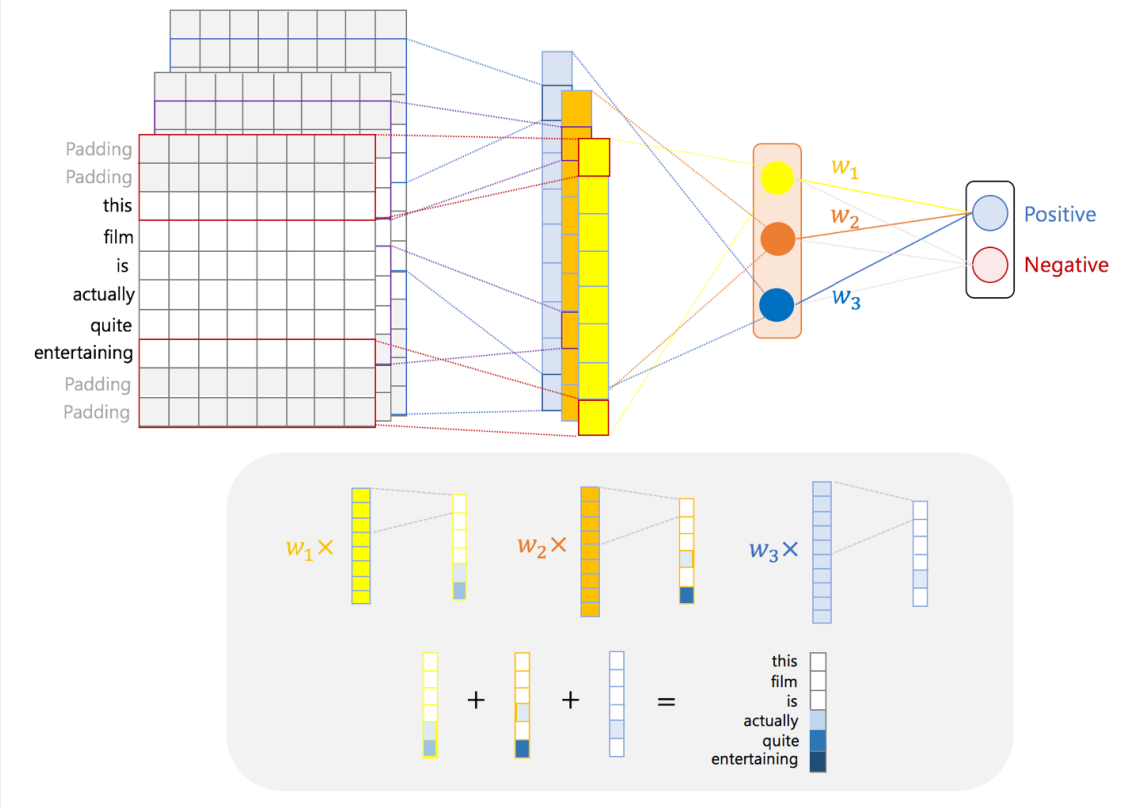


Lecture 8: Document Classification II

Pilsung Kang

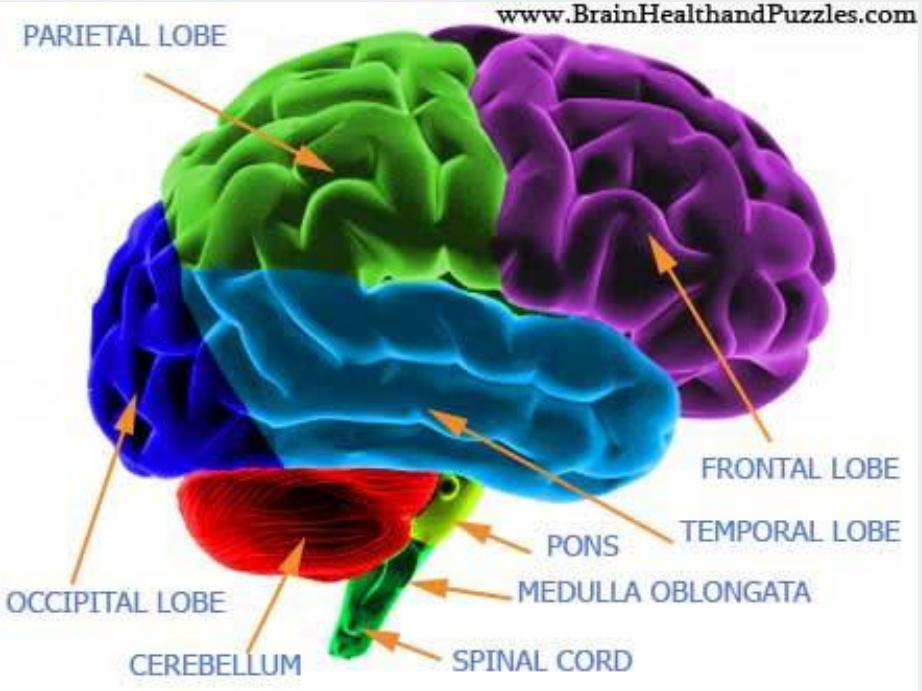
School of Industrial Management Engineering

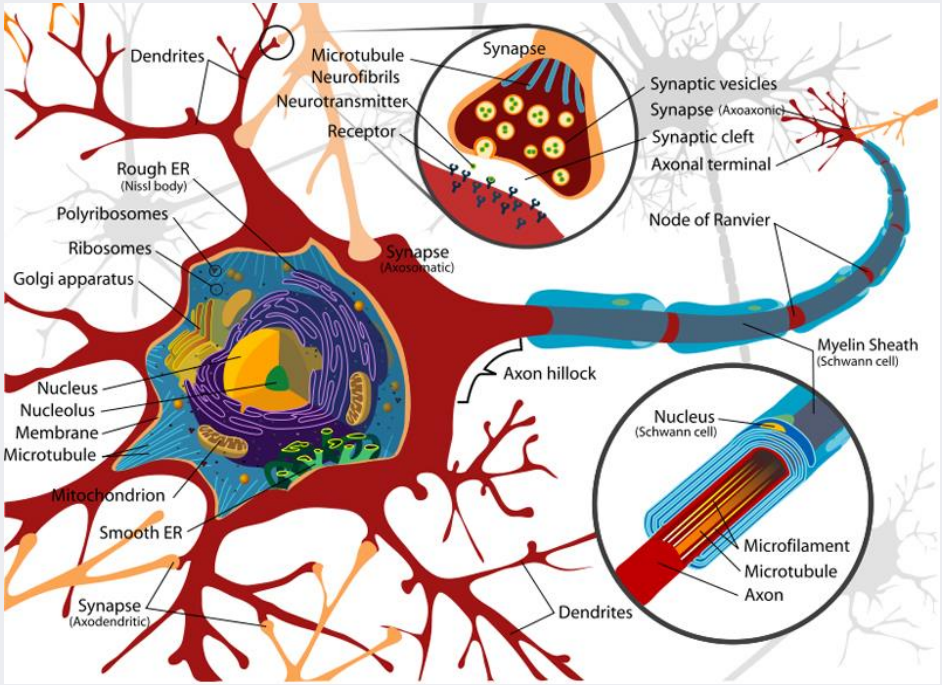
Korea University

AGENDA

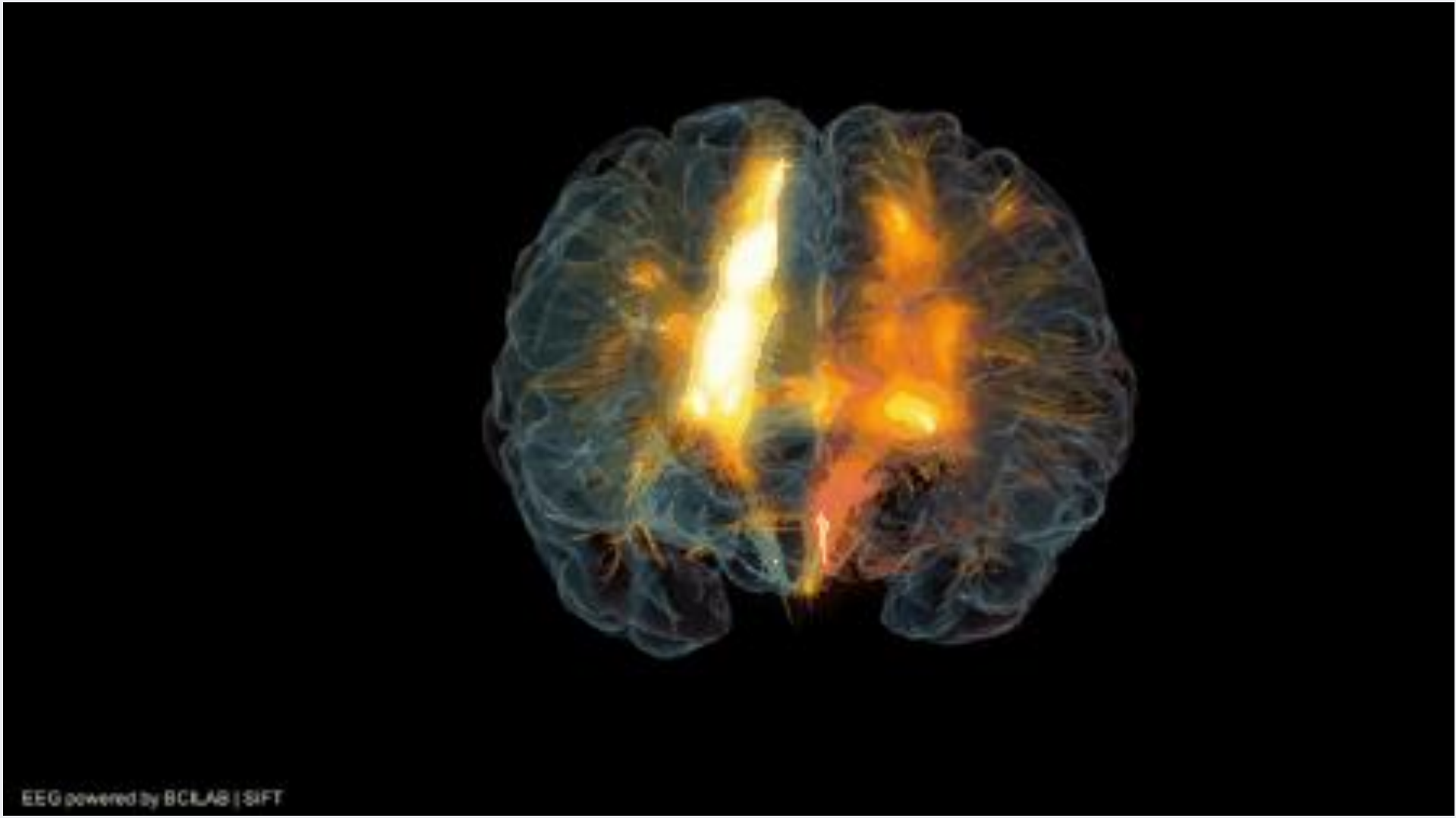
Brain Structure

- How our brain works…
  - ✓ Neurons transmit and analyze communication within the brain and other parts of the nervous system
  - ✓ A message within the brain is converted to electronic signs





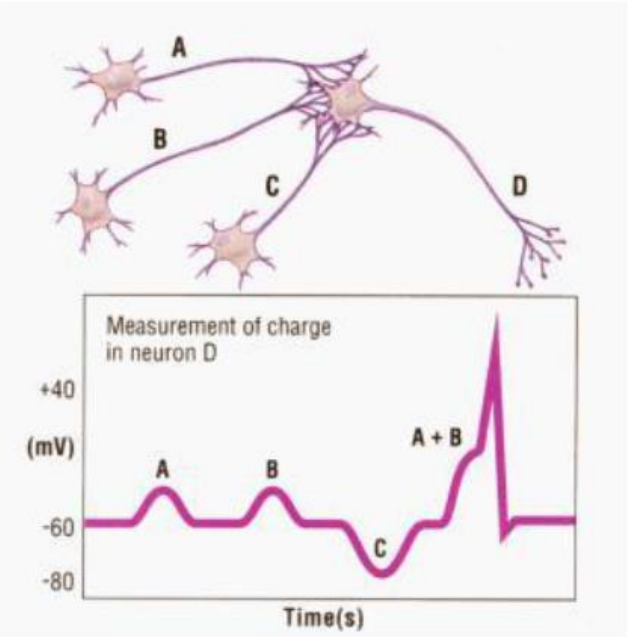
Neuron Firing Off in Real-Time

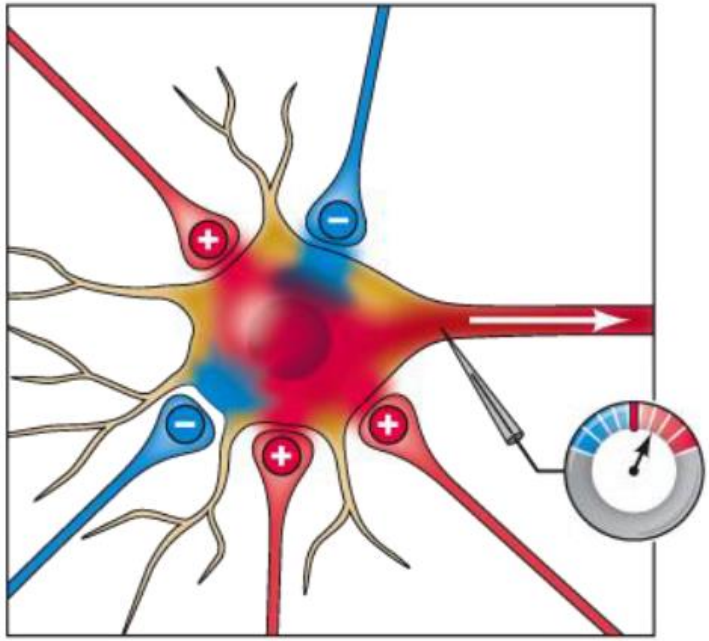


http://www.dailymail.co.uk/sciencetech/article-2581184/The-dynamic-mind-Stunning-3D-glass-brain-shows-neurons-firing-real-time.html

The Way Our Brain Works

- The way neurons work



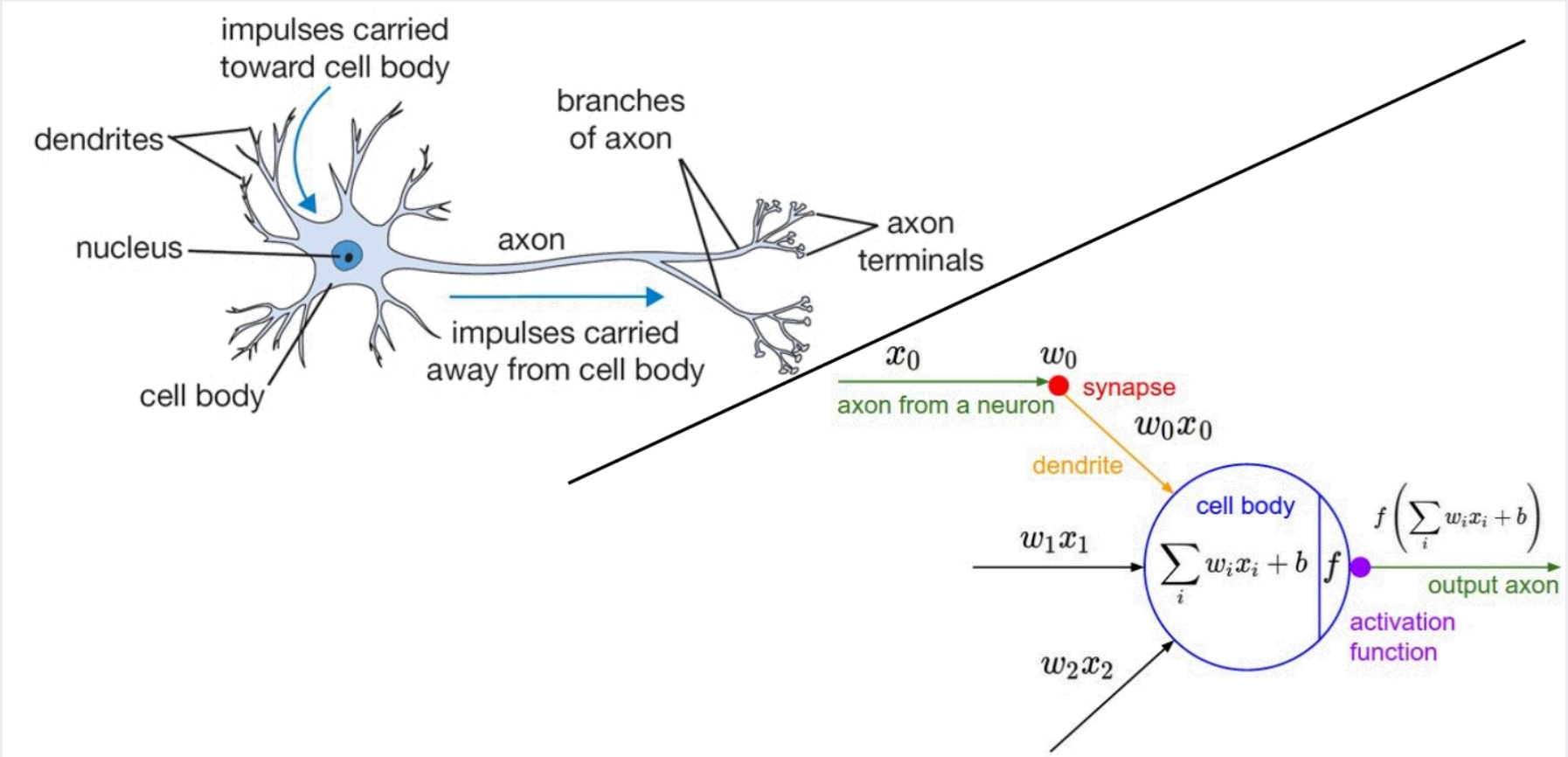


Neurons receive signals

- ✓ Combine those signals – **sum** the stimulus
- ✓ Fire if the stimulus exceed the **threshold**

Perceptron

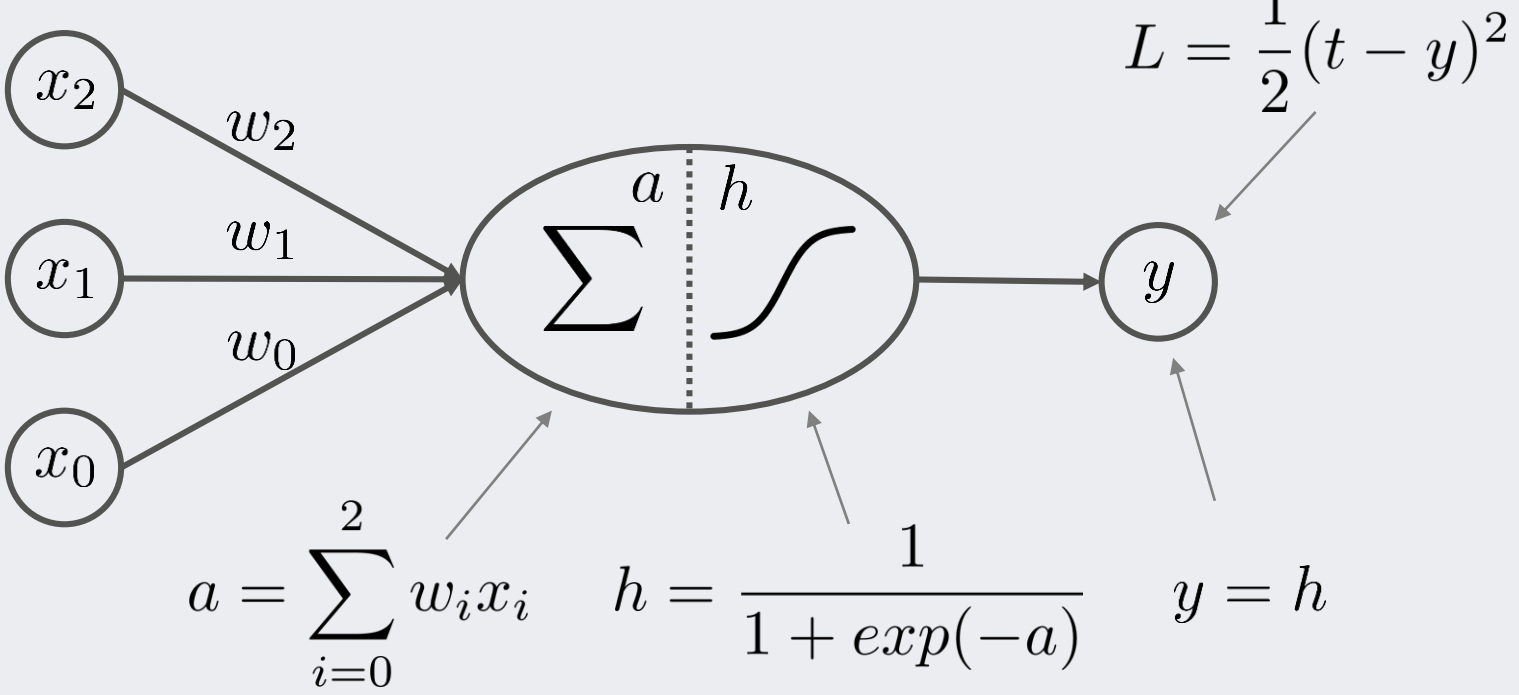
- Imitate a single neuron



Perceptron

- Perceptron
  - ✓ An organism with only 1 neuron

원하는 값(t)과
예측값(y)의 차이를
손실함수로 정의

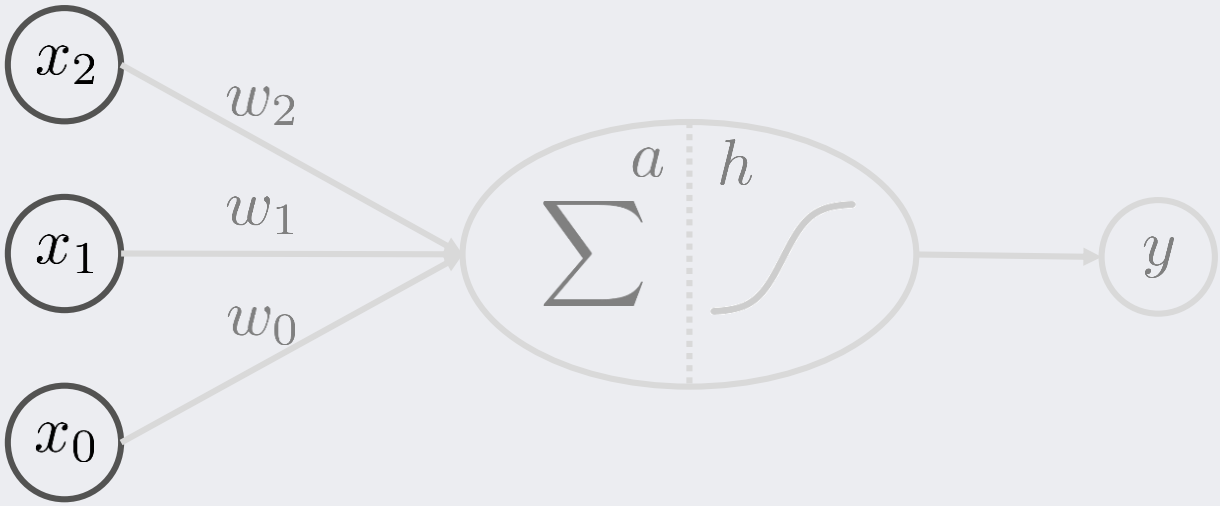


여러 변수들의 정보를
나름대로 취합해서

얼마만큼 다음 단계로
전달할지 결정한다
(활성화)

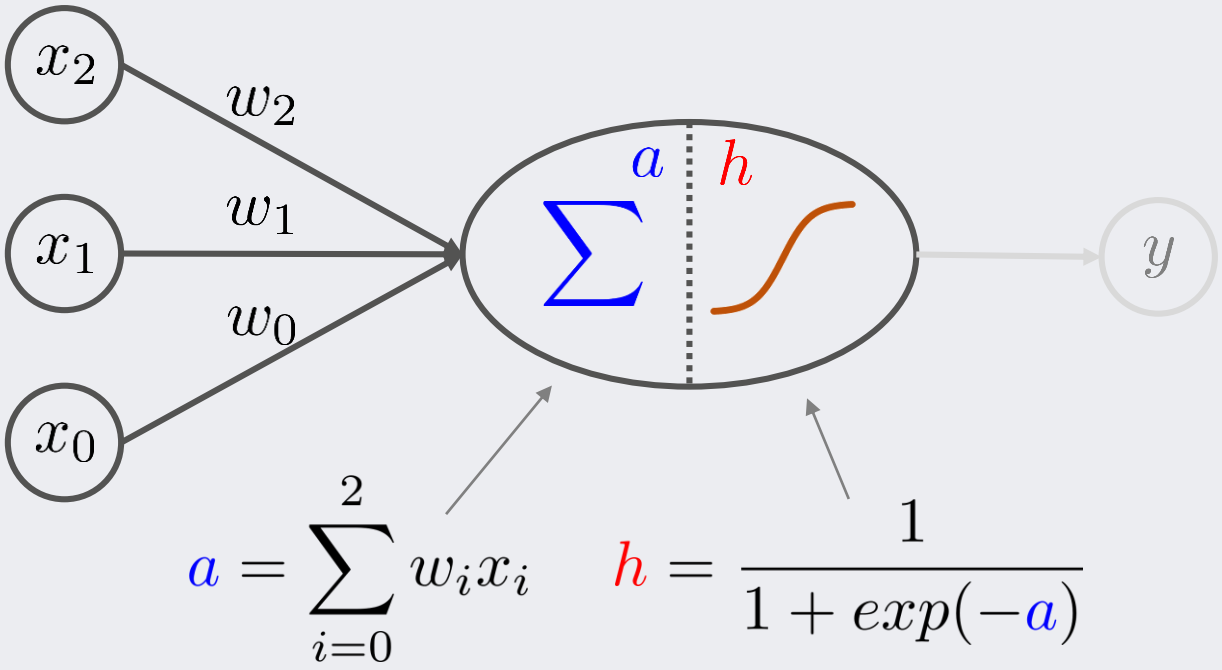
Perceptron

- Input node
  - ✓ Input (predictor, explanatory) variables



Perceptron

- Hidden node
  - ✓ Take the weighted sum of input values and perform a non-linear activation



Perceptron

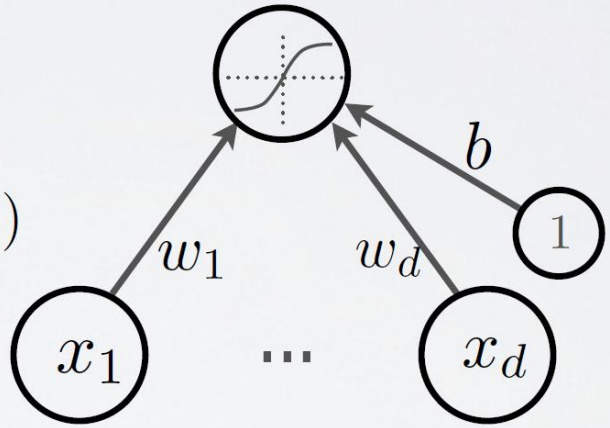
- Role of activation
  - ✓ Determine how much information from the previous layer is forward to the next layer

- Neuron pre-activation (or input activation):

$$a(\mathbf{x}) = b + \sum_i w_i x_i = b + \mathbf{w}^\top \mathbf{x}$$

- Neuron (output) activation

$$h(\mathbf{x}) = g(a(\mathbf{x})) = g(b + \sum_i w_i x_i)$$



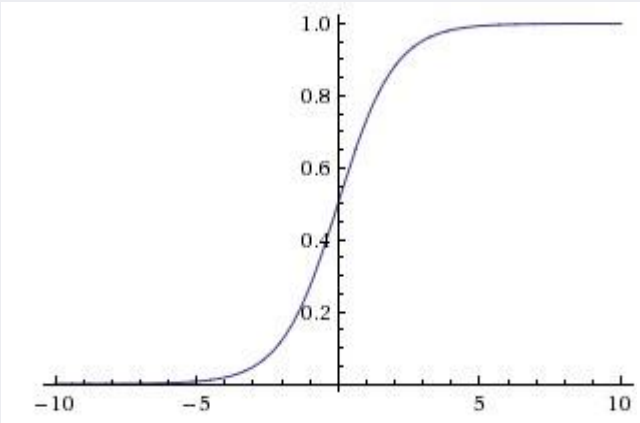
- $\mathbf{w}$ are the connection weights
- b is the neuron bias
- $g(\cdot)$ is called the activation function

http://info.usherbrooke.ca/hlarochelle/cours/ift725_A2013/contenu.html

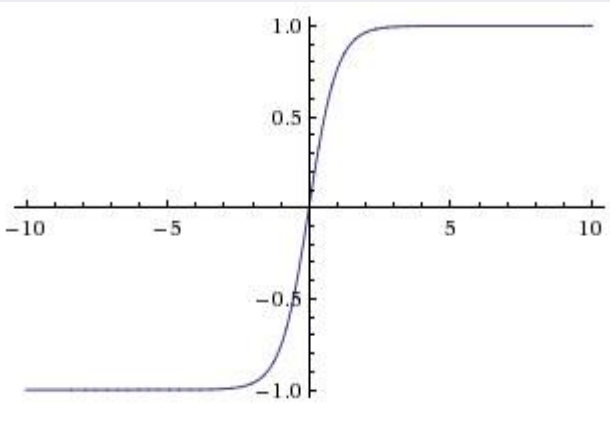
Perceptron

- Representative activation functions
  - ✓ Sigmoid: the most commonly used activation, [0, 1] range, learning speed is relatively slow
  - ✓ Tanh: Similar to sigmoid but [-1, 1] range, learning speed is relatively fast
  - ✓ ReLU (Rectified linear unit): very fast learning speed, easy to compute (without exponential function)

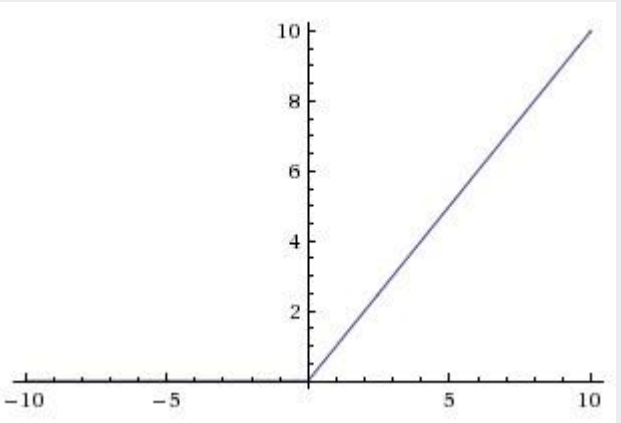
[Sigmoid]



[Tanh]



[ReLU]



$$g(a) = \text{sigm}(a) = \frac{1}{1+\exp(-a)}$$

$$g(a) = \tanh(a) = \frac{\exp(a)-\exp(-a)}{\exp(a)+\exp(-a)} = \frac{\exp(2a)-1}{\exp(2a)+1}$$

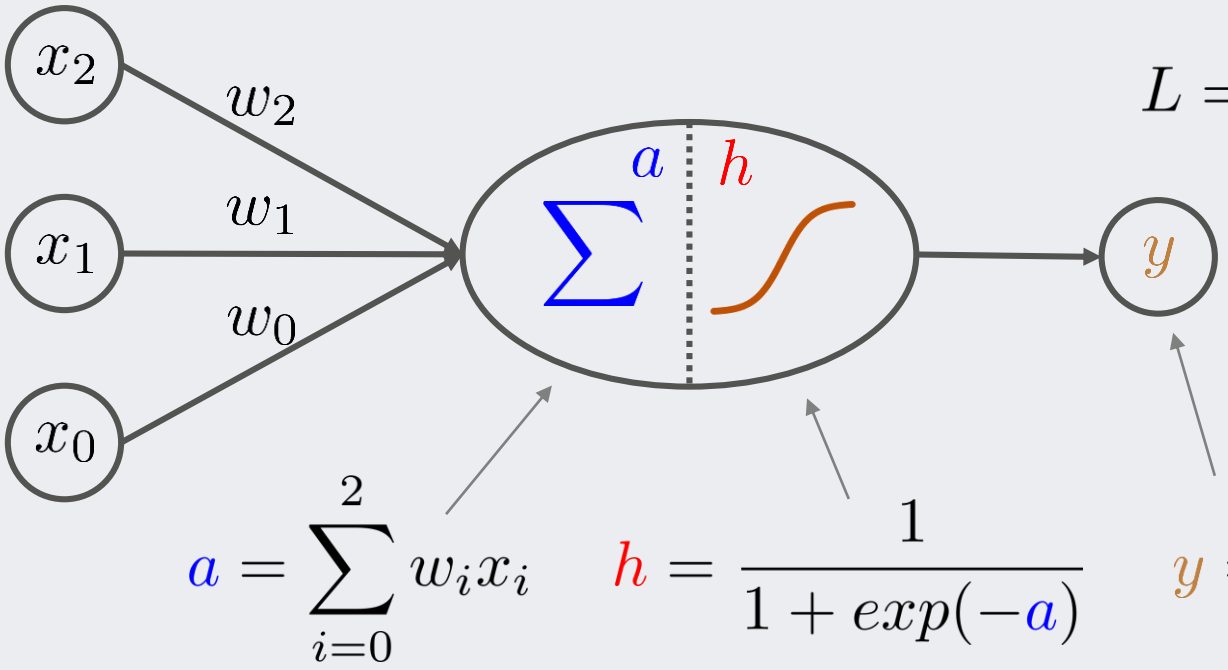
$$g(a) = \text{reclin}(a) = \max(0, a)$$

Perceptron

- Output node
  - ✓ Take the value from the hidden node (Perceptron has only one hidden node)
  - ✓ It takes a weighted sum of hidden nodes in a multi-layer perceptron

원하는 값(t)과 예측값(y)의 차이를 손실함수로 정의

$$L = \frac{1}{2}(t - y)^2$$



$$a = \sum_{i=0}^{2} w_i x_i$$

여러 변수들의 정보를 나름대로 취합해서

$$h = \frac{1}{1 + exp(-a)}$$

얼마만큼 다음 단계로 전달할지 결정한다 (활성화)

$$y = h$$

Perceptron

- Purpose of perceptron
  - ✓ Find the weight w that can best match the input (x) and the target (t)
- How do we know that the relationship is accurately found?
  - ✓ Use a loss function (how the output y is close to the target t)
    - Regression: squared loss is commonly used

$$L = \frac{1}{2}(t - y)^2$$

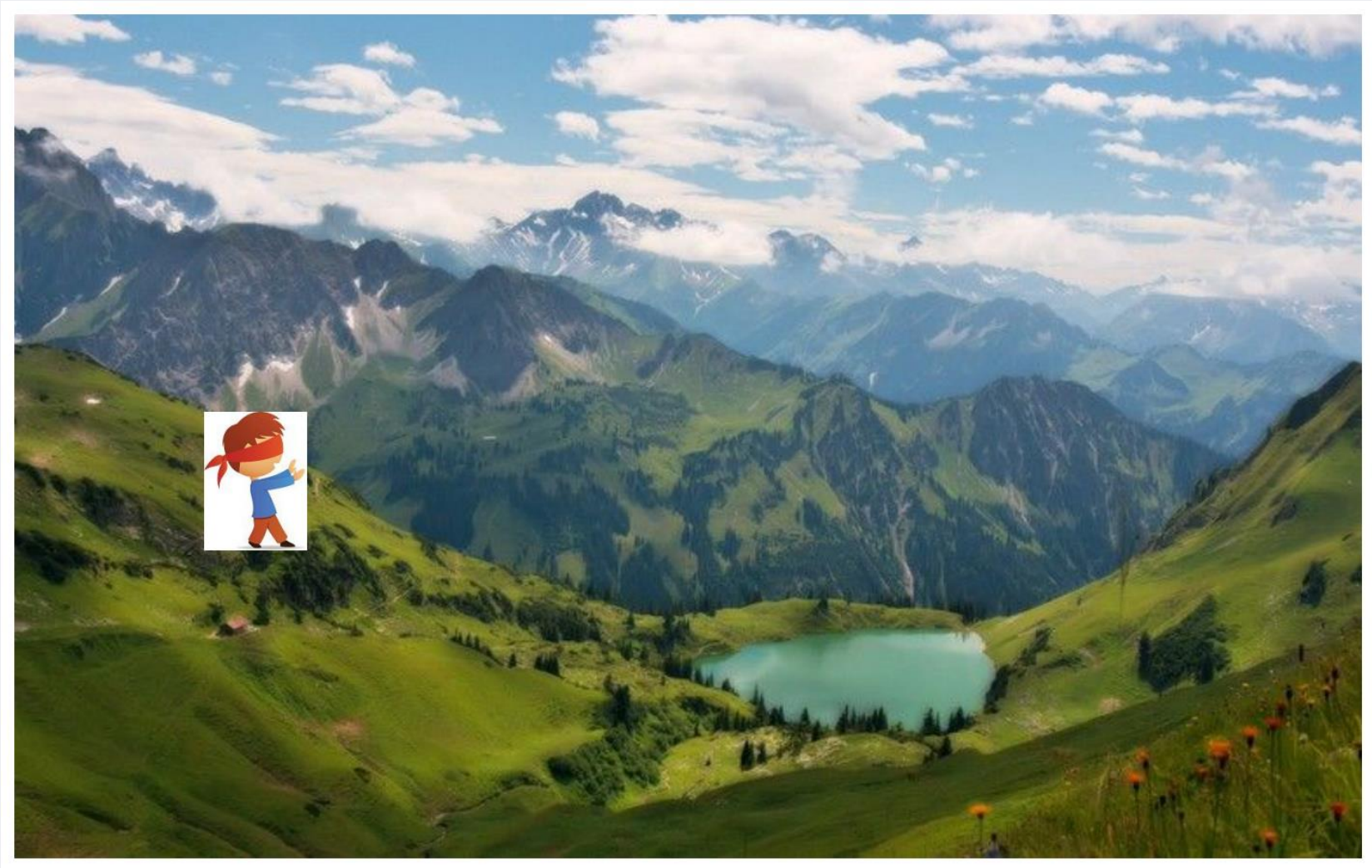
    - Classification: cross-entropy is usually used (only binary classification is possible with perceptron)

$$L = \sum_{i=1}^{2} t_i \log p_i$$

  - ✓ Cost function
    - Computes how inaccurate the current model is (the average of loss function values is commonly used)

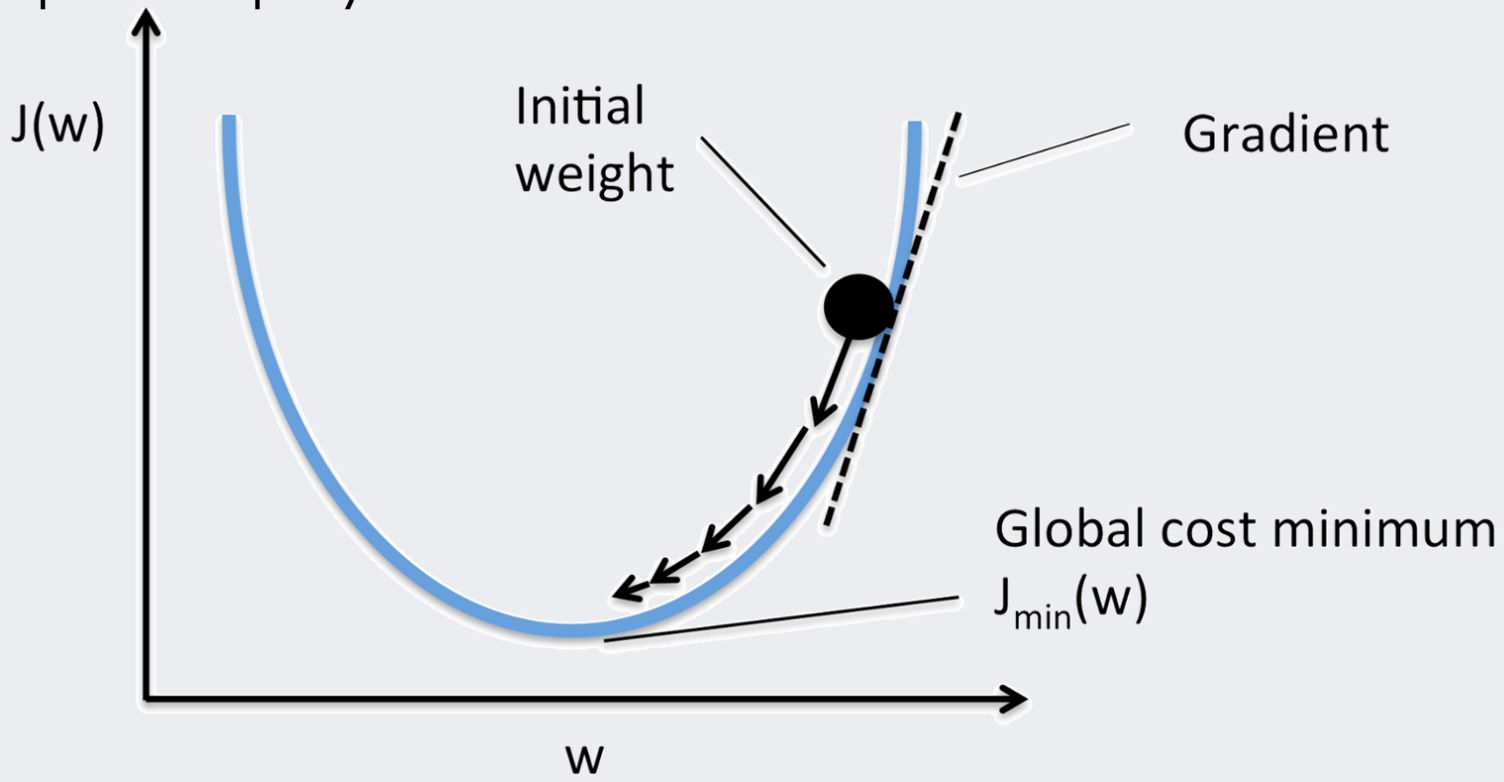
Learning: Gradient Descent

- Gradient Descent



Learning: Gradient Descent

- Gradient Descent Algorithm
  - ✓ Blue line: the objective function to be minimized
  - ✓ Black circle: the current solution
  - ✓ Direction of the arrows: the direction that the current solution should move to improve the quality of solution



Learning: Gradient Descent

- Take the first derivative of the cost function w.r.t the current weight w
  - ✓ Is the gradient 0?
    - Yes: Current weights are the optimum! → end of learning
    - No: Current weights can be improved → learn more
  - ✓ How can we improve the current weights if the gradient is not 0?
    - Move the current weight toward to the opposite direction of the gradient
  - ✓ How much should the weights be moved?
    - Not sure
    - Move them a little and compute the gradient again
    - It will converge

Learning: Gradient Descent

- Theoretical Background
  - ✓ Taylor expansion

$$f(x+\Delta x) = f(x) + \frac{f'(x)}{1!}\Delta x + \frac{f''(x)}{2!}\Delta x^2 + \cdots$$

  - ✓ If the first derivative is not zero, we can decrease the function value by moving x toward the opposite direction of its first derivative

$$x_{new} = x_{old} - \eta f'(x), \quad where \;\; 0 < \eta < 1$$

  - ✓ Then the function value of the new x is always smaller than that of the old x

$$f(x_{new}) = f(x_{old} - \eta f'(x)) \cong f(x_{old}) - \eta |f'(x)|^2 < f(x_{old})$$

https://www.youtube.com/watch?v=3d6DsjIBzJ4&t=322s

Learning: Gradient Descent

- Use chain rule

$$\frac{\partial L}{\partial y} = y - t$$

$$\frac{\partial y}{\partial h} = \frac{exp(-h)}{(1 + exp(-h))^2} = \frac{1}{1 + exp(-h)} \cdot \frac{exp(-h)}{1 + exp(-h)} = y(1 - y)$$

$$\frac{\partial h}{\partial w_i} = x_i \qquad \frac{\partial h}{\partial x_i} = w_i$$

- Gradients for w and x

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial h} \cdot \frac{\partial h}{\partial w_i} = (y - t) \cdot y(1 - y) \cdot x_i$$

$$\frac{\partial L}{\partial x_i} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial h} \cdot \frac{\partial h}{\partial x_i} = (y - t) \cdot y(1 - y) \cdot w_i$$

