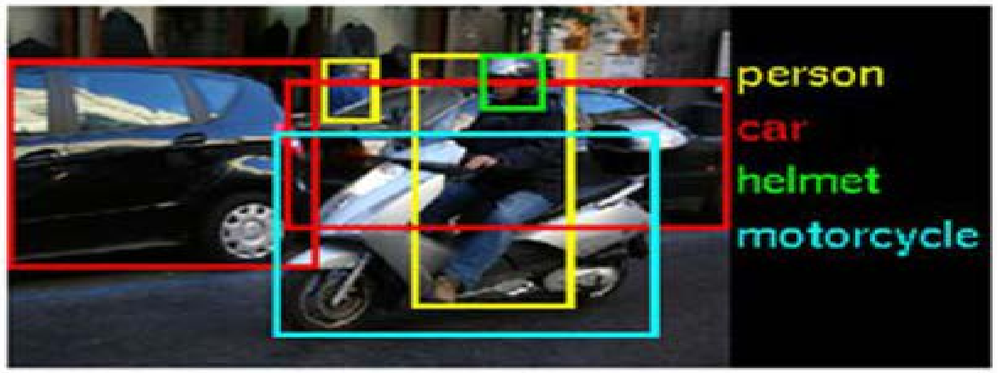
Abs. Expressionism

Color Field Painting

Karayev, Sergey, et al. "Recognizing image style." *arXiv preprint arXiv:1311.3715* (2013).

Object Detection (R-CNN)

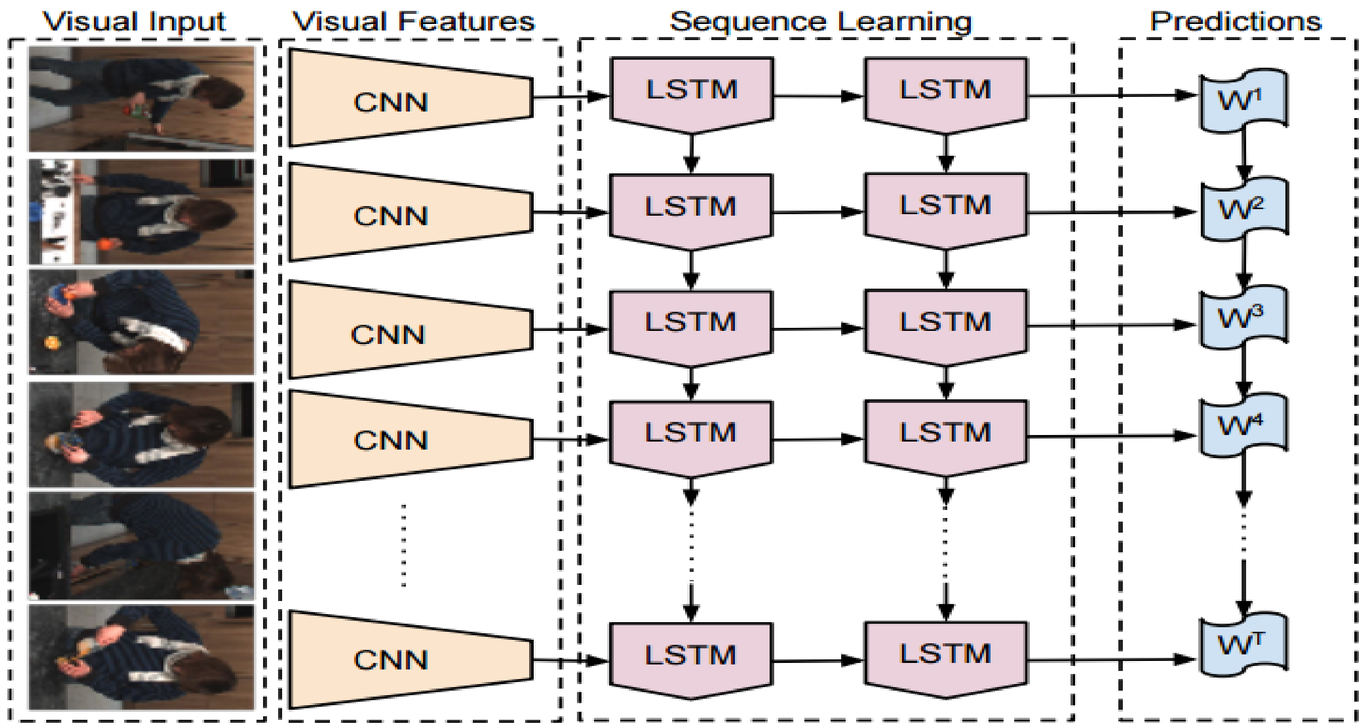




Detection ≈ Localization + Classification

Girshick, Ross, et al. "Region-based convolutional networks for accurate object detection and segmentation." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 38.1 (2016): 142-158.

