Deep Learning Seminar

3. Backpropagation

Contents

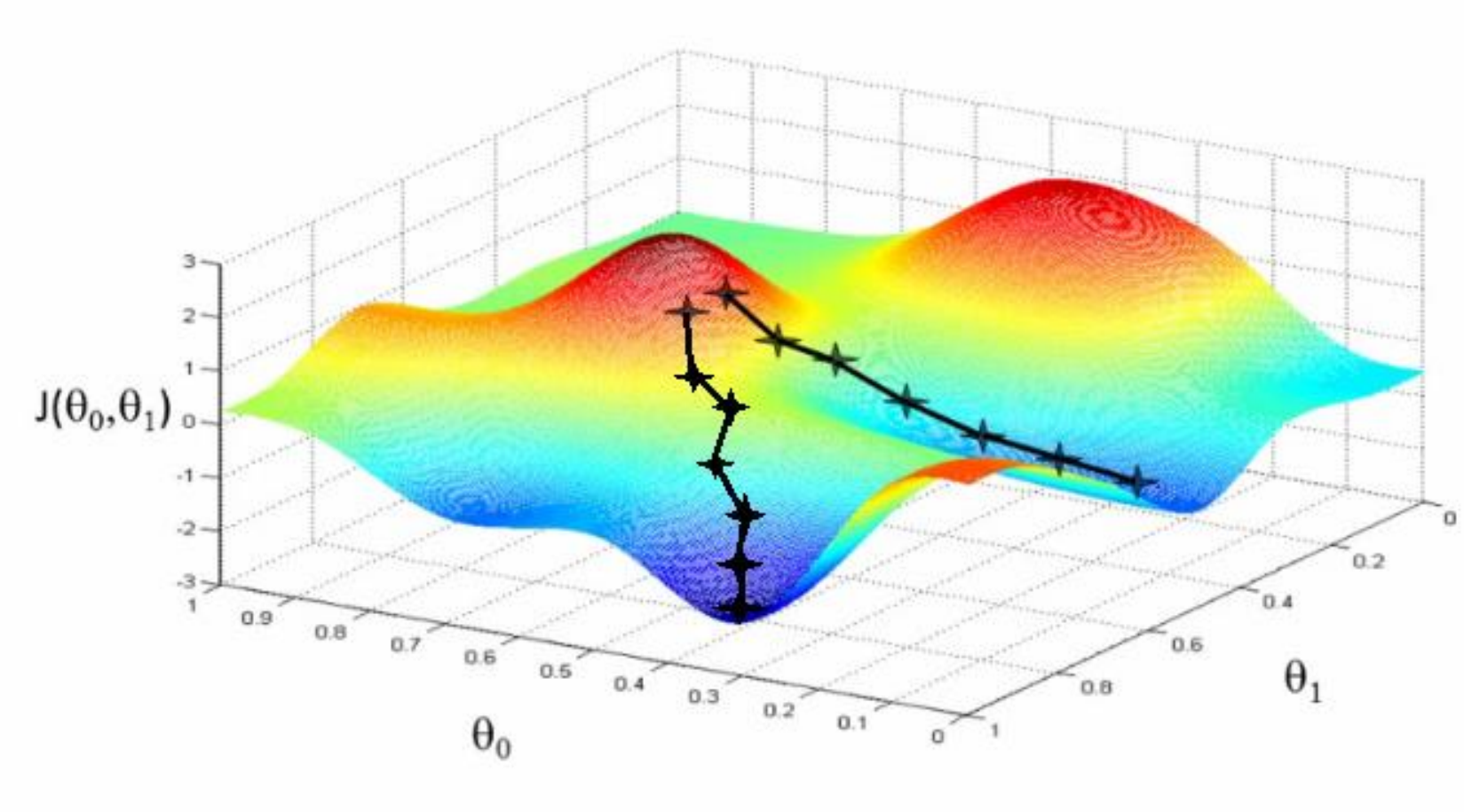
Reference: lecture note (Fei-Fei Li)
lecture note (Andrew Ng)
모두를 위한 머신러닝 (Sung kim)

1. Optimization

Optimization

Optimization

Optimization



Optimization

1) Random Search

무작위로 W 값을 여러 번 넣은 뒤, 그 중에서 최고의 값을 설정

-> 정상에서 눈 가리고 여러 번 하산 해본 뒤, 그 중 최고의 길을 선택

Optimization

2) Random Local Search

무작위 방향으로 W 값 바꿔서 Loss가 감소하는지 확인 한 후, 감소하면 W값을 업데이트

-> 눈을 가리고 무작위 방향으로 정해서 발을 살짝 뻗어서 더듬어 보고 그게 내리막 길이면 한 발짝 내딛음

Optimization

3) Gradient Descent

가장 가파르게 Loss를 감소하는 W 방향을 수학적으로 계산 한 뒤, 해당 방향으로 이동

-> 가장 가파르게 내려갈 수 있는 방향을 계산해서 해당 방향으로 한 발짝 이동

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Optimization

3) Gradient Descent

$$\mathrm{L(y,p)} = -\sum_k y_k log p_k \qquad p_t = \frac{e^{o_t}}{\sum_k e^{o_k}}$$

$$\frac{\partial L}{\partial o_i} = -\sum_k y_k \frac{\partial \log p_k}{\partial o_i} = -\sum_k y_k \frac{1}{p_k}\frac{\partial p_k}{\partial o_i}$$

$$= -y_i(1-p_i) - \sum_{k\neq i} y_k \frac{1}{p_k}(-p_k p_i)$$

$$= -y_i(1-p_i) + \sum_{k\neq i} y_k (p_i)$$

$$= -y_i + y_i p_i + \sum_{k\neq i} y_k (p_i)$$

$$= p_i \left(\sum_k y_k\right) - y_i = p_i - y_i$$

$$\frac{dO}{dW} = x \qquad \because \mathrm{O} = \mathrm{Wx} + \mathrm{b}$$

$$\nabla_W L = \frac{dL}{dW} = \frac{dL}{dO} \times \frac{dO}{dW} = (p-y) \times (x)$$