Learning: Gradient Descent

- Weight update by Gradient Descent

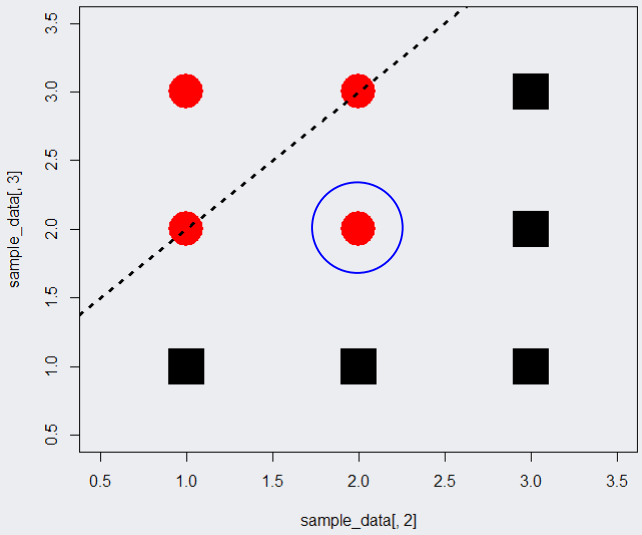
$$w_i^{new} = w_i^{old} - \alpha \times (y - t) \cdot 1 \cdot h(1 - h) \cdot x_i$$

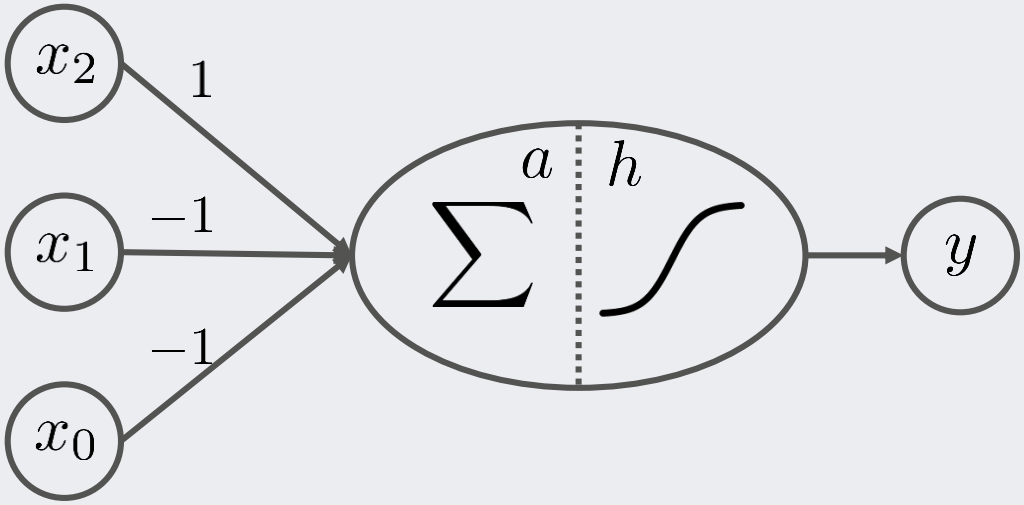
현재의 출력값(y)과 정답(t)이
차이가 많이 날 수록
가중치를 많이 업데이트 하라

대상 가중치와 연결된 입력
변수의 값이 클 수록
가중치를 많이 업데이트 하라

Training Perceptron: Example 1 (alpha = 1)

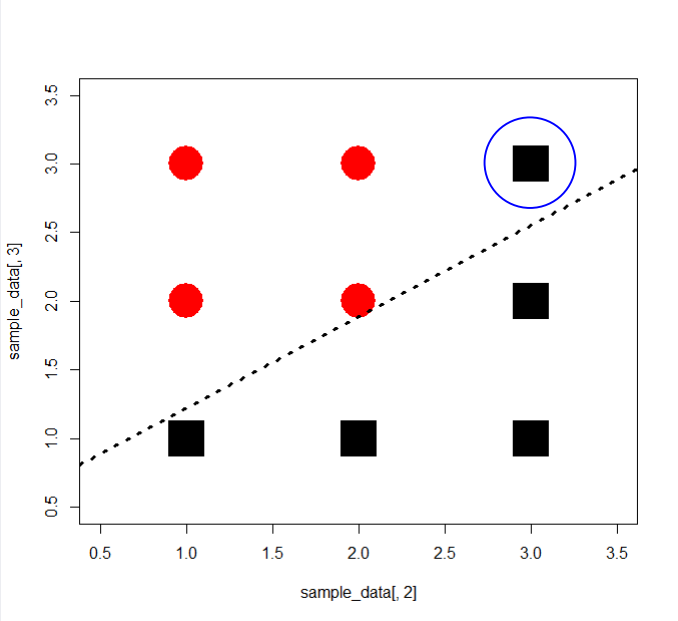
- Initialize and select the first training example (x_1=2, x_2=2, t=1)

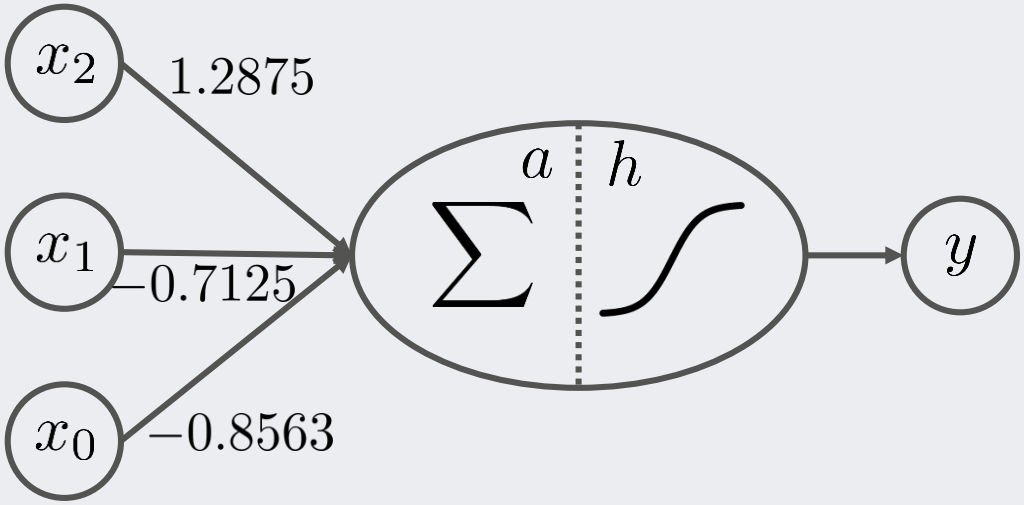




Training Perceptron: Example 1 (alpha = 1)

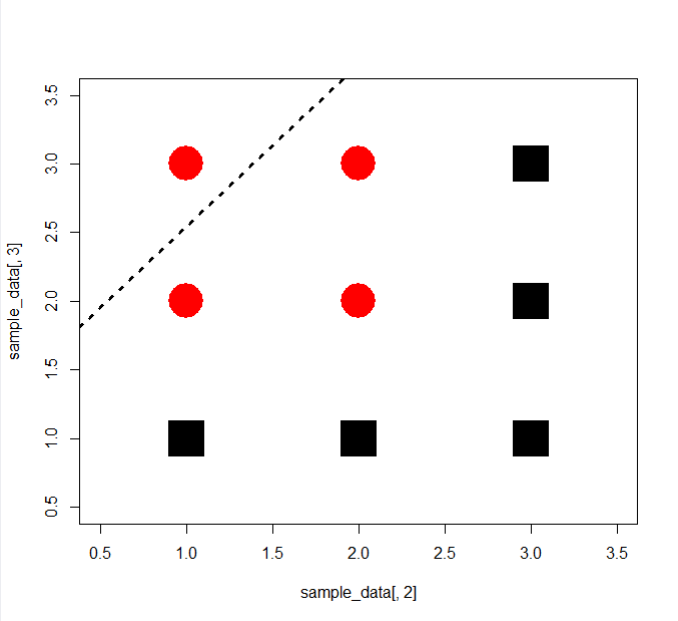
- Training result and selection of the second training example (x_1=3, x_2=3, t=0)

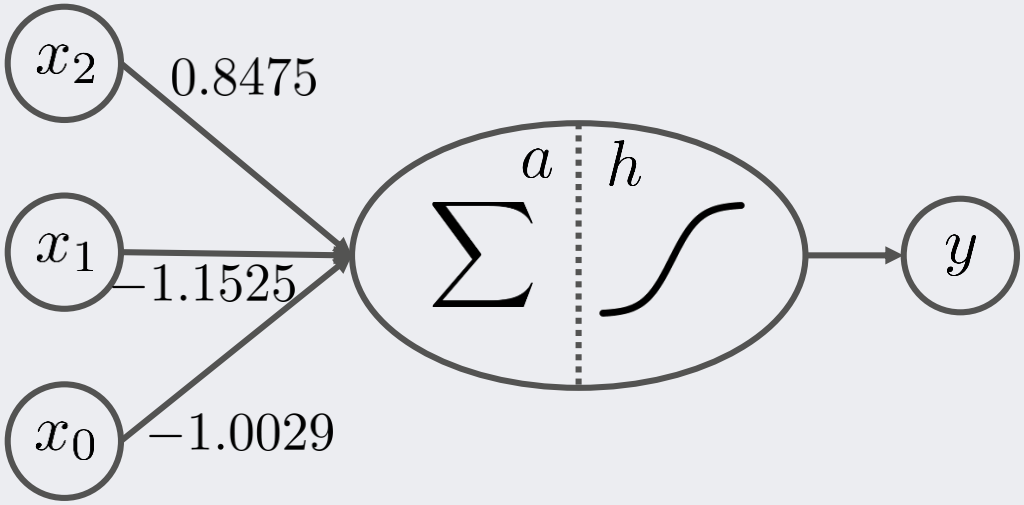




Training Perceptron: Example 1 (alpha = 1)

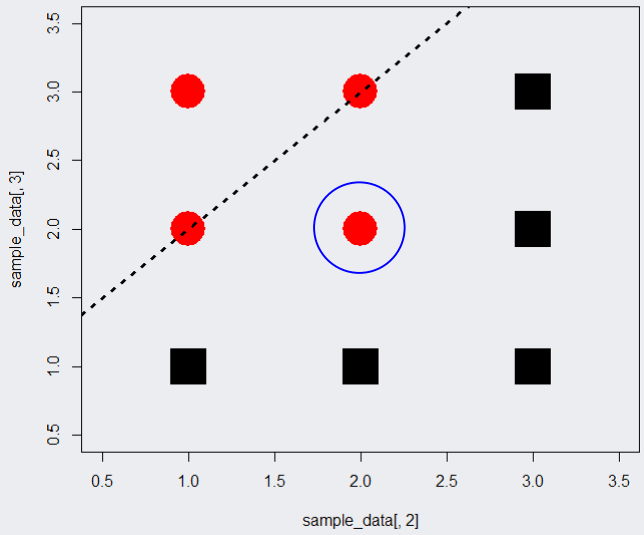
- Training result

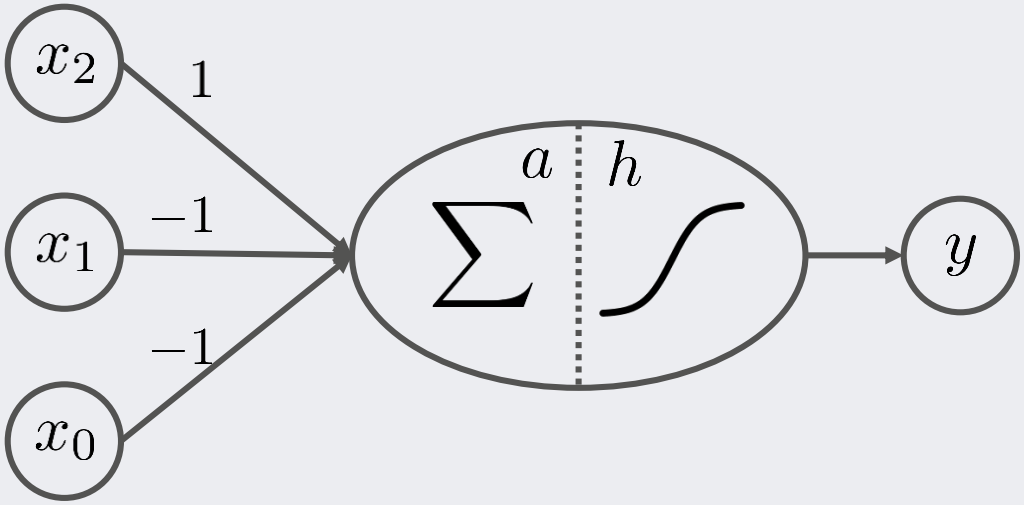




Training Perceptron: Example 1 (alpha = 0.5)

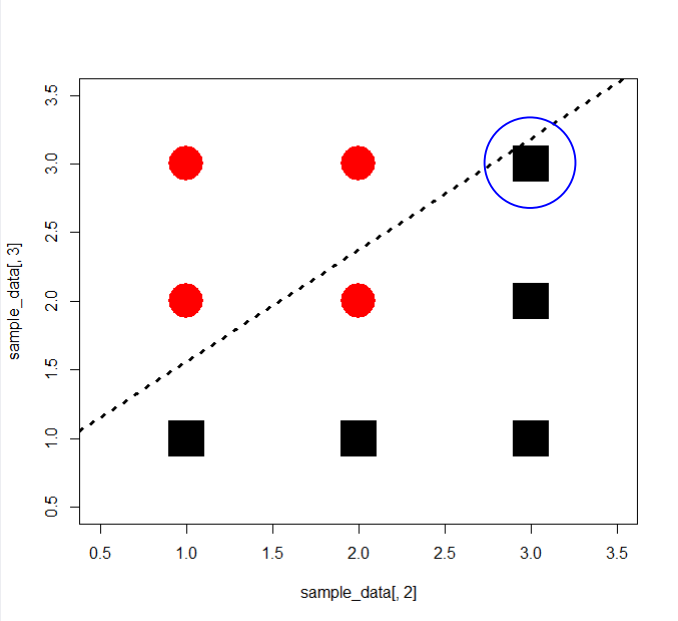
- Initialize and select the first training example (x_1=2, x_2=2, t=1)

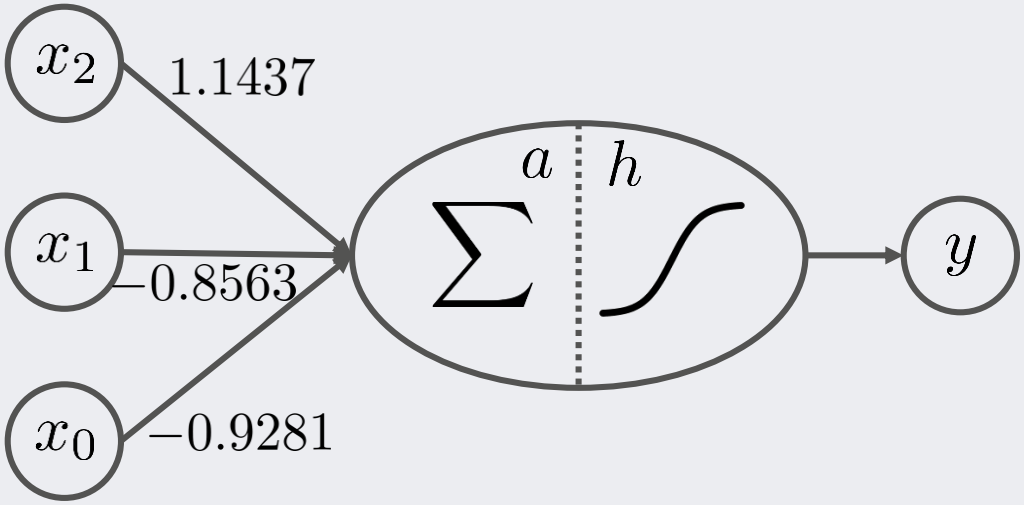




Training Perceptron: Example 1 (alpha = 0.5)

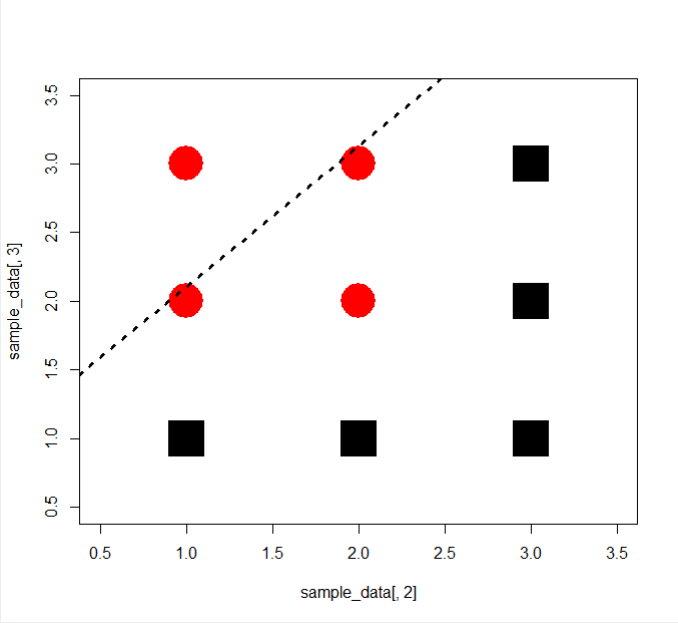
- Training result and selection of the second training example (x_1=3, x_2=3, t=0)

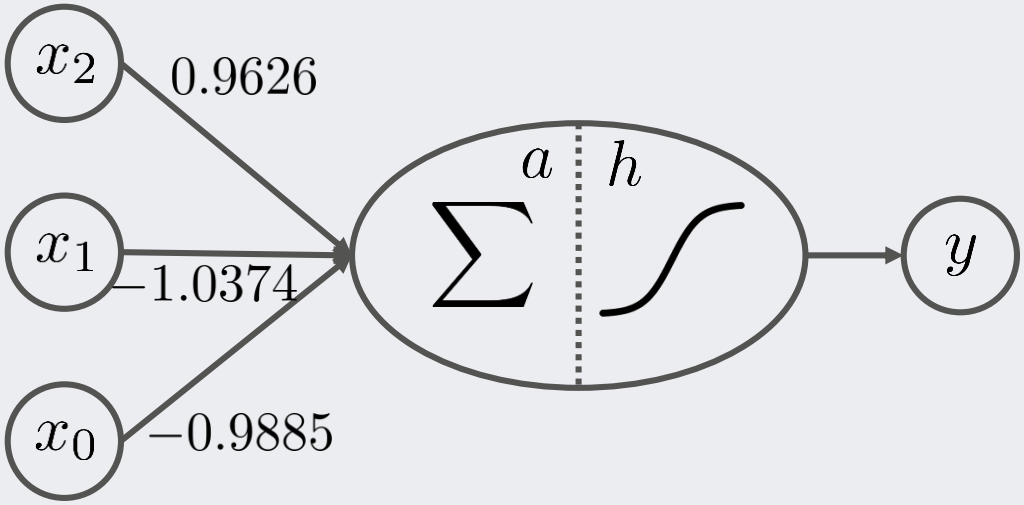




Training Perceptron: Example 1 (alpha = 0.5)

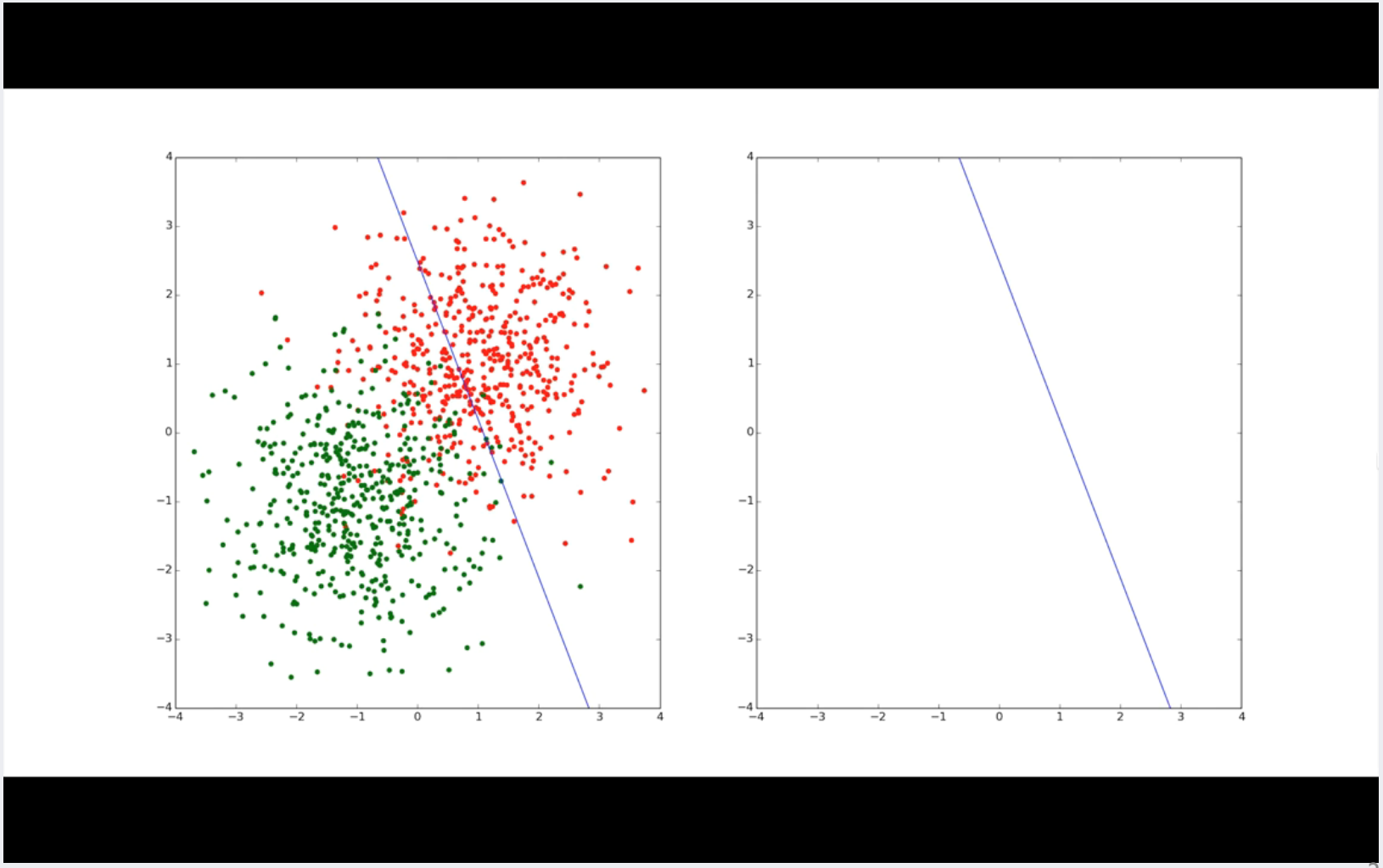
- Training result





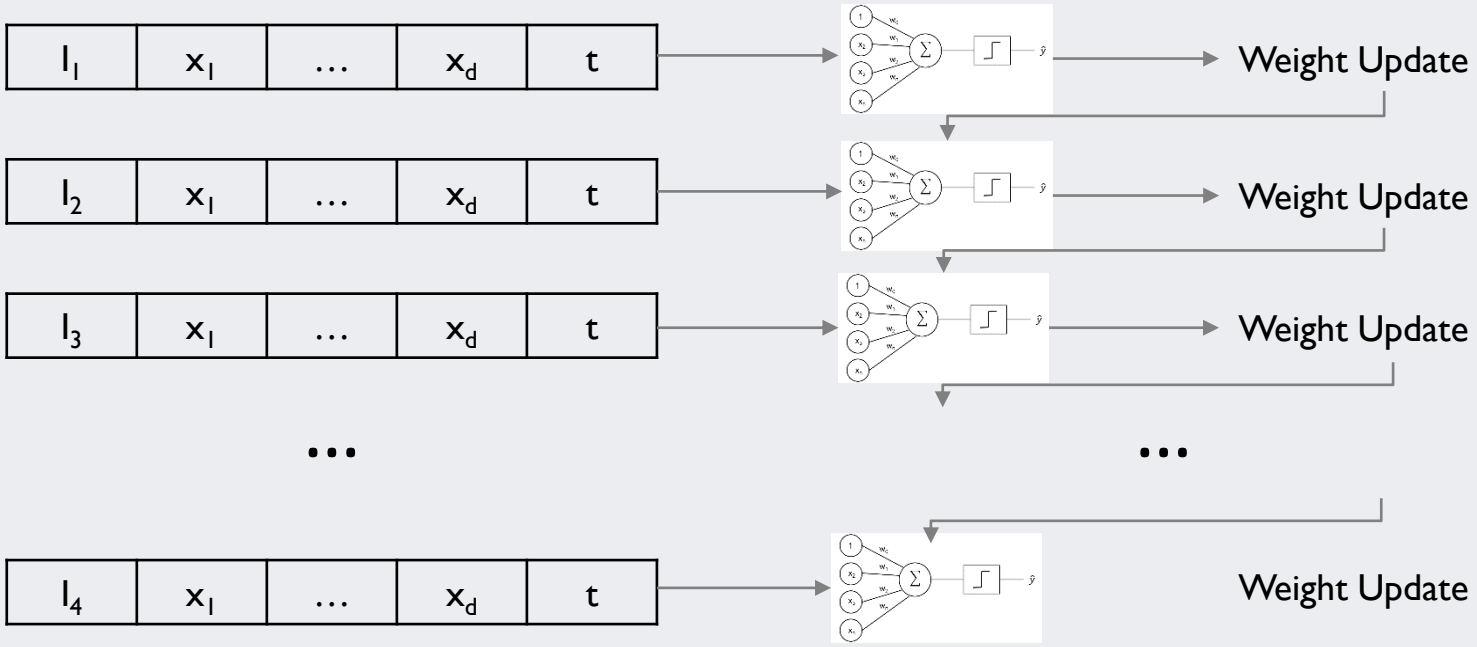
Perceptron

- Training Perceptron



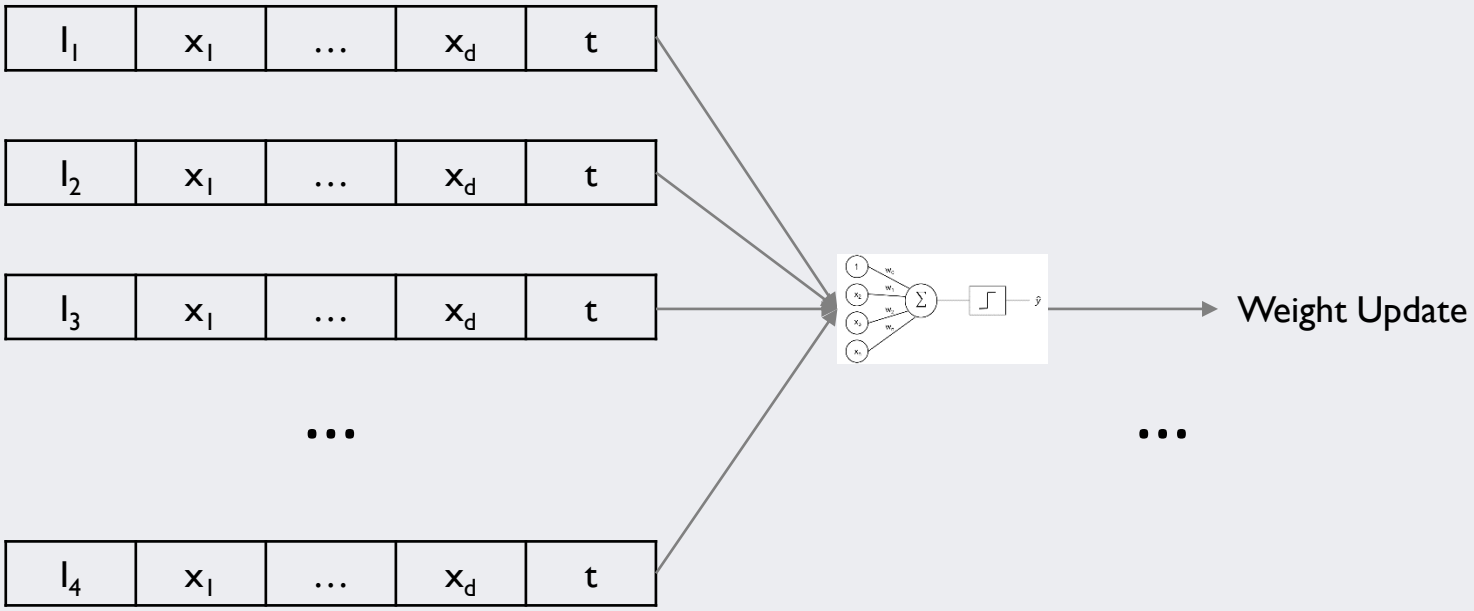
Issues on Gradient Descent

- Issue 1: How frequently should the weights be updated?
  - ✓ Stochastic Gradient Descent (SGD)
    - Compute the loss function for an individual training example and update the gradients



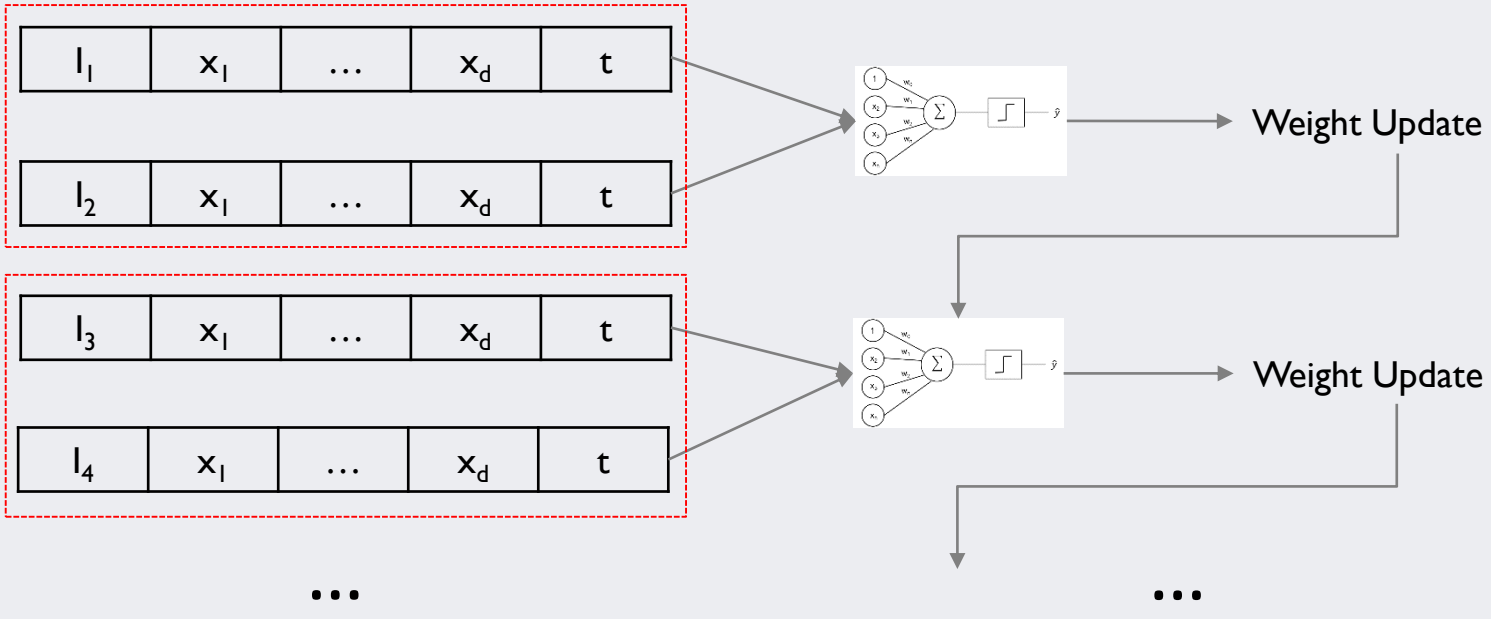
Issues on Gradient Descent

- Issue 1: How frequently should the weights be updated?
  - ✓ Batch Gradient Descent (BGD)
    - Fix the network weights, compute the cost function using all training examples, compute the gradient, and update the weights



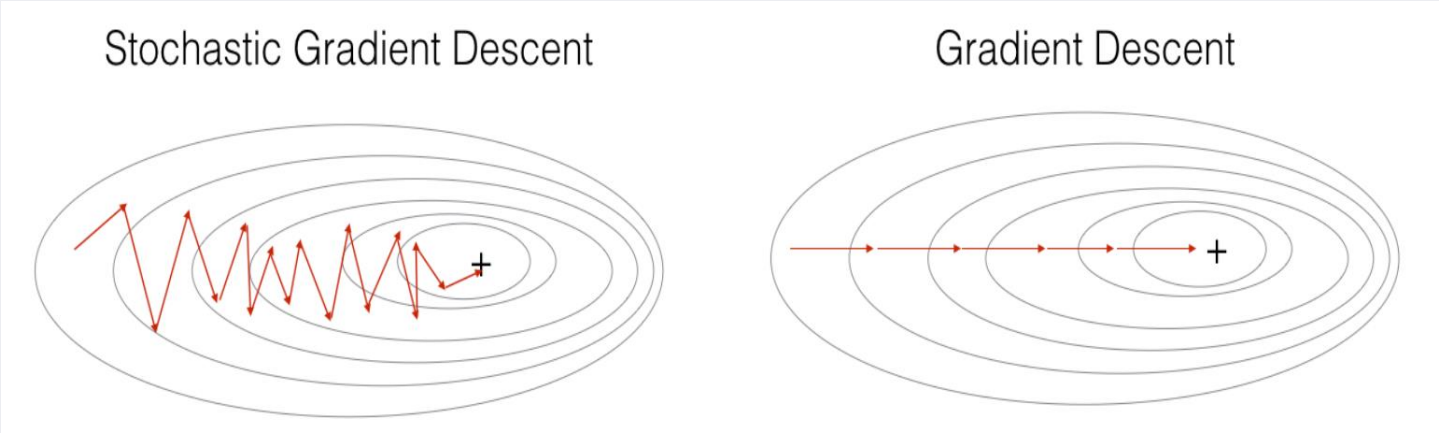
Issues on Gradient Descent

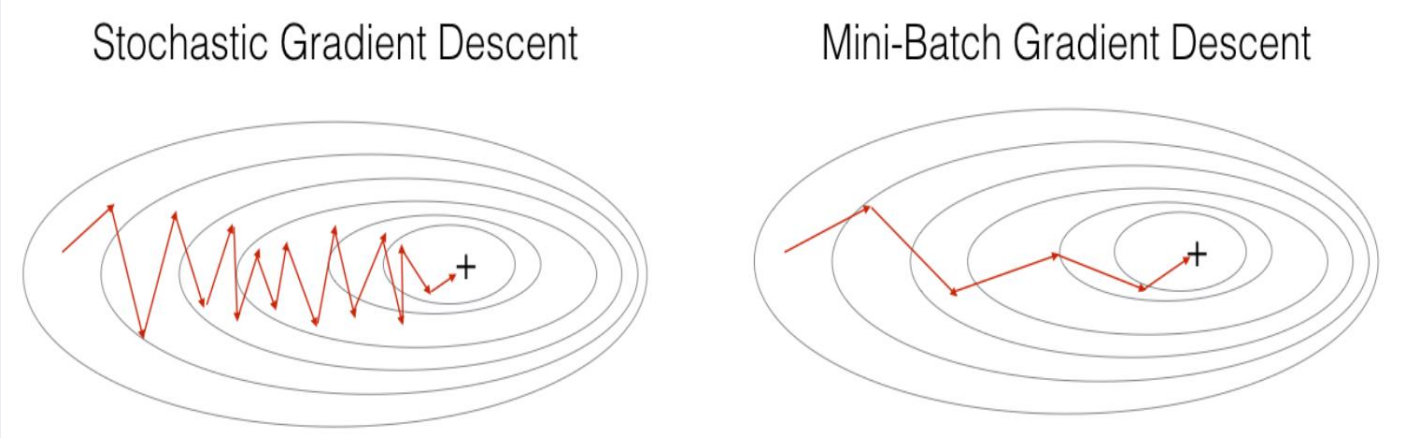
- Issue 1: How frequently should the weights be updated?
  - ✓ Mini-Batch Gradient Descent
    - A strategy between SGD and BGD
    - Construct a mini-batch with n examples from N training examples and compute the gradient using the n examples (when the mini-batch size is set to 2)



Issues on Gradient Descent

- Issue 1: How frequently should the weights be updated?

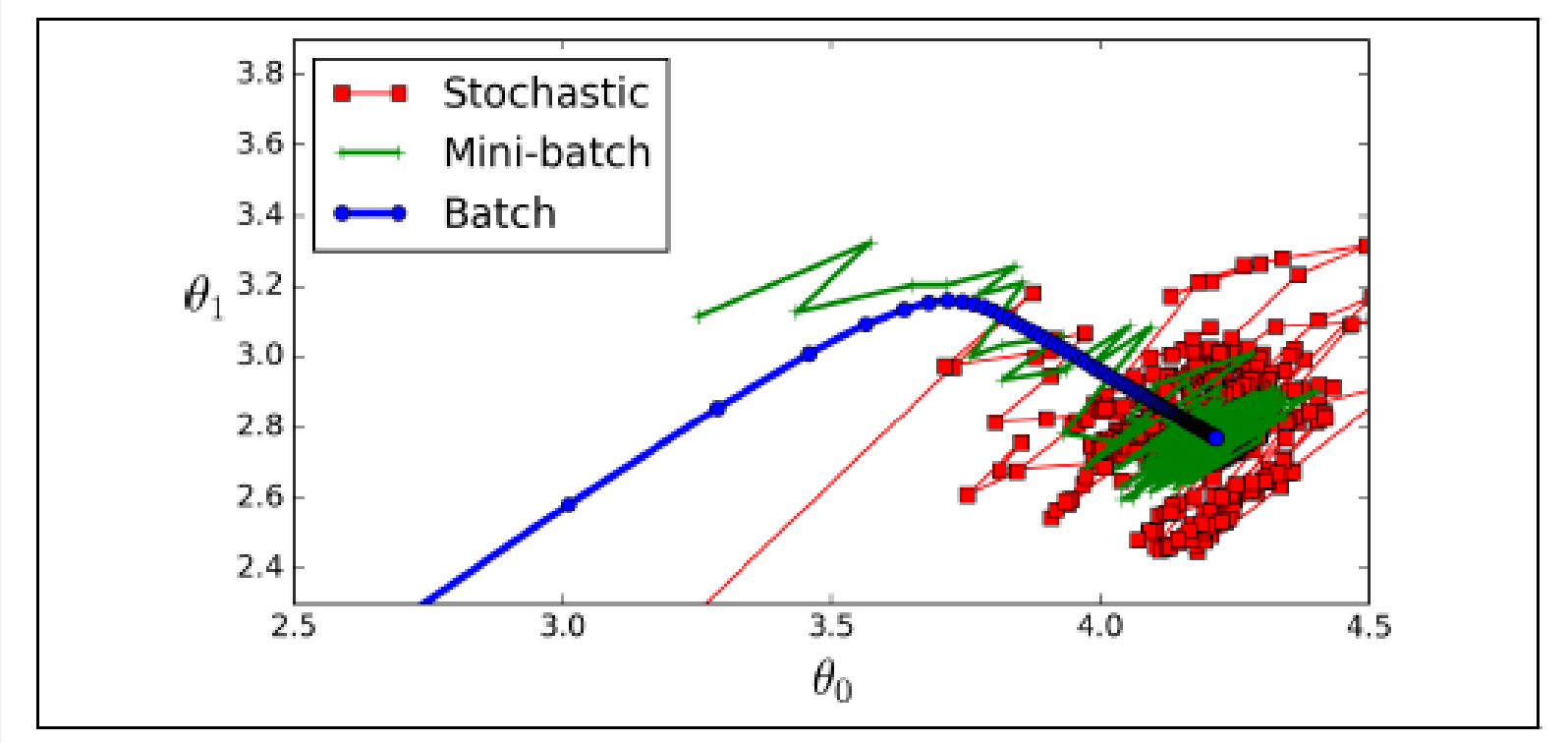




http://pengcheng.tech/2017/09/28/gradient-descent-momentum-and-adam/

Issues on Gradient Descent

- Issue 1: How frequently should the weights be updated?

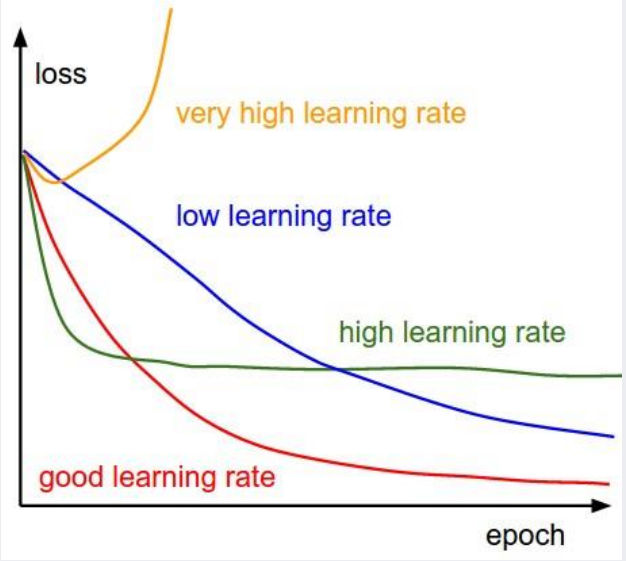


Issues on Gradient Descent

- Issue 2: How much should the weights be updated?
  - ✓ A problem of deciding learning rate

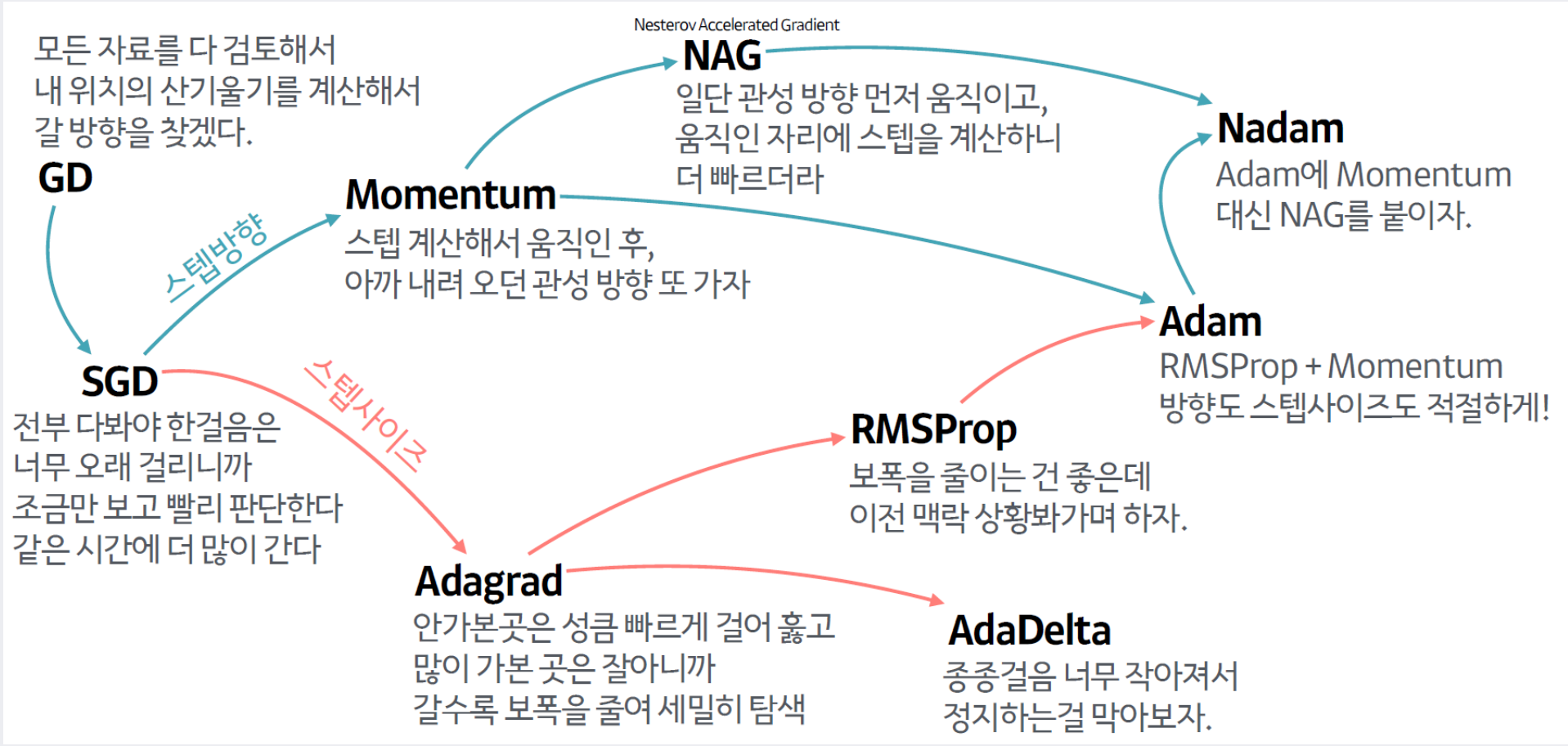
$$w_{new} = w_{old} - \alpha f'(w)$$

    - If the learning rate is too large, the network will not converge
    - If the learning rate is too small, it takes too long time to converge
  - ✓ An appropriate (?) learning rate is required



Issues on Gradient Descent

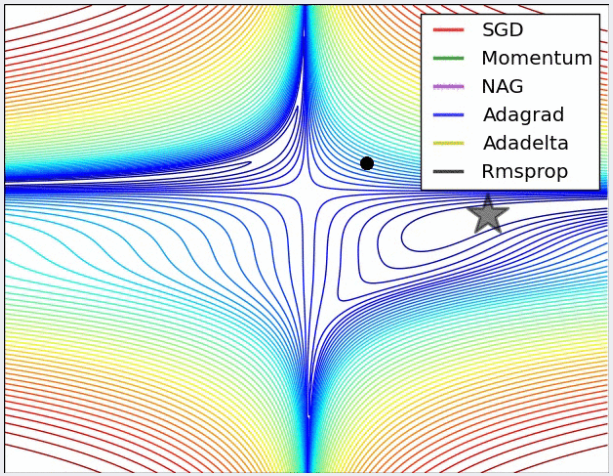
- Issue 2: How much should the weights be updated?