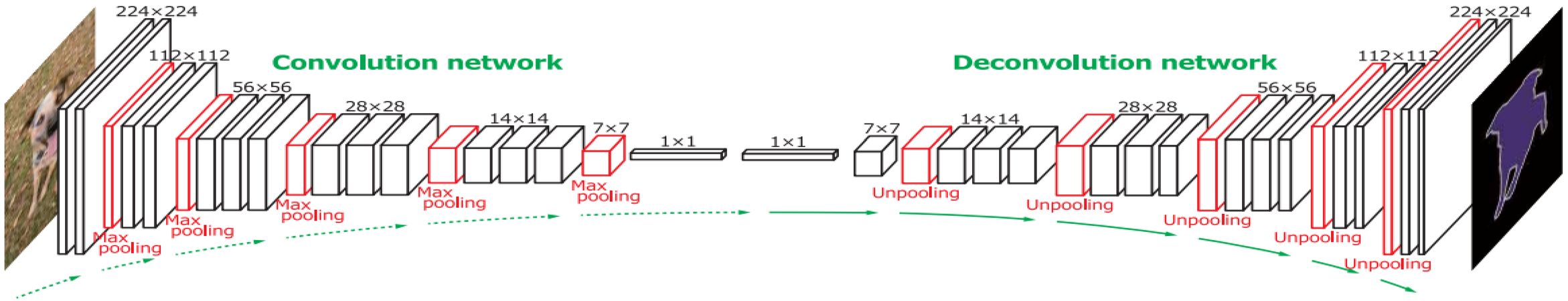
Image Captioning (CNN+LSTM)

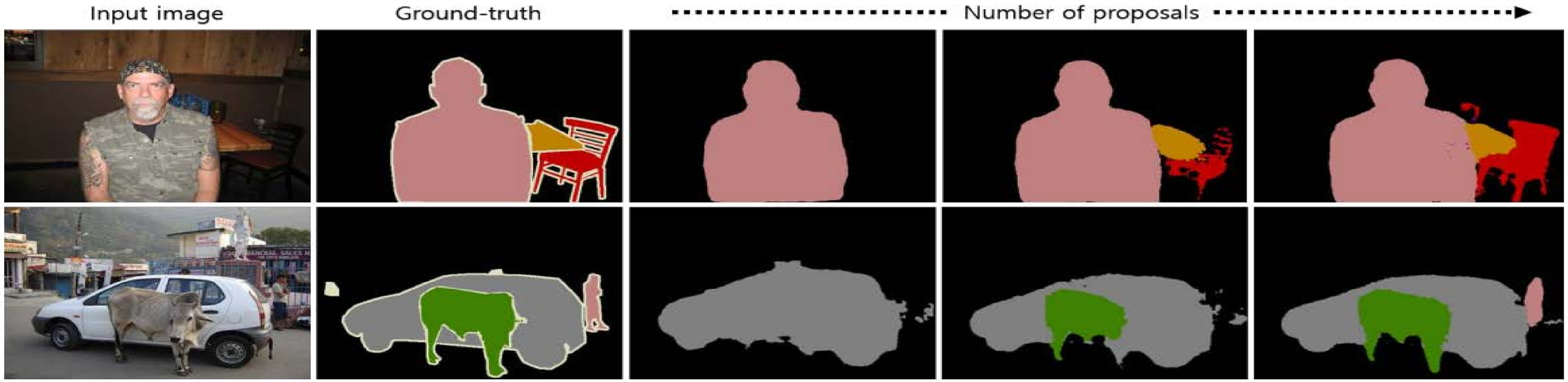


A black and white cat is sitting on a chair.

Donahue, Jeffrey, et al. "Long-term recurrent convolutional networks for visual recognition and description." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

Segmentation (DeconvNet)





Hyeonwoo Noh, Seunghoon Hong, Bohyung Han, Learning Deconvolution Network for Semantic Segmentation, arXiv:1505.04366.

Neural Style (CNN)

Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A neural algorithm of artistic style." arXiv, 2015

Deep Learning Tools

	Caffe	Torch	Theano	TensorFlow
Language	C++, Python	Lua	Python	Python
Pretrained	Yes ++	Yes ++	Yes (Lasagne)	Inception
Multi-GPU: Data parallel	Yes	Yes cunn.DataParallelTable	Yes platoon	Yes
Multi-GPU: Model parallel	No	Yes fbcunn.ModelParallel	Experimental	Yes (best)
Readable source code	Yes (C++)	Yes (Lua)	No	No
Good at RNN	No	Mediocre	Yes	Yes (best)

MLP & CNN TensorFlow 실습

감사합니다.