Optimization

- Stochastic Gradient Descent (SGD)

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(x_i, y_i, W) + \lambda R(W)$$

$$\nabla_W L(W) = \frac{1}{N}\sum_{i=1}^{N} \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W)$$

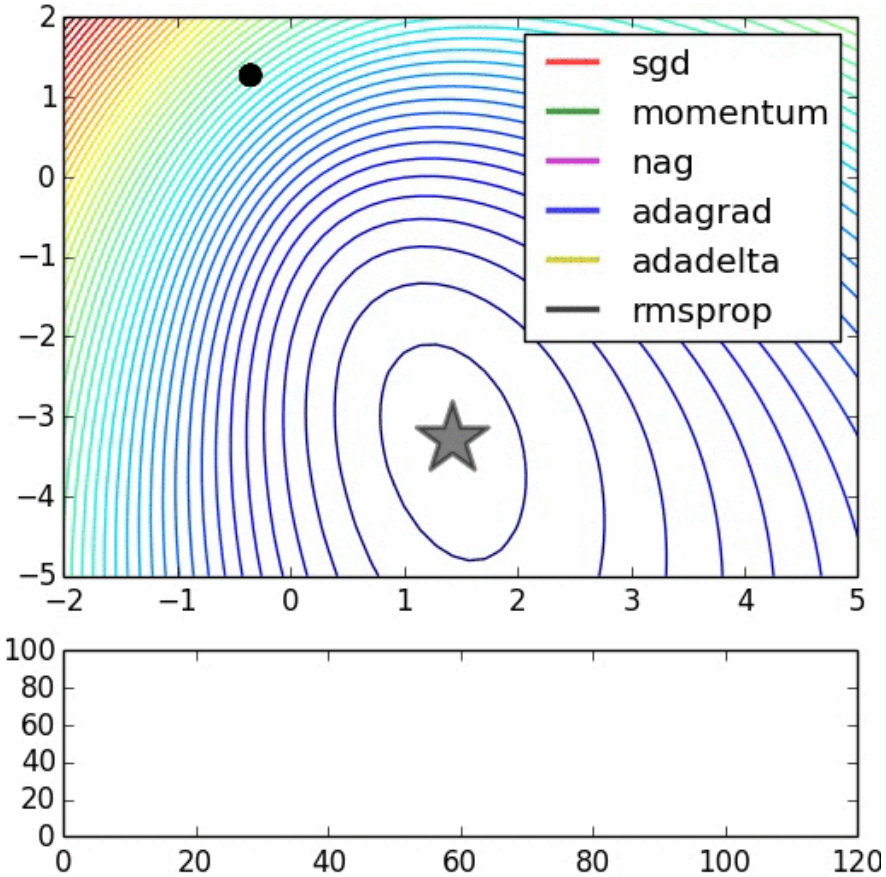
$$w_i^{k+1} = w_i^k - \eta \frac{dL}{dw_i}$$

Learning Rate > 0

(Hyperparameter)