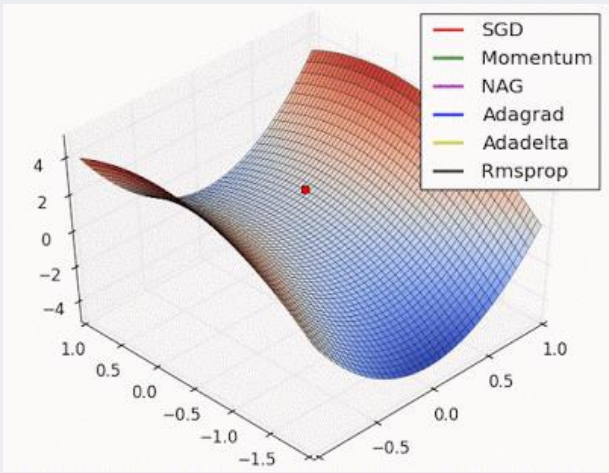
하용호. 자습해도 모르겠던 딥러닝, 머리속에 인스톨 시켜드립니다. (https://www.slideshare.net/yongho/ss-79607172)

Issues on Gradient Descent

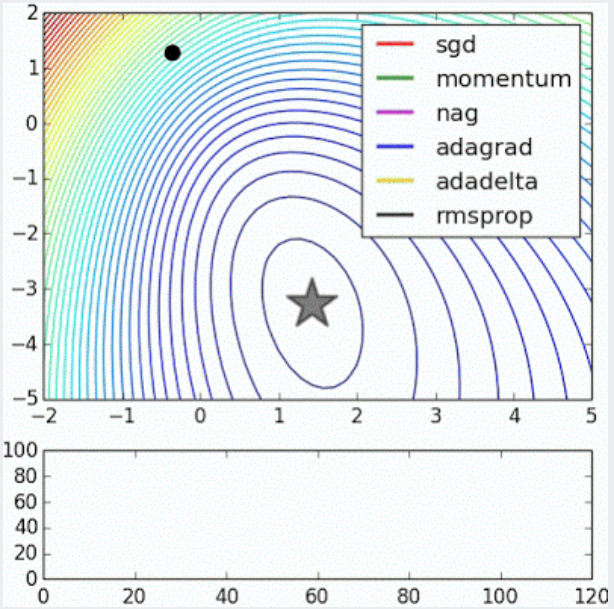
- Issue 2: How much should the weights be updated?



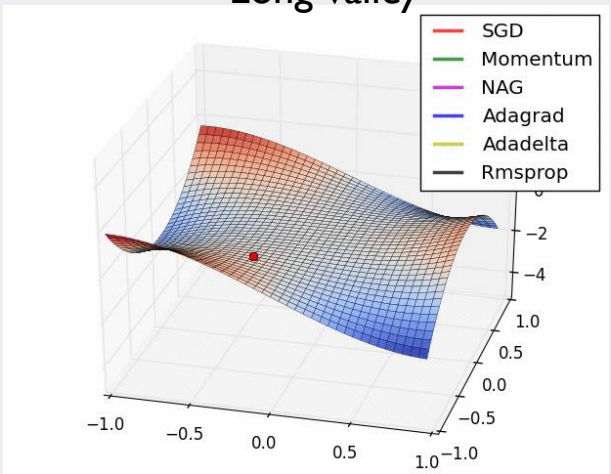
Beale's function



Long valley



Noisy moons



Saddle point

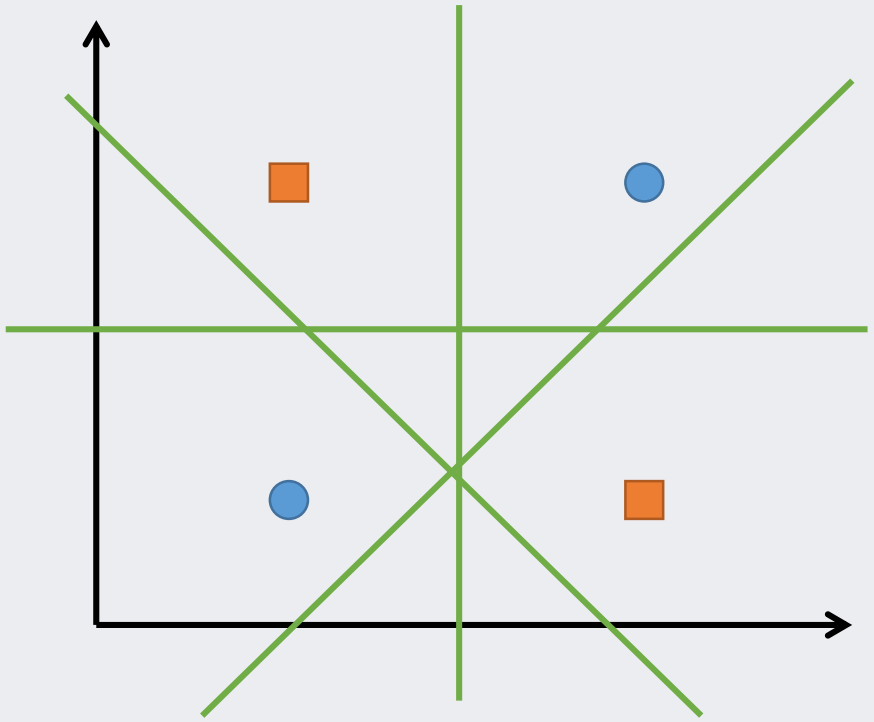
http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html

Limitation of Perceptron

- The Limitation of Linear Models
  - ✓ **Classification:**
    - Linear (Fisher) discriminant analysis, logistic regression, etc.
    - Can only produce a linear class boundary
  - ✓ **Regression:**
    - Multiple linear regression
    - Can only capture the linear relationship between the predictors and the outcome
  - ✓ Cannot results in good prediction performance ***when the classification boundary or the predictor/outcome relationship is not linear***

Limitation of Perceptron

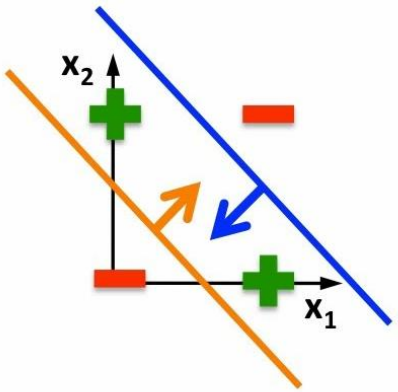
- The Limitation of Linear Models
  - ✓ Draw a straight line that perfectly separates the circles and crosses (XOR)

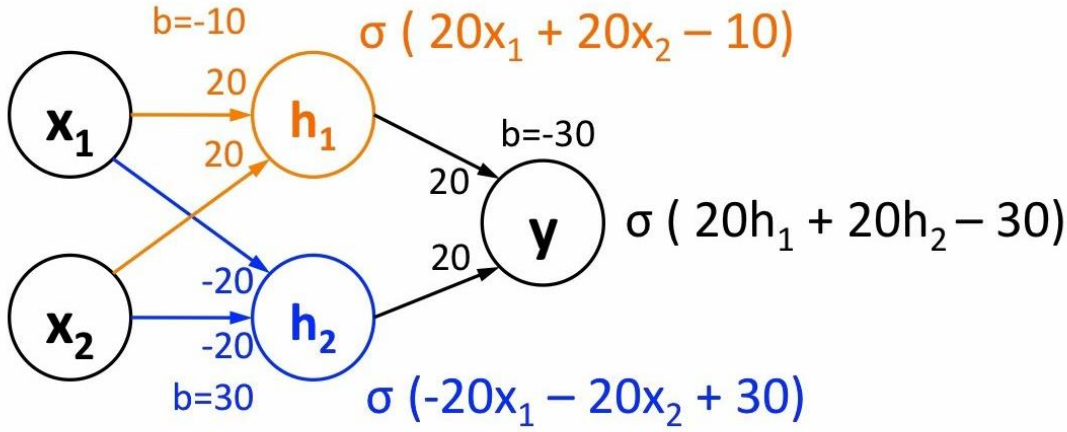


Multi-Layer Perceptron (MLP)

- Combine multiple perceptrons!
  - ✓ If we cannot solve a complex problem directly, then it is better to decompose it into some small and simple problems that can be solved!

Linear classifiers cannot solve this





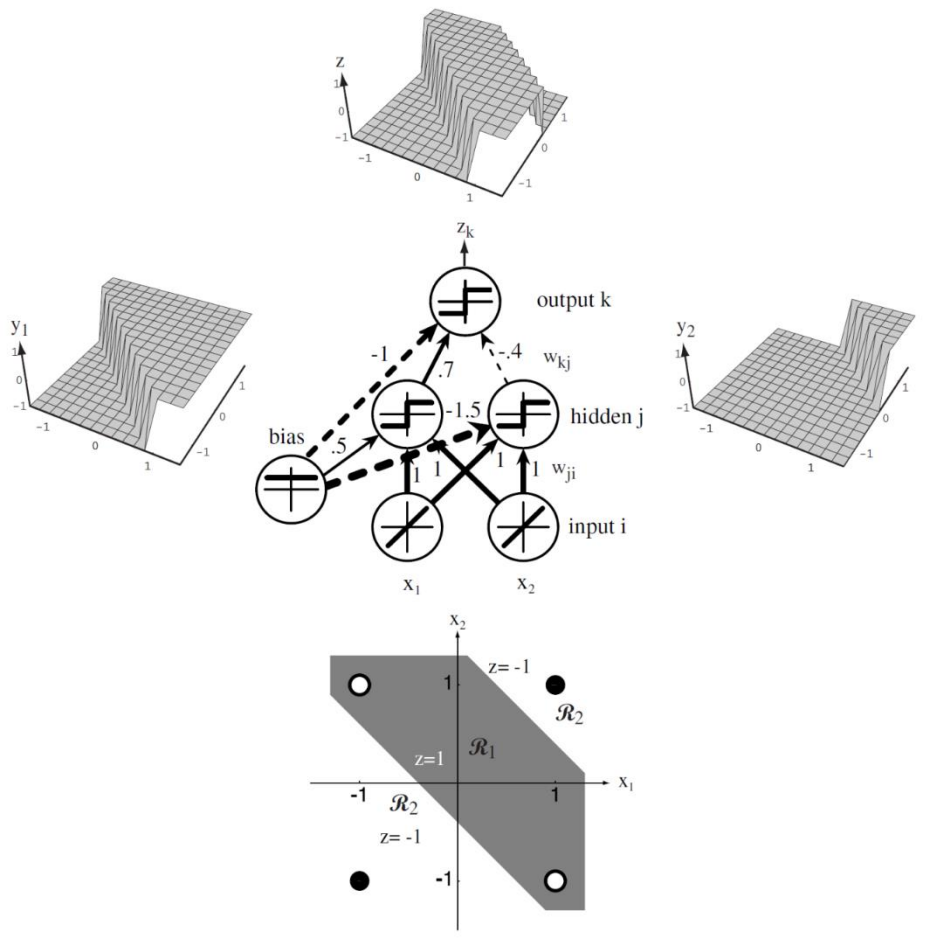
σ(20*0 + 20*0 – 10) ≈ 0
σ(20*1 + 20*1 – 10) ≈ 1
σ(20*0 + 20*1 – 10) ≈ 1
σ(20*1 + 20*0 – 10) ≈ 1

σ (-20*0 – 20*0 + 30) ≈ 1
σ (-20*1 – 20*1 + 30) ≈ 0
σ (-20*0 – 20*1 + 30) ≈ 1
σ (-20*1 – 20*0 + 30) ≈ 1

σ (20*0 + 20*1 – 30) ≈ 0
σ (20*1 + 20*0 – 30) ≈ 0
σ (20*1 + 20*1 – 30) ≈ 1
σ (20*1 + 20*1 – 30) ≈ 1

Multi-Layer Perceptron (MLP)

- Non-linear model
  - ✓ Can find an arbitrary shape of class boundary or regression functions

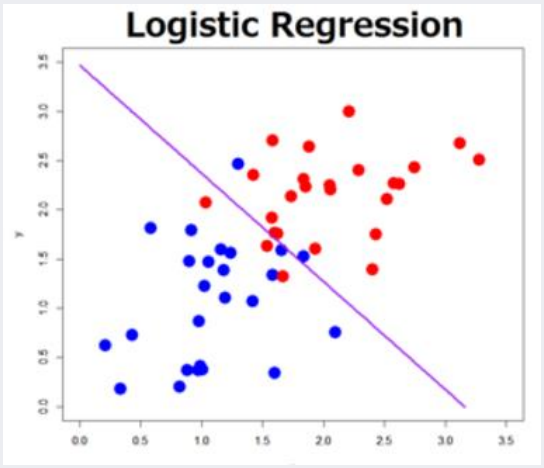


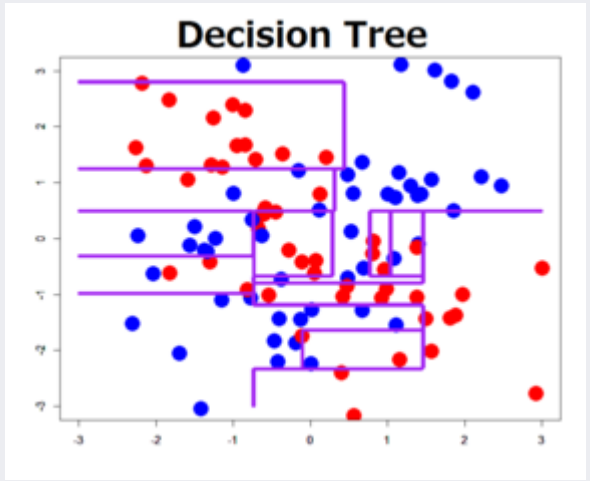
Multi-Layer Perceptron (MLP)

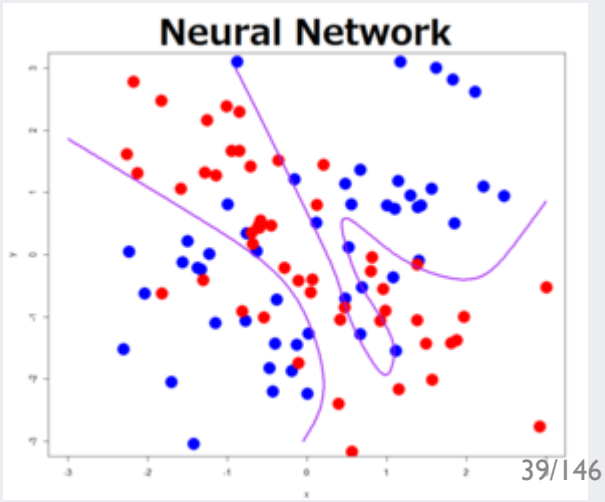
- Decision boundary of MLP
  - ✓ Assume that the class decision boundary can be regarded as a combination of piecewise linear boundaries

	Logstic Regression	Decision Tree	MLP
No. of lines	1	No restriction	User defined (No of hidden layers and hidden nodes)
Direction of lines	No restriction	Vertical to an axis	No restriction

  - MLP has the highest degree of freedom to defined the decision boundary

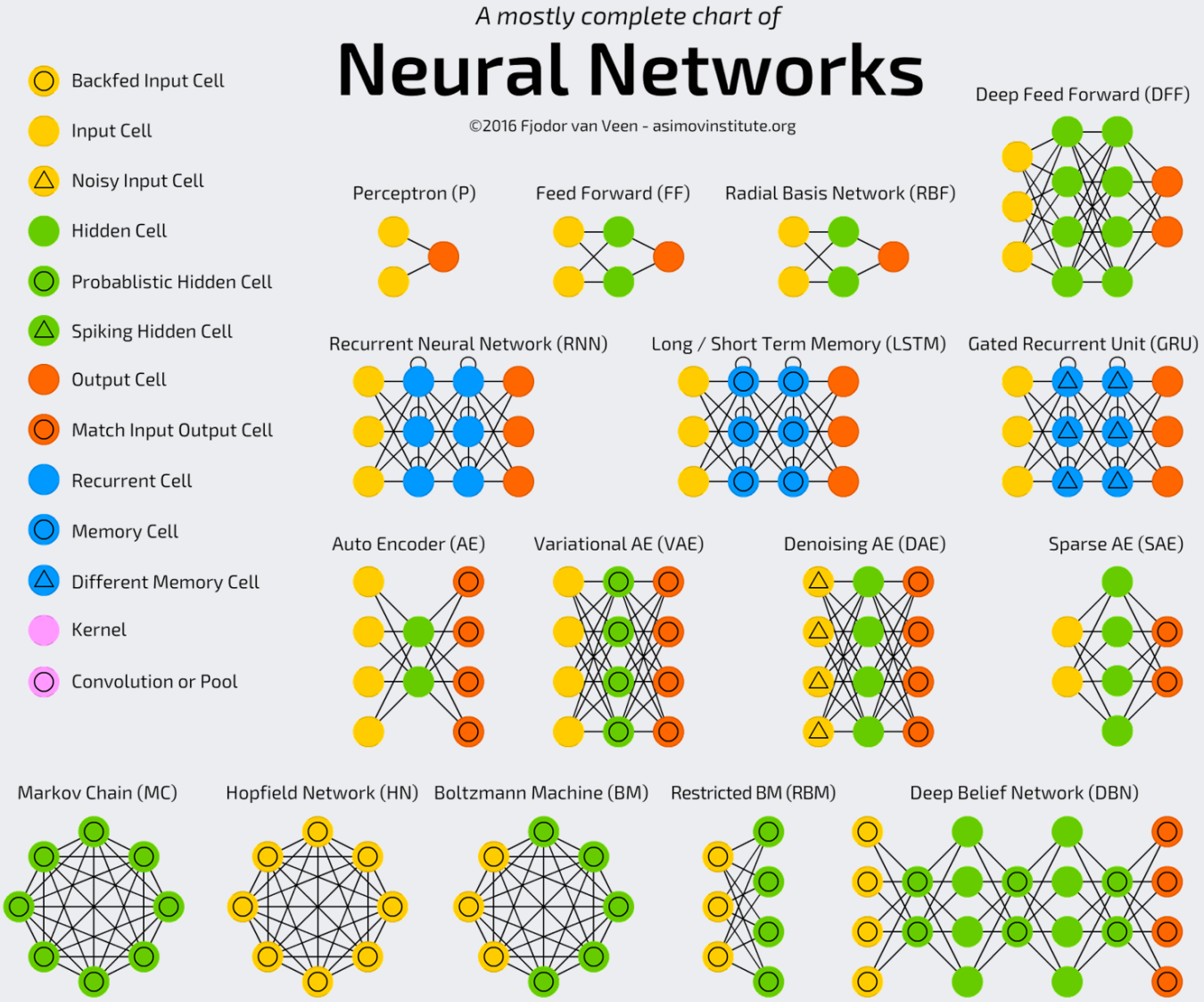






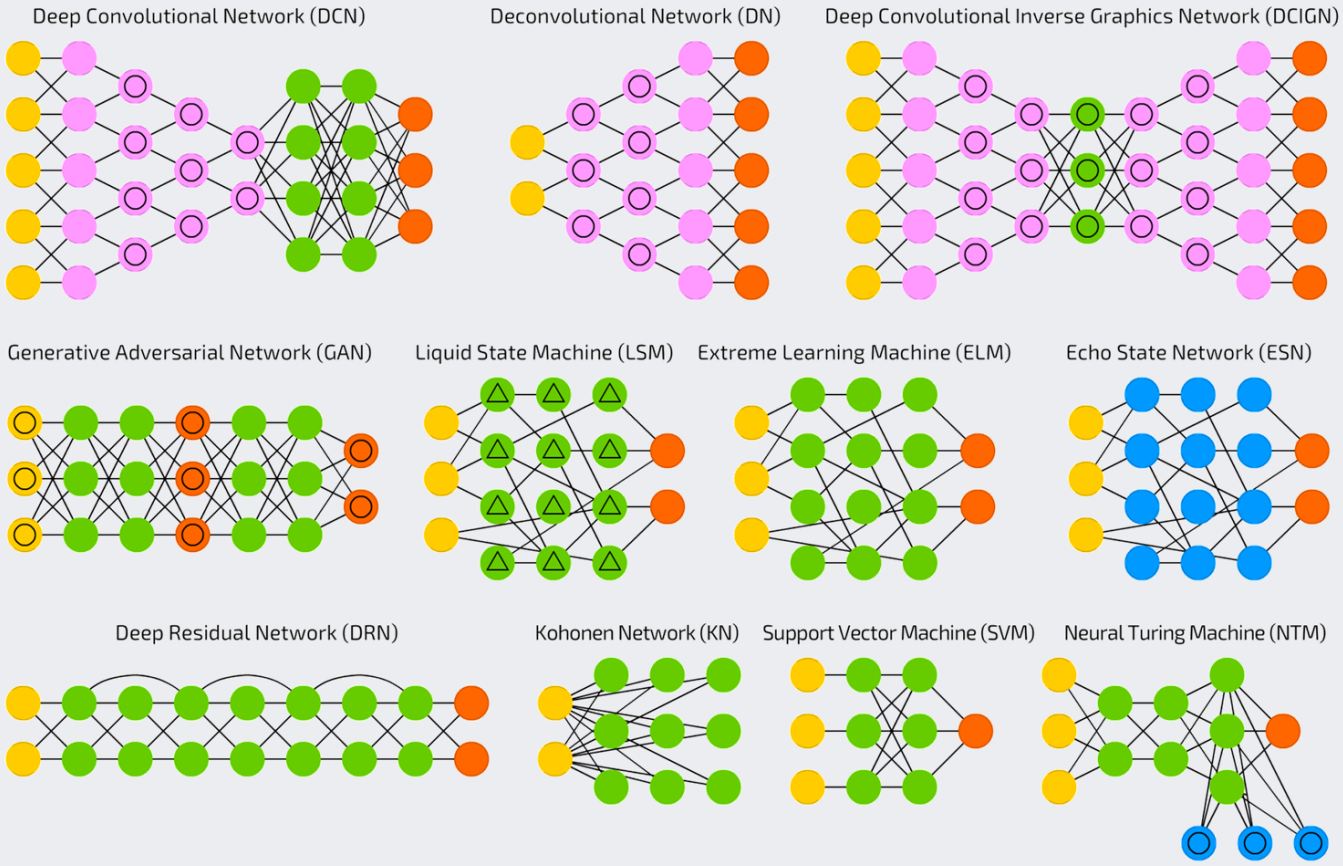
Multi-Layer Perceptron (MLP)

- Various Structure of Artificial Neural Networks



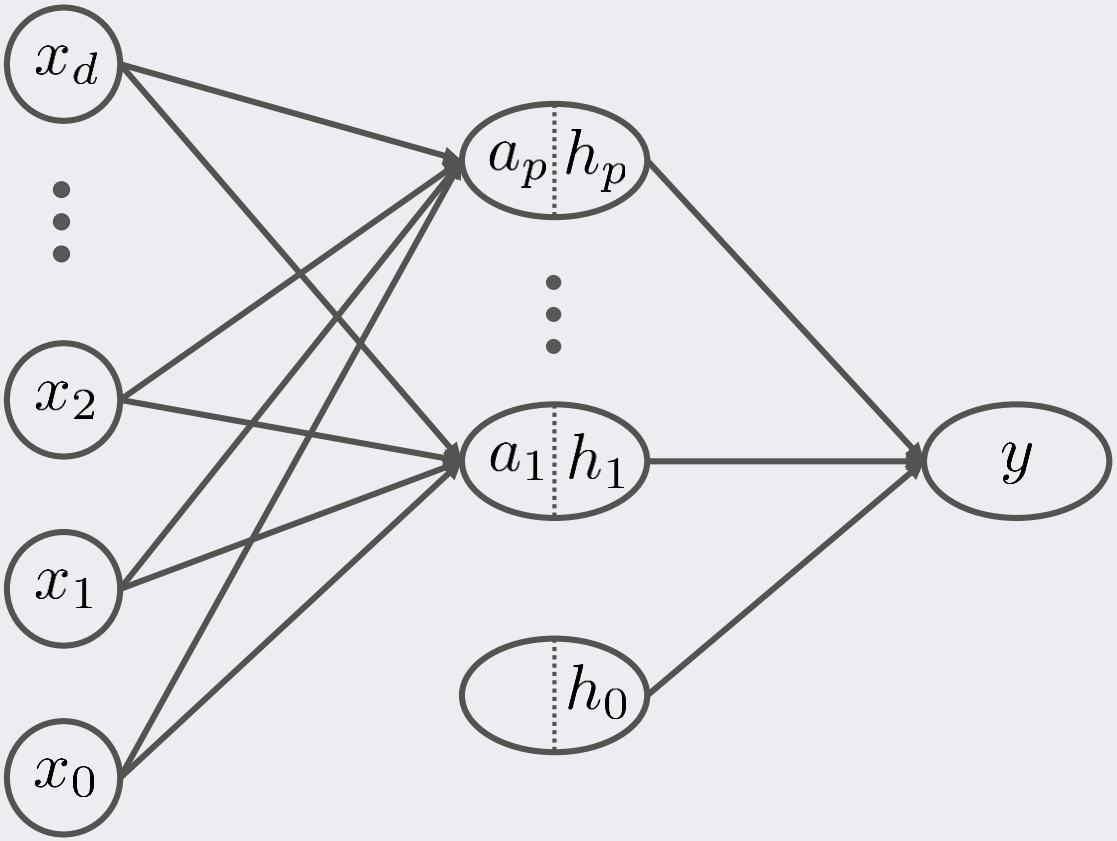
Multi-Layer Perceptron (MLP)

- Various Structure of Artificial Neural Networks



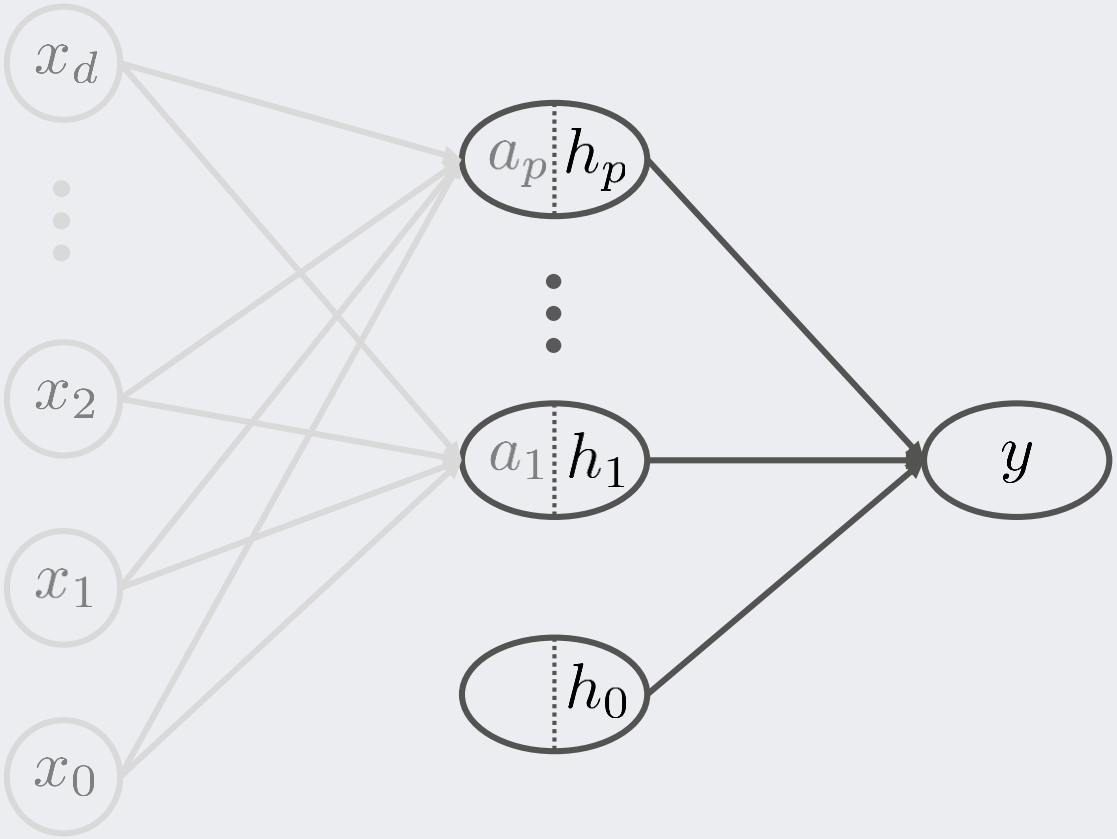
MLP with One Hidden Layer

- Basic Structure: Feed-forward Neural Network with One Hidden Layer
  - ✓ Each hidden node can be considered as an independent perceptron
  - ✓ The output node is a combination of all perceptrons



MLP with One Hidden Layer

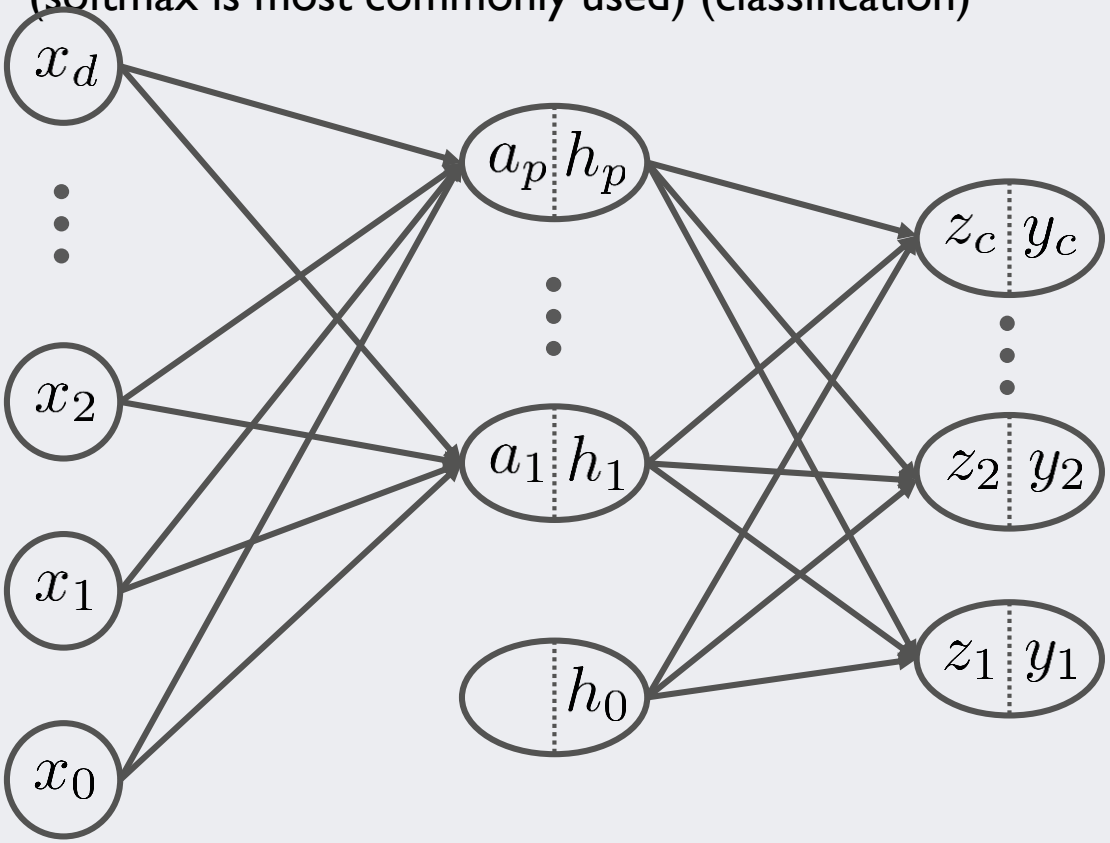
- Basic Structure: Feed-forward Neural Network with One Hidden Layer
  - ✓ Each hidden node can be considered as an independent perceptron
  - ✓ The output node is a linear combination of all perceptrons (regression)



$$y = \sum_{i=0}^{p} w_p^{(2)} h_p$$

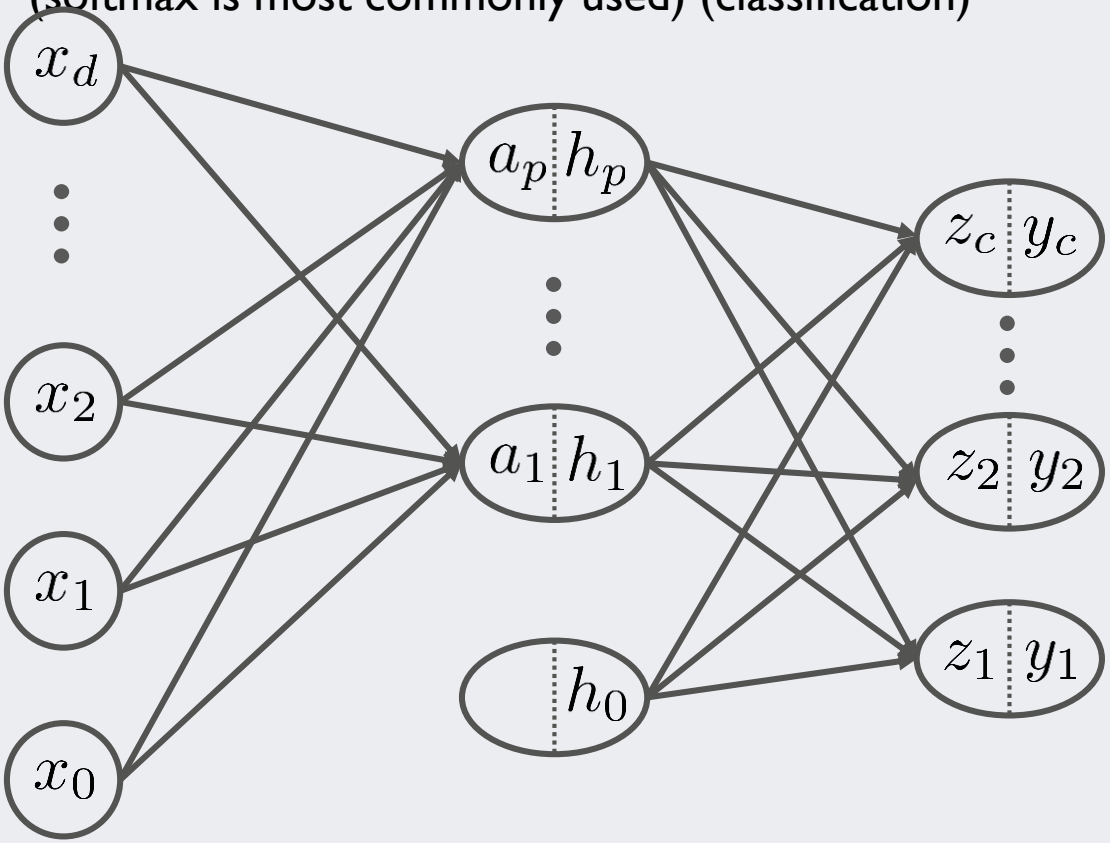
MLP with One Hidden Layer

- Basic Structure: Feed-forward Neural Network with One Hidden Layer
  - ✓ Each hidden node can be considered as an independent perceptron
  - ✓ The output node is a combination of all perceptrons with a non-linear transformation (softmax is most commonly used) (classification)



MLP with One Hidden Layer

- Basic Structure: Feed-forward Neural Network with One Hidden Layer
  - ✓ Each hidden node can be considered as an independent perceptron
  - ✓ The output node is a combination of all perceptrons with a non-linear transformation (softmax is most commonly used) (classification)



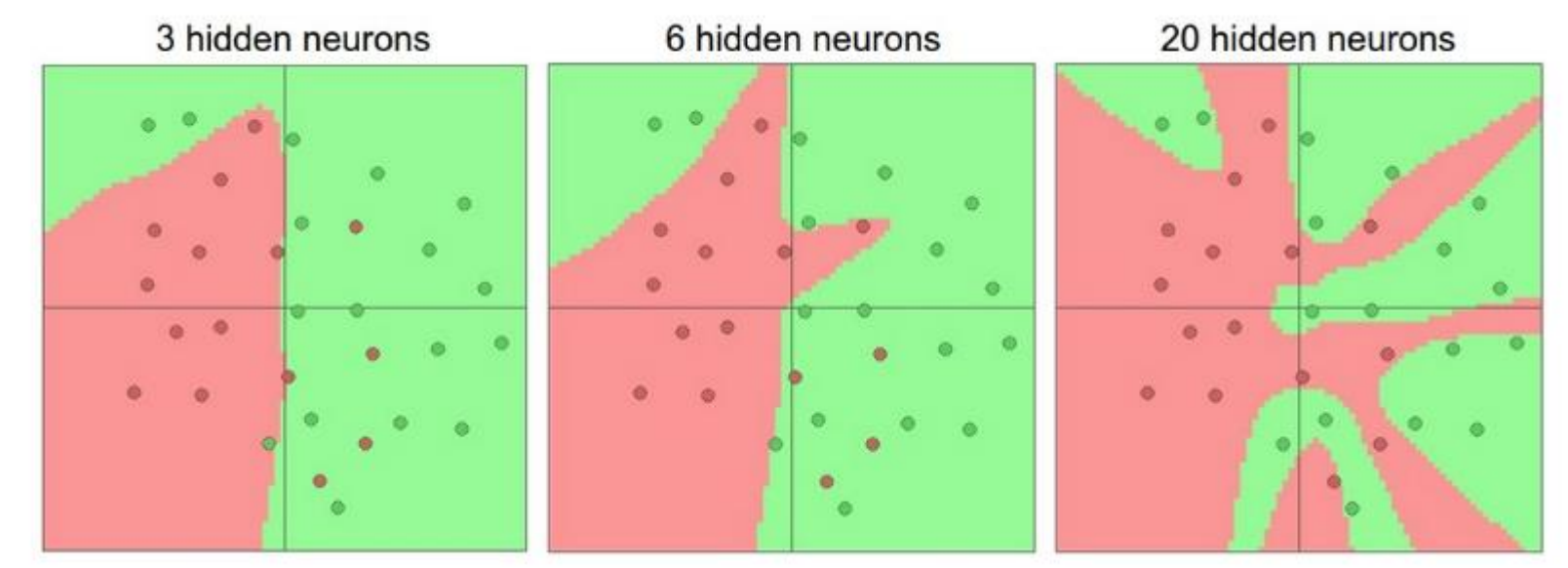
$$z_j = \sum_{i=0}^{p} w_{jp}^{(2)} h_p$$

$$y_j = \frac{e^{z_j}}{\sum_{k=1}^{c} e^{z_k}}$$

각 출력 노드의 출력값을 해당 범주에 속하는 확률로 해석할 수 있음

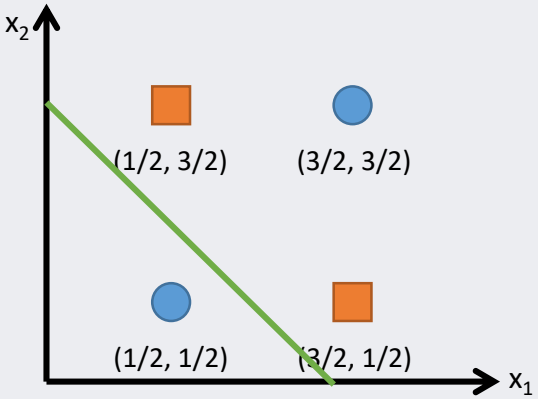
MLP: Hidden Nodes

- The role of hidden nodes
  - ✓ Determines the complexity of ANN
  - ✓ If we use more number of hidden nodes, we can find a more sophisticated decision boundary (classification) or an arbitrary shape of function (regression)

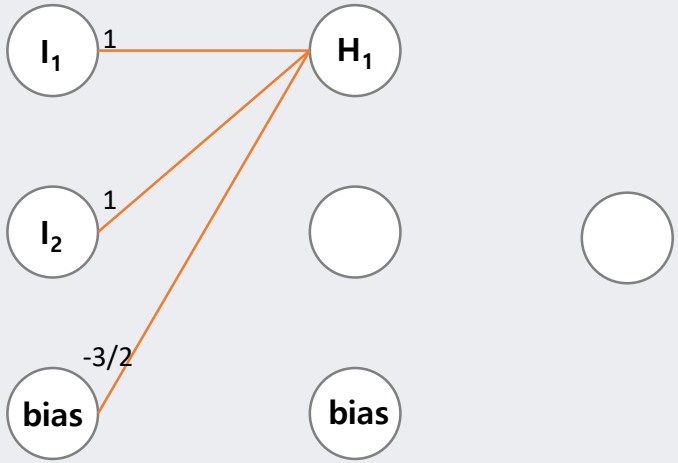


MLP: How it works?

- XOR problem revisited



$$h_1 = x_1 + x_2 - \frac{3}{2}$$

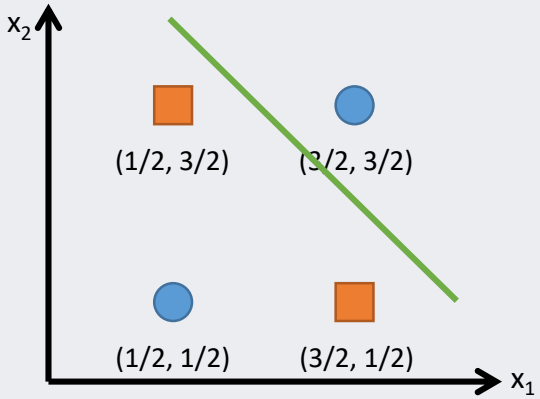


$$z_1 = g(h_1) = \begin{cases} 1 & if\ h_1 \geq 0 \\ -1 & if\ h_1 < 0 \end{cases}$$

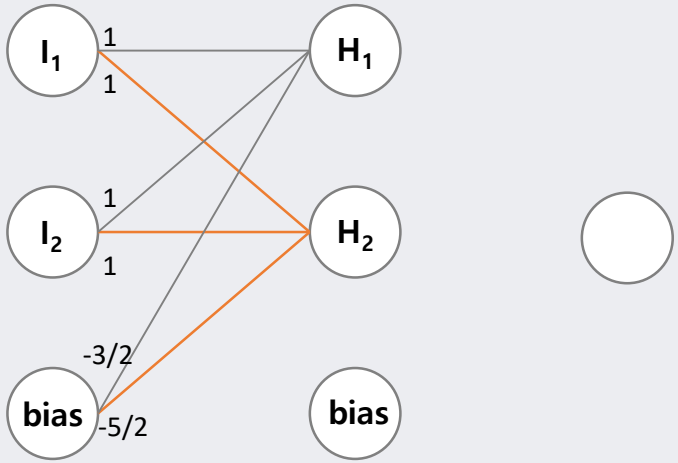
x_1	x_2	h_1	z_1
1/2	1/2	-1/2	-1
3/2	1/2	1/2	1
1/2	3/2	1/2	1
3/2	3/2	3/2	1

MLP: How it works?

- XOR problem revisited (cont')



$$h_2 = x_1 + x_2 - \frac{5}{2}$$

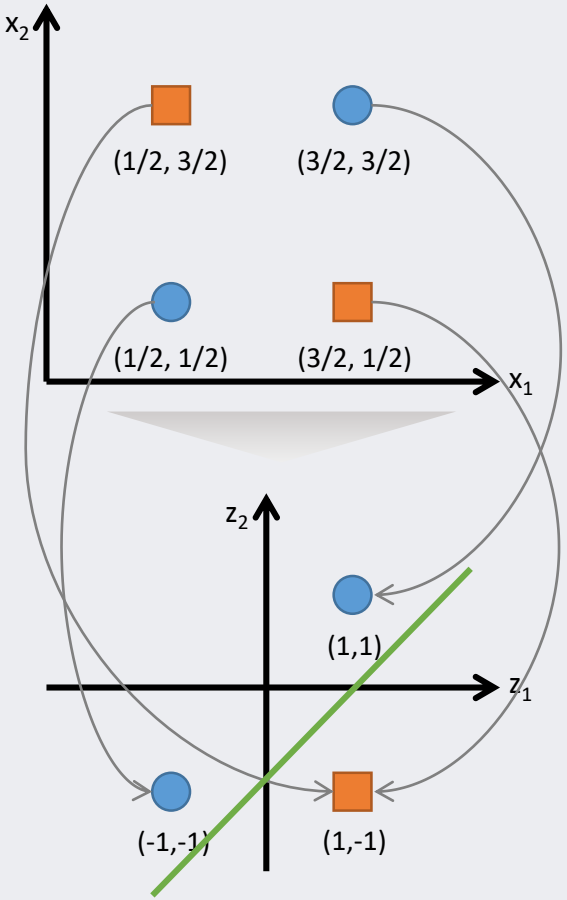


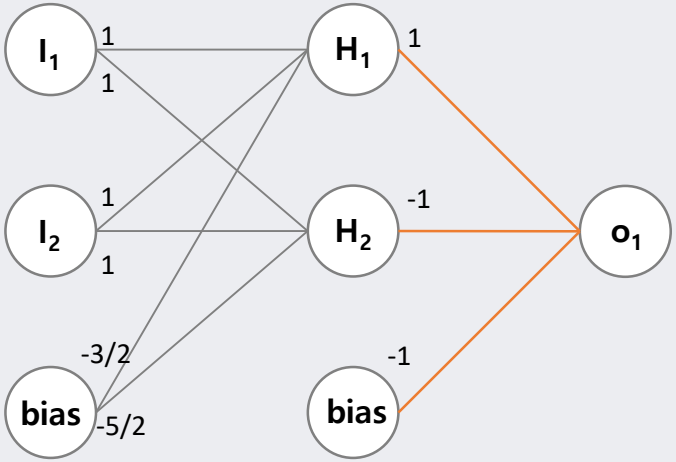
$$z_2 = g(h_2) = \begin{cases} 1 & if\ h_2 \geq 0 \\ -1 & if\ h_2 < 0 \end{cases}$$

x_1	x_2	h_2	z_2
1/2	1/2	-3/2	-1
3/2	1/2	-1/2	-1
1/2	3/2	-1/2	-1
3/2	3/2	1/2	1

MLP: How it works?

- XOR problem revisited (cont')





x_1	x_2	h_1	z_1	h_2	z_2	o_1	z
1/2	1/2	-1/2	-1	-3/2	-1	-1	-1
3/2	1/2	1/2	1	-1/2	-1	1	1
1/2	3/2	1/2	1	-1/2	-1	1	1
3/2	3/2	3/2	1	1/2	1	-1	-1

$$o_1 = z_1 - z_2 - 1$$

$$z = g(o_1) = \begin{cases} 1 & if\ o_1 \geq 0 \\ -1 & if\ o_1 < 0 \end{cases}$$

MLP: Formulation

- General formulation
  - ✓ The output of the hidden node j (when the activation function is sigmoid):

$$h_j = \sum_{i=1}^{d+1} w_{ji}^{(1)} x_i, \quad z_j = g(h_j) = \frac{1}{1+\exp(-h_j)}$$

  - ✓ The output of the output node (when the activation function is sigmoid):

$$o = \sum_{j=1}^{p+1} w_j^{(2)} g(h_j), \quad z = g(o) = \frac{1}{1+\exp(-o)}$$

  - ✓ The final outcome of the neural network:

$$\hat{y} = g(\sum_{j=1}^{p+1} w_j^{(2)} g(\sum_{i=1}^{d+1} w_{ji}^{(1)} x_i))$$

MLP: Training

- Gradient descent algorithm
  - ✓ Taylor expansion of a function

$$f(x+\Delta x) = f(x) + \frac{f'(x)}{1!}\Delta x + \frac{f''(x)}{2!}\Delta x^2 + \cdots$$

  - ✓ For the minimization problem, we can reduce the objective function value by moving the current solution to the opposite direction of the first derivative if it is not zero

$$x_{new} = x_{old} - \eta f'(x), \quad where \;\; 0 < \eta < 1$$

  - ✓ The objective function value become lower compared to the previous solution

$$f(x_{new}) = f(x_{old} - \eta f'(x)) \cong f(x_{old}) - \eta |f'(x)|^2 < f(x_{old})$$

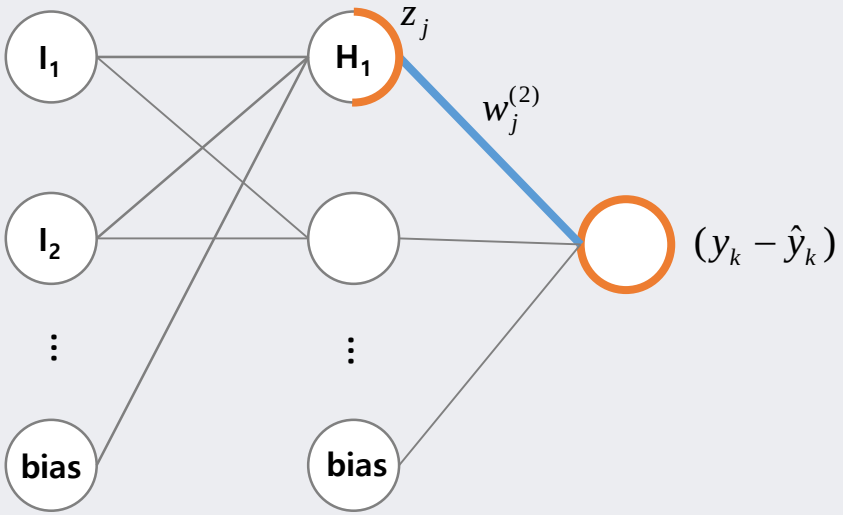
MLP: Training

- Error Back-Propagation
  - ✓ The error of kth observation

$$Err_k = \frac{1}{2}(y_k - \hat{y}_k)^2, \quad \hat{y}_k = \sum_{j=1}^{p+1} w_j^{(2)} g\left(\sum_{i=1}^{d+1} w_{ji}^{(1)} x_i\right)$$

  - ✓ The weight $w_j^{(2)}$ which connects the jth hidden node

$$\frac{\partial Err_k}{\partial w_j^{(2)}} = \frac{\partial Err_k}{\partial \hat{y}_k} \cdot \frac{\partial \hat{y}_k}{\partial w_j^{(2)}} = (y_k - \hat{y}_k) \cdot z_j$$

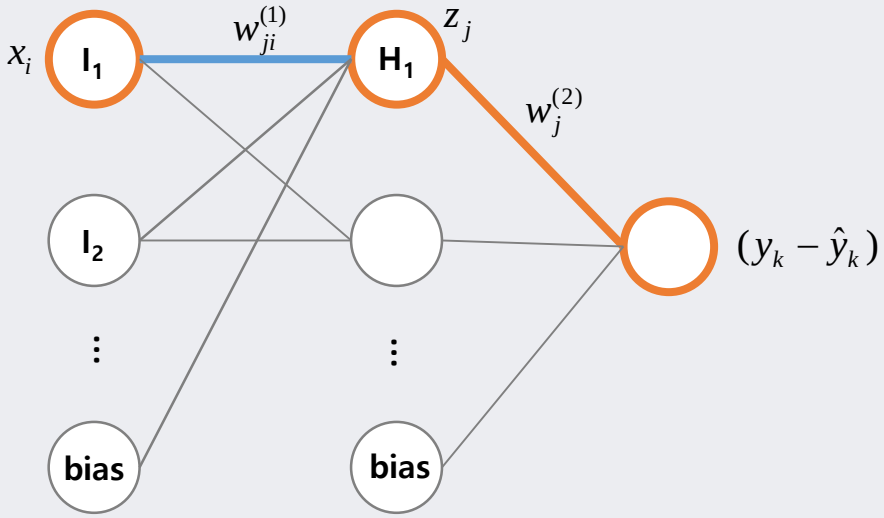


최종 결과물을 얻고	**Feed Forward and Prediction**
그 결과물과 우리가 원하는 결과물과의 차이점을 찾은 후	**Cost Function**
그 차이가 무엇으로 인해 생기는 지	**Differentiation (미분)**
역으로 내려가면서 추정하여	**Back Propagation**
새로운 Parameter 값을 배움	**Weight Update**

MLP: Training

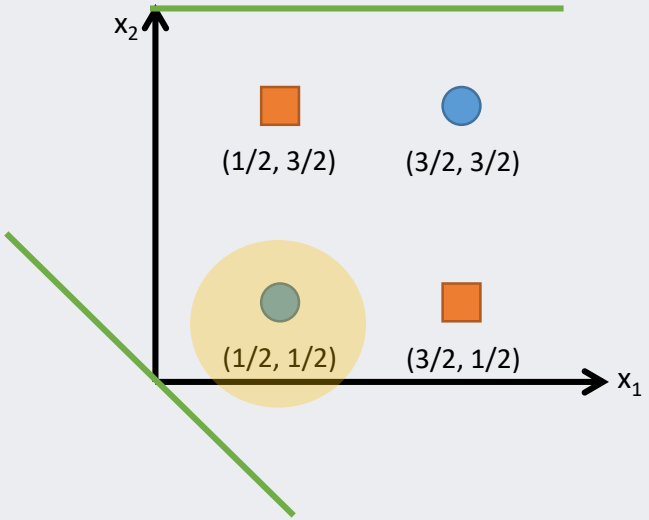
- Error Back-Propagation
  - ✓ The weight $w_{ji}^{(1)}$ which connects the i^{th} input node and j^{th} hidden node

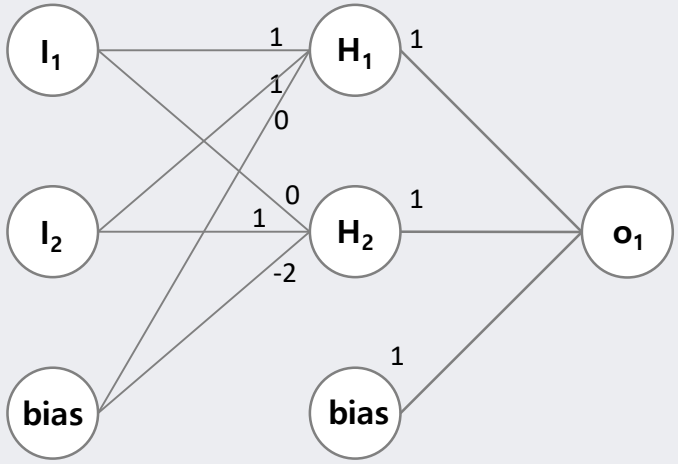
$$\frac{\partial Err_k}{\partial w_{ji}^{(1)}} = \frac{\partial Err_k}{\partial \hat{y}_k} \cdot \frac{\partial \hat{y}_k}{\partial z_j} \cdot \frac{\partial z_j}{\partial h_j} \cdot \frac{\partial h_j}{\partial w_{ji}^{(1)}} = (y_k - \hat{y}_k) \cdot w_j^{(2)} \cdot z_j \cdot (1 - z_j) \cdot x_i$$



MLP: Training

- Error Back-Propagation: Example
  - ✓ Initial weight: Random generation





$$h_1 = \sum w_{1i}^{(1)} x_i = 1 \times 0.5 + 1 \times 0.5 + 0 \times 1 = 1 \qquad z_1 = \frac{1}{1 + \exp(1)} = 0.269$$

$$h_2 = \sum w_{2i}^{(1)} x_i = 0 \times 0.5 + 1 \times 0.5 + (-2) \times 1 = -1.5 \qquad z_2 = \frac{1}{1 + \exp(-1.5)} = 0.818$$

$$\hat{y} = \sum w_j^{(2)} z_j = 1 \times 0.269 + 1 \times 0.818 + 1 \times 1 = 2.087$$

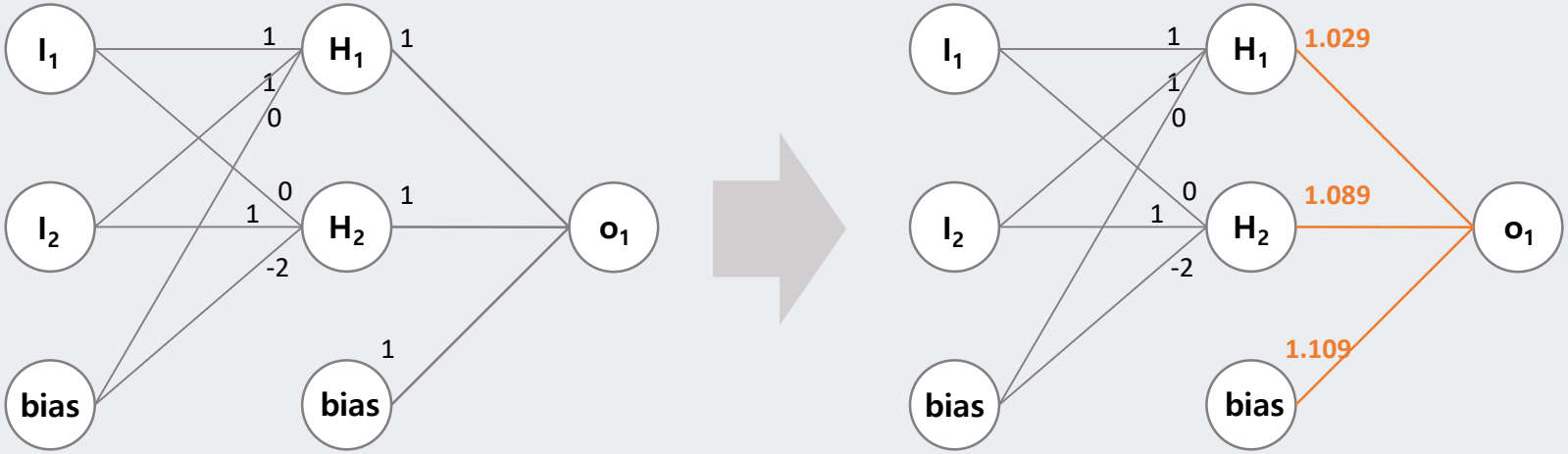
MLP: Training

- Error Back-Propagation: Example
  - ✓ Update the weights between the output and the hidden nodes

$$w_1^{(2)}(new) = w_1^{(2)}(old) - \eta \times (y - \hat{y}) \times z_1 = 1 - 0.1 \times (1 - 2.087) \times 0.269 = 1.029$$

$$w_2^{(2)}(new) = w_2^{(2)}(old) - \eta \times (y - \hat{y}) \times z_2 = 1 - 0.1 \times (1 - 2.087) \times 0.818 = 1.089$$

$$w_0^{(2)}(new) = w_0^{(2)}(old) - \eta \times (y - \hat{y}) \times b^{(2)} = 1 - 0.1 \times (1 - 2.087) \times 1 = 1.109$$



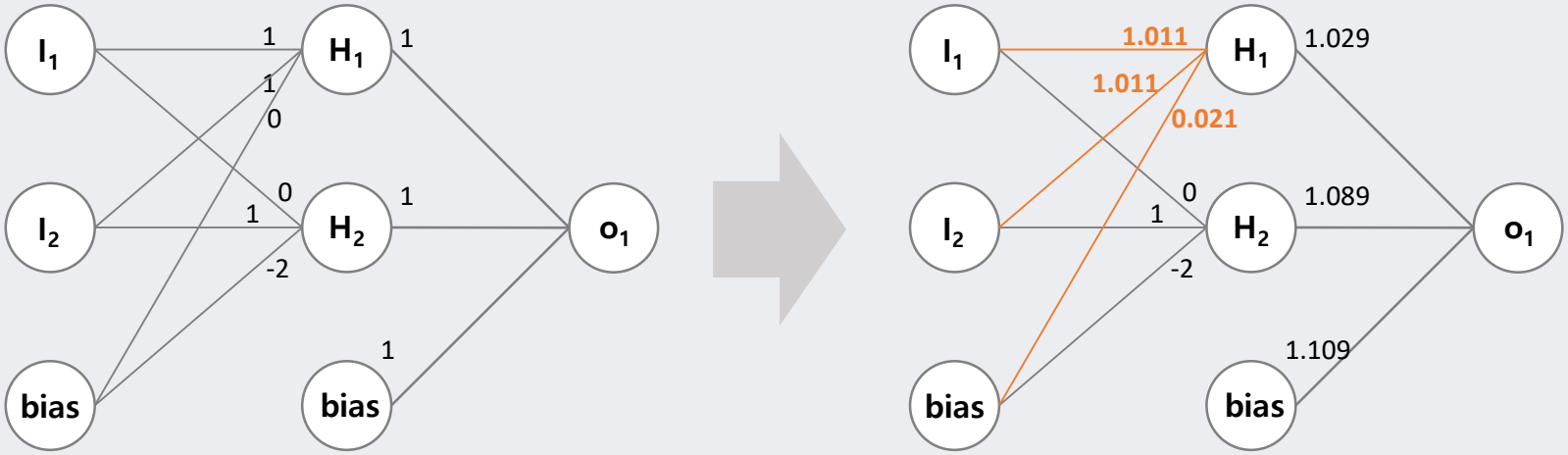
MLP: Training

- Error Back-Propagation: Example
  - ✓ Update the weights between the H_1 and the input nodes

$$w_{11}^{(1)}(new) = w_{11}^{(1)}(old) - \eta \times (y - \hat{y}) \times w_1^{(2)} \times z_1 \times (1 - z_1) \times x_1 = 1.011$$

$$w_{12}^{(1)}(new) = w_{12}^{(1)}(old) - \eta \times (y - \hat{y}) \times w_1^{(2)} \times z_1 \times (1 - z_1) \times x_2 = 1.011$$

$$w_{10}^{(1)}(new) = w_{10}^{(1)}(old) - \eta \times (y - \hat{y}) \times w_1^{(2)} \times z_1 \times (1 - z_1) \times b^{(1)} = 0.021$$



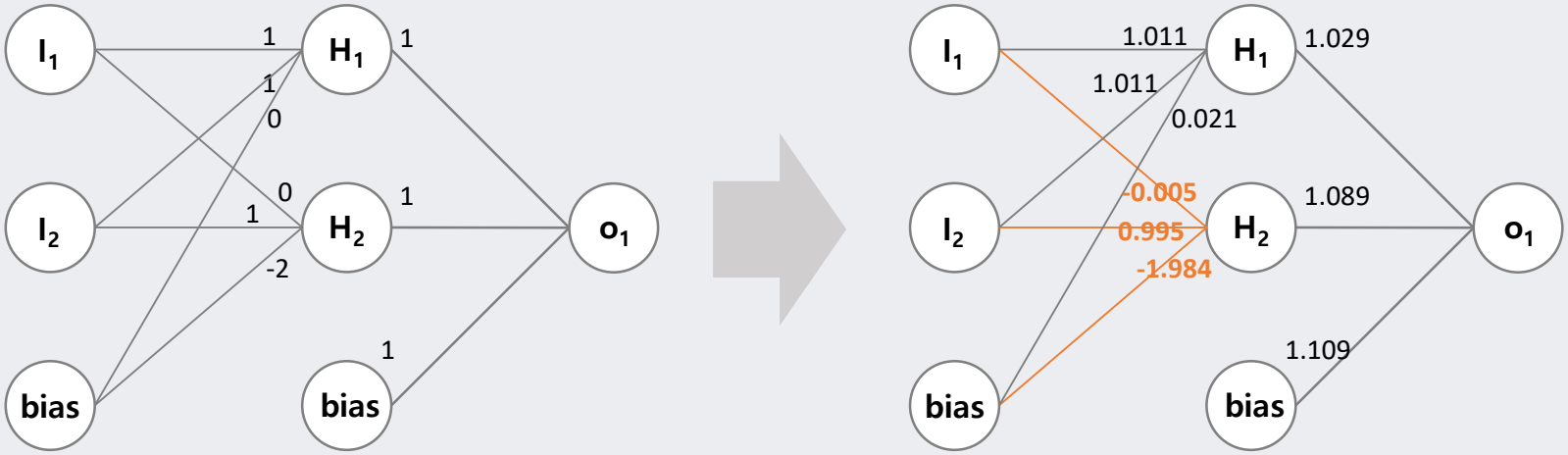
MLP: Training

- Error Back-Propagation: Example
  - ✓ Update the weights between the H_1 and the input nodes

$$w_{21}^{(1)}(new) = w_{21}^{(1)}(old) - \eta \times (y - \hat{y}) \times w_2^{(2)} \times z_2 \times (1 - z_2) \times x_1 = -0.005$$

$$w_{22}^{(1)}(new) = w_{22}^{(1)}(old) - \eta \times (y - \hat{y}) \times w_2^{(2)} \times z_2 \times (1 - z_2) \times x_2 = 0.995$$

$$w_{20}^{(1)}(new) = w_{20}^{(1)}(old) - \eta \times (y - \hat{y}) \times w_2^{(2)} \times z_2 \times (1 - z_2) \times b^{(1)} = -1.984$$



MLP: Training

- Goal
  - ✓ Find the weights that yield best predictions
- Features
  - ✓ The process described before is repeated for all records
  - ✓ At each record, compare the prediction to the actual target
  - ✓ Difference is the error for the output node
  - ✓ Error is propagated back and distributed to all the hidden nodes and used to update their weights

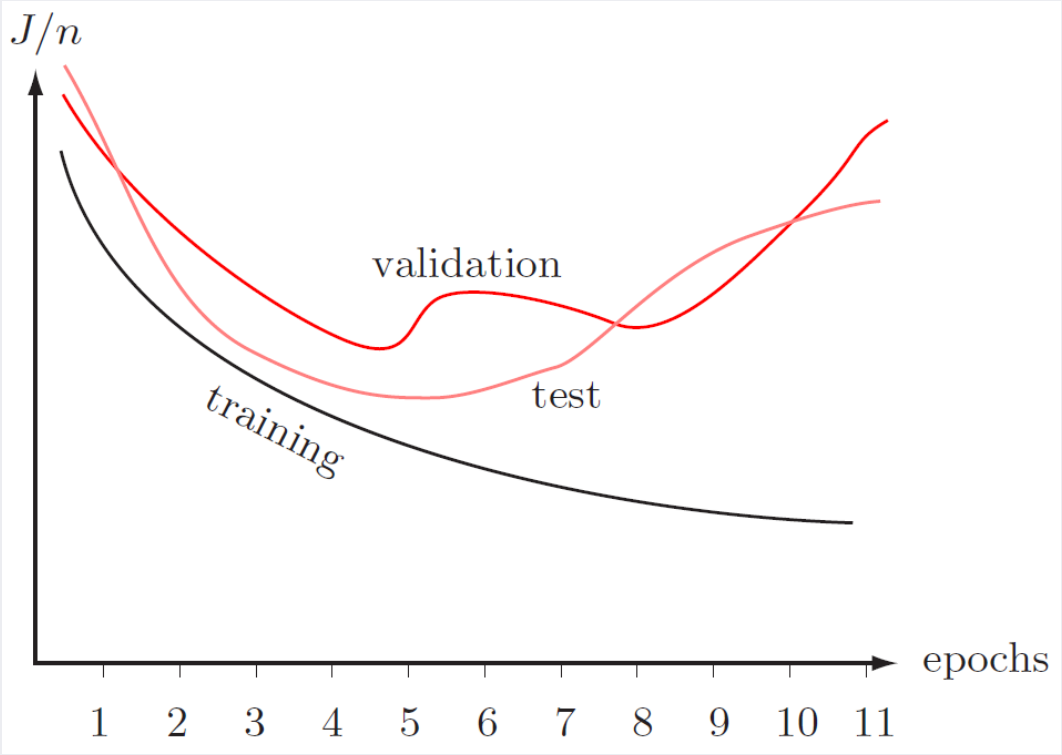
MLP: Training

- Why it works
  - ✓ Big errors lead to big changes in weights
  - ✓ Small errors leave weights relatively unchanged
  - ✓ Over thousand of updates, a given weight keeps changing until the error associated with it is negligible

- Common criteria to stop updating
  - ✓ When weights change very little from one epoch to the next
  - ✓ When the misclassification rate reaches a required threshold
  - ✓ When a limit on runs is reached

MLP: Training

- With sufficient iterations, neural networks can easily over-fit the data.
- To avoid over-fitting,
  - ✓ Track error in validation data
  - ✓ Limit iterations
  - ✓ Limit complexity of network
  - ✓ N. of hidden layers, nodes, etc.



AGENDA

01 Neural Network: Overview

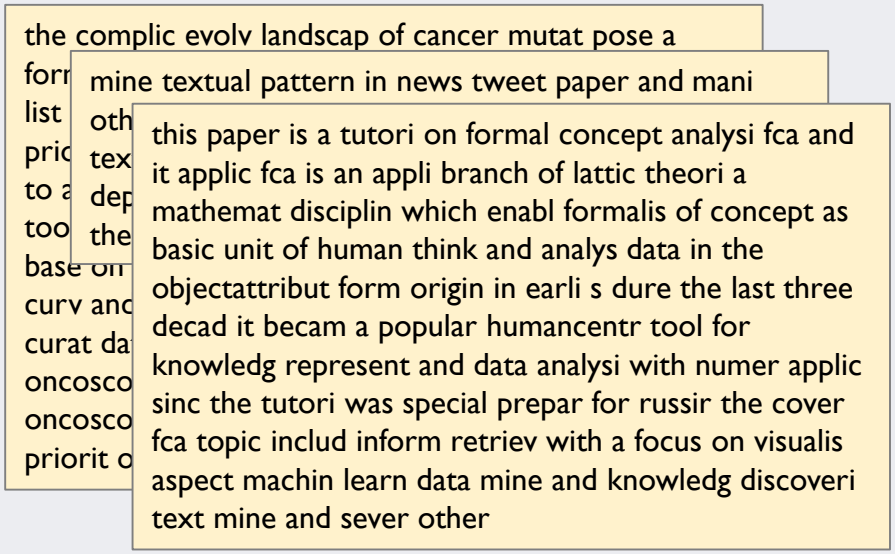
02 Convolutional Neural Network for Document Classification

03 Recurrent Neural Network for Document Classification

MLP for Document Classification

- MLP for Document Classification

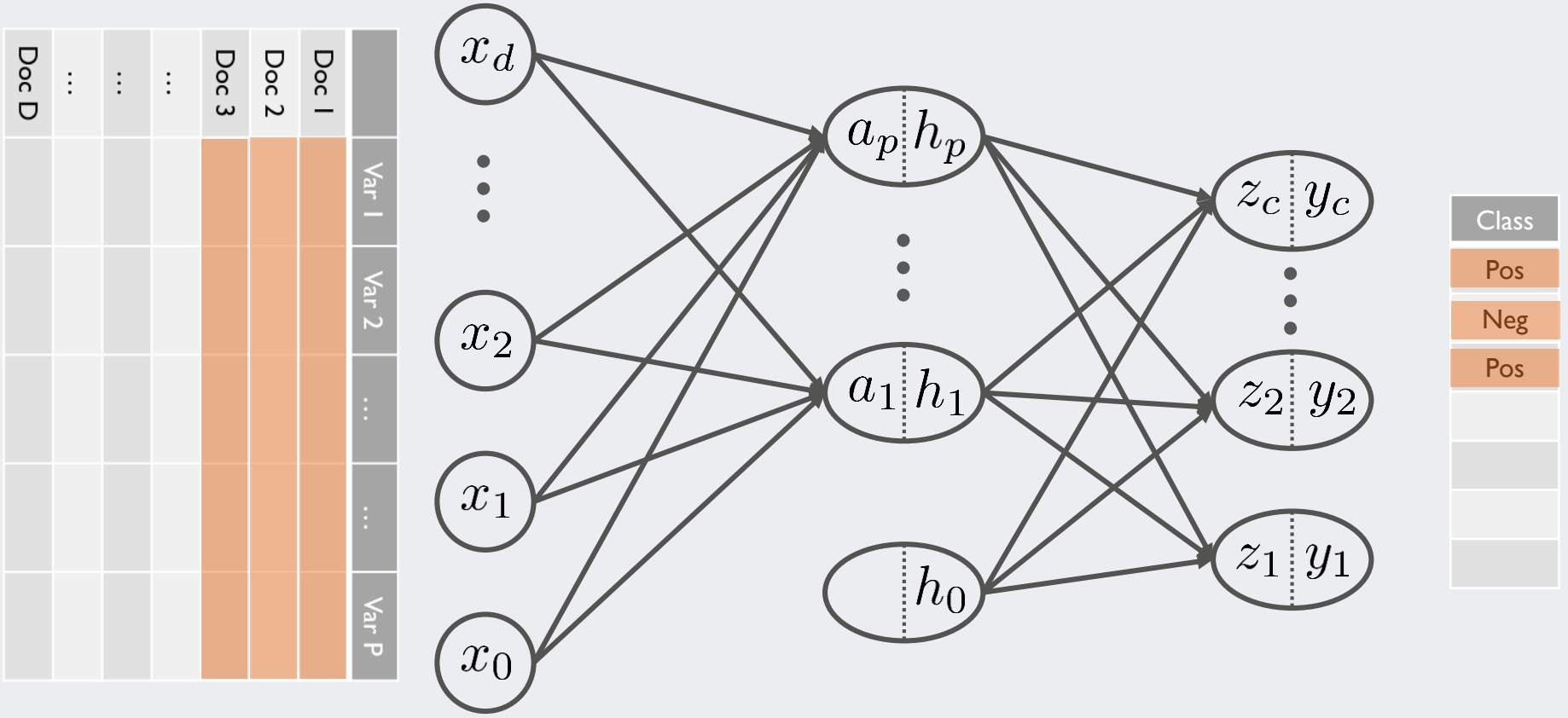
Transform unstructured data into structured data



	Var 1	Var 2	…	…	Var P	Class
Doc 1						Pos
Doc 2						Neg
Doc 3						Pos
…						
…						
…						
Doc D						

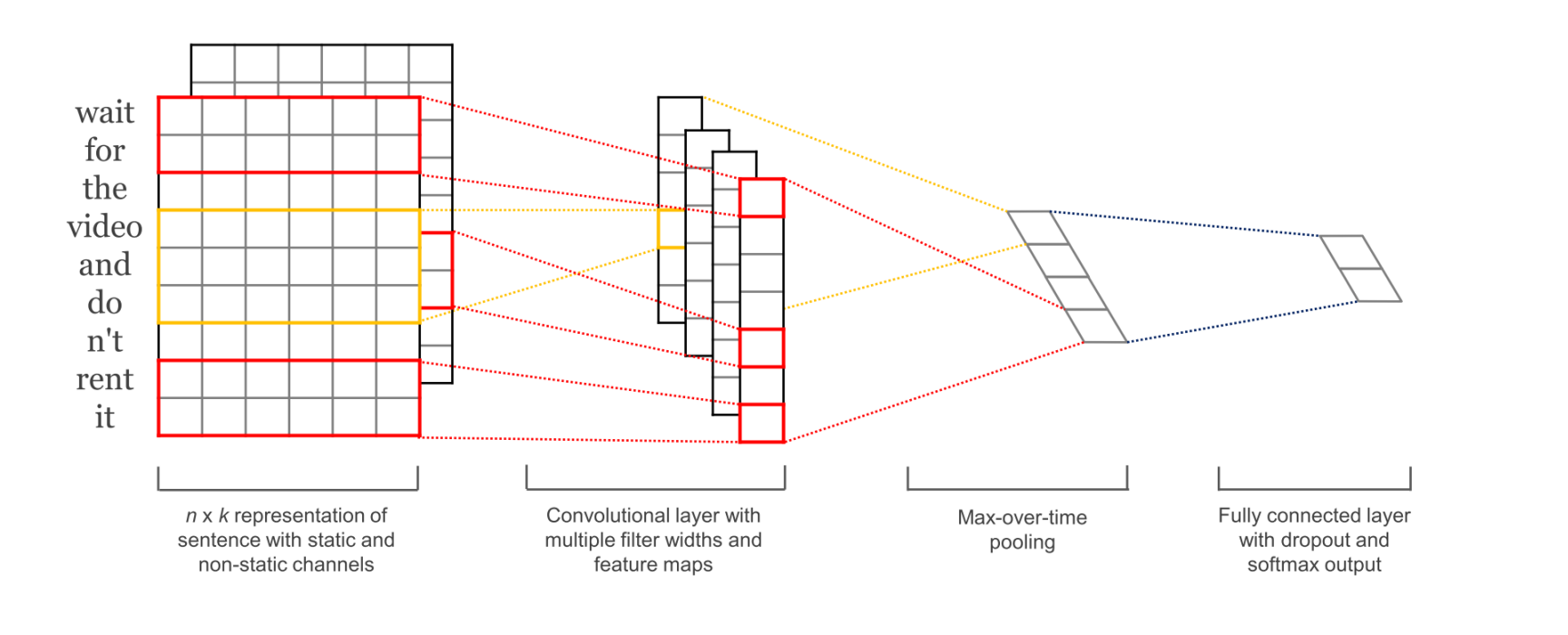
MLP for Document Classification

- MLP for Document Classification



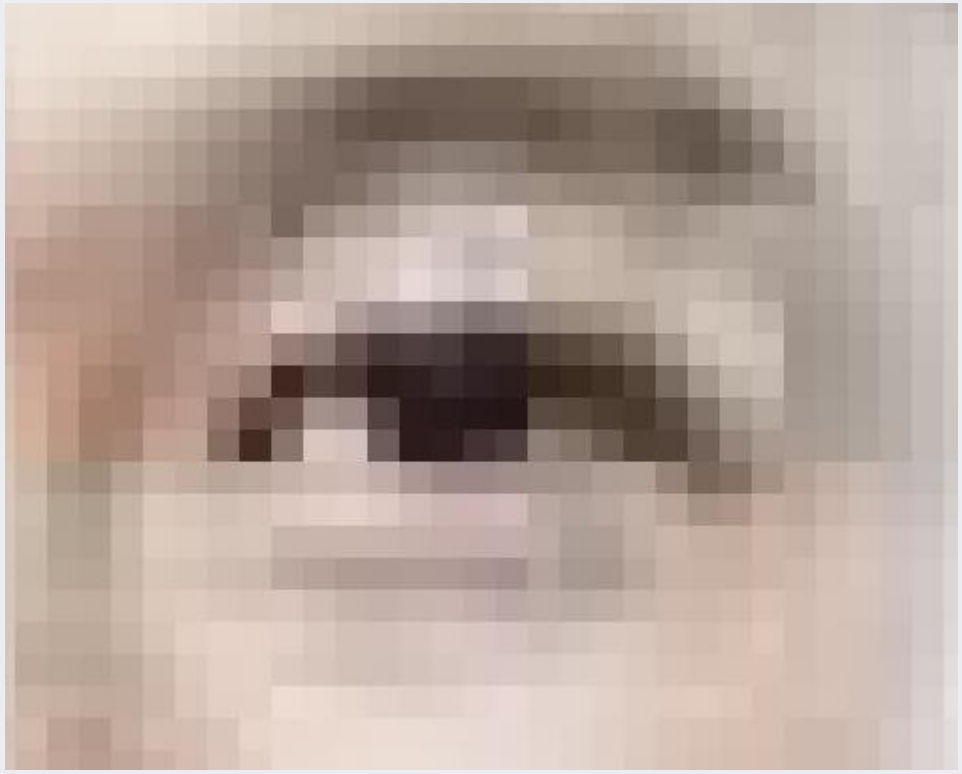
Convolutional Neural Network (CNN)

- CNN for Sentence Classification (Kim, 2014)



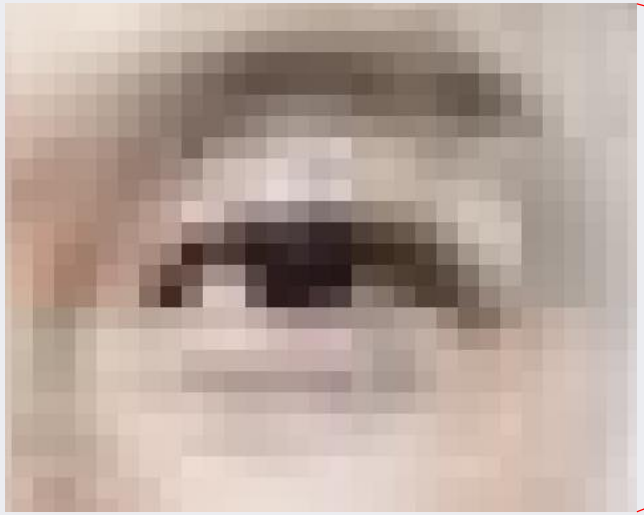
CNN Basics: Image Representation

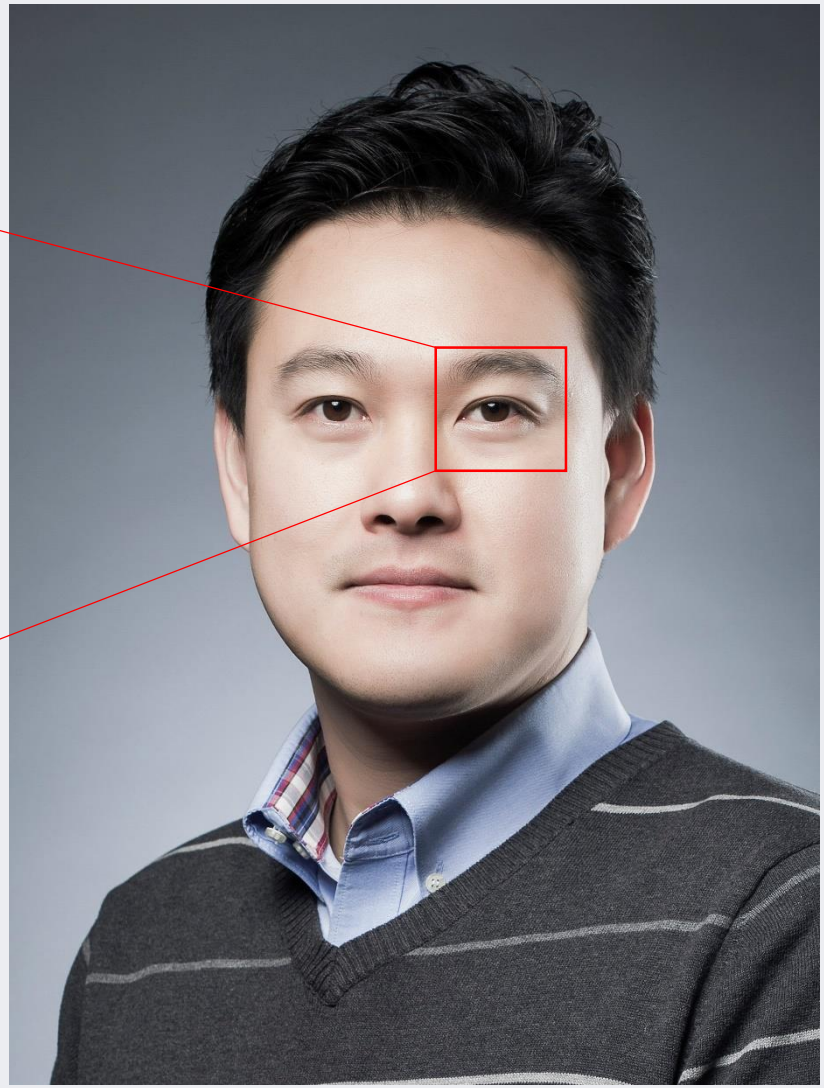
- Whose eye is it?



CNN Basics: Image Representation

- Whose eye is it?