Optimization

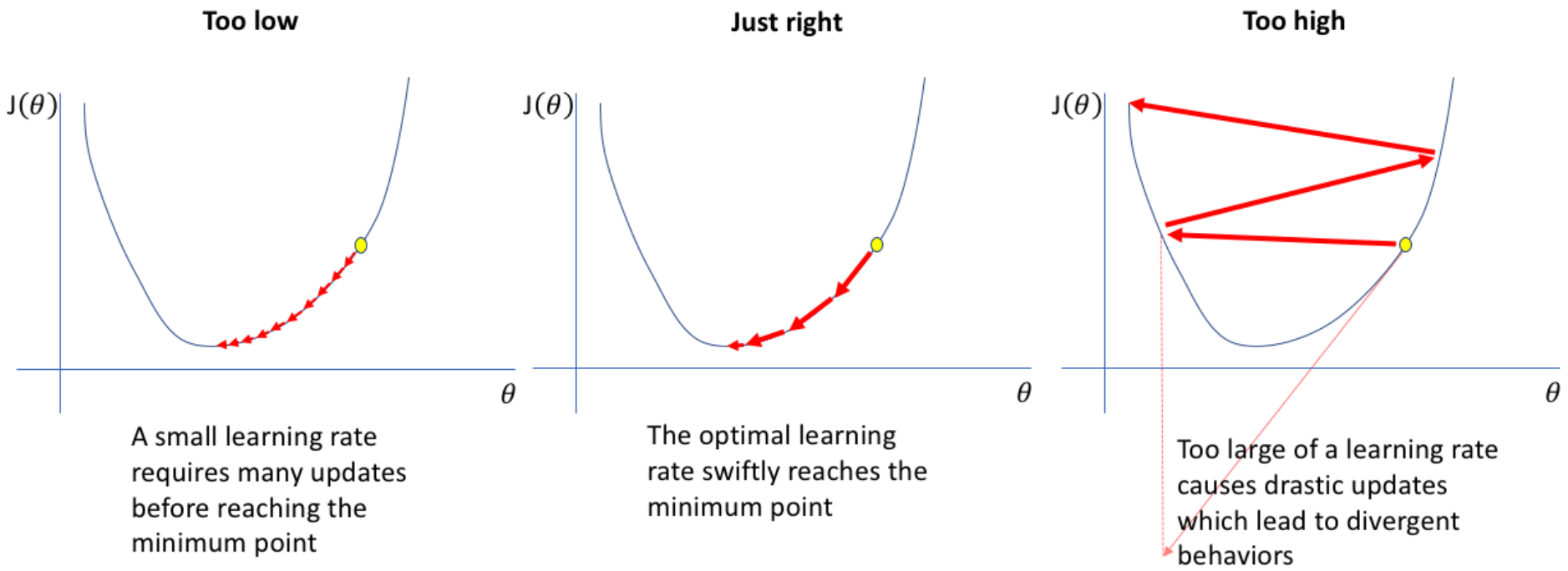
- Optimizers Comparison



Optimization

- Learning rate

하강보폭



Optimization

- Learning rate tuning

SGD Optimizer default lr:	1e-2 (=0.1)
Adam Optimizer default lr:	1e-4 (=0.001)
Learning rate decay:	½ decay / n epoch

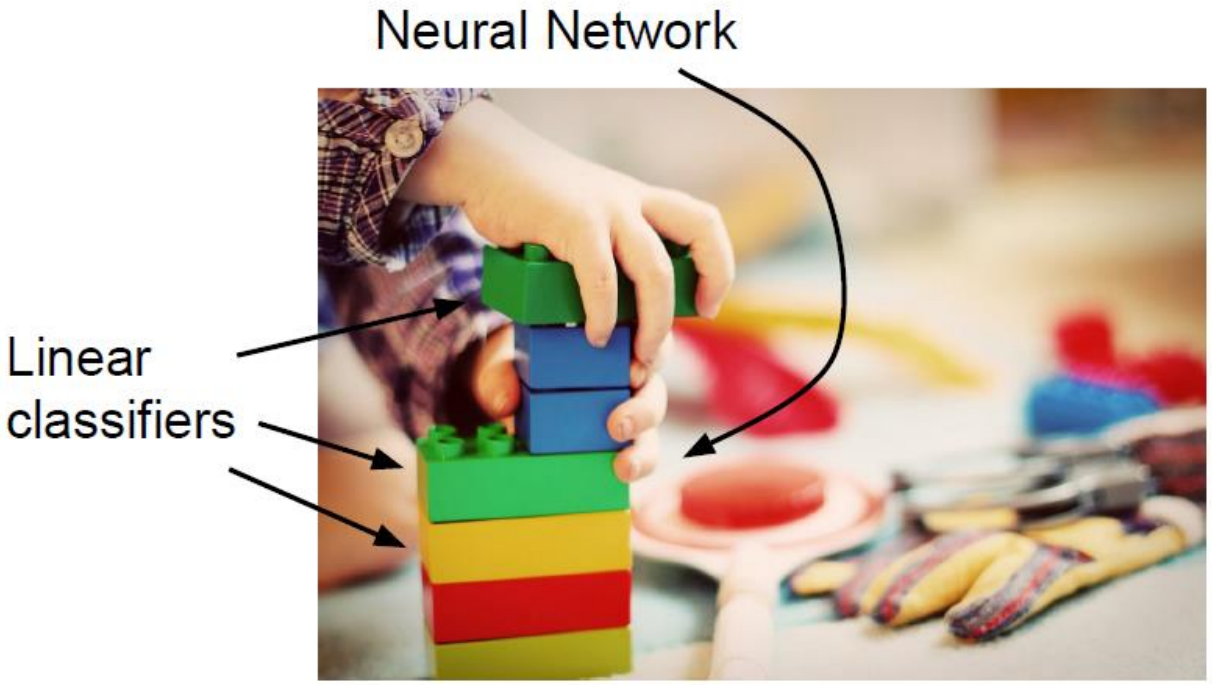
- Optimizer selection

1~10 epoch:	Adam Optimizer	(Fast, Rough)
10~ epoch:	SGD Optimizer	(Slow, Accurate)