CNN Basics: Image Representation

- A color image is a tensor with the size of (Width X Height X 3 (RGB))

1,685

2,247

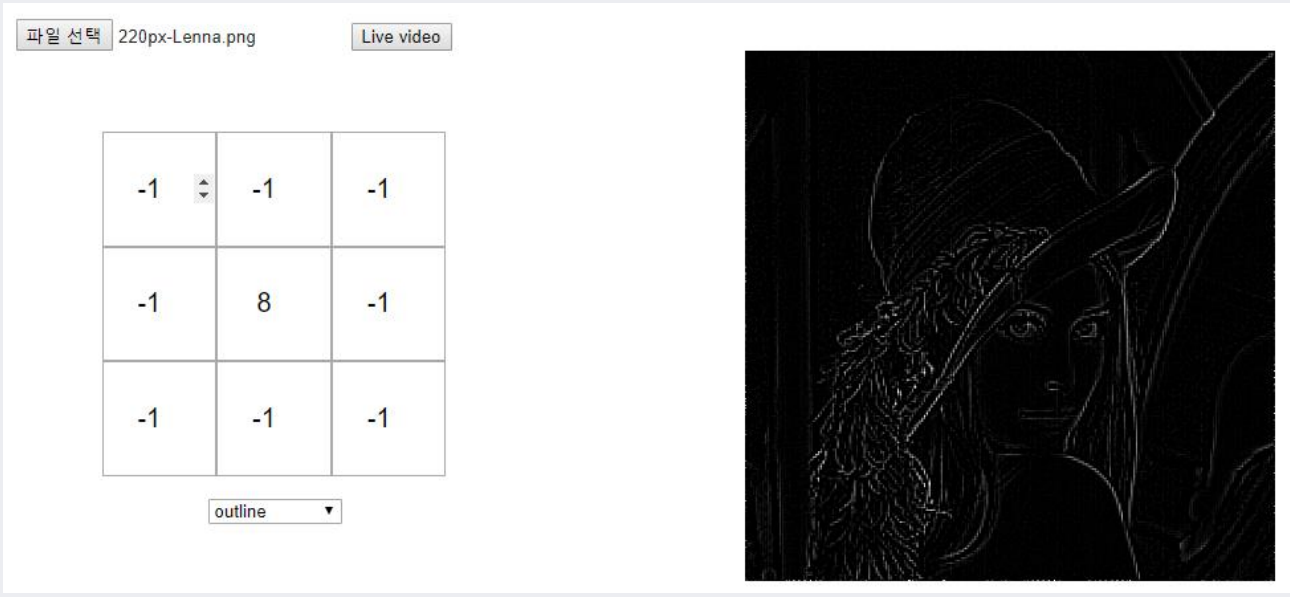
```
> pskang[1:10, 1:10, 1]
           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]      [,9]     [,10]
 [1,] 0.2784314 0.2784314 0.2745098 0.2745098 0.2745098 0.2745098 0.2705882 0.2705882 0.2862745 0.2901961
 [2,] 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2666667 0.2705882
 [3,] 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2705882 0.2745098
 [4,] 0.2705882 0.2705882 0.2745098 0.2745098 0.2745098 0.2745098 0.2784314 0.2784314 0.2823529 0.2823529
 [5,] 0.2705882 0.2705882 0.2705882 0.2745098 0.2745098 0.2784314 0.2784314 0.2784314 0.2784314 0.2784314
 [6,] 0.2705882 0.2705882 0.2745098 0.2745098 0.2745098 0.2745098 0.2784314 0.2784314 0.2823529 0.2784314
 [7,] 0.2705882 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2784314 0.2941176 0.2862745
 [8,] 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2901961 0.2823529
 [9,] 0.2745098 0.2705882 0.2745098 0.2784314 0.2823529 0.2823529 0.2745098 0.2666667 0.2784314 0.2784314
[10,] 0.2784314 0.2784314 0.2784314 0.2823529 0.2862745 0.2862745 0.2784314 0.2745098 0.2745098 0.2745098
```

```
> pskang[1:10, 1:10, 2]
           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]      [,9]     [,10]
 [1,] 0.2980392 0.2980392 0.2941176 0.2941176 0.2941176 0.2941176 0.2901961 0.2901961 0.2980392 0.3019608
 [2,] 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2784314 0.2823529
 [3,] 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2823529 0.2862745
 [4,] 0.2901961 0.2901961 0.2941176 0.2941176 0.2941176 0.2941176 0.2980392 0.2980392 0.2941176 0.2941176
 [5,] 0.2901961 0.2901961 0.2901961 0.2941176 0.2941176 0.2980392 0.2980392 0.2980392 0.2901961 0.2901961
 [6,] 0.2901961 0.2901961 0.2941176 0.2941176 0.2941176 0.2941176 0.2980392 0.2980392 0.2941176 0.2901961
 [7,] 0.2901961 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2980392 0.3058824 0.2980392
 [8,] 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.3019608 0.2941176
 [9,] 0.2862745 0.2823529 0.2862745 0.2901961 0.2941176 0.2941176 0.2862745 0.2784314 0.2901961 0.2901961
[10,] 0.2901961 0.2901961 0.2901961 0.2941176 0.2980392 0.2980392 0.2901961 0.2862745 0.2862745 0.2862745
```

```
> pskang[1:10, 1:10, 3]
           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]      [,9]     [,10]
 [1,] 0.3215686 0.3215686 0.3176471 0.3176471 0.3176471 0.3176471 0.3137255 0.3137255 0.3254902 0.3294118
 [2,] 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3058824 0.3098039
 [3,] 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3098039 0.3137255
 [4,] 0.3137255 0.3137255 0.3176471 0.3176471 0.3176471 0.3176471 0.3215686 0.3215686 0.3215686 0.3215686
 [5,] 0.3137255 0.3137255 0.3137255 0.3176471 0.3176471 0.3215686 0.3215686 0.3215686 0.3176471 0.3176471
 [6,] 0.3137255 0.3137255 0.3176471 0.3176471 0.3176471 0.3176471 0.3215686 0.3215686 0.3215686 0.3176471
 [7,] 0.3137255 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3215686 0.3333333 0.3254902
 [8,] 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3294118 0.3215686
 [9,] 0.3058824 0.3019608 0.3058824 0.3098039 0.3137255 0.3137255 0.3058824 0.2980392 0.3098039 0.3098039
[10,] 0.3098039 0.3098039 0.3098039 0.3137255 0.3176471 0.3176471 0.3098039 0.3058824 0.3058824 0.3058824
```

CNN Basics: Convolution

- Problem
  - ✓ If all pixels are connected to the first hidden layer, we need to optimize too many network weights
- Image Convolution (Filter, Kernel)
  - ✓ A matrix that capture a certain type of image feature (ex: edge detection)

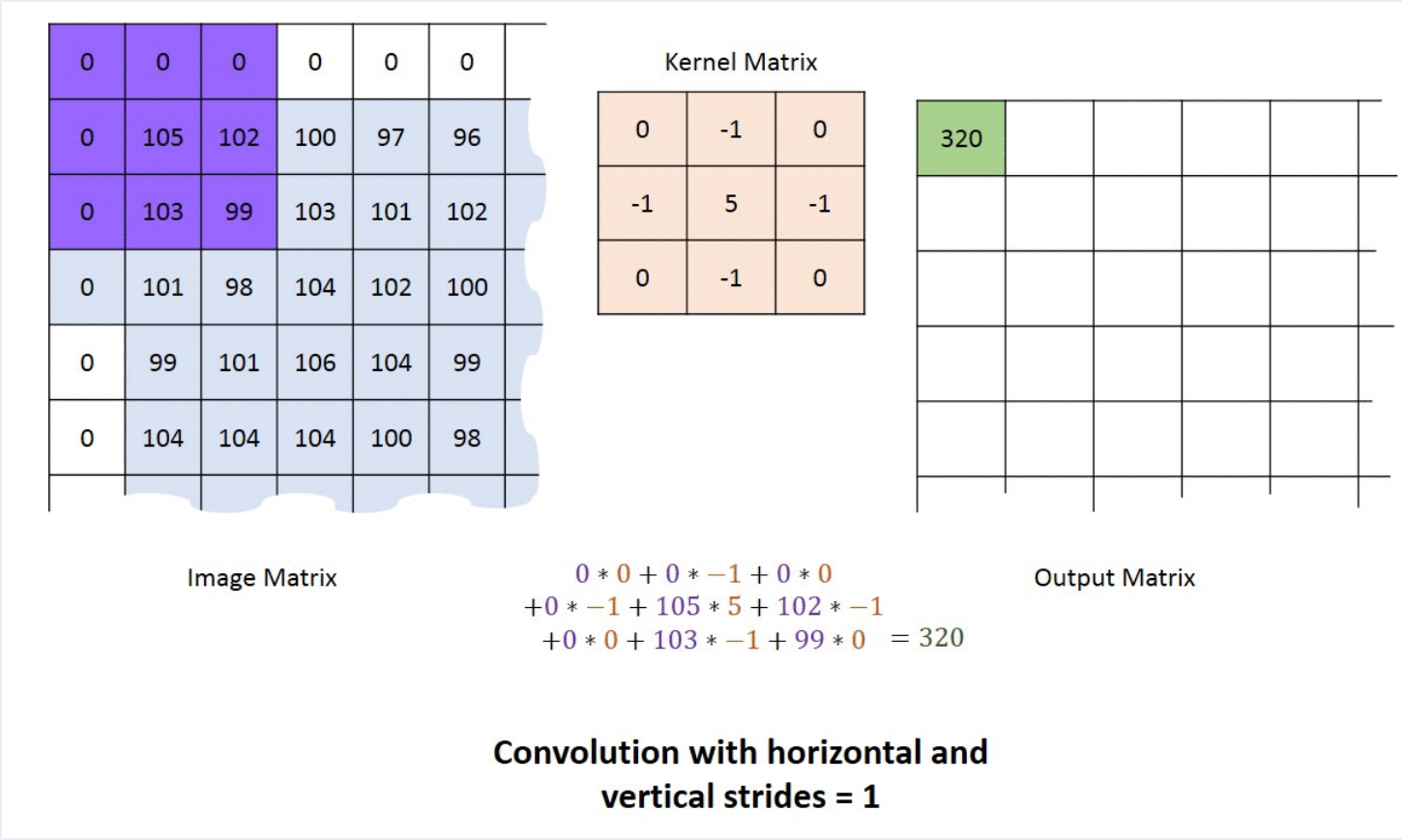


She appeared on the cover of the journal *Optical Engineering* in 1991, and the model herself was invited to a conference of the Image Science and Technology society in 1997.

https://www.software.ac.uk/blog/2016-10-03-how-photo-playboy-became-part-scientific-culture

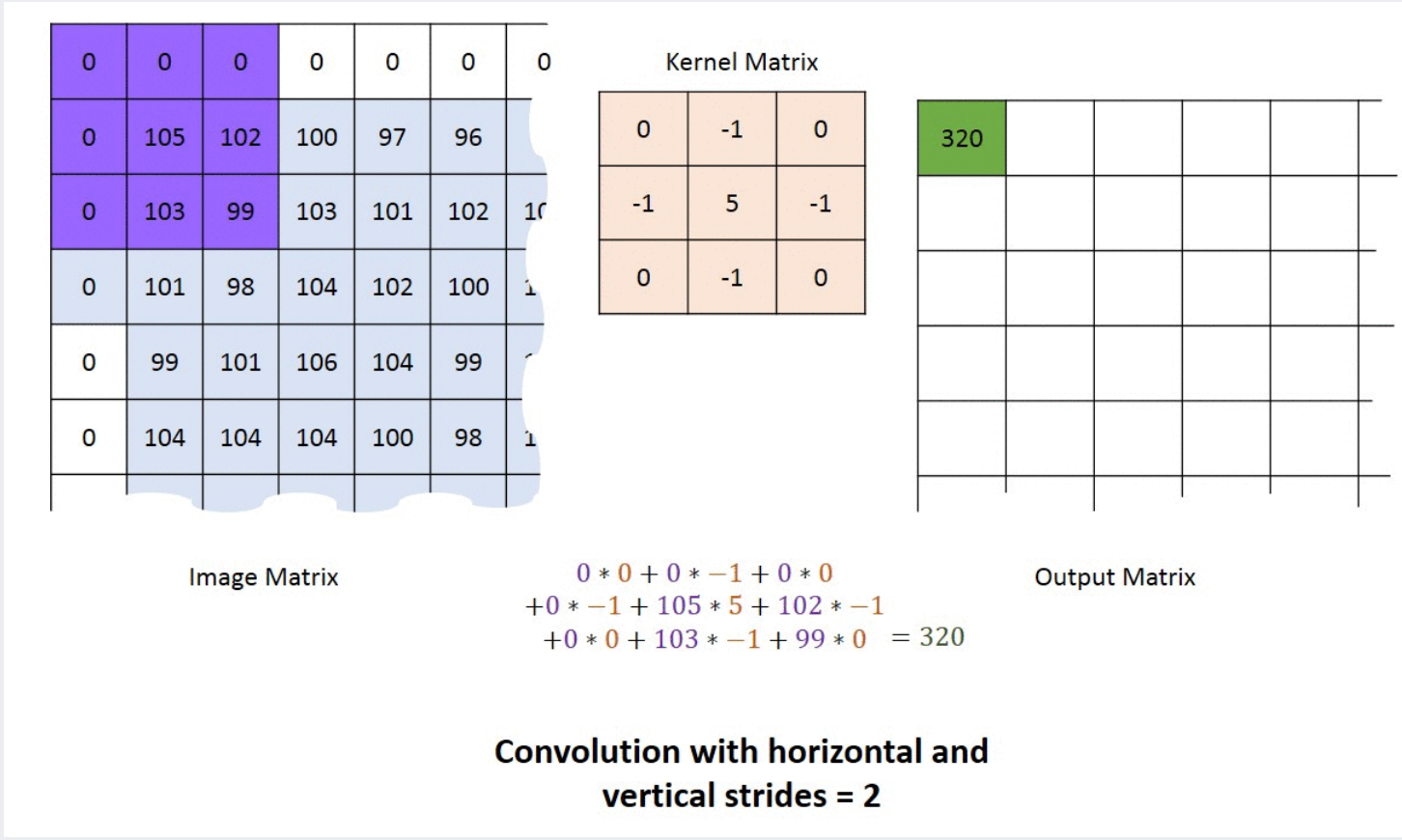
CNN Basics: Convolution

- Image Convolution (Filter, Kernel)
  - ✓ A matrix that capture a certain type of image feature



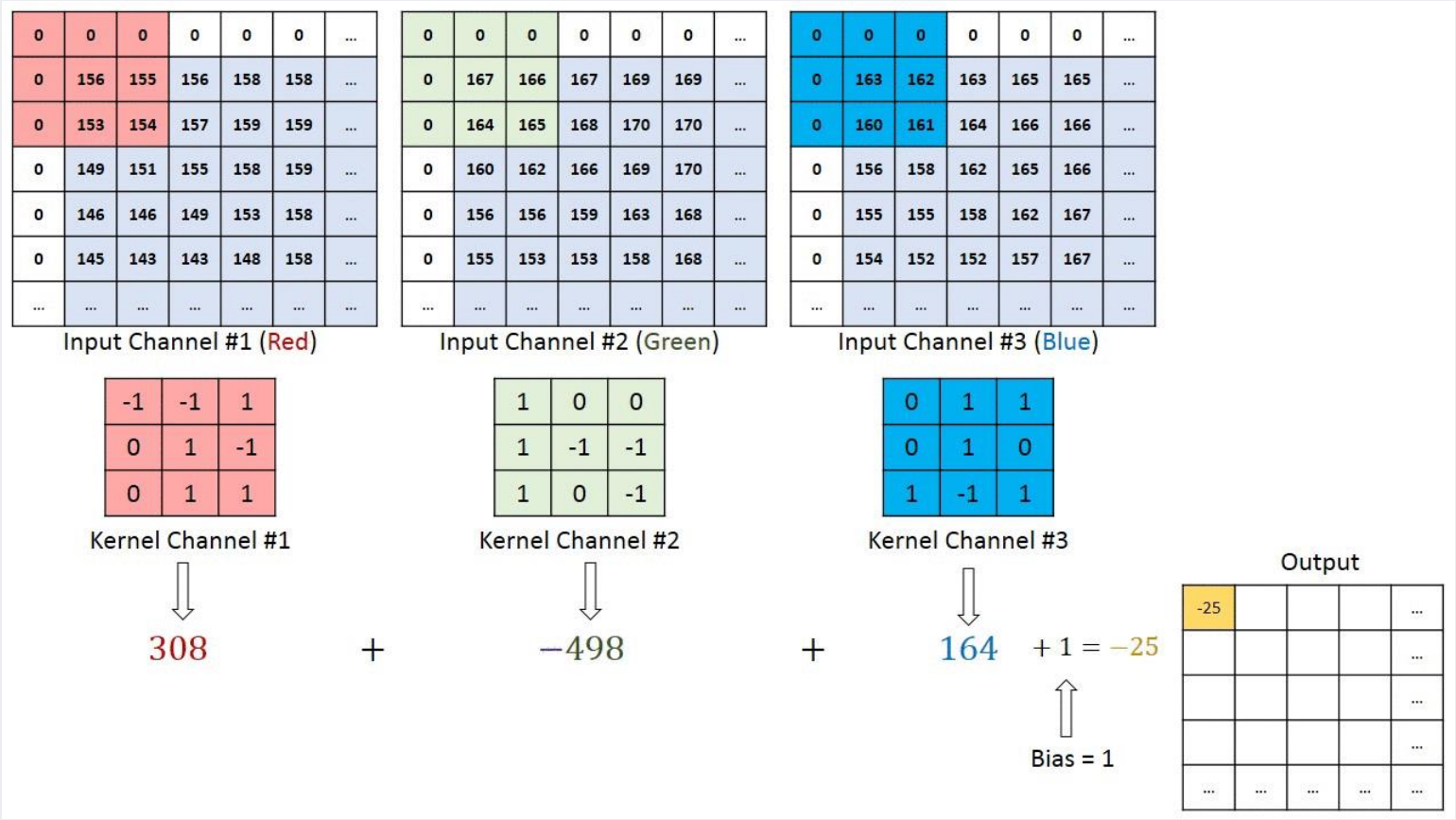
CNN Basics: Convolution

- Image Convolution (Filter, Kernel)
  - ✓ A matrix that capture a certain type of image feature



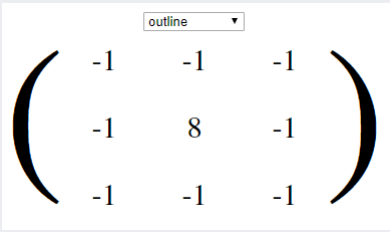
CNN Basics: Convolution

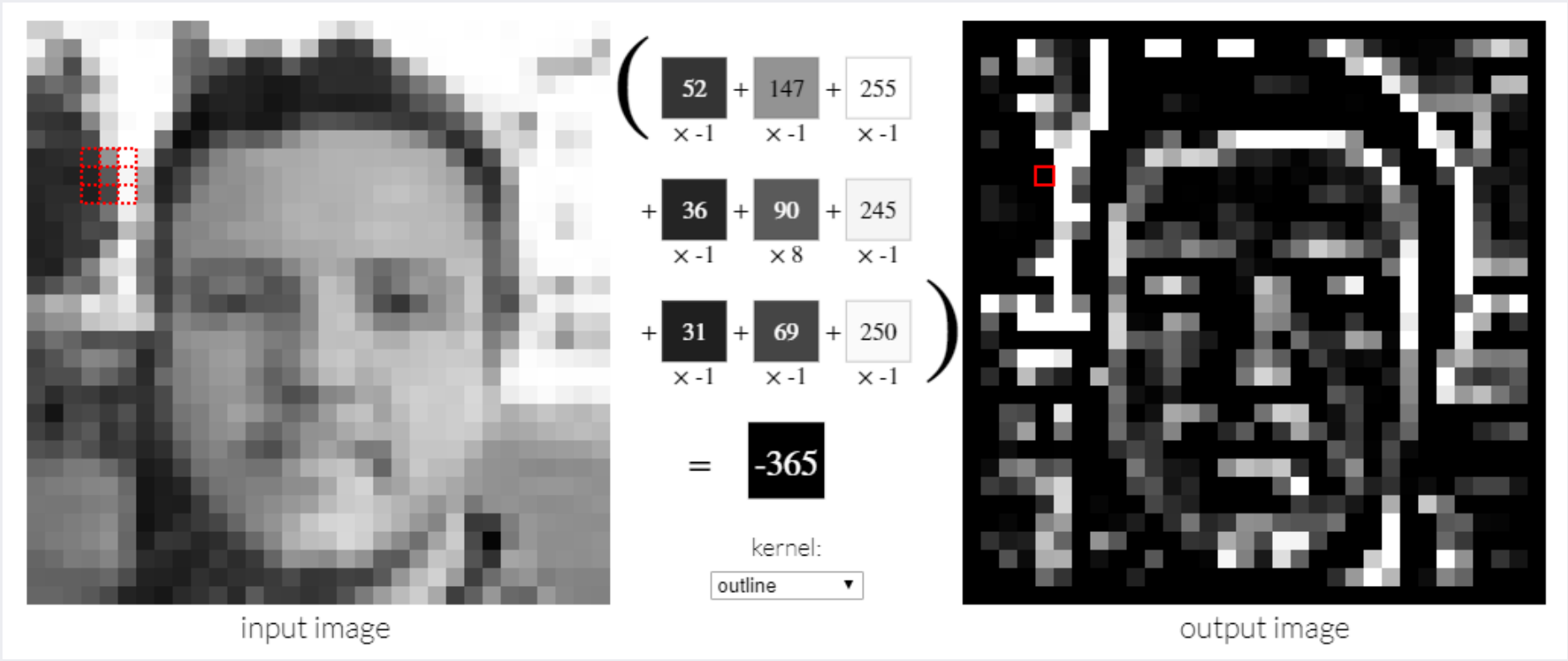
- Image Convolution (Filter, Kernel)
  - ✓ A matrix that capture a certain type of image feature



CNN Basics: Convolution

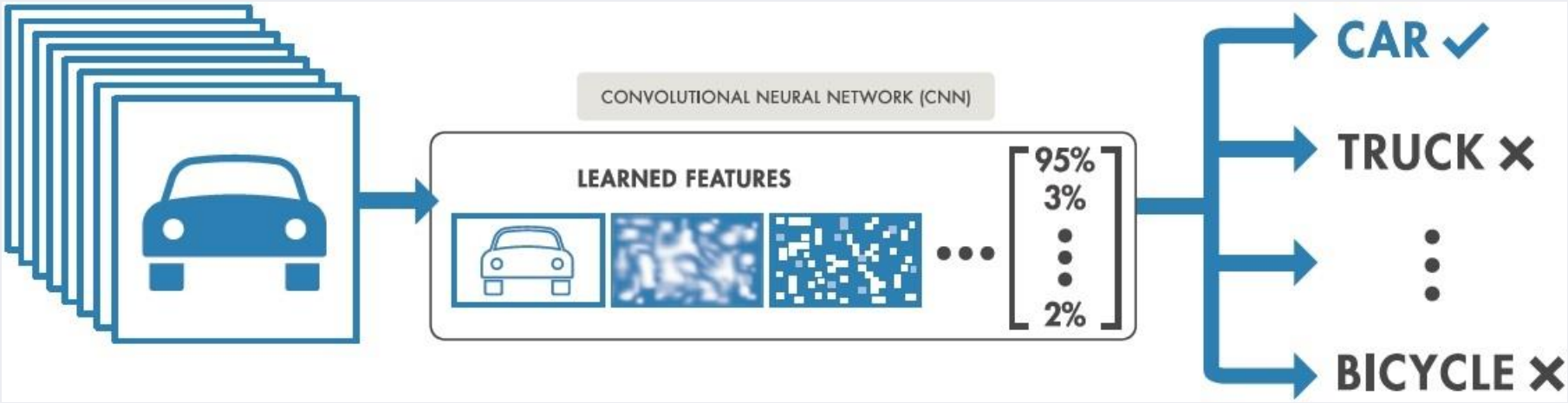
- A matrix that capture a certain type of image feature
  - ✓ Image filter example: http://setosa.io/ev/image-kernels/





CNN: Main Idea

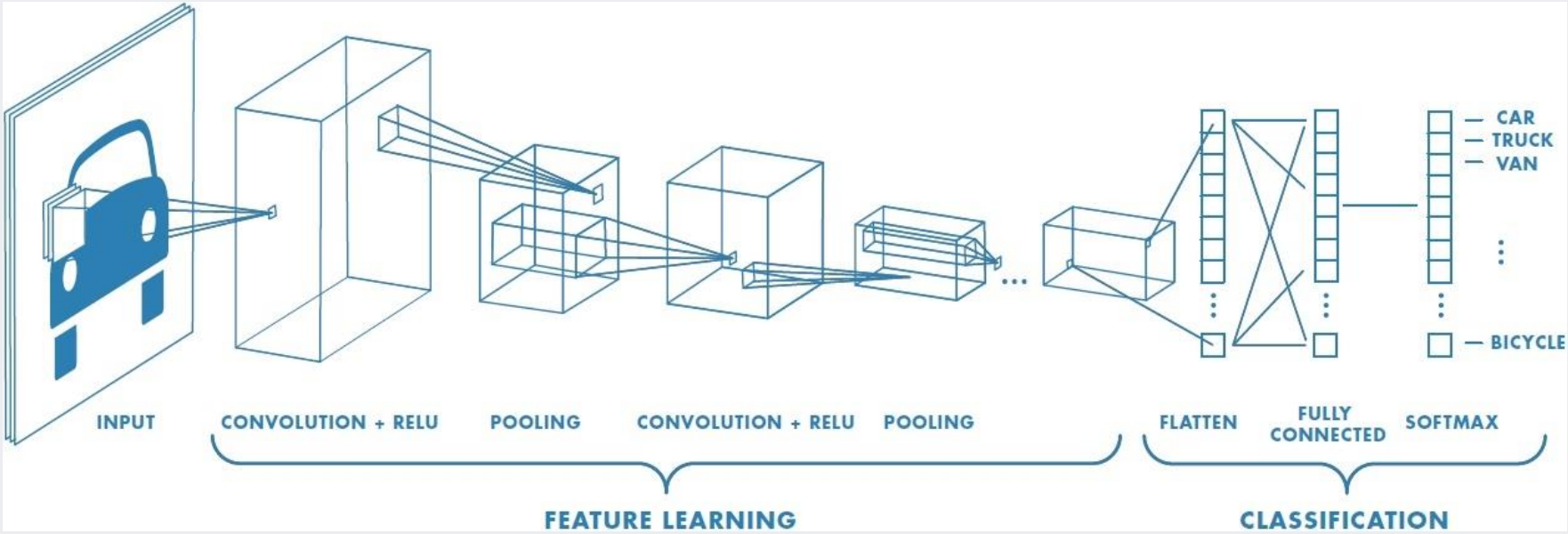
- What types of convolutions should be used?
  - ✓ Let the network learn a set of appropriate convolutions



https://kr.mathworks.com/discovery/convolutional-neural-network.html

CNN: Main Idea

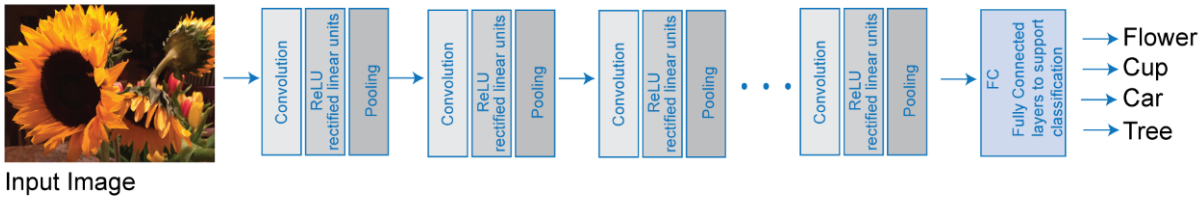
- What types of convolutions should be used?
  - ✓ Let the network learn a set of appropriate convolutions

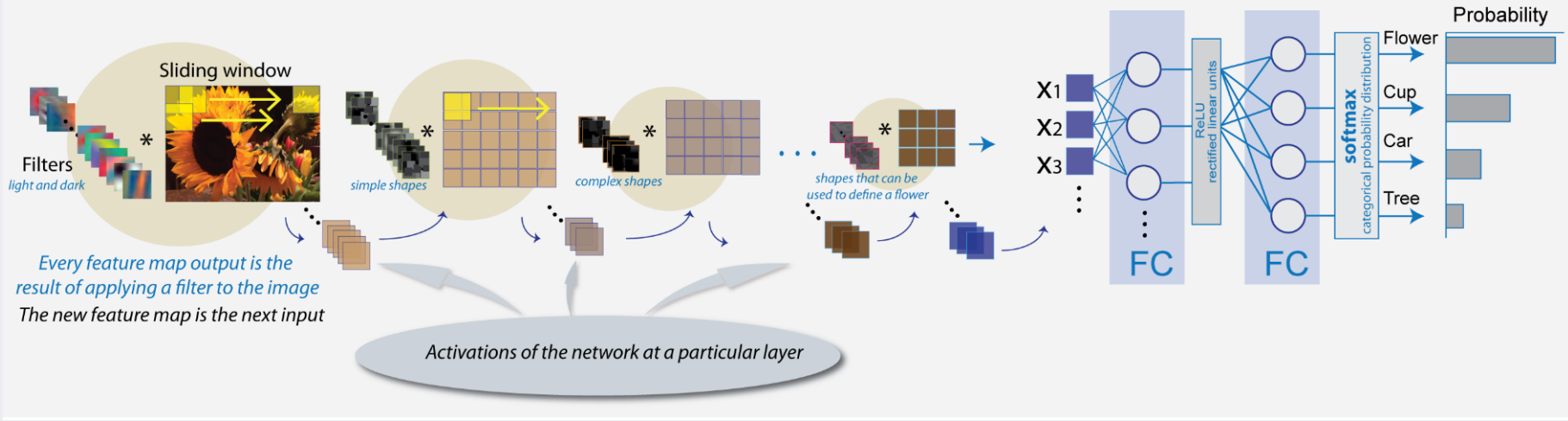


https://kr.mathworks.com/discovery/convolutional-neural-network.html

CNN: Basic Structure

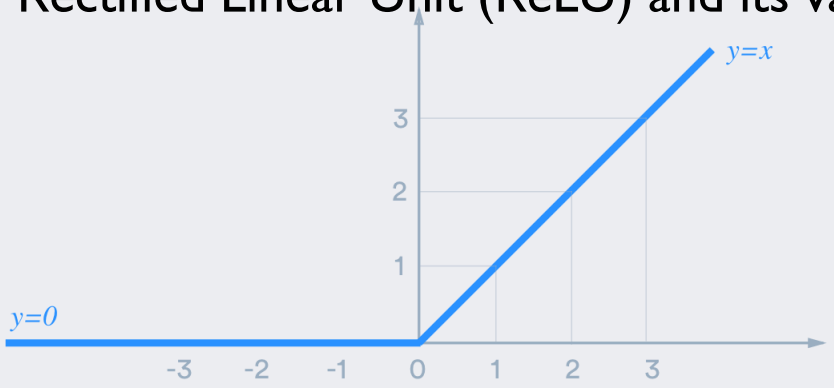
- In general, CNN consists of three main parts
  - ✓ Convolution: learn data-specific features
  - ✓ Activation (usually ReLU is used): non-linear mapping
  - ✓ Pooling: abstract the information

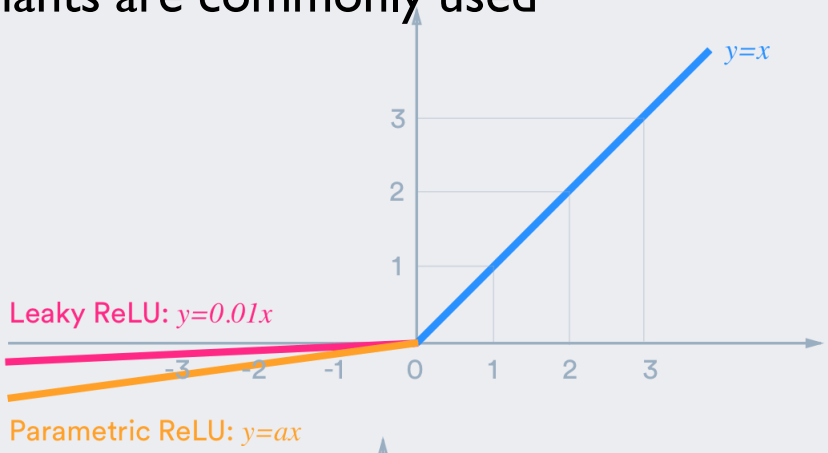


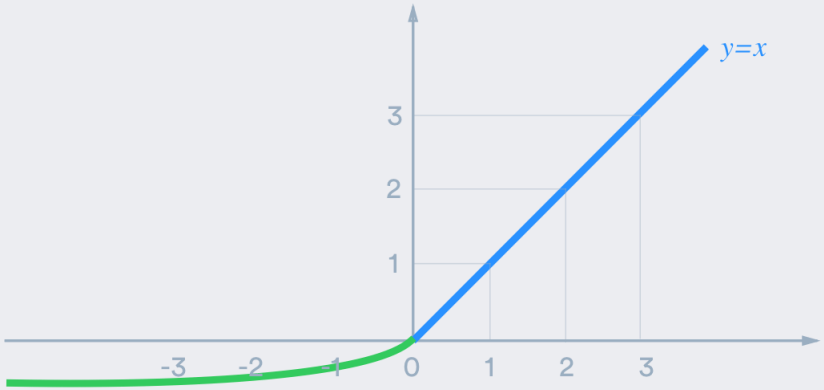


CNN: Activation

- Two main properties of an activation function
  - ✓ Non-linear mapping
  - ✓ Avoiding vanishing gradient problem
- Rectified Linear Unit (ReLU) and its variants are commonly used



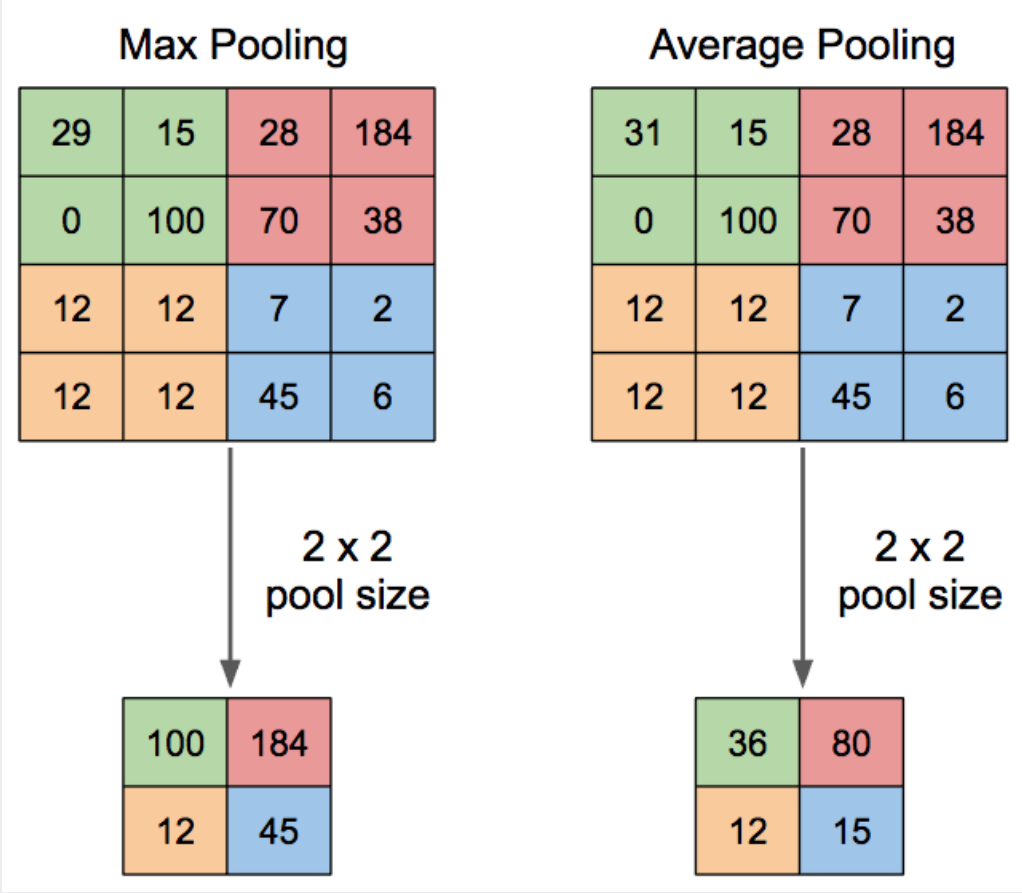




https://www.tinymind.com/learn/terms/relu

CNN: Pooling

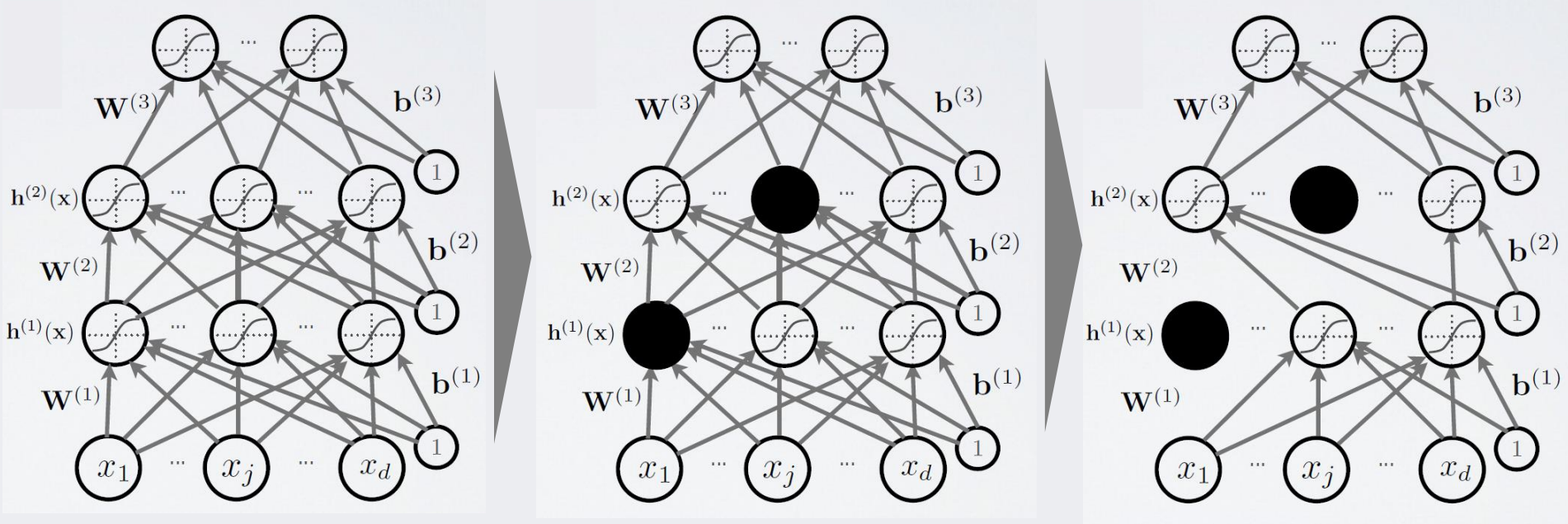
- Down-sampling operation to reduce the size of feature maps
  - ✓ Max pooling and Average pooling are two commonly used pooling scheme



https://pythonmachinelearning.pro/introduction-to-convolutional-neural-networks-for-vision-tasks/

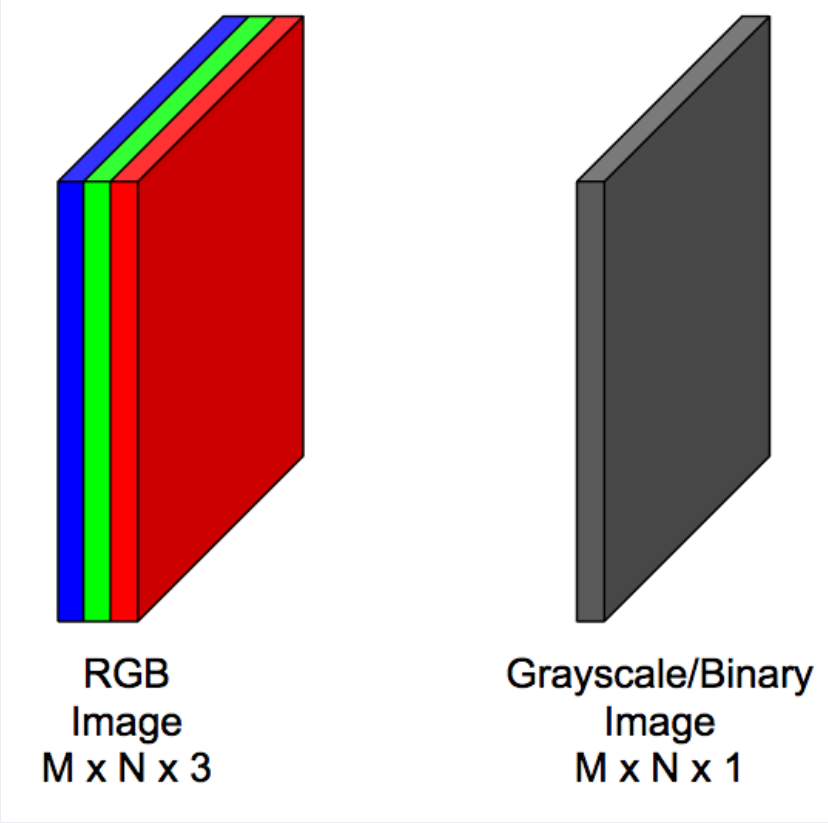
CNN: Dropout

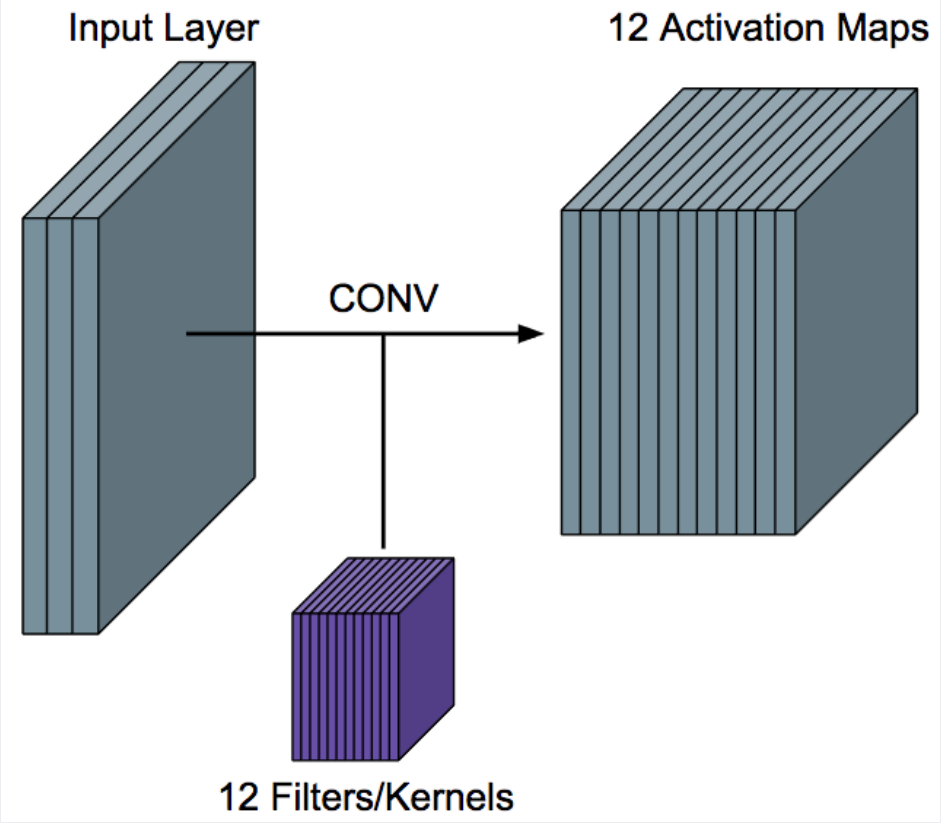
- Dropout (Srivastava et al. 2014)
  - ✓ Intentionally omit some nodes during the training
    - Weights connected to these nodes are not updated
  - ✓ These nodes are randomly selected for each batch



CNN: Summary

- Main Operations in CNNs
  - ✓ Begins with a raw input image
  - ✓ Applying convolutions for the previous layer



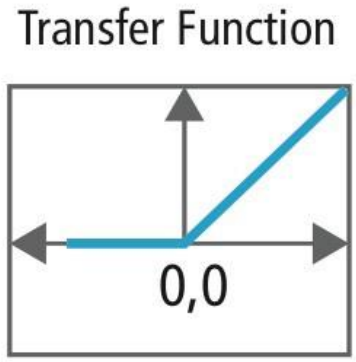


https://pythonmachinelearning.pro/introduction-to-convolutional-neural-networks-for-vision-tasks/

CNN: Summary

- ReLU activation for non-linear mapping

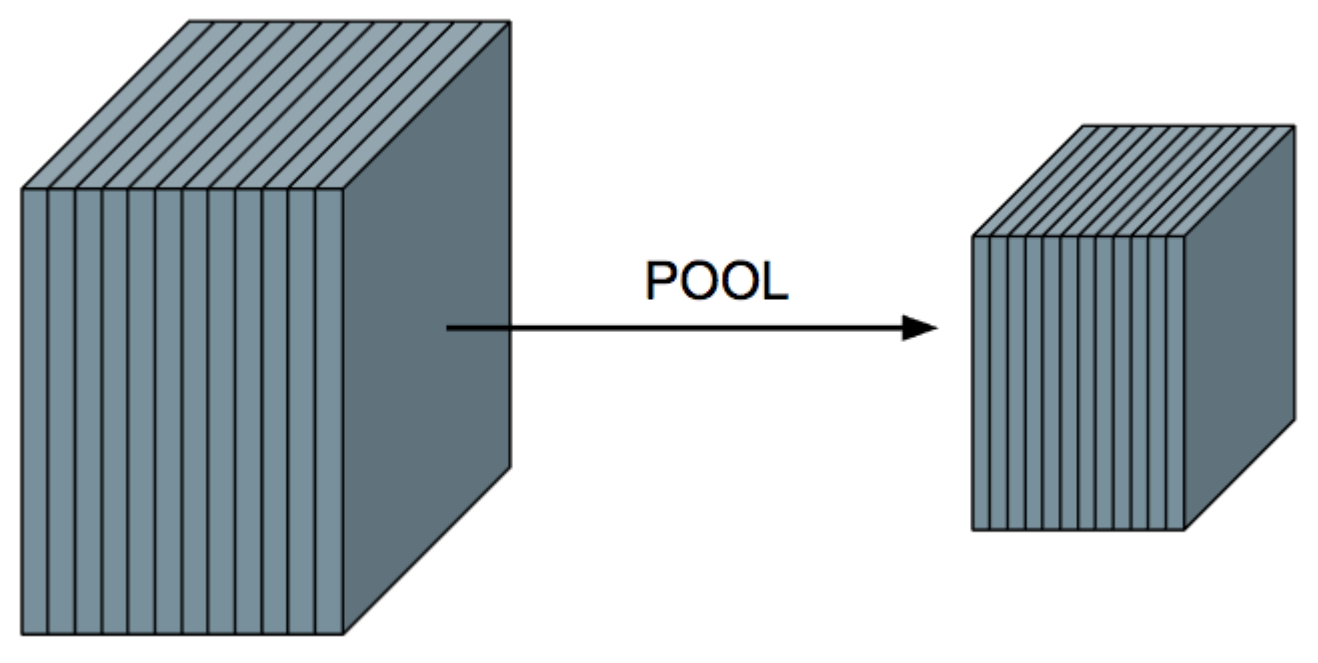
15	20	-10	35
18	-110	25	100
20	-15	25	-10
101	75	18	23



15	20	0	35
18	0	25	100
20	0	25	0
101	75	18	23

CNN: Summary

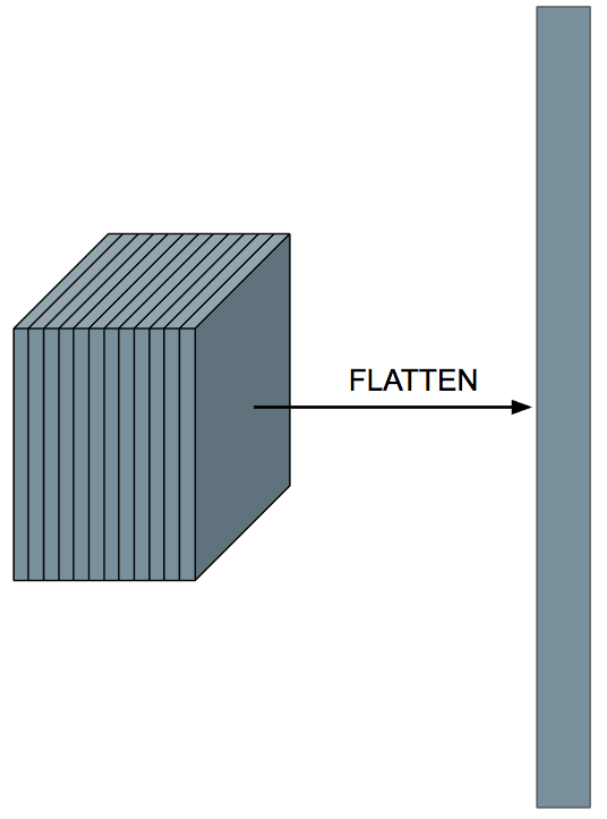
- Pooling operation for information abstraction



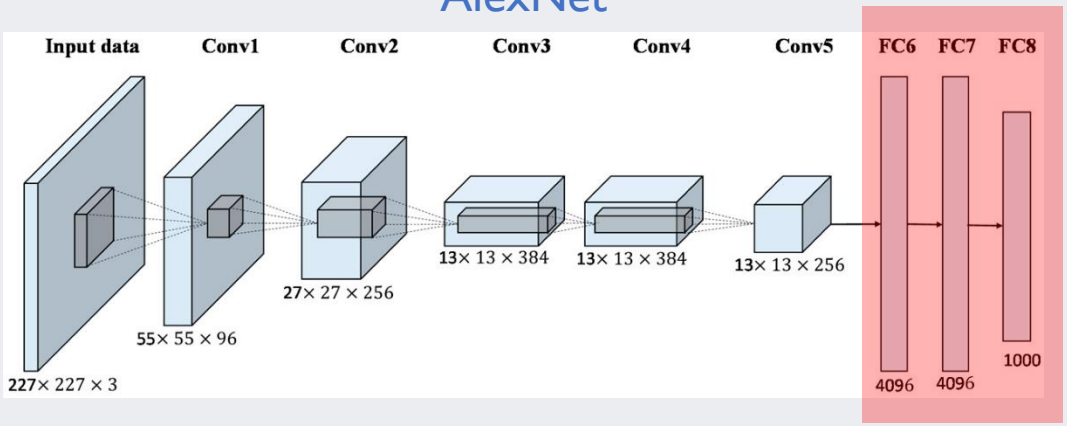
https://pythonmachinelearning.pro/introduction-to-convolutional-neural-networks-for-vision-tasks/

CNN: Summary

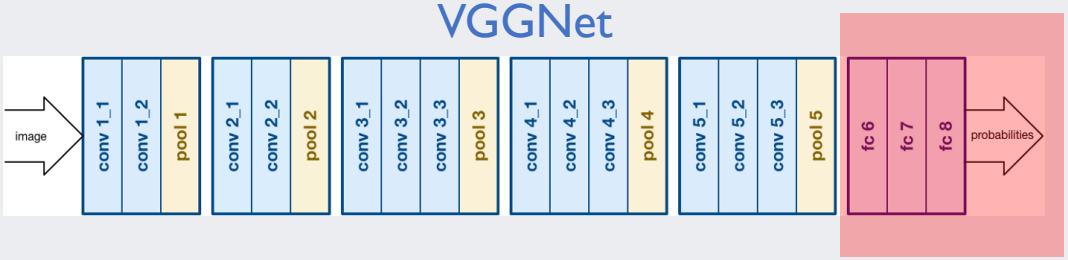
- Fully connected layer(s) to the output layer



AlexNet

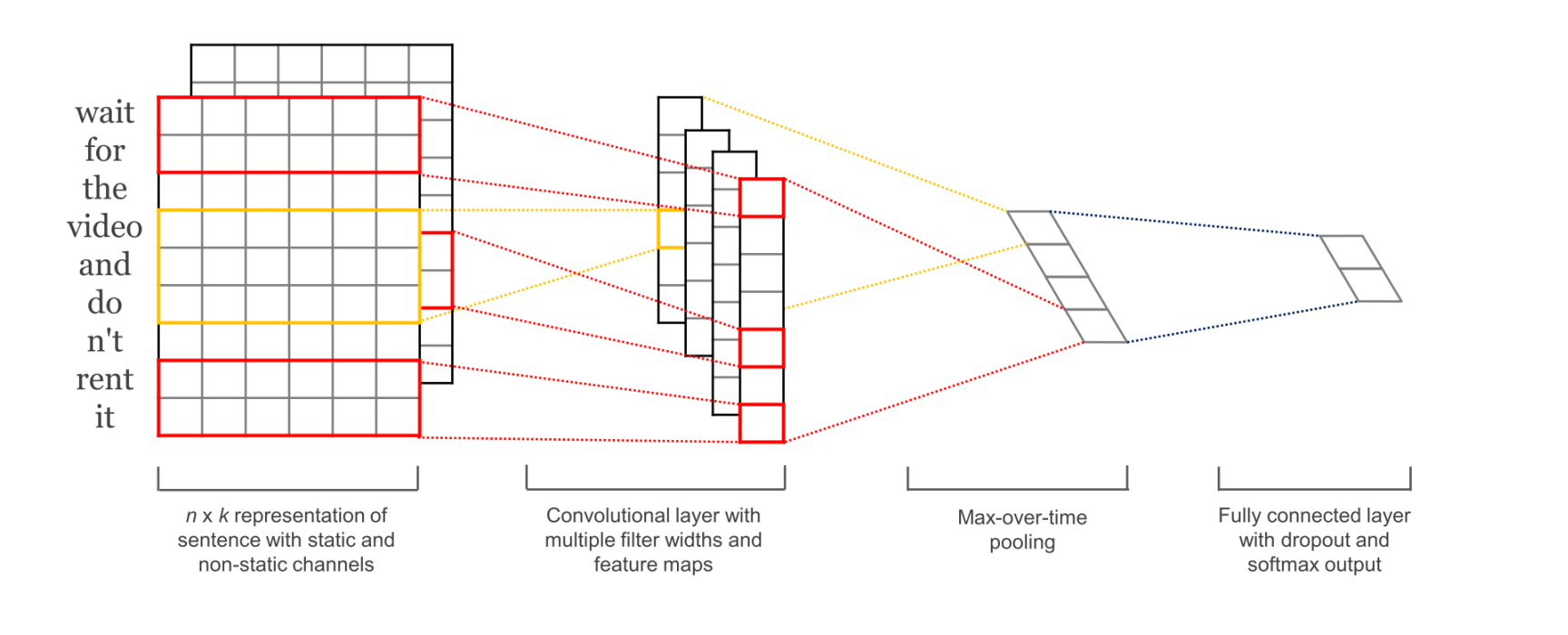


VGGNet

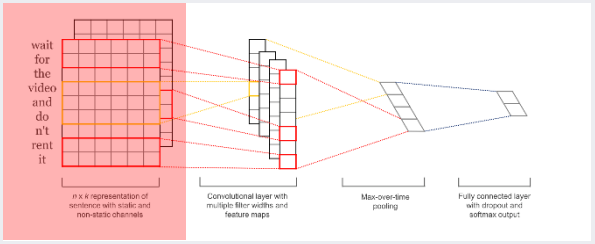


CNN for Sentence Classification

- Yoon Kim (2014)
  - ✓ A simple CNN structure with only one convolution layer

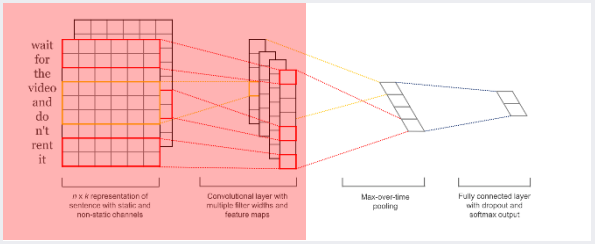


CNN for Sentence Classification



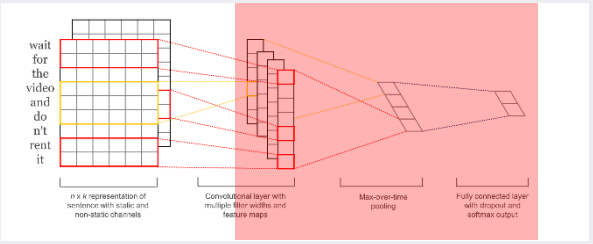
- Yoon Kim (2014)
  - ✓ Input: n by k (by m) representation of sentence
    - n: the number of words in a sentence (parameter)
      - Shorter sentences are zero-padded and longer sentences are trimmed
    - k: word embedding dimensions
      - Pre-trained word embedding vectors can be used
      - These vectors can be static (not updated during the training) or non-static (fine-tuned for the task-specific corpus)
    - Multi-channel input is also possible → an input sentence becomes a tensor
      - Pre-trained word embedding by Word2Vec (Static)
      - Pre-trained word embedding by Word2Vec (Non-static)
      - Pre-trained word embedding by Glove (Static)
      - Randomly initialized embedding
      - …

CNN for Sentence Classification



- Yoon Kim (2014)
  - ✓ Convolution
    - Different size of convolutions can be used
      - A squared convolution is common for image processing, but a rectangular convolution with the width of k is used for text processing
      - Convolution stride is usually set to 1
      - The larger the height, the more words are considered by the convolution at a single time

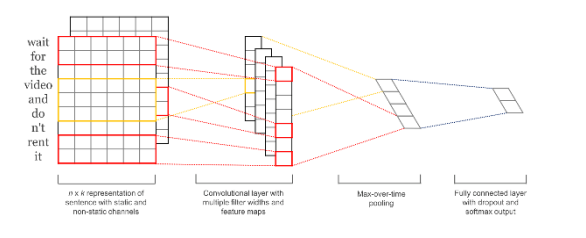
CNN for Sentence Classification



- Yoon Kim (2014)
  - ✓ Max pooling
    - To capture the most important feature
    - Can deal with variable sentence lengths
  - ✓ Fully connected operation
    - Connected to two output nodes (positive and negative)
  - ✓ Learning strategies
    - Filter window of 3, 4, and 5 with 100 feature maps each
    - Dropout (rate = 0.5) is used between the fully connected layer and the output layer
    - L_2 regularization (3) for the weight is used
    - Mini-batch size of 50
    - These values were chosen via a grid search on the SST-2 dev set

CNN for Sentence Classification

- Yoon Kim (2014): Experiments
  - ✓ Datasets

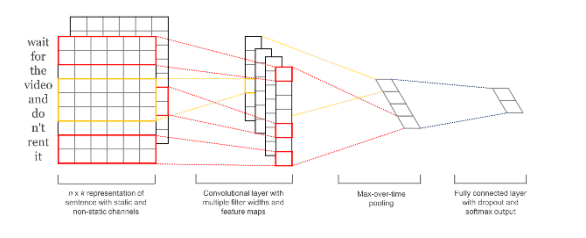


Data	c	l	N	$\|V\|$	$\|V_{pre}\|$	*Test*
MR	2	20	10662	18765	16448	CV
SST-1	5	18	11855	17836	16262	2210
SST-2	2	19	9613	16185	14838	1821
Subj	2	23	10000	21323	17913	CV
TREC	6	10	5952	9592	9125	500
CR	2	19	3775	5340	5046	CV
MPQA	2	3	10606	6246	6083	CV

Table 1: Summary statistics for the datasets after tokenization. c: Number of target classes. l: Average sentence length. N: Dataset size. $|V|$: Vocabulary size. $|V_{pre}|$: Number of words present in the set of pre-trained word vectors. *Test*: Test set size (CV means there was no standard train/test split and thus 10-fold CV was used).

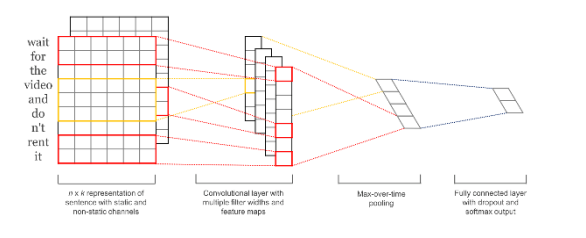
CNN for Sentence Classification

- Yoon Kim (2014): Experiments
  - ✓ Classification performance



Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	**89.6**
CNN-non-static	**81.5**	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	**88.1**	93.2	92.2	**85.0**	89.4
RAE (Socher et al., 2011)	77.7	43.2	82.4	–	–	–	86.4
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	–	–	–	–
RNTN (Socher et al., 2013)	–	45.7	85.4	–	–	–	–
DCNN (Kalchbrenner et al., 2014)	–	48.5	86.8	–	93.0	–	–
Paragraph-Vec (Le and Mikolov, 2014)	–	**48.7**	87.8	–	–	–	–
CCAE (Hermann and Blunsom, 2013)	77.8	–	–	–	–	–	87.2
Sent-Parser (Dong et al., 2014)	79.5	–	–	–	–	–	86.3
NBSVM (Wang and Manning, 2012)	79.4	–	–	93.2	–	81.8	86.3
MNB (Wang and Manning, 2012)	79.0	–	–	**93.6**	–	80.0	86.3
G-Dropout (Wang and Manning, 2013)	79.0	–	–	93.4	–	82.1	86.1
F-Dropout (Wang and Manning, 2013)	79.1	–	–	**93.6**	–	81.9	86.3
Tree-CRF (Nakagawa et al., 2010)	77.3	–	–	–	–	81.4	86.1
CRF-PR (Yang and Cardie, 2014)	–	–	–	–	–	82.7	–
SVM_S (Silva et al., 2011)	–	–	–	–	**95.0**	–	–

CNN for Sentence Classification



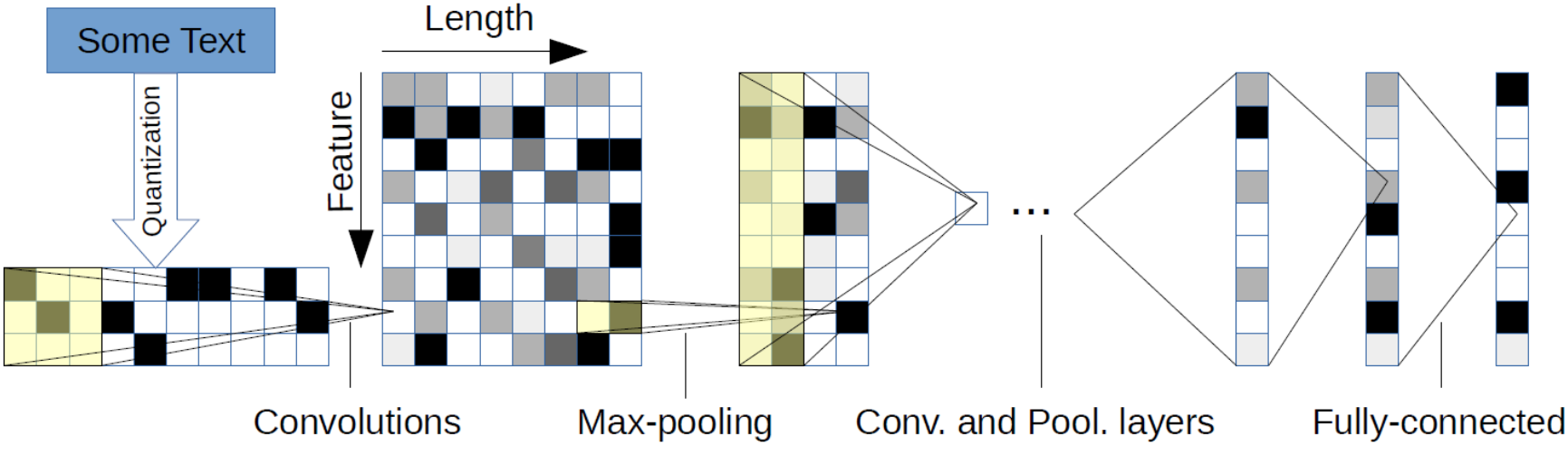
- Yoon Kim (2014): Experiments
  - ✓ Findings
    - A simple model with static vectors performs well, implying that the pre-trained vectors are good and universal feature extractors and can be utilized across datasets
    - Fine-tuning the pre-trained vector for each task gives further improvements
    - Multi-channel (static + non-static) does not work well as the author expected
    - Dropout proved to be a good regularizer; one can use a larger than necessary network and simply let dropout regularize it
      - Dropout consistently added 2%-4% relative performance
    - Adadelta gave similar results to Adagrad but required fewer epochs

CNN for Text Classification

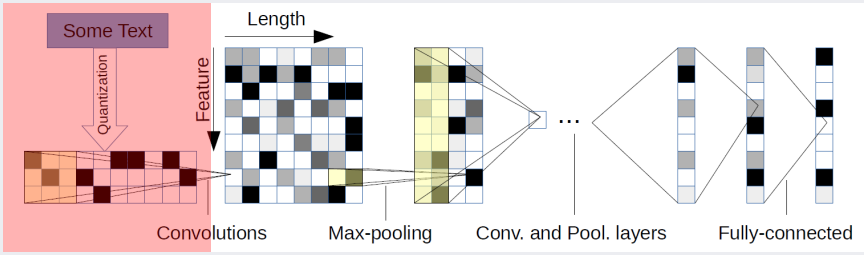
- Character-level CNN (Zhang et al., 2015)
  - ✓ A total of 70 characters are quantized
    - 26 English letters, 10 digits, and 33 other special characters (1 space)

```
abcdefghijklmnopqrstuvwxyz0123456789
-,;.!?:'''/\|_@#$%^&*~`+-=<>()[]{}
```

  - ✓ Model Structure



CNN for Text Classification



- Character-level CNN (Zhang et al., 2015)
  - ✓ Network structure
    - Large and Small models are designed
      - Convolution layers

Layer	Large Feature	Small Feature	Kernel	Pool
1	1024	256	7	3
2	1024	256	7	3
3	1024	256	3	N/A
4	1024	256	3	N/A
5	1024	256	3	N/A
6	1024	256	3	3

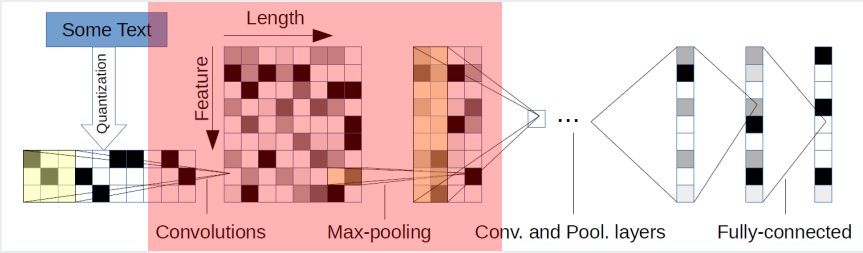
      - Fully connected layers

Layer	Output Units Large	Output Units Small
7	2048	1024
8	2048	1024
9	Depends on the problem	

    - Input: 70 by 1014 matrix
      - Each column in an one-hot vector for the corresponding character (not distributed representation)

CNN for Text Classification

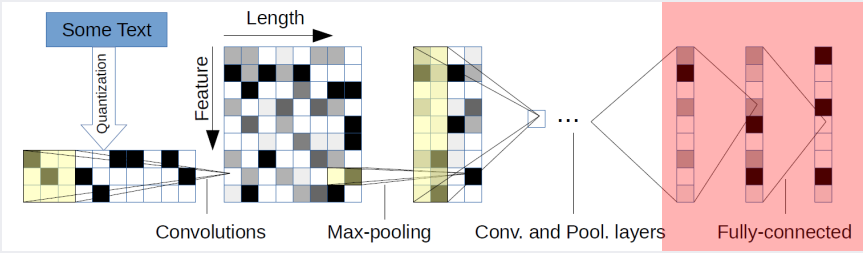
- Character-level CNN (Zhang et al., 2015)



  - ✓ Convolution
    - Feature size by Kernel size of convolutions are performance
      - (Large model) From the input to the first hidden layer: 70 by 7
      - (Large model) From the first to the second hidden layer: 1024 by 7
      - (Large model) From the second to the third layer: 1024 by 3
  - ✓ Max pooling
    - Max pooling with the size of (1 by 3) with the stride of 3 is performed for the first, second, and sixth layers
      - In Yoon Kim (2014), max pooling is performed only once for each feature

CNN for Text Classification

- Character-level CNN (Zhang et al., 2015)



✓ Fully connected layer

- The number of output nodes differs depending on the problem
- Dropout (rate = 0.5) is used between fully connected layers

✓ Data augmentation using thesaurus

- Augmentation methods used for image processing are not appropriate for text processing
- Human can augment text data well, but requires many resources
- Replace a word with its synonym

CNN for Text Classification

- Character-level CNN (Zhang et al., 2015)
  - ✓ Large-scale Datasets

Dataset	Classes	Train Samples	Test Samples	Epoch Size
AG's News	4	120,000	7,600	5,000
Sogou News	5	450,000	60,000	5,000
DBPedia	14	560,000	70,000	5,000
Yelp Review Polarity	2	560,000	38,000	5,000
Yelp Review Full	5	650,000	50,000	5,000
Yahoo! Answers	10	1,400,000	60,000	10,000
Amazon Review Full	5	3,000,000	650,000	30,000
Amazon Review Polarity	2	3,600,000	400,000	30,000

CNN for Text Classification

- Character-level CNN (Zhang et al., 2015)
  - ✓ Classification Performances

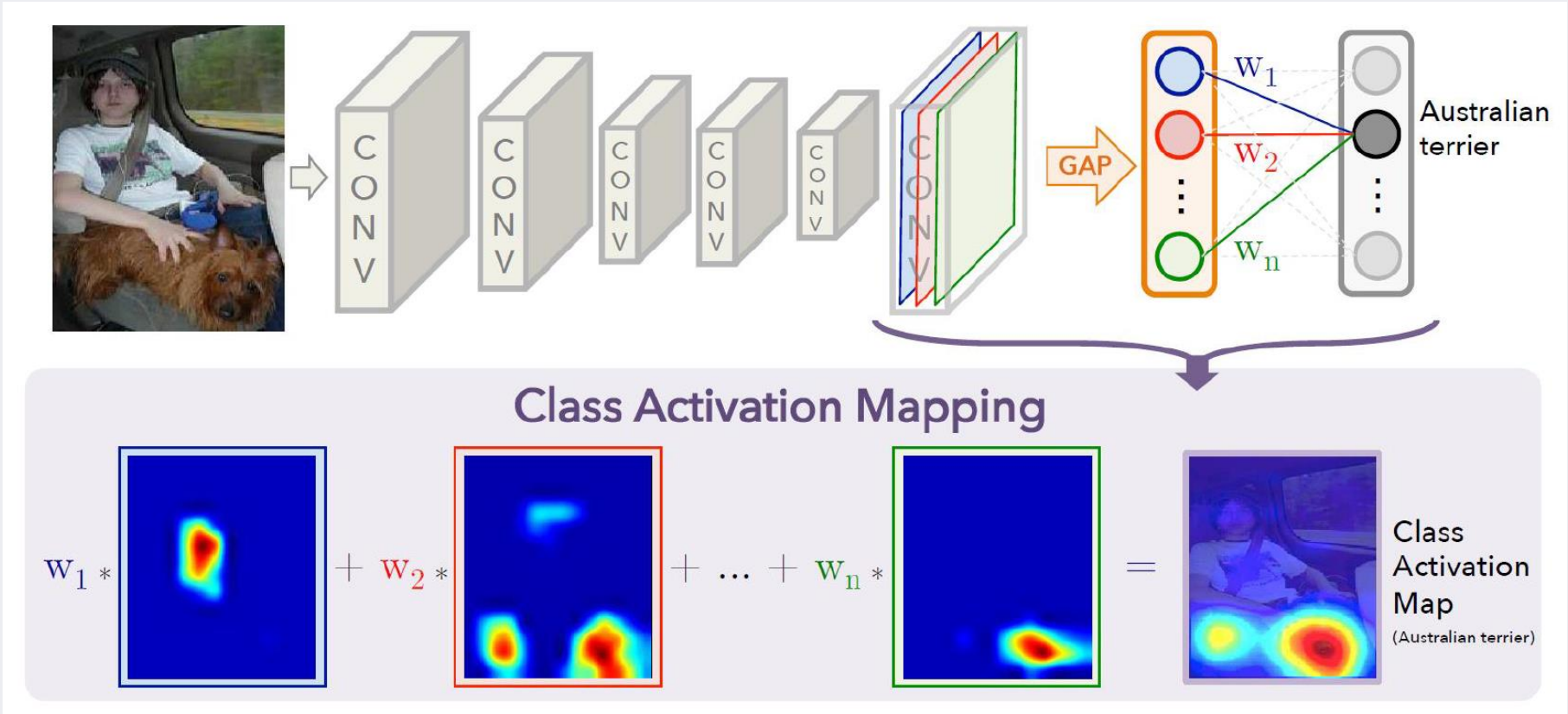
Model	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
BoW	11.19	7.15	3.39	7.76	42.01	31.11	45.36	9.60
BoW TFIDF	10.36	6.55	2.63	6.34	40.14	28.96	44.74	9.00
ngrams	7.96	2.92	1.37	**4.36**	43.74	31.53	45.73	7.98
ngrams TFIDF	**7.64**	**2.81**	**1.31**	4.56	45.20	31.49	47.56	8.46
Bag-of-means	**16.91**	**10.79**	**9.55**	**12.67**	**47.46**	**39.45**	**55.87**	**18.39**
LSTM	13.94	4.82	1.45	5.26	41.83	29.16	40.57	6.10
Lg. w2v Conv.	9.92	4.39	1.42	4.60	40.16	31.97	44.40	5.88
Sm. w2v Conv.	11.35	4.54	1.71	5.56	42.13	31.50	42.59	6.00
Lg. w2v Conv. Th.	9.91	-	1.37	4.63	39.58	31.23	43.75	5.80
Sm. w2v Conv. Th.	10.88	-	1.53	5.36	41.09	29.86	42.50	5.63
Lg. Lk. Conv.	8.55	4.95	1.72	4.89	40.52	29.06	45.95	5.84
Sm. Lk. Conv.	10.87	4.93	1.85	5.54	41.41	30.02	43.66	5.85
Lg. Lk. Conv. Th.	8.93	-	1.58	5.03	40.52	28.84	42.39	5.52
Sm. Lk. Conv. Th.	9.12	-	1.77	5.37	41.17	28.92	43.19	5.51
Lg. Full Conv.	9.85	8.80	1.66	5.25	38.40	29.90	40.89	5.78
Sm. Full Conv.	11.59	8.95	1.89	5.67	38.82	30.01	40.88	5.78
Lg. Full Conv. Th.	9.51	-	1.55	4.88	38.04	29.58	40.54	5.51
Sm. Full Conv. Th.	10.89	-	1.69	5.42	**37.95**	29.90	40.53	5.66
Lg. Conv.	12.82	4.88	1.73	5.89	39.62	29.55	41.31	5.51
Sm. Conv.	15.65	8.65	1.98	6.53	40.84	29.84	40.53	5.50
Lg. Conv. Th.	13.39	-	1.60	5.82	39.30	**28.80**	40.45	**4.93**
Sm. Conv. Th.	14.80	-	1.85	6.49	40.16	29.84	**40.43**	5.67

CNN for Text Classification

- Character-level CNN (Zhang et al., 2015)
  - ✓ Findings
    - Character-level CNN is an effective method
    - Dataset size forms a dichotomy between traditional and CNN models
    - CNN may work well for user-generated data
    - Choice of alphabet makes a difference
    - Semantics of tasks may not matter
    - Bag-of-means is a misuse of word2vec
    - There is no free lunch

CNN for Text Classification: Localization

- Image Localization
  - ✓ Class Activation Map (CAM) (Zhou et al., 2016)
    - Localize significant areas based only on class label information

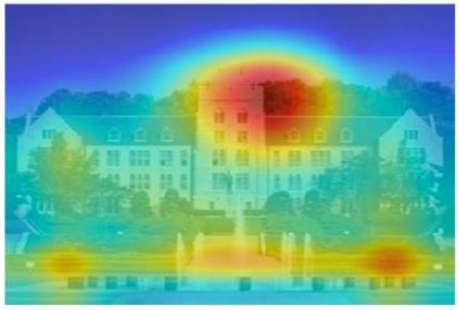


CNN for Text Classification: Localization

- Image Localization
  - ✓ Class Activation Map (CAM)
    - Localize significant areas based only on class label information

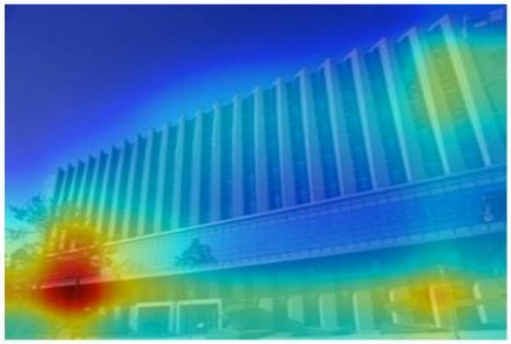
Predictions:

- **Type of environment:** outdoor
- **Semantic categories:** palace:0.23, formal_garden:0.20, mansion:0.15, castle:0.07, courthouse:0.06
- **SUN scene attributes:** man-made, openarea, naturallight, grass, vegetation, foliage, leaves, directsunsunny, trees, vacationingtouring
- **Informative region for the category *palace* is:**



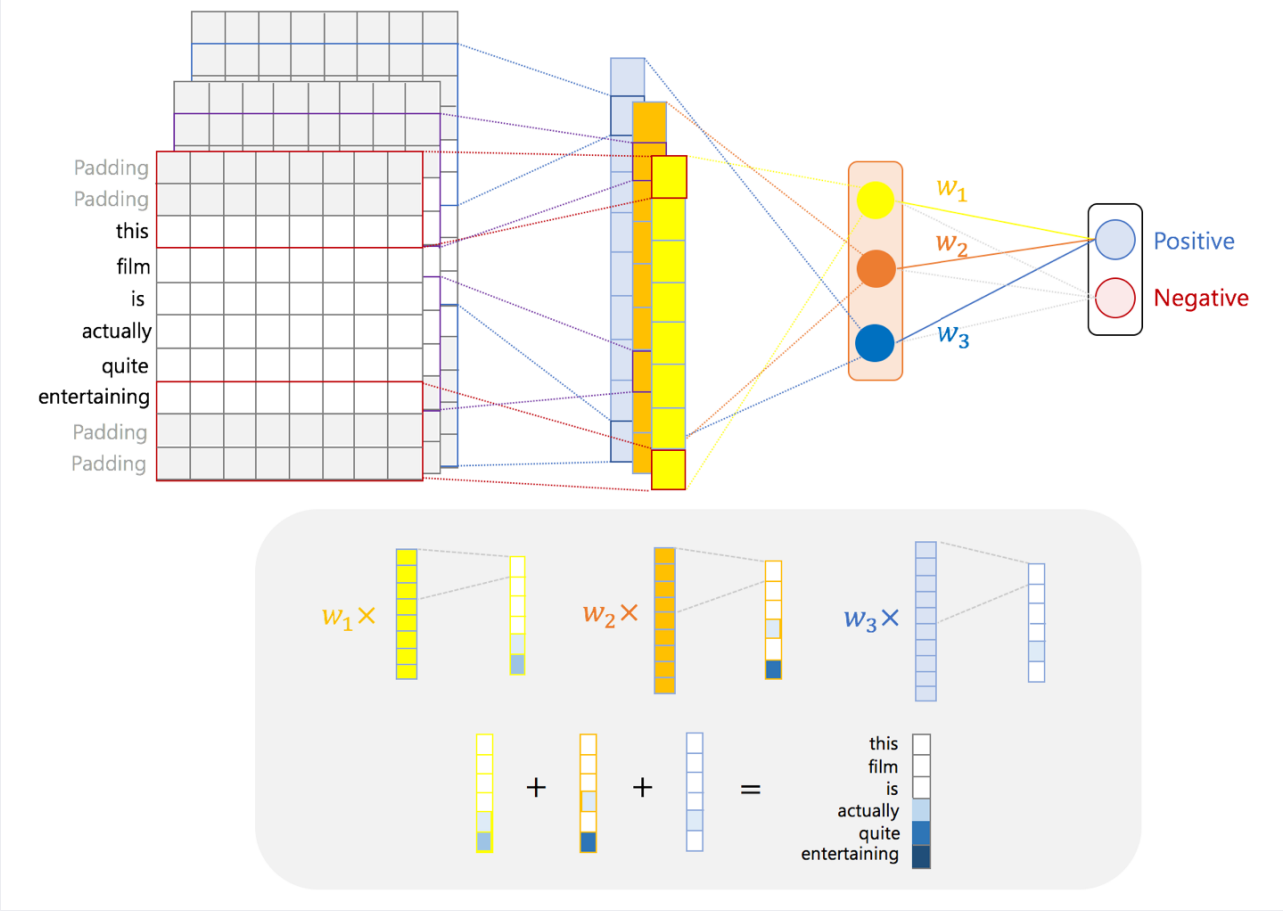
Predictions:

- **Type of environment:** outdoor
- **Semantic categories:** office_building:0.33, building_facade:0.24, hospital:0.17, skyscraper:0.06,
- **SUN scene attributes:** man-made, naturallight, openarea, mostlyverticalcomponents, directsunsunny, clouds, glass, mostlyhorizontalcomponents, semi-enclosedarea, metal
- **Informative region for the category *office_building* is:**



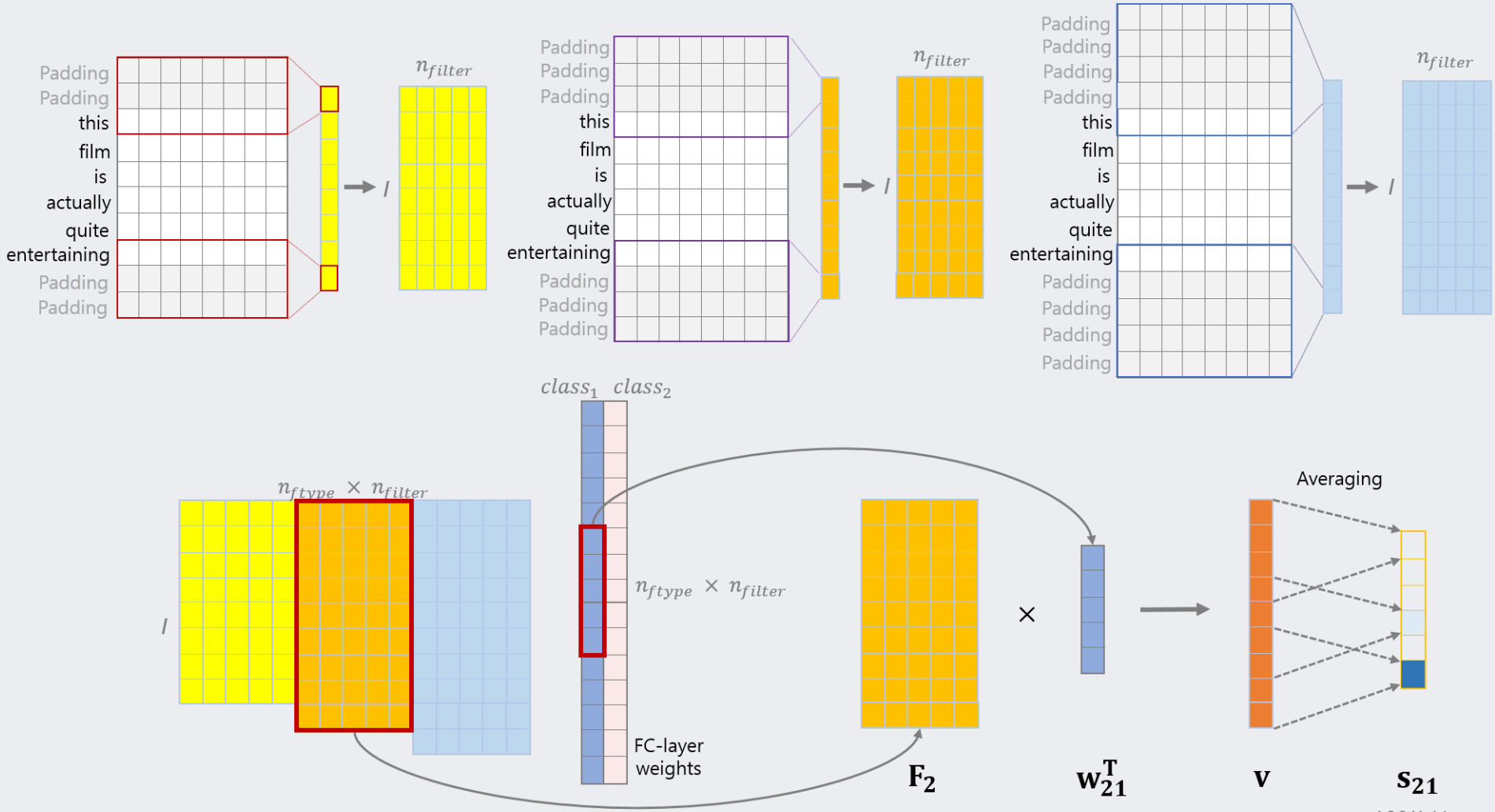
CNN for Text Classification: Localization

- Class Activation Map (CAM) for Sentiment Classification (Lee et al., 2018)



CNN for Text Classification: Localization

- Class Activation Map (CAM) for Sentiment Classification



CNN for Text Classification: Localization

- Class Activation Map (CAM) for Sentiment Classification
  - ✓ Two datasets: IMDB (English), WATCHA (Korean)

Table 1. Rating distribution for IMDB dataset

Rating	**1**	**2**	**3**	**4**	**7**	**8**	**9**	**10**
Reviews	10,122	4,586	4,961	5,531	4,803	5,859	4,607	9,731
Class	Negative				Positive			

Table 2. Rating distribution for the WATCHA dataset

Rating	**0.5**	**1**	**1.5**	**2**	**2.5**	**3**	**3.5**	**4**	**4.5**	**5.0**
Reviews	50,660	66,184	62,094	163,272	173,650	411,757	424,378	652,250	297,327	416,096
Class	Negative				Not used					Positive

Table 3. Information on datasets

Dataset	Number of tokens	Average number of words per document	Maximum number of words per document
IMDB	115,205	227.03	2,192
WATCHA	424,027	9.52	1,853

Table 4. CNN hyper-parameters

Filter type (window size)	**N. of Filters**	**Learning rate**	**Input Dimension**	**Dropout rate**	**L_2 regularization (λ)**	**Optimizer**	**Batch size**
3 (tri-gram) 4 (quad-gram) 5 (5-gram)	128 each	0.001	100	0.5	0.1	Adam	64

CNN for Text Classification: Localization

- Class Activation Map (CAM) for Sentiment Classification

Table 9. Frequently appearing words in positive/negative sentences in the IMDB test dataset (semantically positive or negative words are in blue and red fonts, respectively).

Positive					Negative				
CAM²-Rand	*CAM²-Static*	*CAM²-Non-Static*	*CAM²-2channel*	*CAM²-4channel*	*CAM²-Rand*	*CAM²-Static*	*CAM²-Non-Static*	*CAM²-2channel*	*CAM²-4channel*
and	and	and	and	and	the	is	the	the	and
is	great	is	is	is	and	and	and	and	the
the	very	excellent	excellent	the	worst	was	worst	worst	worst
excellent	a	the	the	a	of	the	of	of	of
a	is	a	a	excellent	a	bad	a	a	is
great	well	perfect	great	perfect	is	just	is	is	a
perfect	as	great	perfect	amazing	boring	this	awful	awful	awful
amazing	it	amazing	wonderful	enjoyed	waste	acting	waste	waste	waste
wonderful	good	wonderful	amazing	great	awful	plot	boring	boring	boring
of	i	enjoyed	enjoyed	of	was	a	was	was	was
s	film	i	as	i	this	of	this	this	this
i	wonderful	of	hilarious	well	to	boring	bad	to	bad
enjoyed	excellent	as	superb	it	i	script	movie	i	movie
superb	beautiful	hilarious	of	wonderful	movie	awful	to	movie	to
it	of	superb	i	s	bad	waste	terrible	terrible	i
hilarious	the	s	s	superb	terrible	movie	i	bad	acting
today	favorite	an	fun	hilarious	poor	stupid	poor	poor	poor
an	movie	well	today	fun	poorly	terrible	poorly	poorly	poorly
loved	comedy	definitely	well	loved	in	so	horrible	in	s
in	fun	it	an	an	s	no	dull	acting	just
job	in	enjoyable	was	very	it	completely	stupid	s	stupid
was	story	today	it	to	just	lame	acting	worse	in
as	my	was	definitely	was	film	poor	worse	dull	scirpt
fun	worth	very	job	today	with	are	s	with	horrible
to	highly	with	enjoyable	definitely	are	i	script	film	dull
with	enjoyed	surprised	in	with	acting	that	in	script	film
touching	loved	in	with	most	script	an	film	just	it
enjoyable	entertaining	fun	movie	enjoyable	stupid	or	lame	it	as
movie	also	entertaining	loved	good	save	crap	just	stupid	worse

CNN for Text Classification: Localization

- Class Activation Map (CAM) for Sentiment Classification

Positive					Negative				
CAM²-Rand	CAM²-Static	CAM²-Non-Static	CAM²-2channel	CAM²-4channel	CAM²-Rand	CAM²-Static	CAM²-Non-Static	CAM²-2channel	CAM²-4channel
영화	영화	영화	영화	영화	영화	영화	영화	영화	영화
최고의 (best)	수	그	수	너무	너무	너무	너무	너무	너무
수	최고의 (best)	너무	그	최고의 (best)	왜	왜	왜	왜	왜
그	그	최고의 (best)	너무	그	이	그냥	이	그냥	이
정말	이	그리고	최고의 (best)	수	그냥	없고 (none)	그냥	없고 (none)	없고 (none)
그리고	정말	수	그리고	그리고	더	없는 (none)	없고 (none)	이	그냥
너무	너무	더	이	정말	그	없다 (none)	없는 (none)	없다 (none)	없는 (none)
이	다시	가장	정말	진짜	없고 (none)	더	영화는	영화는	없다 (none)
가장	잘	정말	더	이	없는 (none)	느낌	그	없는 (none)	영화는
더	가장	다시	가장	가장	수	영화는	영화를	영화를	더
다시	그리고	진짜	잘	더	영화는	좀	영화가	다	영화가
잘	있는	최고 (best)	있는	잘	이런	뻔한 (obvious)	수	그	이런
최고 (best)	진짜	있는	최고 (best)	최고 (best)	없다 (none)	이	정말	더	수
내	내	이	보고	다시	영화를	영화가	없다 (none)	이런	영화를
다	모든	내	내	있는	정말	안 (not)	다	보는	내가
진짜	좋다 (good)	잘	진짜	좋다 (good)	것	차라리 (rather)	그나마	그나마	보는
완벽한 (perfect)	더	보고	다시	다	내가	것	내가	안 (not)	정말
있는	영화를	대한	영화를	보고	영화가	스토리	더	정말	것
영화를	또	내내	모든	완벽한 (perfect)	건	내	내	좀	그
본	아름다운 (beautiful)	한	다	내	다	영화를	좀	진짜	내
모든	보고	마지막	대한	봐도	이렇게	보는	이런	영화가	다
것	내가	모든	내가	모든	좀	이런	잘	것	스토리
보고	최고 (best)	것	마지막	마지막	한	건	건	차라리 (rather)	이렇게
한	한	내가	한	아름다운 (beautiful)	보는	무슨	봤는데	내	건
좋다 (good)	함께	또	것	또	느낌	듯	모르겠다	건	별
대한	꼭	다	아름다운 (beautiful)	영화를	진짜	모르겠다	것	만든	한
없는 (none)	작품	완벽한 (perfect)	영화가	좋았다 (good)	않는 (not)	전개	이렇게	수	좀
영화가	마지막	이런	이렇게	본	대한	전혀 (never)	차라리 (rather)	한	차라리 (rather)
봐도	완벽한 (perfect)	아름다운 (beautiful)	완벽한 (perfect)	것	내	본	안 (not)	내가	전혀 (never)

CNN for Text Classification: Localization

- Class Activation Map (CAM) for Sentiment Classification
  - ✓ Localization example for the IMDB dataset

CAM²-4channel Seeing as the vote average was pretty low and the fact that the clerk in the video store thoug ht was just OK I didn t t have much expectations when renting this film But contrary to the a bove I **enjoyed it** a lot This is acharming movie It didn t need to grow on me **I enjoyed it** fro m the beginning Mel Brooks gives a great performance as thelead character I think somewha t different from his usual persona in his movies There s not a lot of knockout jokes or someth inglike that but there are some rather hilarious scenes and overall this is a **very enjoyable an d very easy** to watch film **Very recommended** **Positive**

CAM²-4channel I hate this movie It is a horrid movie Sean Young s character is completely unsympathetic He r performance is wooden at best The storyline is completely **predictable and completely uni nteresting** I would never recommend this film to anyone It is one of the **worst** movies I have ever had the misfortune to see **Negative**

CNN for Text Classification: Localization

- Class Activation Map (CAM) for Sentiment Classification
  - ✓ Localization example for the WATCH dataset

CAM²-4channel	여지껏 봤던 영화중 **단연 최고라고** 할만한 작품이다 스토리 배경 시대배경과 영화배경 모두 감독의 카메라 기법 배우의 연기력 뭐하나 빠지는 것이 없다 **Positive**
CAM²-4channel	예술의 기본은 낯설게 하기다 그런 시도조차 보이지 않는다는 점 영화 중간중간 나오는 조명이 엄청나게 거슬렸다는 점이 **너무 싫었다** **Negative**