2. Backpropagation

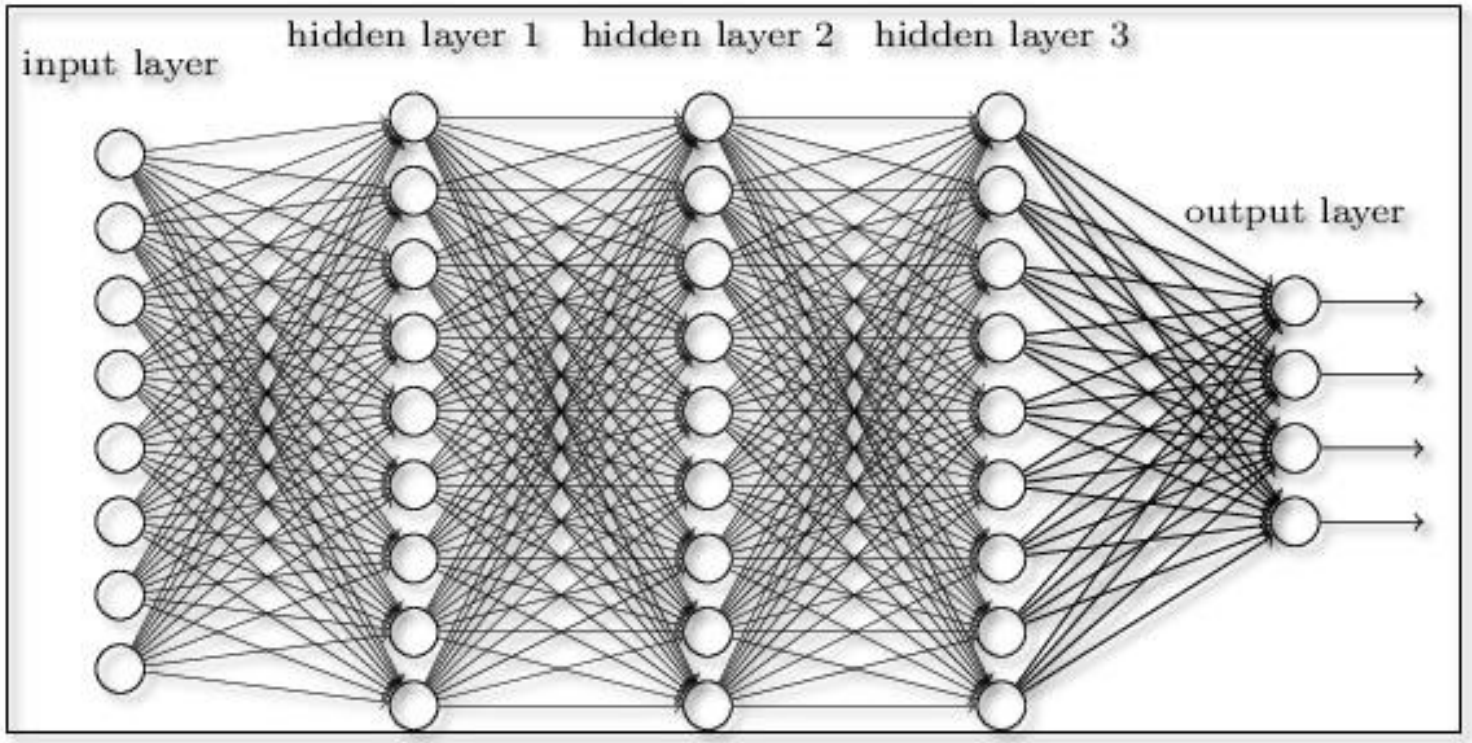
Backpropagation

- Neural Network



Backpropagation

- Neural Network



Backpropagation

- Backpropagation

  - How to update weight ?

  - W 값이 Loss에 얼마나 영향을 주었는지를 수치화 한 뒤, Loss를 줄이는 방향으로 W값을 업데이트
    (Loss에 영향을 많이 주었으면 크게 W가 크게 변화)

  - Neural Network 의 학습을 위한 핵심적인 개념

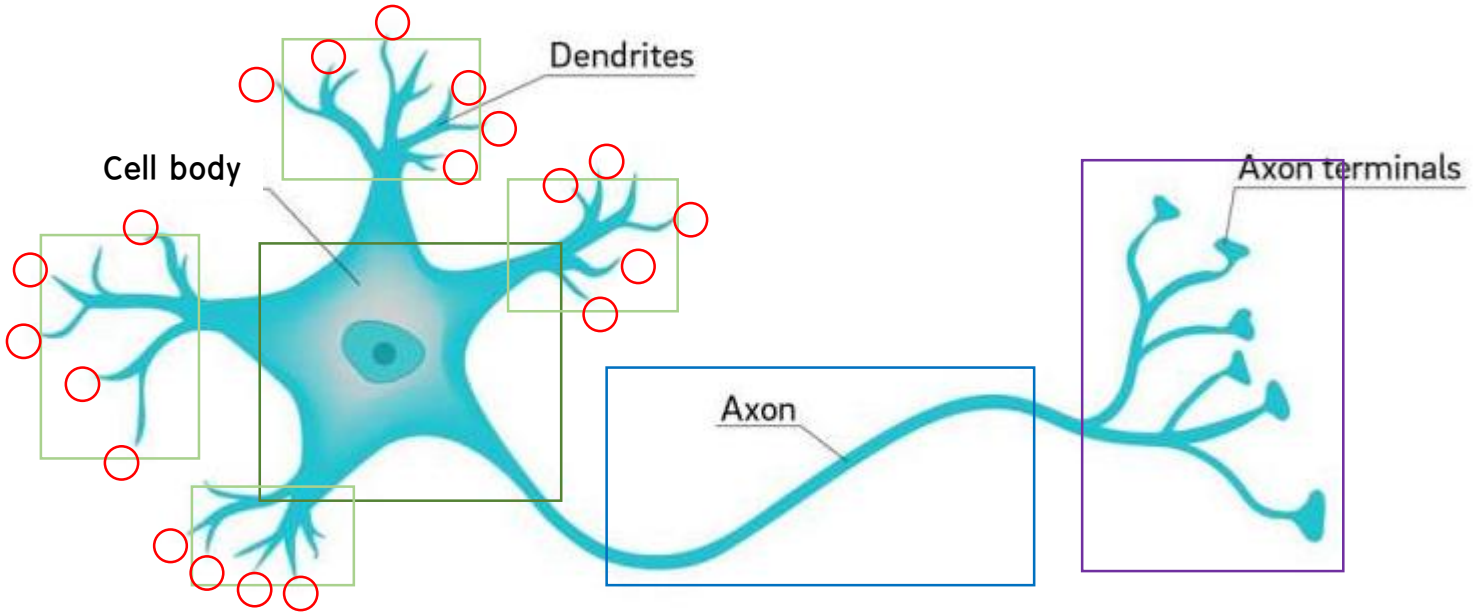
3. Neural Network

3-1) Neural Network (Fully-connection network)

3-2) Convolutional Neural Network

Neuron

- Structure of a neuron



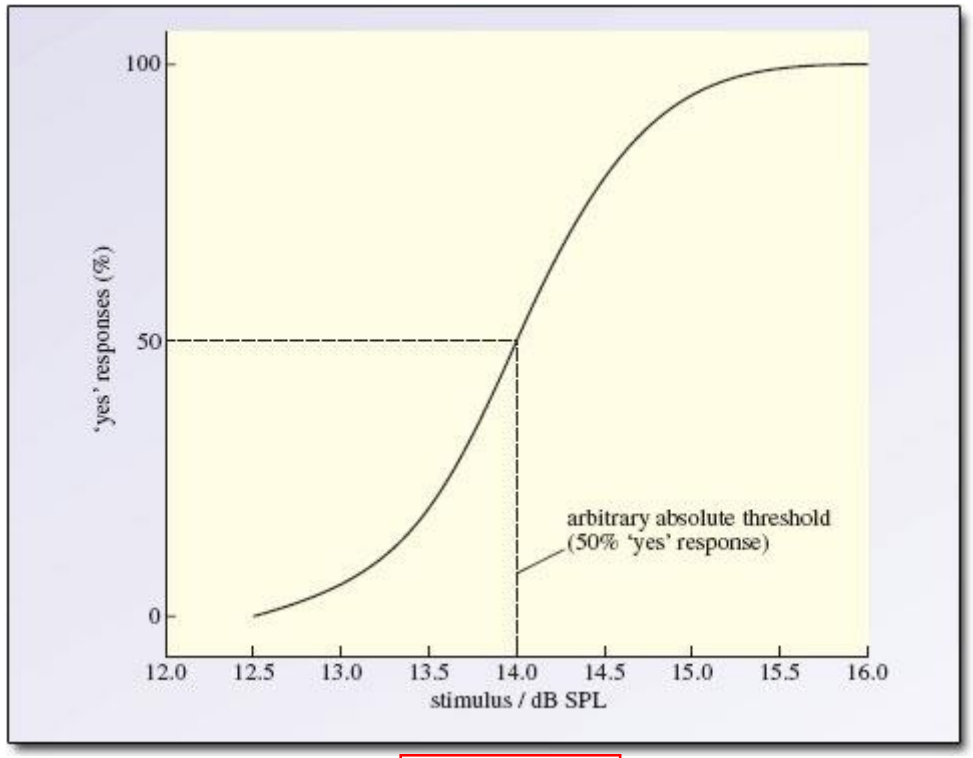
Input (x)	Functions 1 (wx)	Functions 2 (wx +b)	Functions 3 (activ (wx+b))	Output (active (wx+b))

Neuron

- Neuron threshold

Activation Function

Output
(activ (wx + b))



Input
(wx + b)

Neuron

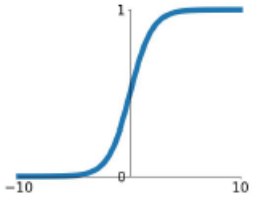
- Activation functions

Recurrent Neural Network
Auto-Encoder (Segmentation)

Convolutional Neural Network

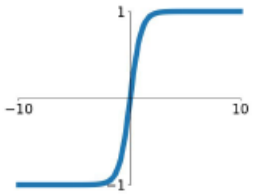
Sigmoid

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



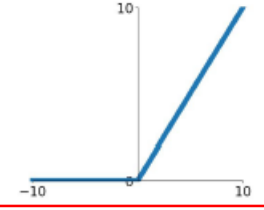
tanh

$$\tanh(x)$$



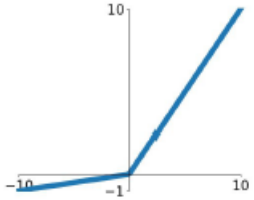
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