AGENDA

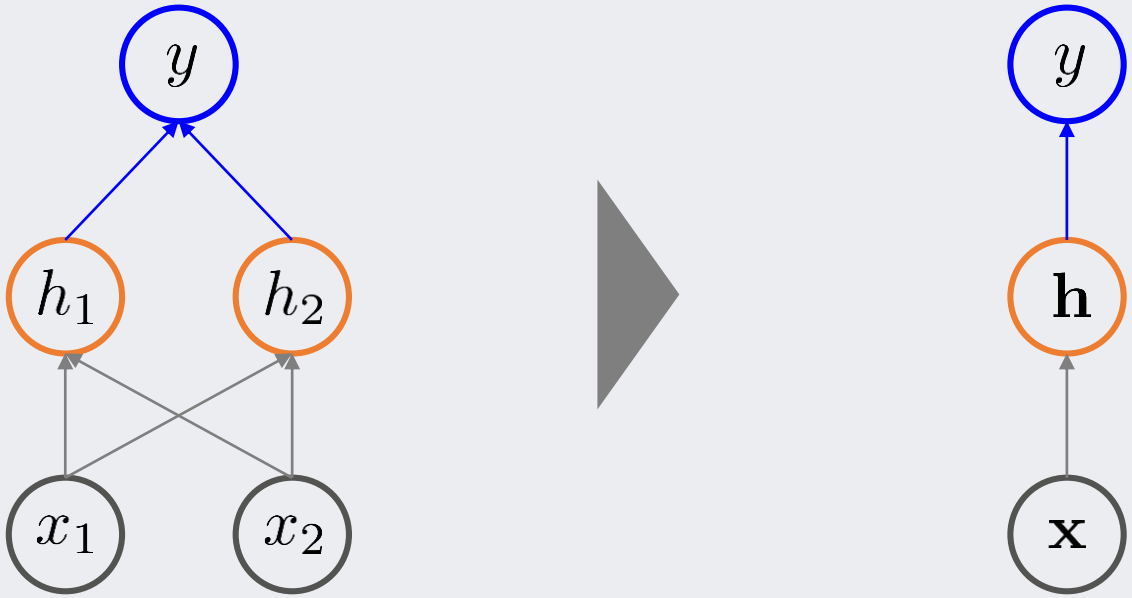
01 Neural Network: Overview

02 Convolutional Neural Network for Document Classification

03 Recurrent Neural Network for Document Classification

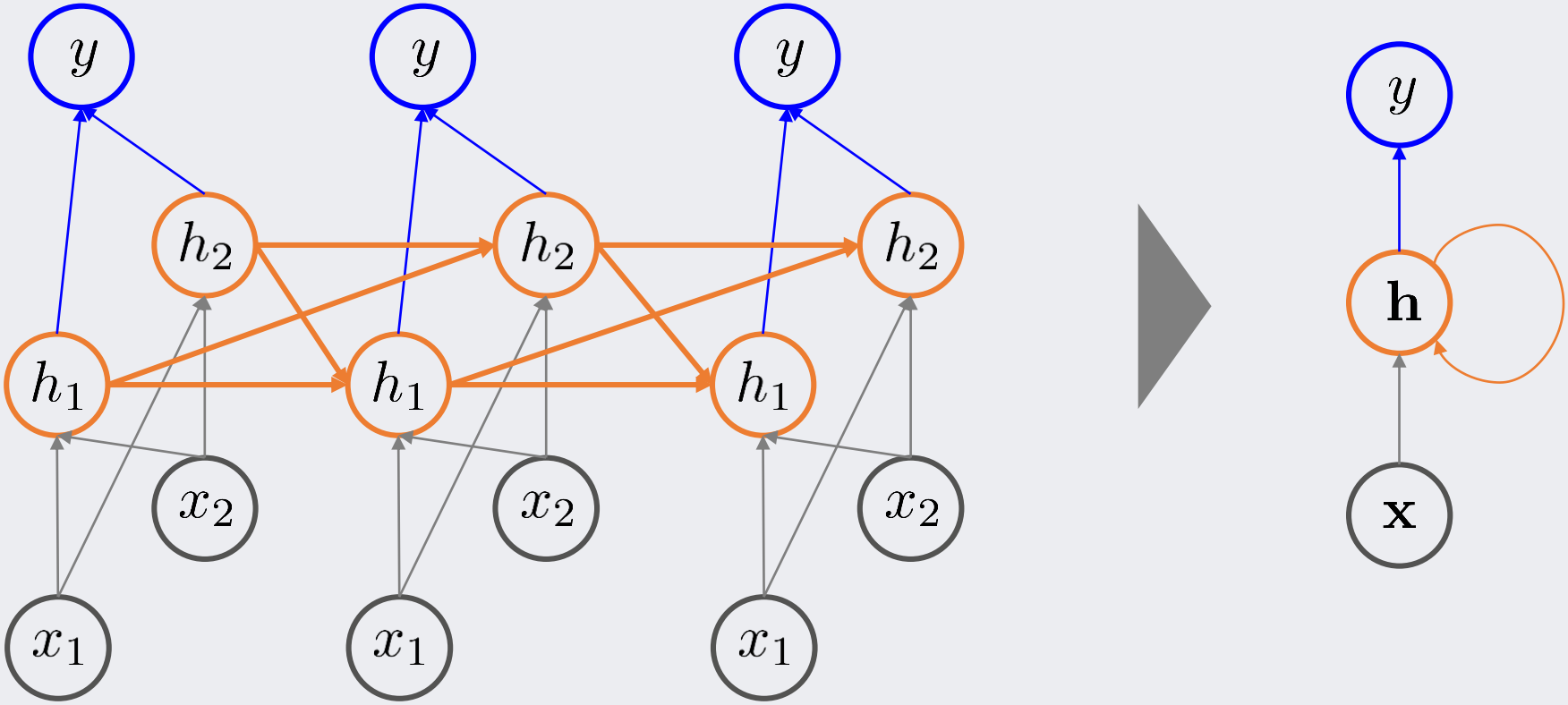
RNN Basics: Sequence

- NN structure without sequence



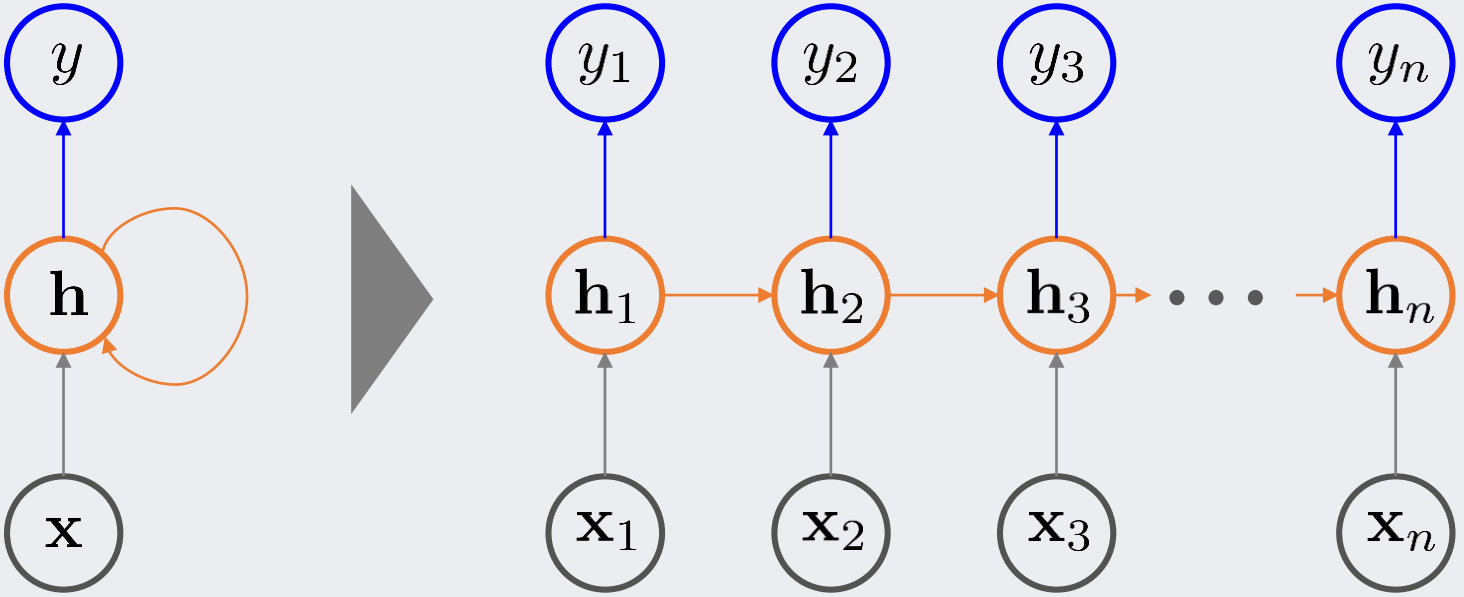
RNN Basics: Sequence

- NN structure with sequence



RNN Basics: Sequence

- NN structure with sequence



RNN Basics: Sequence

- NN structure with sequence
  - ✓ Input-Output structure

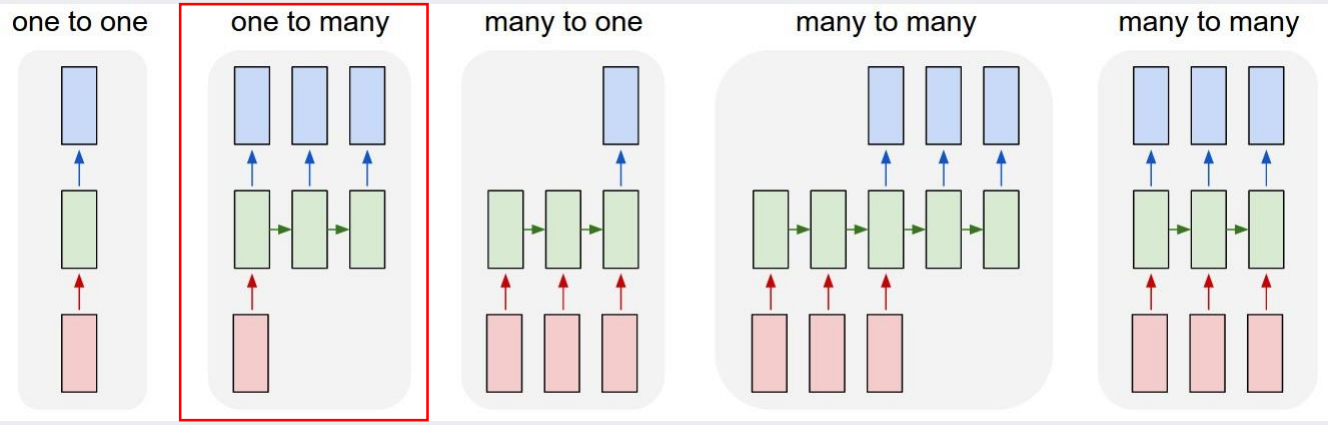
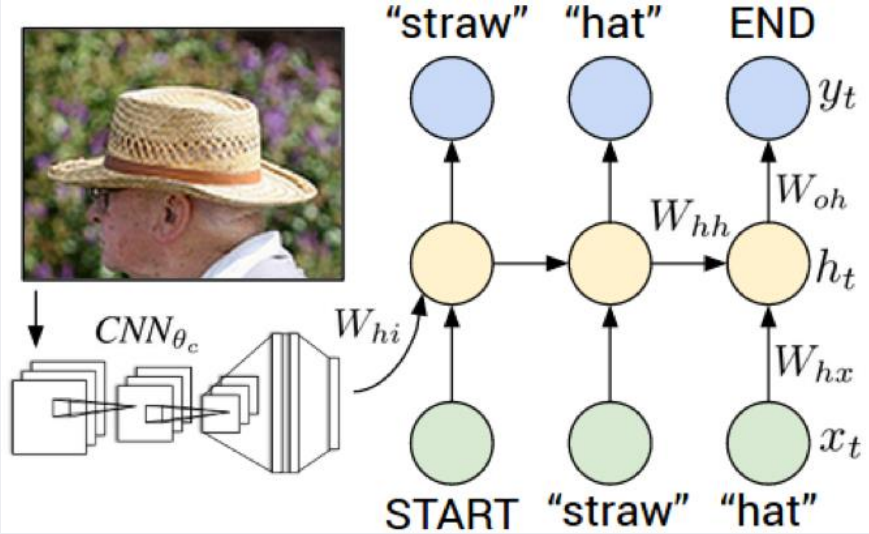
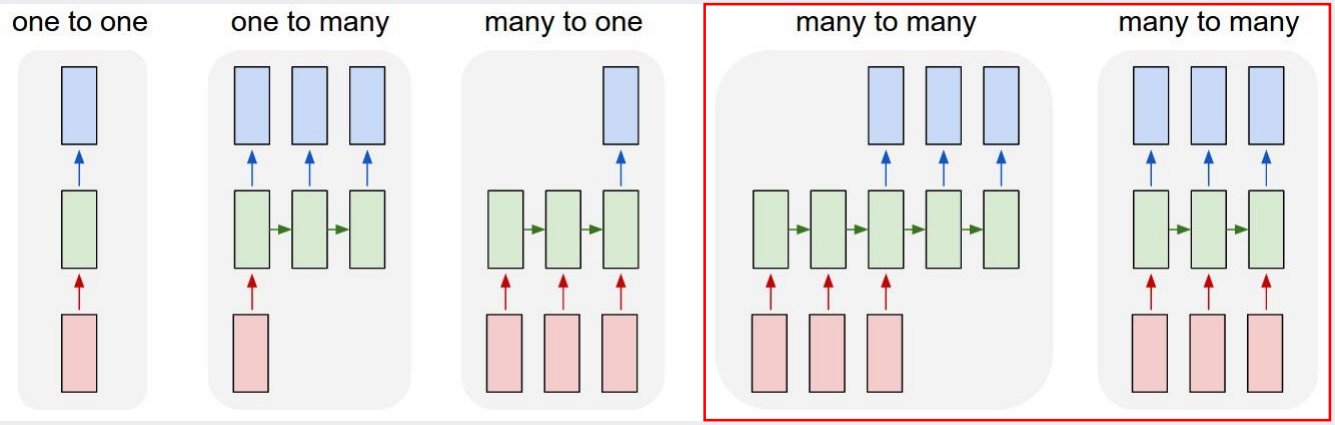


Image captioning

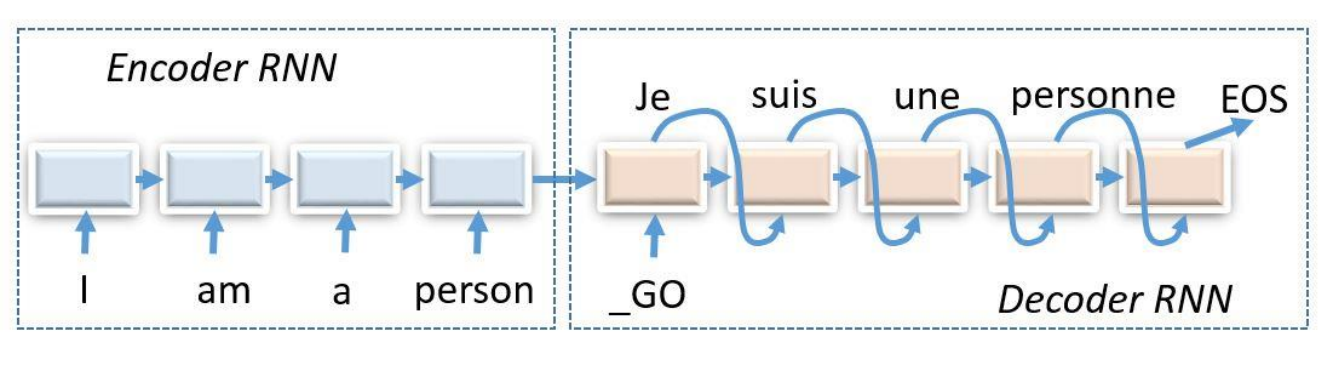


RNN Basics: Sequence

- NN structure with sequence
  - ✓ Input-Output structure



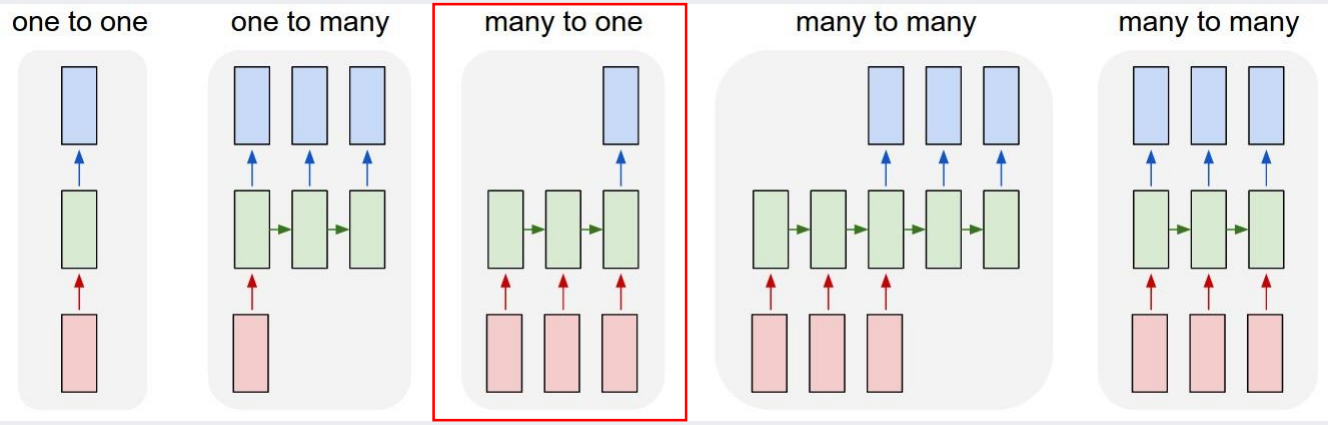
Machine translation/Dialog system



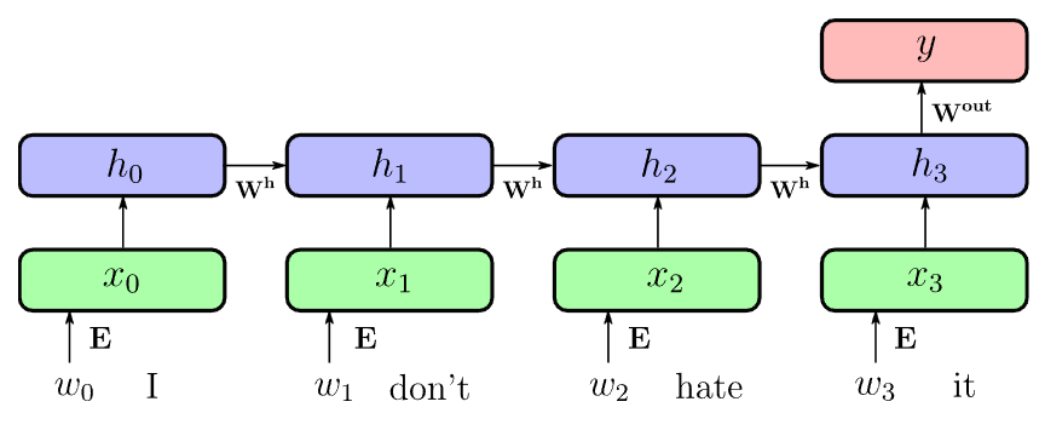
https://esciencegroup.com/2016/03/04/fun-with-recurrent-neural-nets-one-more-dive-into-cntk-and-tensorflow/

RNN Basics: Sequence

- NN structure with sequence
  - ✓ Input-Output structure

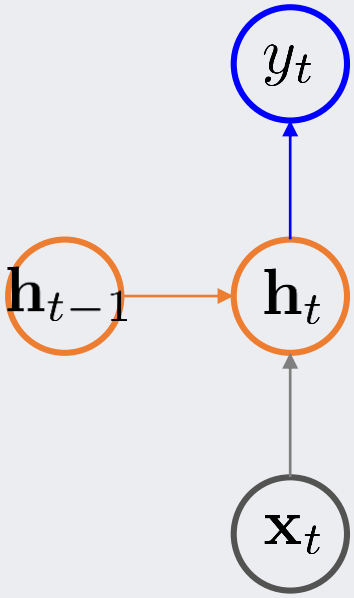


Text classification/Sentiment analysis



RNN Basics: Forward Path

- Information flow in Vanilla RNN
  - ✓ f( ) and g( ) are activation functions

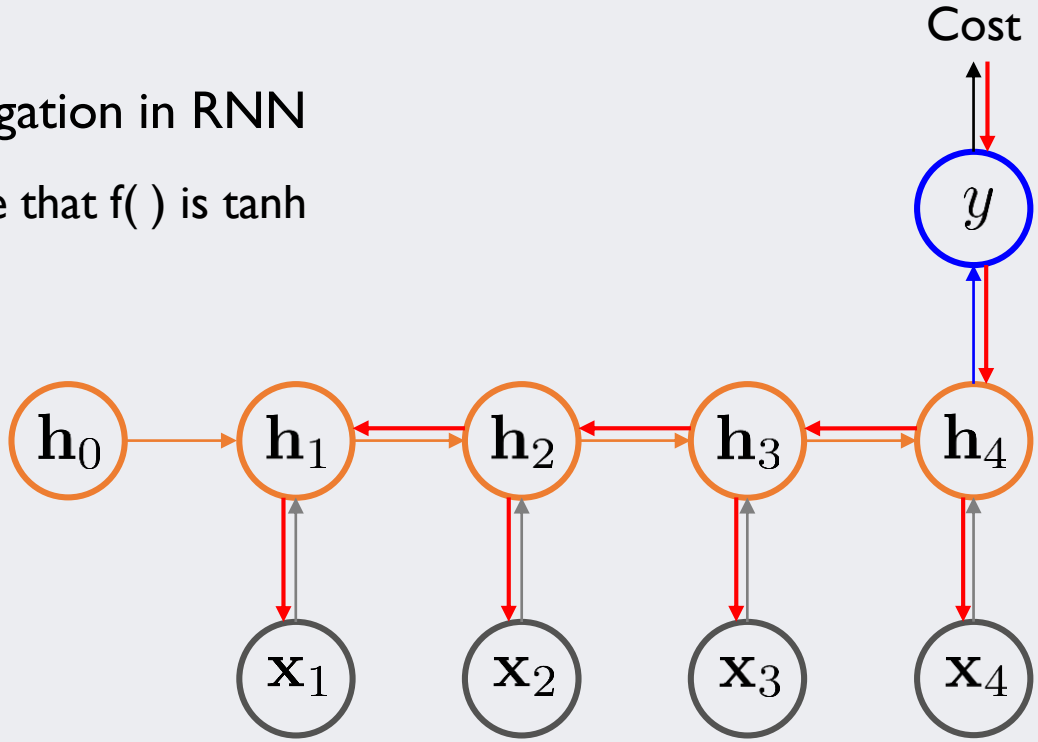


$$y_t = g(\mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y)$$

$$\mathbf{h}_t = f(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{b}_x)$$

RNN Basics: Gradient Vanishing/Exploding Problem

- Backpropagation in RNN
  - ✓ Assume that f( ) is tanh



$$\frac{\partial Cost}{\partial \mathbf{W}_{xh}} = \frac{\partial Cost}{\partial y} \times \frac{\partial y}{\partial \mathbf{h}_4} \times \frac{\partial \mathbf{h}_4}{\partial \mathbf{W}_{xh}} + \frac{\partial Cost}{\partial y} \times \frac{\partial y}{\partial \mathbf{h}_4} \times \frac{\partial \mathbf{h}_4}{\partial \mathbf{h}_3} \times \frac{\partial \mathbf{h}_3}{\partial \mathbf{W}_{xh}}$$
$$+ \frac{\partial Cost}{\partial y} \times \frac{\partial y}{\partial \mathbf{h}_4} \times \frac{\partial \mathbf{h}_4}{\partial \mathbf{h}_3} \times \frac{\partial \mathbf{h}_3}{\partial \mathbf{h}_2} \times \frac{\partial \mathbf{h}_2}{\partial \mathbf{W}_{xh}}$$
$$+ \frac{\partial Cost}{\partial y} \times \frac{\partial y}{\partial \mathbf{h}_4} \times \frac{\partial \mathbf{h}_4}{\partial \mathbf{h}_3} \times \frac{\partial \mathbf{h}_3}{\partial \mathbf{h}_2} \times \frac{\partial \mathbf{h}_2}{\partial \mathbf{h}_1} \times \frac{\partial \mathbf{h}_1}{\partial \mathbf{W}_{xh}}$$

RNN Basics: Gradient Vanishing/Exploding Problem

- Backpropagation in RNN
  - ✓ In general

$$\frac{\partial Cost}{\partial \mathbf{W}_{xh}} = \sum_{i=1}^{n} \left(\frac{\partial Cost}{\partial y} \cdot \frac{\partial y}{\partial \mathbf{h}_n} \cdot \left(\prod_{j=i}^{n-1} \frac{\partial \mathbf{h}_{j+1}}{\partial \mathbf{h}_j} \right) \cdot \frac{\partial \mathbf{h}_i}{\partial \mathbf{W}_{xh}} \right)$$

  - ✓ Recall that f( ) is tanh and

$$\mathbf{h}_t = f(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{b}_x) = tanh(\mathbf{z}_t)$$

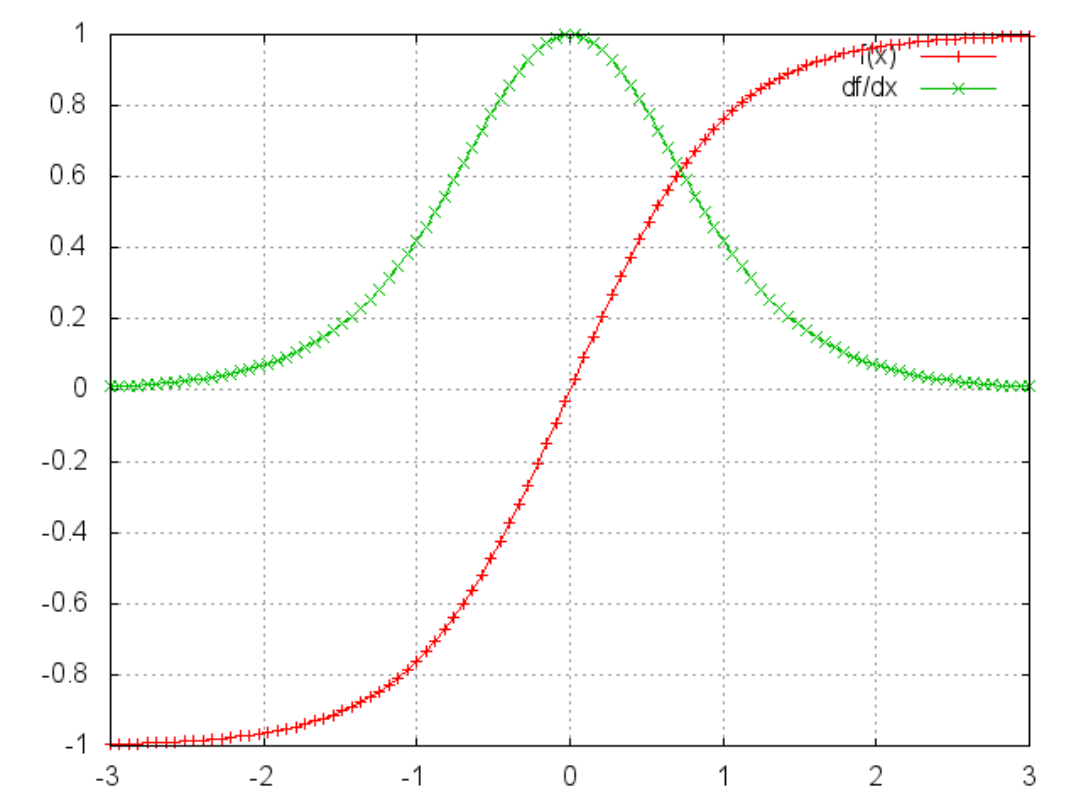
$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} = \frac{\partial \mathbf{h}_t}{\partial \mathbf{z}_t} \times \frac{\partial \mathbf{z}_t}{\partial \mathbf{h}_{t-1}} = (1 - tanh^2(\mathbf{z}_t)) \times \mathbf{W}_{hh}$$

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{W}_{xh}} = \frac{\partial \mathbf{h}_t}{\partial \mathbf{z}_t} \times \frac{\partial \mathbf{z}_t}{\partial \mathbf{W}_{xh}} = (1 - tanh^2(\mathbf{z}_t)) \times \mathbf{x}_t$$

RNN Basics: Gradient Vanishing/Exploding Problem

- Backpropagation in RNN

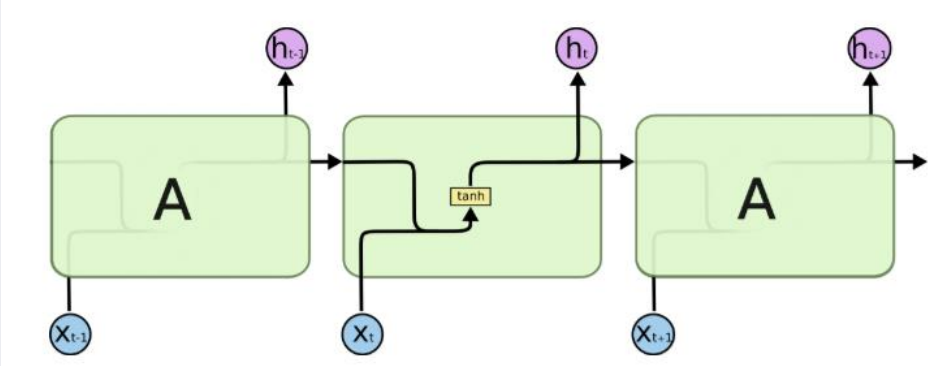
$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} = \frac{\partial \mathbf{h}_t}{\partial \mathbf{z}_t} \times \frac{\partial \mathbf{z}_t}{\partial \mathbf{h}_{t-1}} = \boxed{(1 - tanh^2(\mathbf{z}_t))} \times \mathbf{W}_{hh}$$



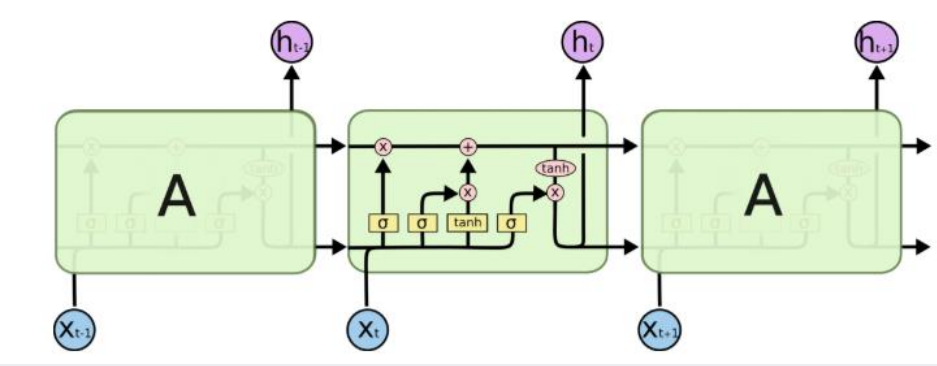
RNN Basic: LSTM

- LSTM: Long Short-Term Memory
  - ✓ Able to learn long-term dependency by preventing gradient exploding/vanishing problem

Vanilla RNN

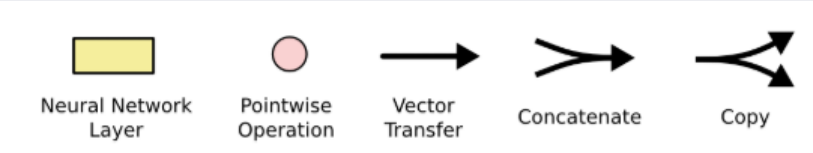


LSTM



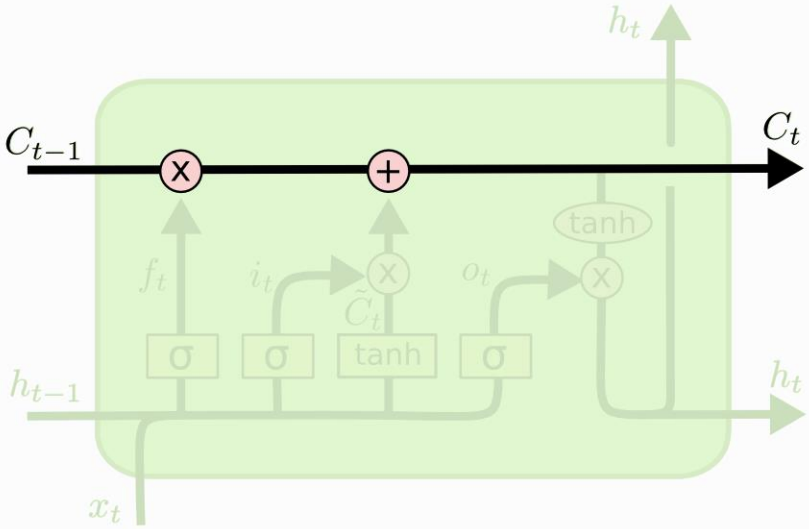
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Operation symbols



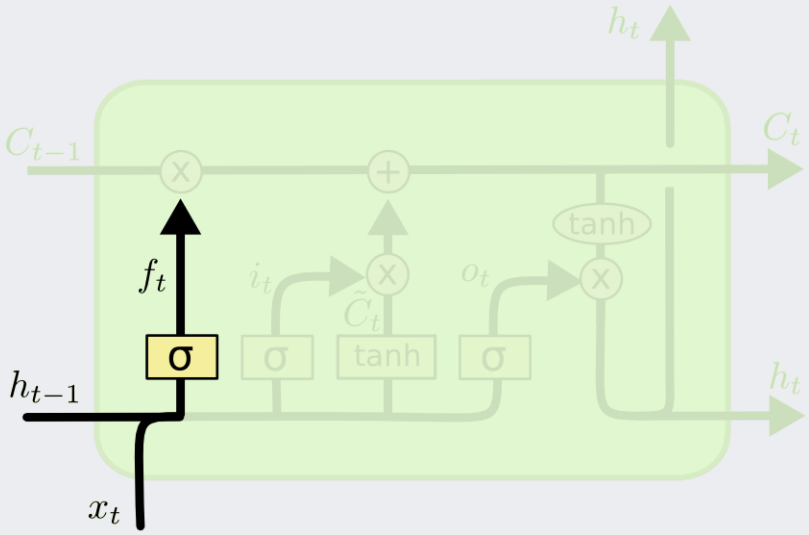
RNN Basic: LSTM

- Cell state
  - ✓ The key to LSTM, the horizontal line running through the top of the diagram



RNN Basic: LSTM

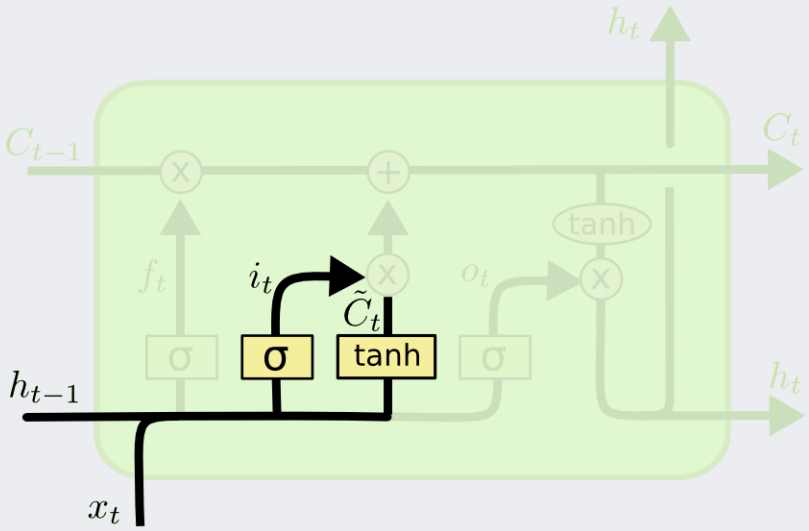
- Step 1: Decide what information we are going to throw away from the cell state
  - ✓ Forget gate: a sigmoid layer that takes the previous hidden state h_{t-1} and the current input x_t and outputs a number between 0 and 1 for the previous cell state
    - 1: completely keep the information in the previous cell state
    - 0: completely ignore the information



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

RNN Basic: LSTM

- Step 2: Decide what new information we are going to store in the cell state
  - ✓ Input gate: a sigmoid layer that decides which values we will update
  - ✓ A tanh layer create a vector of new candidate cell state value $\tilde{C}_t$

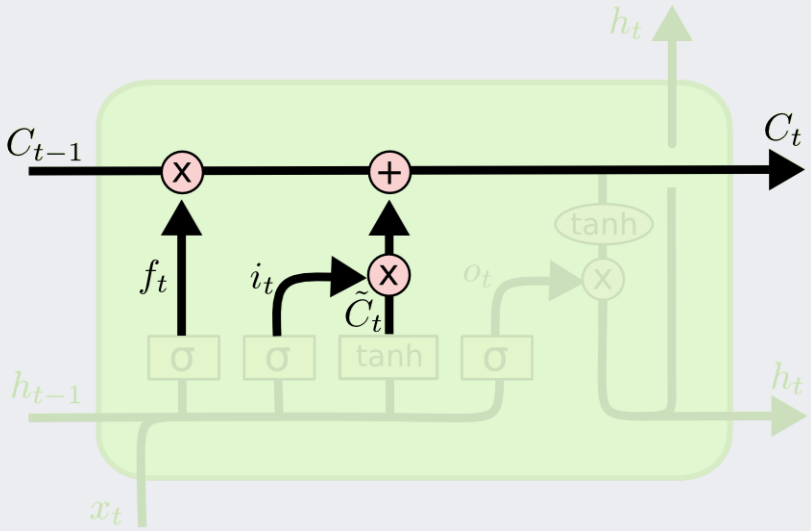


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

RNN Basic: LSTM

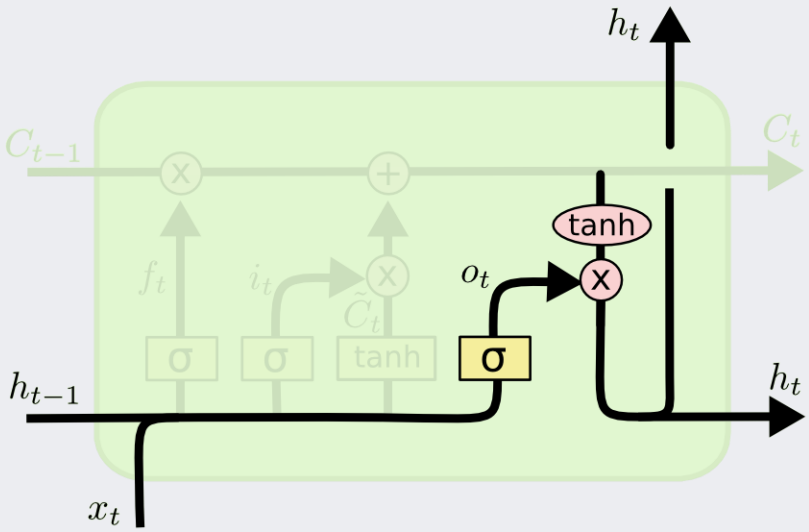
- Step 3: Update the old cell state into the new cell state
  - ✓ Multiply the old state C_{t-1} by f_t and add $i_t * \tilde{C}_t$, which is a new candidate value scaled by how much we decided to update each state value



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

RNN Basic: LSTM

- Step 4: Decide what we are going to output
  - ✓ Based on the current cell state but will be a filtered version

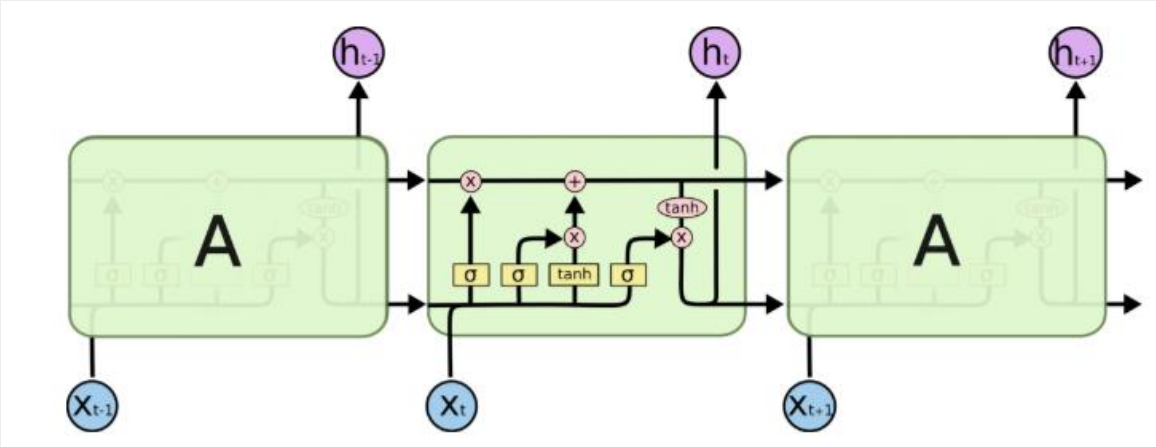


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

RNN Basic: LSTM

- Summary



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

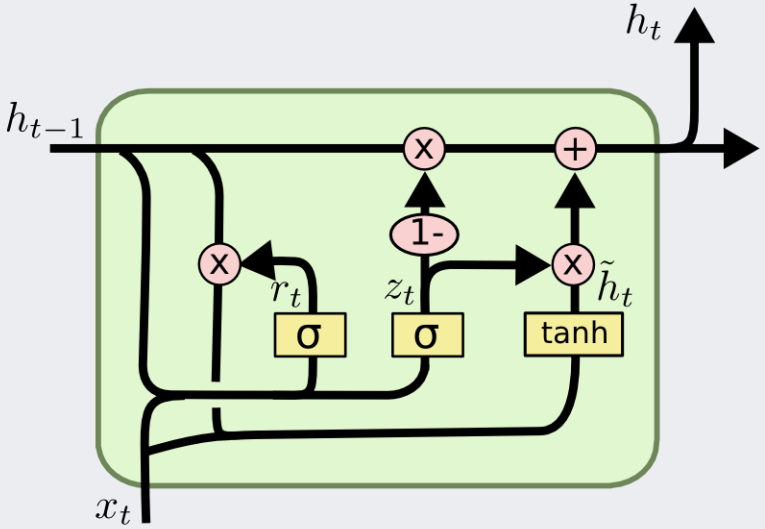
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

RNN Basic: GRU

- GRU: Gated Recurrent Unit



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right) \quad \text{Update gate}$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right) \quad \text{Reset gate}$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

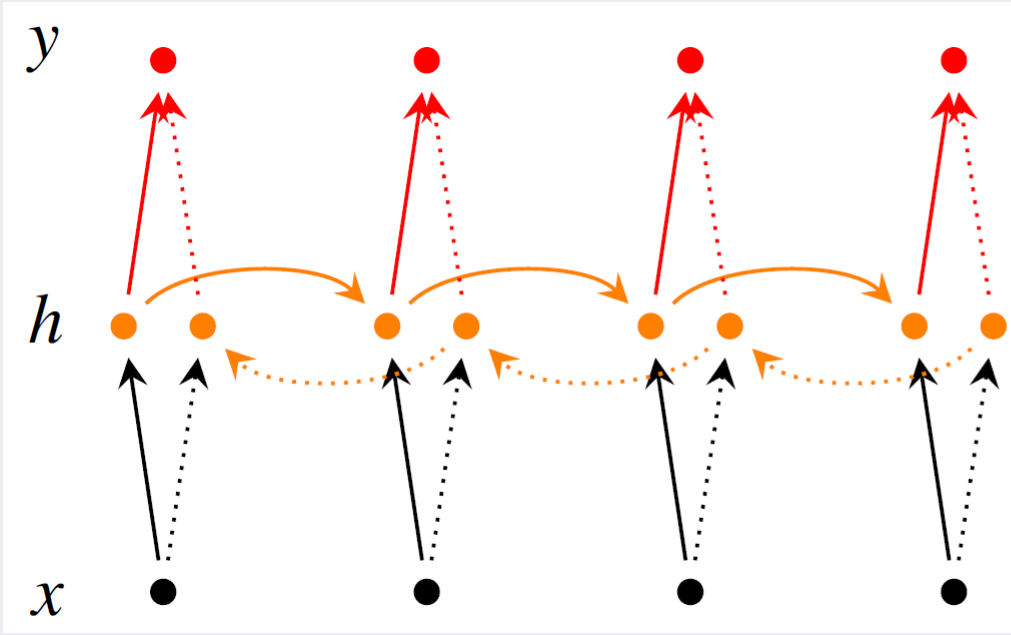
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

✓ Combines the forget and input gates into a single "update gate"

✓ Merges the cell state and hidden state

RNN Variations: Bidirectional RNN

- Bidirectional RNN
  - ✓ Attempts to incorporate information from words both preceding and following



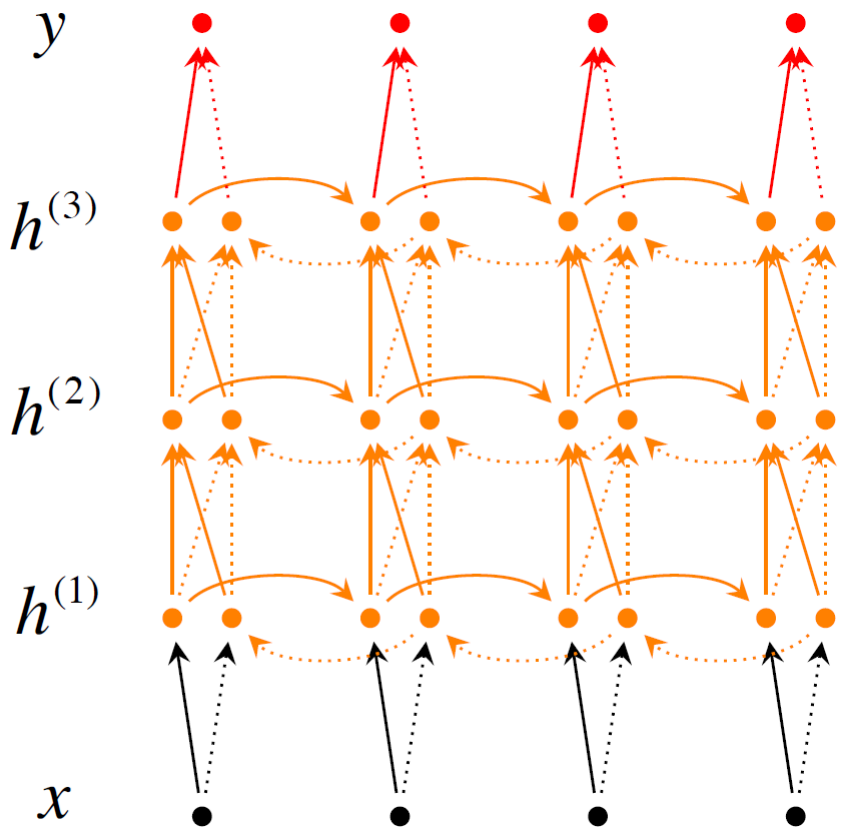
$$\vec{h}_t = f(\vec{W}x_t + \vec{V}\vec{h}_{t-1} + \vec{b})$$

$$\overleftarrow{h}_t = f(\overleftarrow{W}x_t + \overleftarrow{V}\overleftarrow{h}_{t+1} + \overleftarrow{b})$$

$$y_t = g(U[\vec{h}_t; \overleftarrow{h}_t] + c)$$

RNN Variations: Bidirectional RNN

- Deep-Bidirectional RNN
  - ✓ Why not more than one layer of RNN?