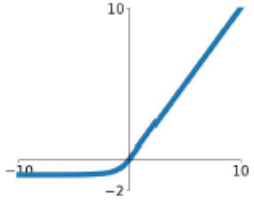


Maxout

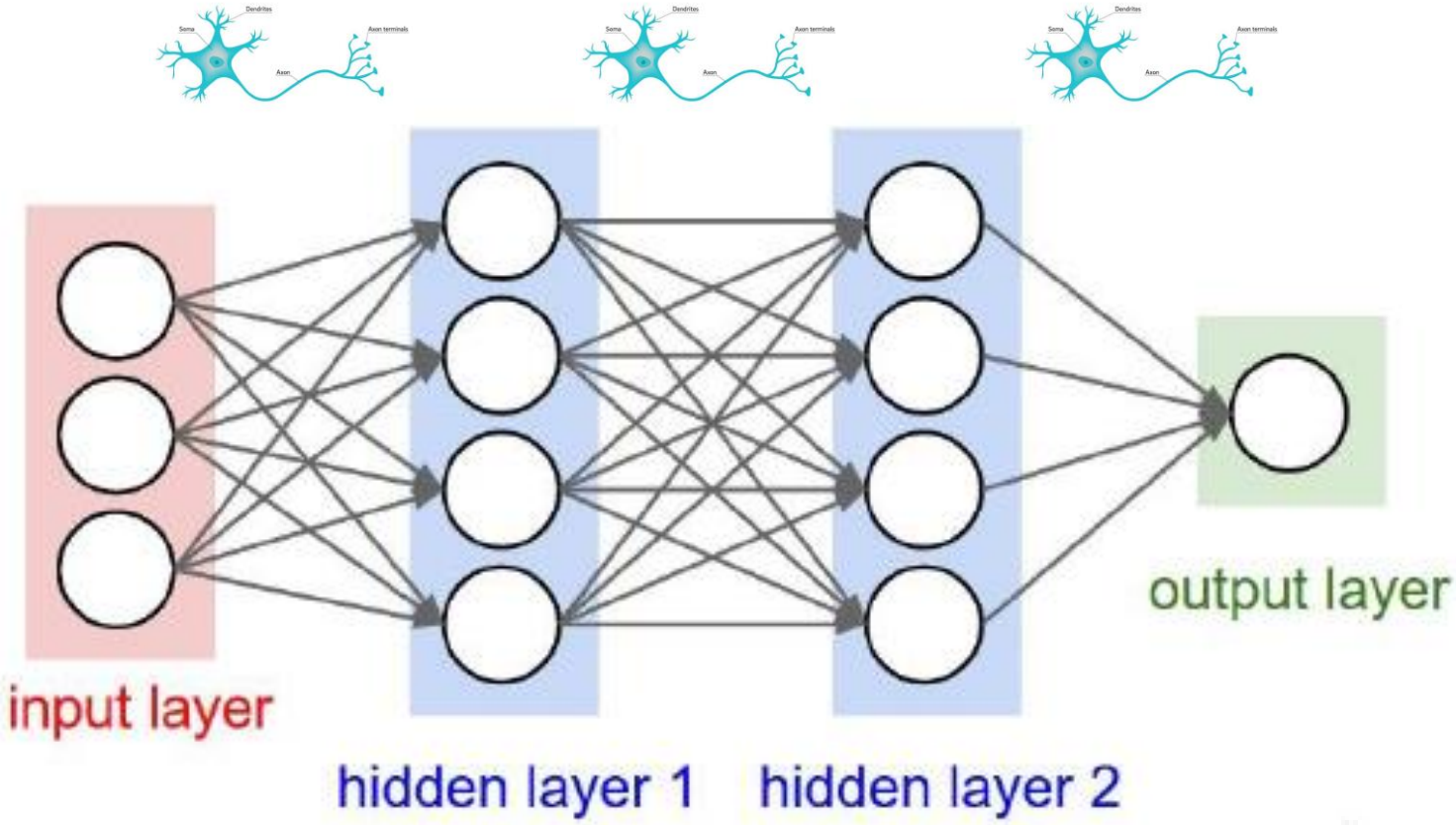
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Neural Networks (Fully-connected network)

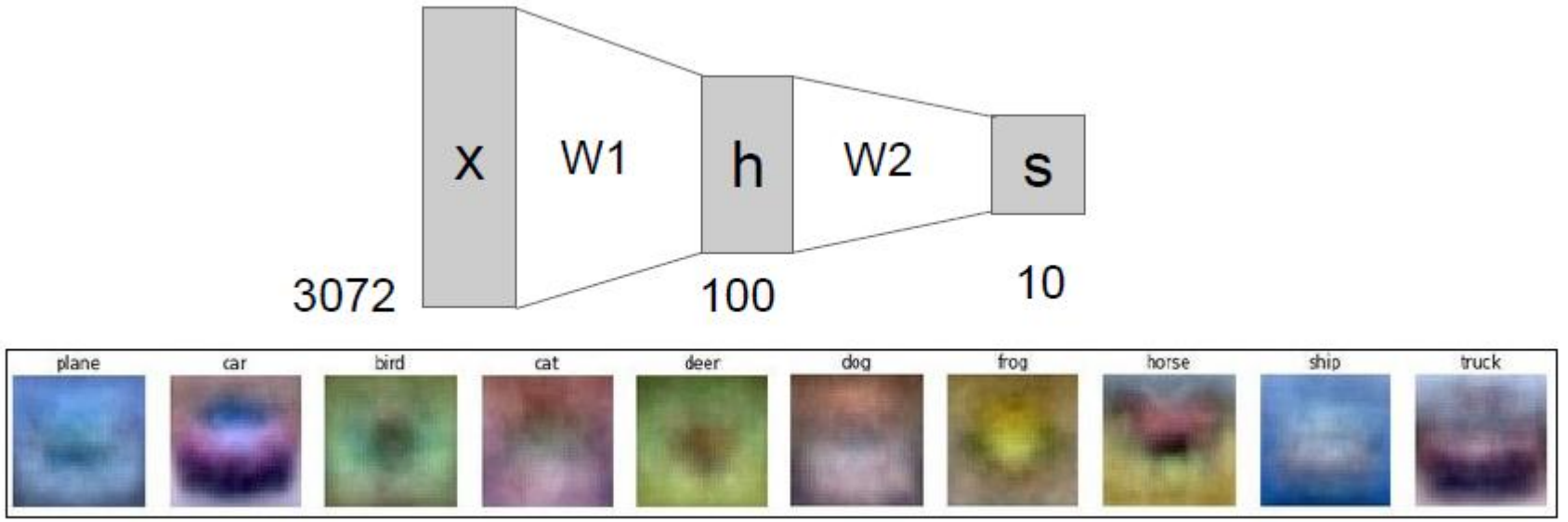


"Fully-connected" layers ⟹ $f\left(\sum_i w_i x_i + b\right)$

Neural Networks (Fully-connected network)

$$f = Wx$$

$$f = W_2 \max(0, W_1 x)$$

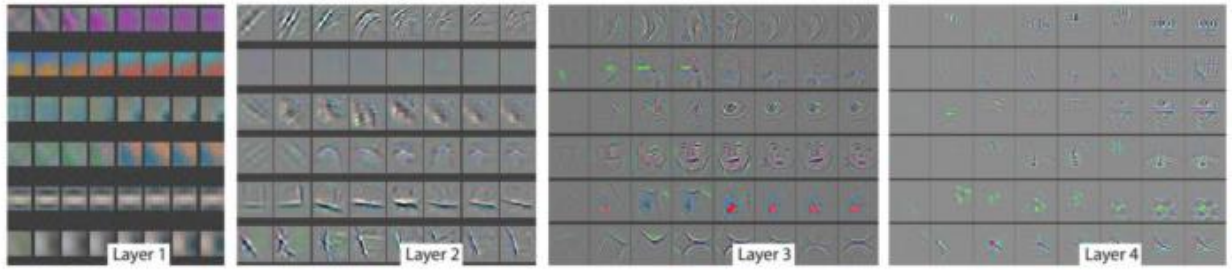


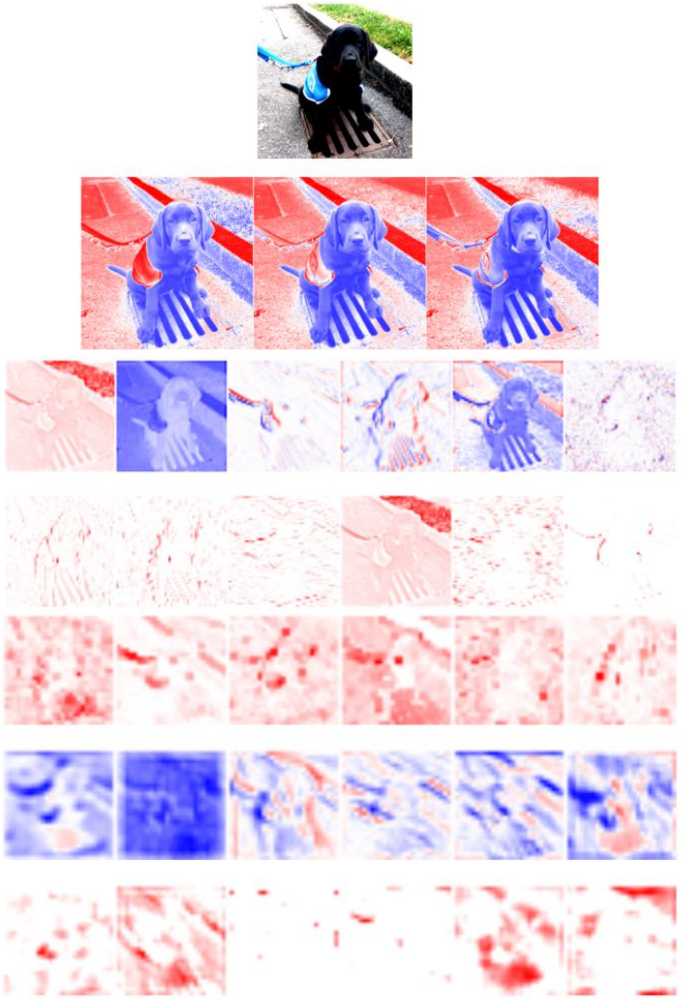
〈 Feature map 〉

Neural Networks (Fully-connection network)