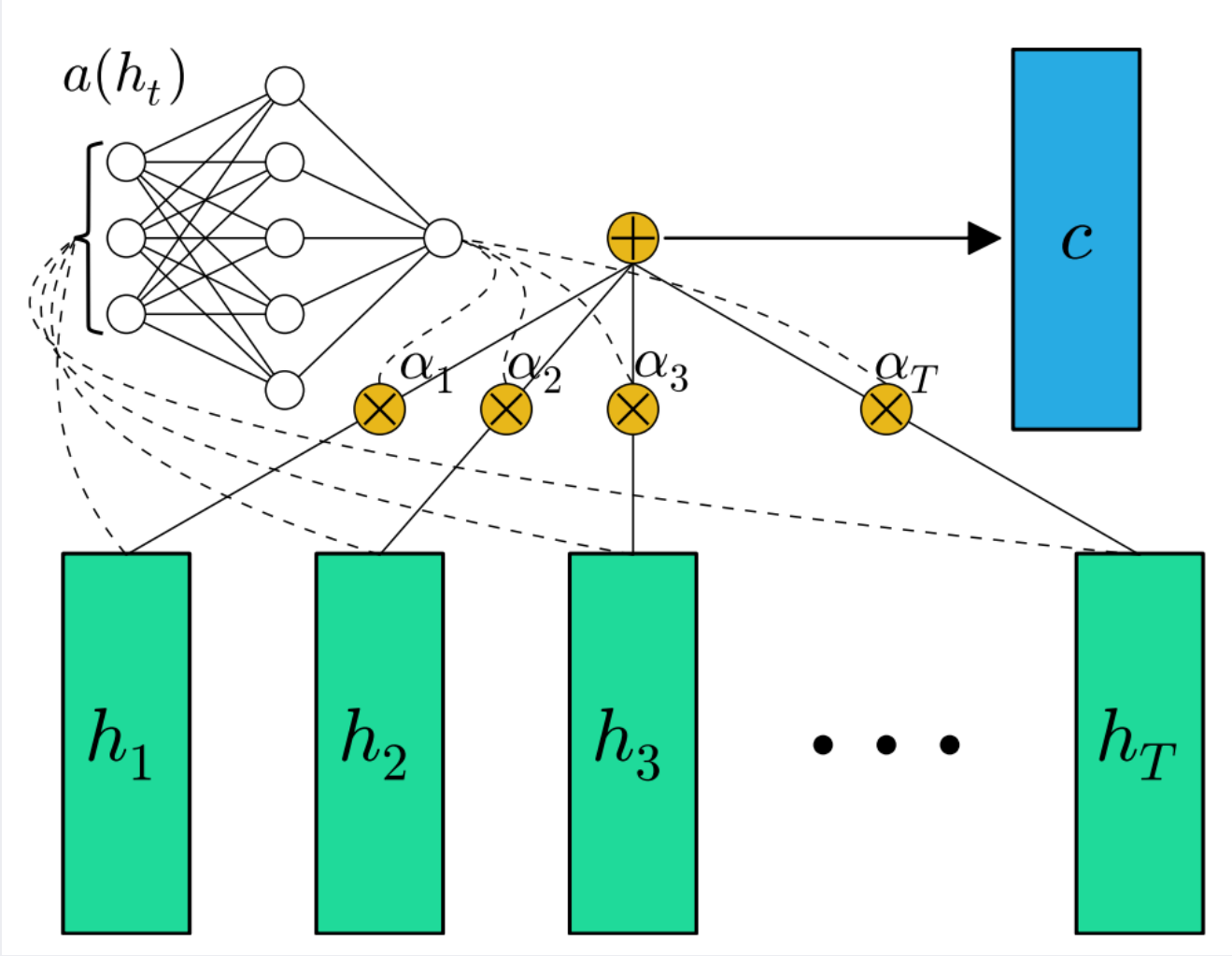
$$\overrightarrow{h}_t^{(i)} = f(\overrightarrow{W}^{(i)} h_t^{(i-1)} + \overrightarrow{V}^{(i)} \overrightarrow{h}_{t-1}^{(i)} + \overrightarrow{b}^{(i)})$$

$$\overleftarrow{h}_t^{(i)} = f(\overleftarrow{W}^{(i)} h_t^{(i-1)} + \overleftarrow{V}^{(i)} \overleftarrow{h}_{t+1}^{(i)} + \overleftarrow{b}^{(i)})$$

$$y_t = g(U[\overrightarrow{h}_t^{(L)}; \overleftarrow{h}_t^{(L)}] + c)$$

RNN: Attention

- Attention mechanism for finding significant words in document classification

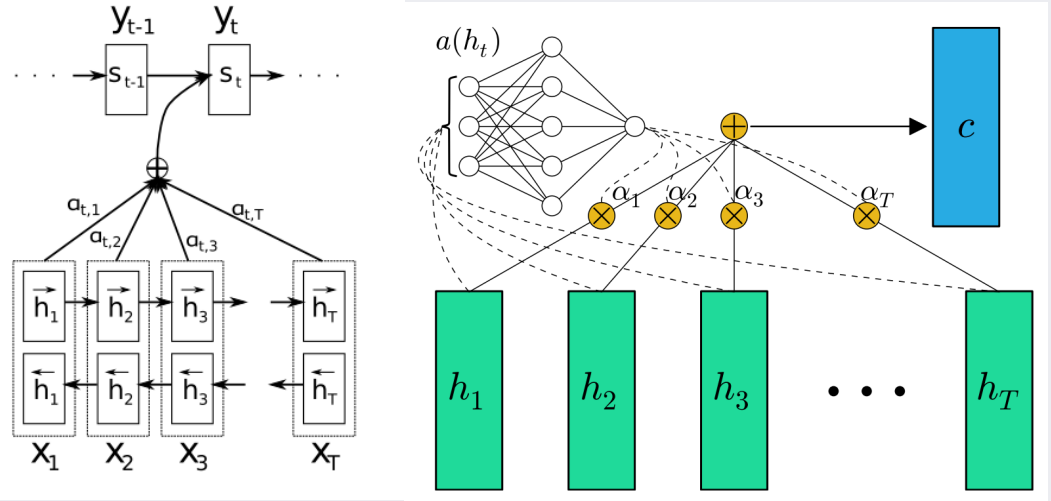


Source: (Raffel and Ellis, 2016)

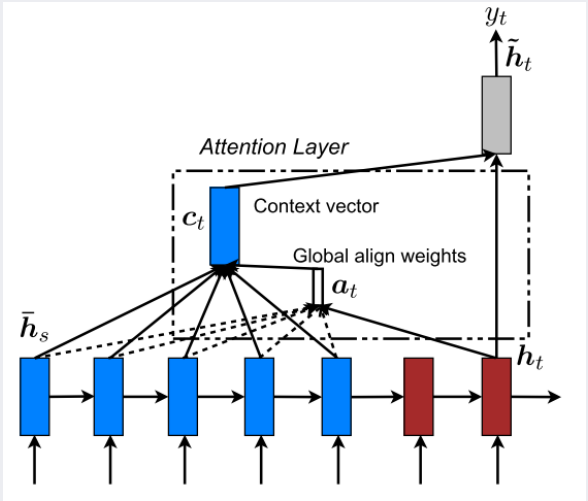
RNN: Attention

- Two main attention mechanisms
  - ✓ Bahadanau attention (Bahdanau et al., 2015)
    - Attention scores are separated trained, the current hidden state is a function of the context vector and the previous hidden state
  - ✓ Luong attention (Luong et al., 2015)
    - Attention scores are not trained, the new current hidden state is the simple tanh of the weigthed concatenation of the context vector and the current hidden state of the decoder

Bahdanau attention



Luong attention



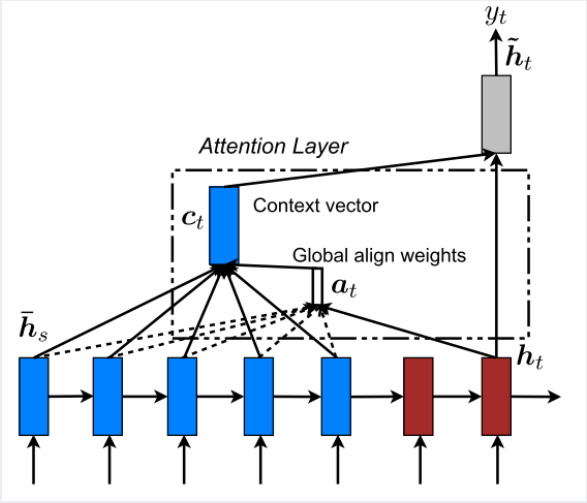
RNN: Attention

- Luong attention
  - ✓ New hidden state of the decoder is the simple tanh of the weigthed concatenation of the context vector and the current hidden state of the decoder:

$$\tilde{\mathbf{h}}_t = tanh(\mathbf{W_c}[\mathbf{c}_t; \mathbf{h}_t])$$

  - ✓ The attention vector is fed throught the softmax layer to produce the predictive distribution:

$$p(y_t|y_{y<t}, x) = \text{softmax}(\mathbf{W_s}\tilde{\mathbf{h}}_t)$$

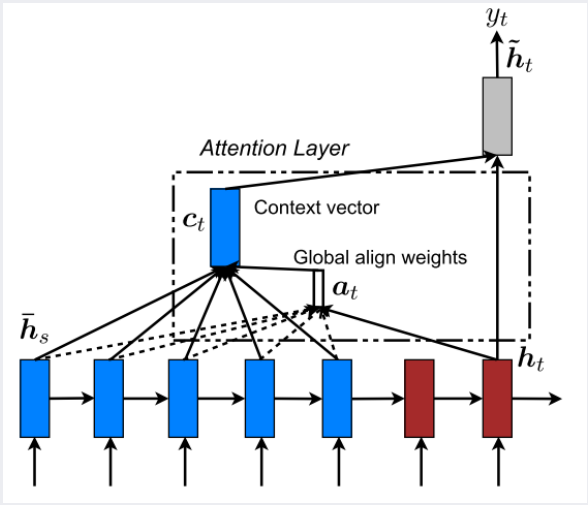


RNN: Attention

- Luong attention
  - ✓ A variable-length alignment vector:

$$\mathbf{a}_t(s) = \text{align}(\mathbf{h}_t, \overline{\mathbf{h}}_s)$$

$$= \frac{exp\Big(\text{score}(\mathbf{h}_t, \overline{\mathbf{h}}_s)\Big)}{\sum_{s'} exp\Big(\text{score}(\mathbf{h}_t, \overline{\mathbf{h}}'_s)\Big)}$$



  - ✓ **score** is referred as a context-based function:

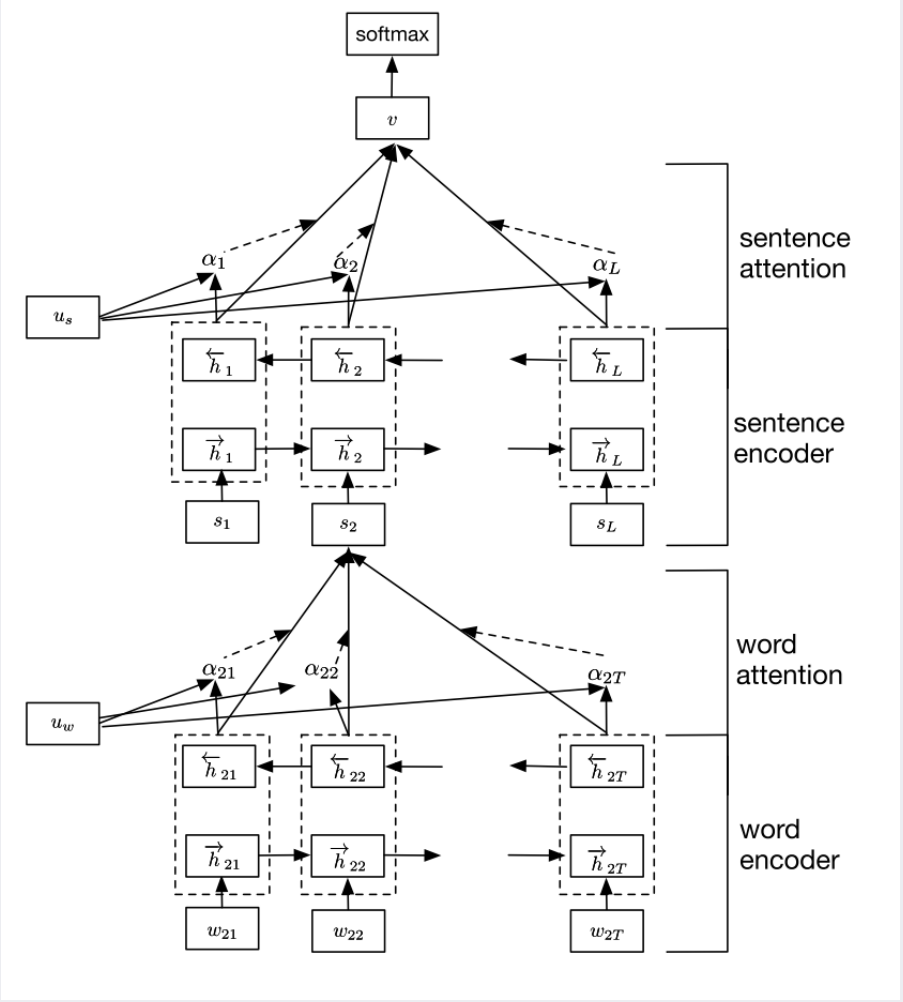
$$\text{score}(\mathbf{h}_t, \overline{\mathbf{h}}_s) = \begin{cases} \mathbf{h}_t{}^T \overline{\mathbf{h}}_s, & dot \\ \mathbf{h}_t{}^T \mathbf{W_a} \overline{\mathbf{h}}_s, & general \\ \mathbf{v}_a^T tanh(\mathbf{W_c}[\mathbf{c}_t; \mathbf{h}_t]), & concat \end{cases}$$

  - ✓ Context vector

$$\mathbf{c}_t = \overline{\mathbf{h}}_s \mathbf{a}_t$$

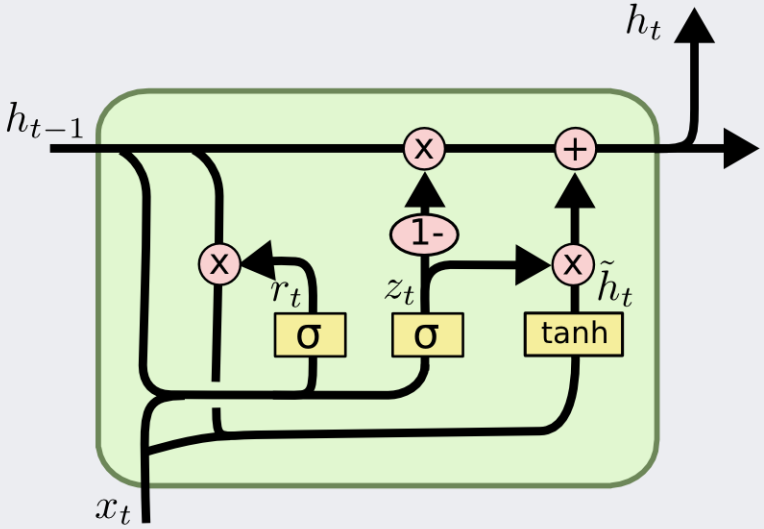
RNN for Document Classification and Attention

- Hierarchical Attention Network (Yang et al., 2016)
  - ✓ Level 1: Word sequence encoder
  - ✓ Level 2: Word-level attention layer
  - ✓ Level 3: Sentence encoder
  - ✓ Level 4: Sentence-level attention layer



RNN for Document Classification and Attention

- Hierarchical Attention Network: Sequence encoder
  - ✓ Bidirectional GRU is employed



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right) \quad \text{Update gate}$$

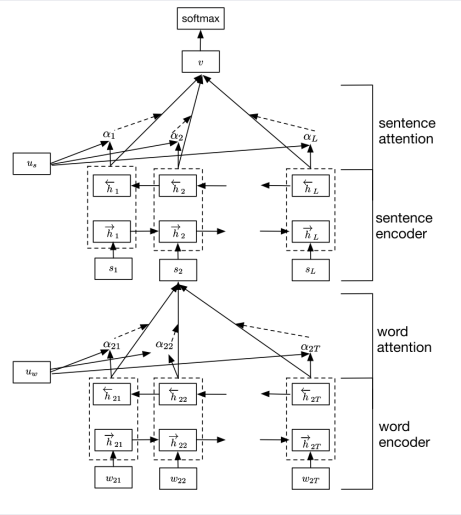
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right) \quad \text{Reset gate}$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

RNN for Document Classification and Attention

- Hierarchical Attention Network: Hierarchical Attention



✓ Word encoder

$$\mathbf{x}_{it} = \mathbf{W}_e w_{it}, \ t \in [1, T],$$
$$\overrightarrow{\mathbf{h}}_{it} = \overrightarrow{GRU}(\mathbf{x}_{it}), \quad t \in [1, T],$$
$$\overleftarrow{\mathbf{h}}_{it} = \overleftarrow{GRU}(\mathbf{x}_{it}), \quad t \in [1, T],$$
$$\mathbf{h}_{it} = [\overrightarrow{\mathbf{h}}_{it}, \overleftarrow{\mathbf{h}}_{it}].$$

✓ Word Attention

$$\mathbf{u}_{it} = tanh(\mathbf{W}_w \mathbf{h}_{it} + \mathbf{b}_w)$$
$$\alpha_{it} = \frac{\exp(\mathbf{u}_{it}^T \mathbf{u}_w)}{\sum_{t'} \exp(\mathbf{u}_{it'}^T \mathbf{u}_w)}$$
$$\mathbf{s}_i = \sum_t \alpha_{it} \mathbf{h}_{it}$$

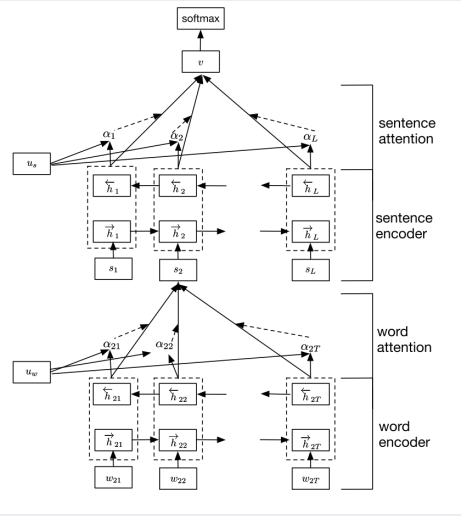
Word context vector

- Can be seen as a high level representation of a fixed query "what is the informative word?"
- Randomly initialized and jointly learned during the training process

RNN for Document Classification and Attention

- Hierarchical Attention Network: Hierarchical Attention
  - ✓ Sentence encoder

$$\overrightarrow{\mathbf{h}}_i = \overrightarrow{GRU}(\mathbf{s}_i), \quad i \in [1, L],$$
$$\overleftarrow{\mathbf{h}}_i = \overleftarrow{GRU}(\mathbf{s}_i), \quad i \in [1, L],$$
$$\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i, \overleftarrow{\mathbf{h}}_i].$$



  - ✓ Sentence attention

$$\mathbf{u}_i = tanh(\mathbf{W}_s \mathbf{h}_i + \mathbf{b}_s)$$
$$\alpha_i = \frac{\exp(\mathbf{u}_i^T \mathbf{u}_s)}{\sum_{i'} \exp(\mathbf{u}_{i'}^T \mathbf{u}_s)}$$
$$\mathbf{v} = \sum_i \alpha_i \mathbf{h}_i$$

Sentence context vector
- Can be seen as a high level representation of a fixed query "what is the informative sentence?"
- Randomly initialized and jointly learned during the training process

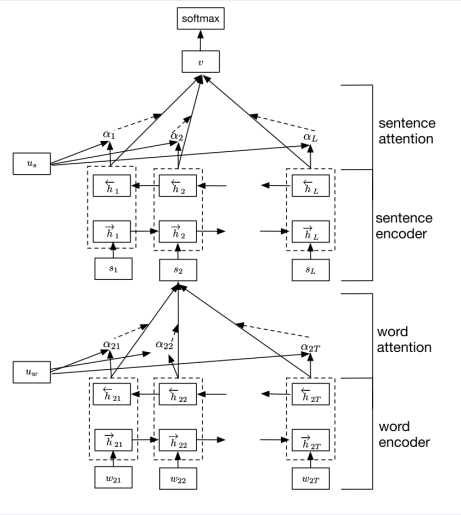
RNN for Document Classification and Attention

- Hierarchical Attention Network
  - ✓ Document classification

$$p = \text{softmax}(\mathbf{W}_c\mathbf{v} + b_c)$$

  - ✓ Loss function: negative log likelihood of the correct labels

$$L = -\sum_d logp_{dj}$$



RNN for Document Classification and Attention

- Hierarchical Attention Network: Experiment
  - ✓ Data description

Data set	classes	documents	average #s	max #s	average #w	max #w	vocabulary
Yelp 2013	5	335,018	8.9	151	151.6	1184	211,245
Yelp 2014	5	1,125,457	9.2	151	156.9	1199	476,191
Yelp 2015	5	1,569,264	9.0	151	151.9	1199	612,636
IMDB review	10	348,415	14.0	148	325.6	2802	115,831
Yahoo Answer	10	1,450,000	6.4	515	108.4	4002	1,554,607
Amazon review	5	3,650,000	4.9	99	91.9	596	1,919,336

RNN for Document Classification and Attention

- Hierarchical Attention Network: Experiment
  - ✓ Classification performance

	Methods	Yelp'13	Yelp'14	Yelp'15	IMDB	Yahoo Answer	Amazon
Zhang et al., 2015	BoW	-	-	58.0	-	68.9	54.4
	BoW TFIDF	-	-	59.9	-	71.0	55.3
	ngrams	-	-	56.3	-	68.5	54.3
	ngrams TFIDF	-	-	54.8	-	68.5	52.4
	Bag-of-means	-	-	52.5	-	60.5	44.1
Tang et al., 2015	Majority	35.6	36.1	36.9	17.9	-	-
	SVM + Unigrams	58.9	60.0	61.1	39.9	-	-
	SVM + Bigrams	57.6	61.6	62.4	40.9	-	-
	SVM + TextFeatures	59.8	61.8	62.4	40.5	-	-
	SVM + AverageSG	54.3	55.7	56.8	31.9	-	-
	SVM + SSWE	53.5	54.3	55.4	26.2	-	-
Zhang et al., 2015	LSTM	-	-	58.2	-	70.8	59.4
	CNN-char	-	-	62.0	-	71.2	59.6
	CNN-word	-	-	60.5	-	71.2	57.6
Tang et al., 2015	Paragraph Vector	57.7	59.2	60.5	34.1	-	-
	CNN-word	59.7	61.0	61.5	37.6	-	-
	Conv-GRNN	63.7	65.5	66.0	42.5	-	-
	LSTM-GRNN	65.1	67.1	67.6	45.3	-	-
This paper	HN-AVE	67.0	69.3	69.9	47.8	75.2	62.9
	HN-MAX	66.9	69.3	70.1	48.2	75.2	62.9
	HN-ATT	**68.2**	**70.5**	**71.0**	**49.4**	**75.8**	**63.6**

RNN for Document Classification and Attention

- Hierarchical Attention Network: Experiment
  - ✓ Attention distribution of two words: "good" and "bad"

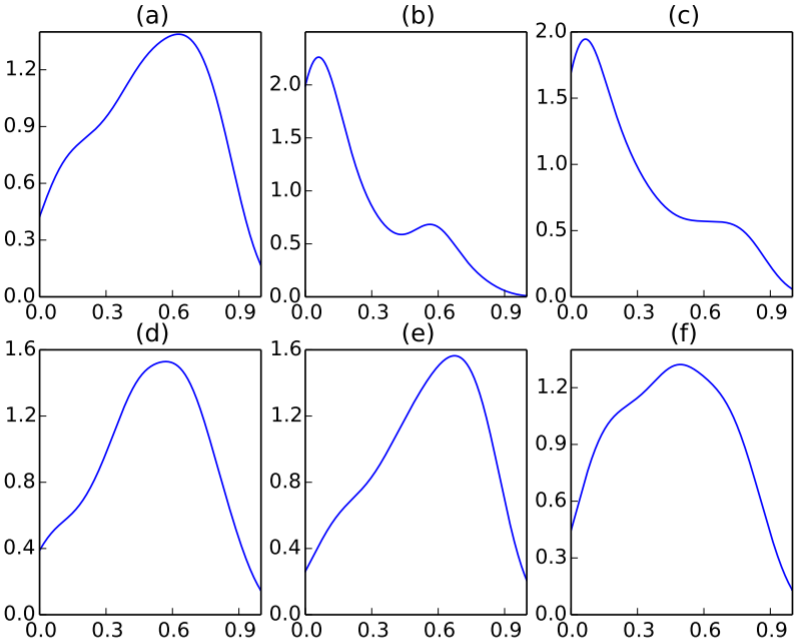


Figure 3: Attention weight distribution of `good`. (a) — aggregate distribution on the test split; (b)-(f) stratified for reviews with ratings 1-5 respectively. We can see that the weight distribution shifts to *higher* end as the rating goes higher.

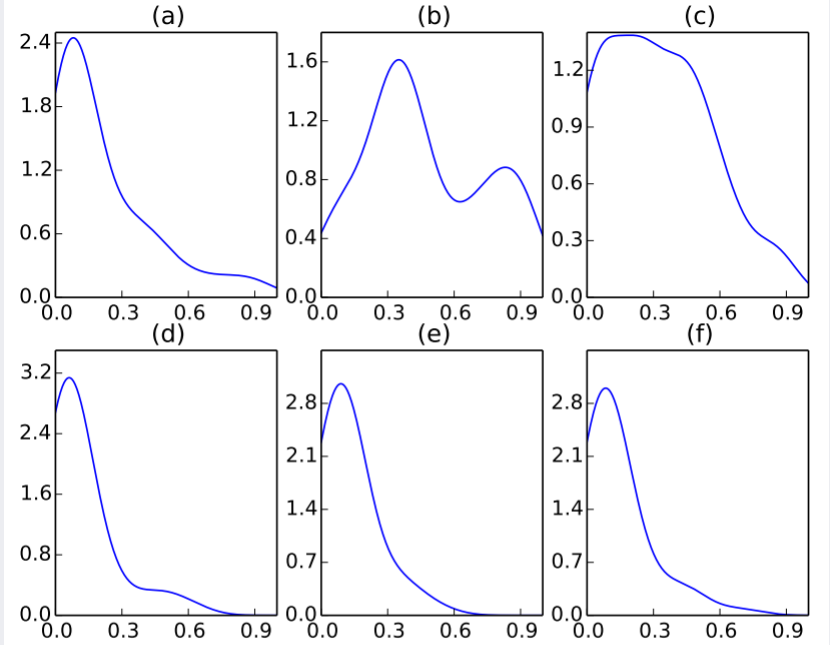


Figure 4: Attention weight distribution of the word `bad`. The setup is as above: (a) contains the aggregate distribution, while (b)-(f) contain stratifications to reviews with ratings 1-5 respectively. Contrary to before, the word `bad` is considered important for poor ratings and less so for good ones.

RNN for Document Classification and Attention

- Hierarchical Attention Network: Experiment
  - ✓ Visualization of attention scores

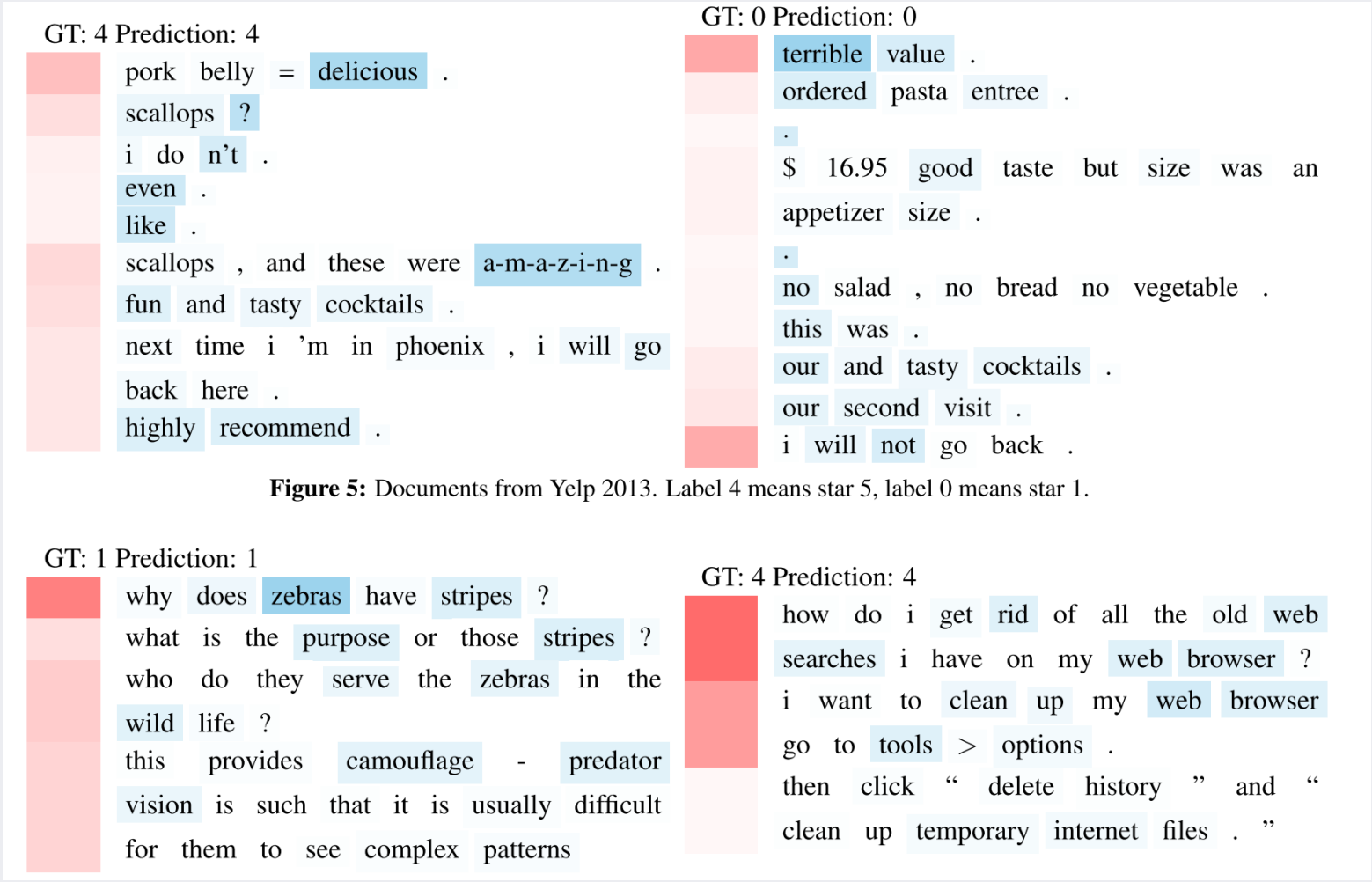


Figure 5: Documents from Yelp 2013. Label 4 means star 5, label 0 means star 1.

RNN for Document Classification and Attention

- Comparison of RNN attention and CNN localization

CAM²-4channel	Seeing as the vote average was pretty low and the fact that the clerk in the video store thoug ht was just OK I didn t t have much expectations when renting this film But contrary to the a bove I **enjoyed it** a lot This is acharming movie It didn t need to grow on me **I enjoyed it** fro m the beginning Mel Brooks gives a great performance as thelead character I think somewha t different from his usual persona in his movies There s not a lot of knockout jokes or someth inglike that but there are some rather hilarious scenes and overall this is a **very enjoyable an d very easy** to watch film **Very recommended** **Positive**
HAN	Seeing as the vote average was **pretty** low and the fact that the clerk in the video store thoug ht it was just OK I didnt have much expectations when renting this film But contrary to the a bove I **enjoyed** it a lot This is a **charming** movie It didnt need to grow on me I **enjoyed** it fro m the beginning Mel Brooks gives a great **performance** as the lead character I think somewh at different from his usual persona in his movies There not a lot of knockoutjokes or somethi ng like that but there are some rather hilarious scenes and **overall** this is a very enjoyable an d veryeasy to watch film **Very** recommended **Positive**

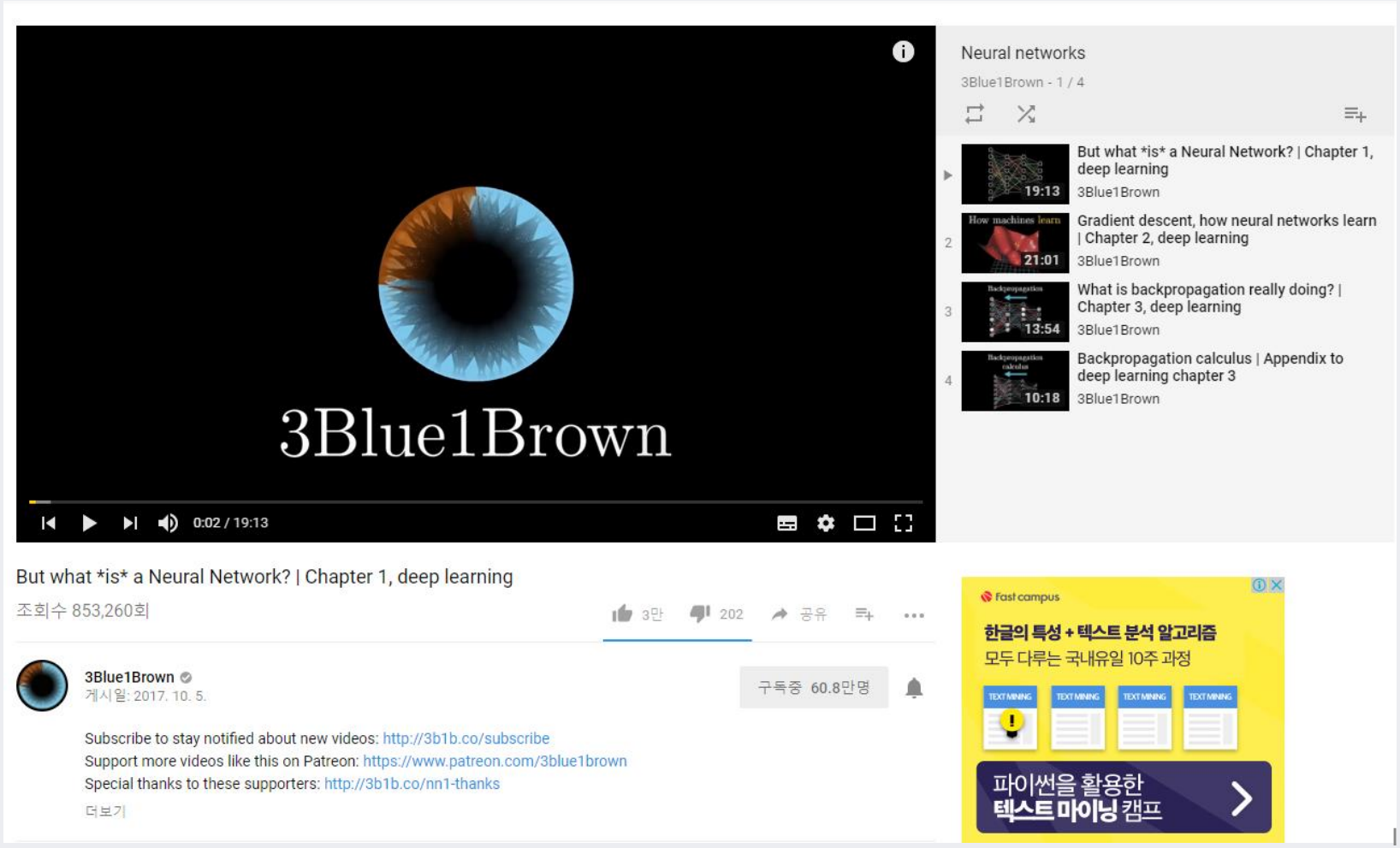
RNN for Document Classification and Attention

- Comparison of RNN attention and CNN localization

CAM²-4channel	I hate this movie It is a horrid movie Sean Young s character is completely unsympathetic He r performance is wooden at best The storyline is completely **predictable and completely uni nteresting** I would never recommend this film to anyone It is one of the **worst** movies I have ever had the misfortune to see **Negative**
HAN	I **hate** this movie It is a **horrid** movie Sean Youngs character **is** completely unsympathetic He rperformance **is** wooden at best The storyline is completely predictable and completely **unin teresting** I would never recommend **this** film to anyone It is one of the **worst** movies I have e ver had the misfortune to see **Negative**

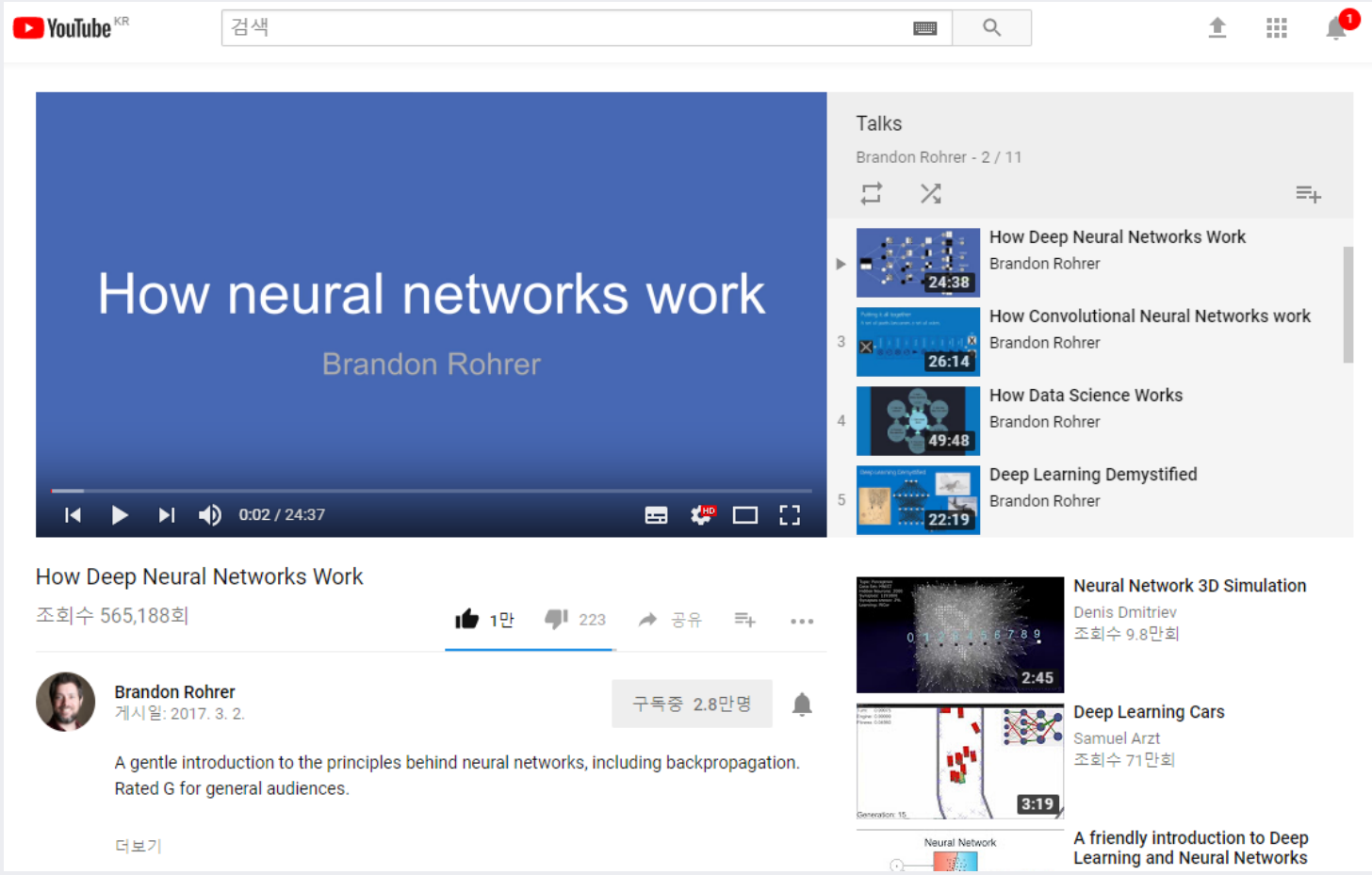
Recommended Video Lectures

- 유튜브 3Blue 1Brown Neural Network 강좌
  - ✓ https://www.youtube.com/channel/UCYO_jab_esuFRV4b17AJtAw



Recommended Video Lectures

- 유튜브 Brandon Rohrer 강좌
  - ✓ https://www.youtube.com/watch?v=ILsA4nyG7I0&list=PLVZqlMpoM6kbaeySxhdtgQPFEC5nV7Faa&index=2




ANY
questions?

References

Research Papers

- Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.
- Kim, Y. (2014). Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.
- Lee, G., Jeong, J., Seo, S., Kim, C, & Kang, P. (2018). Sentiment classification with word localization based on weakly supervised learning with a convolutional neural network. Knowledge-Based Systems, 152, 70-82.
- Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.
- Raffel, C., & Ellis, D. P. (2015). Feed-forward networks with attention can solve some long-term memory problems. arXiv preprint arXiv:1512.08756.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), 1929-1958.
- Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016). Hierarchical attention networks for document classification. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 1480-1489).
- Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In Advances in neural information processing systems (pp. 649-657).

References

Research Papers

- Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016, June). Learning deep features for discriminative localization. In Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on (pp. 2921-2929). IEEE.

Other Materials

- http://machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html
- An overview of gradient descent optimization algorithms: http://ruder.io/optimizing-gradient-descent/
- Convolutional Neural Network: http://cs231n.github.io/convolutional-networks/
- Understanding LSTM Network
  - ✓ http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (English)
  - ✓ https://brunch.co.kr/@chris-song/9 (Korean)