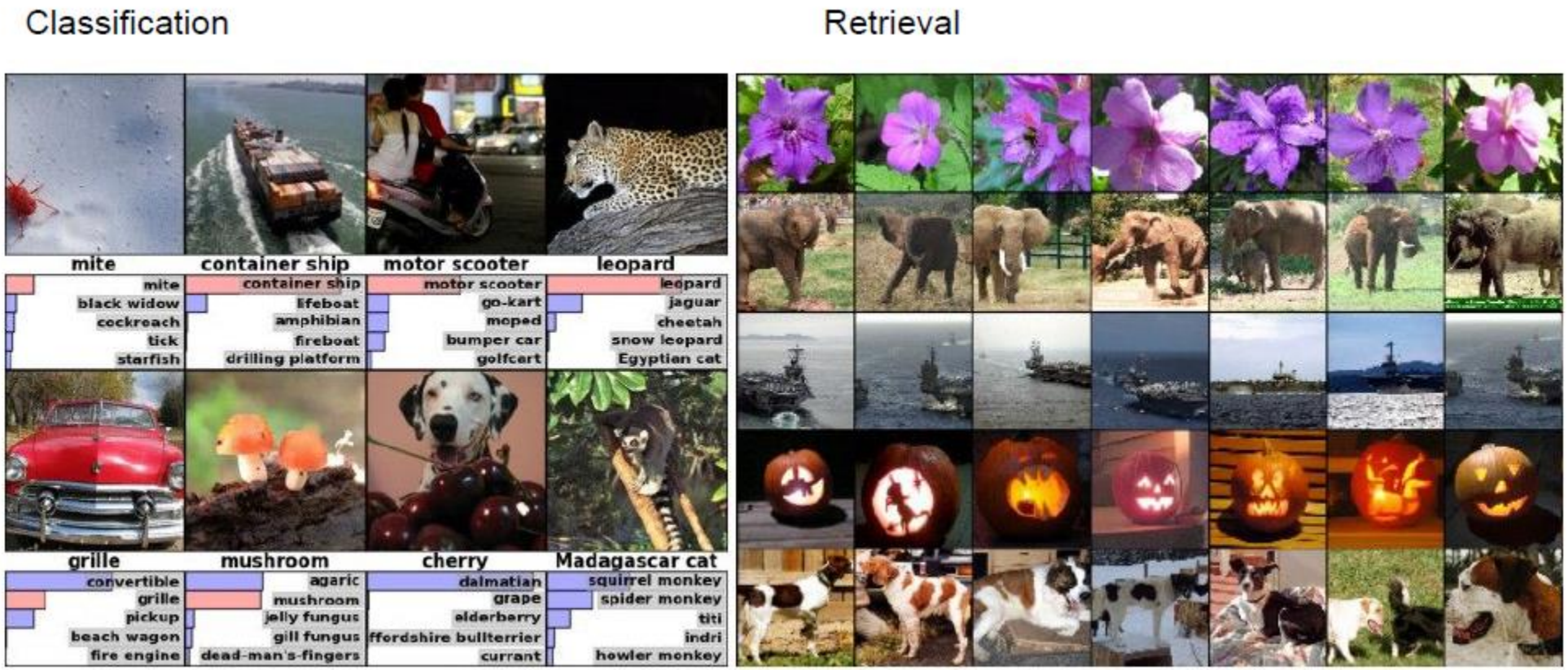
- Feature-map

  - Output of the layer
  - Representation of object



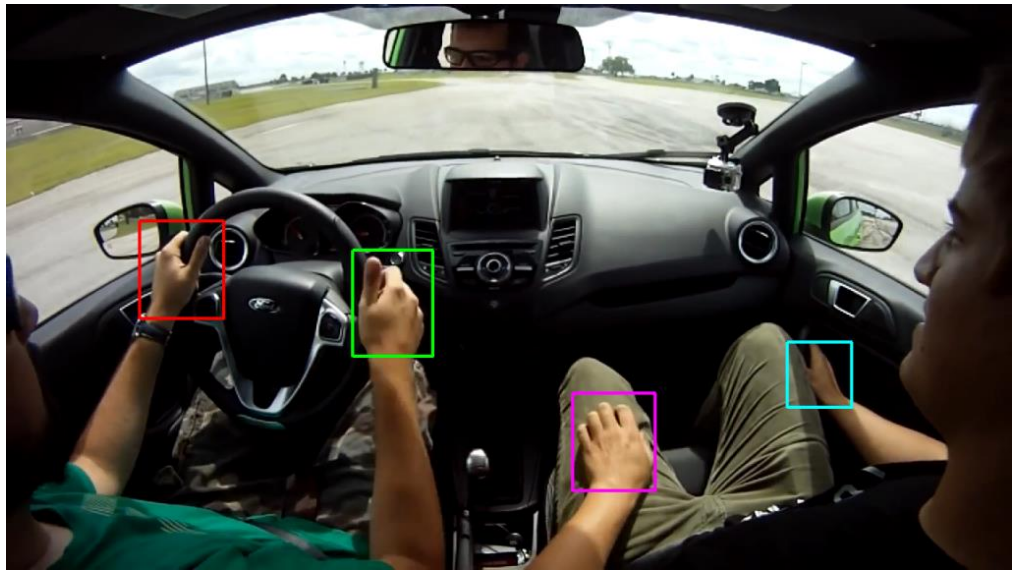


Convolutional Neural Network

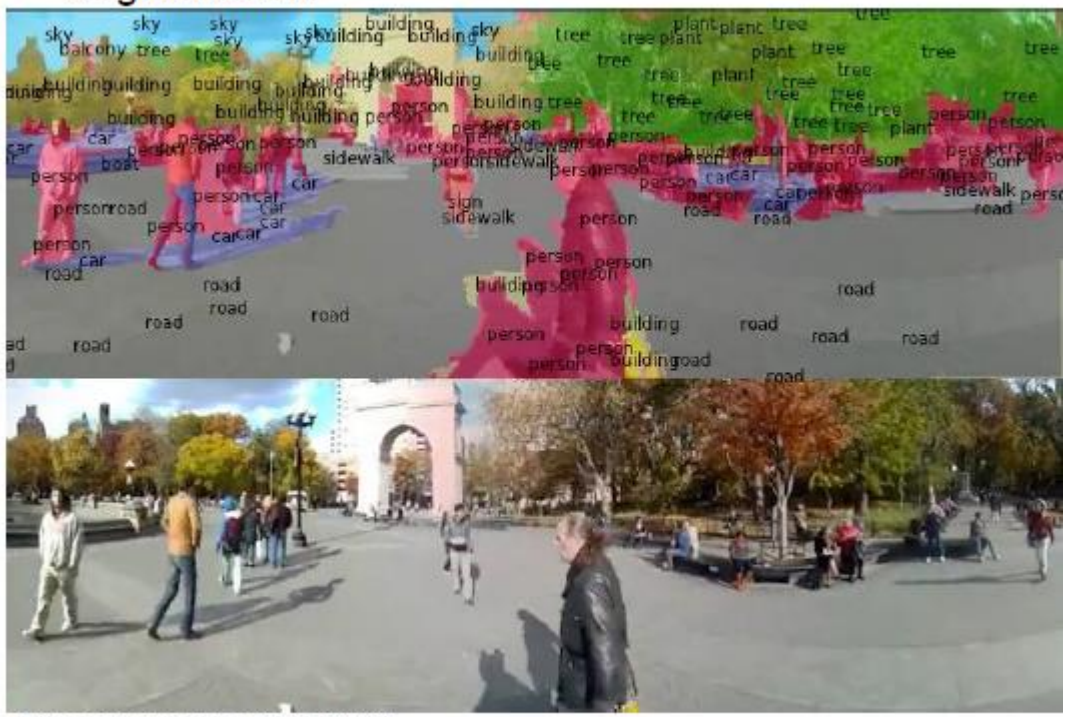


Convolutional Neural Network

Detection



Segmentation



Figures copyright Clement Farabet, 2012.
Reproduced with permission.

[Farabet et al., 2012]

Convolutional Neural Network

Motion Classification