Artificial Neural Network: Perceptron

Pilsung Kang

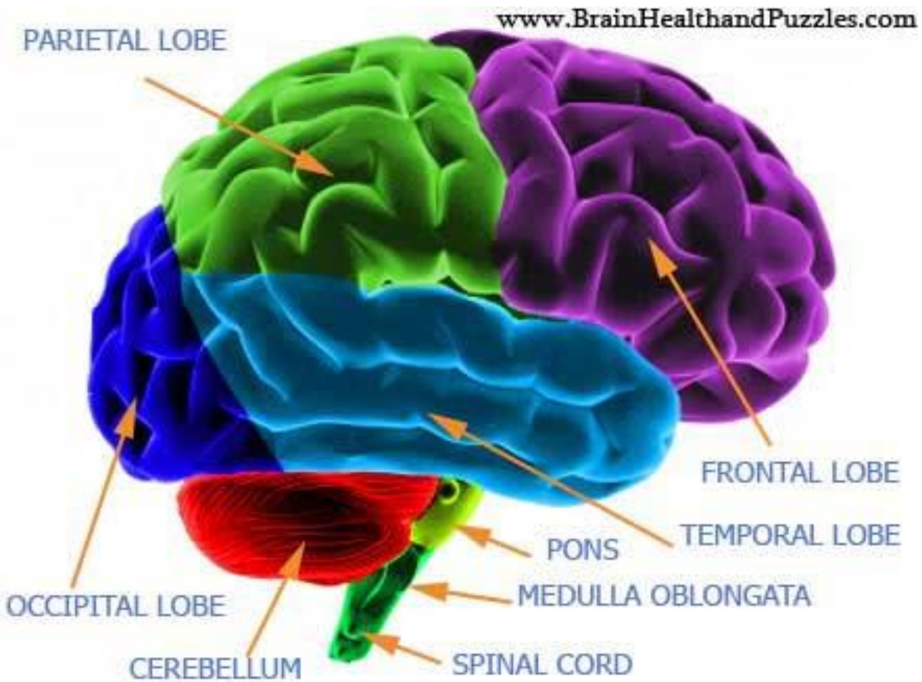
School of Industrial Management Engineering

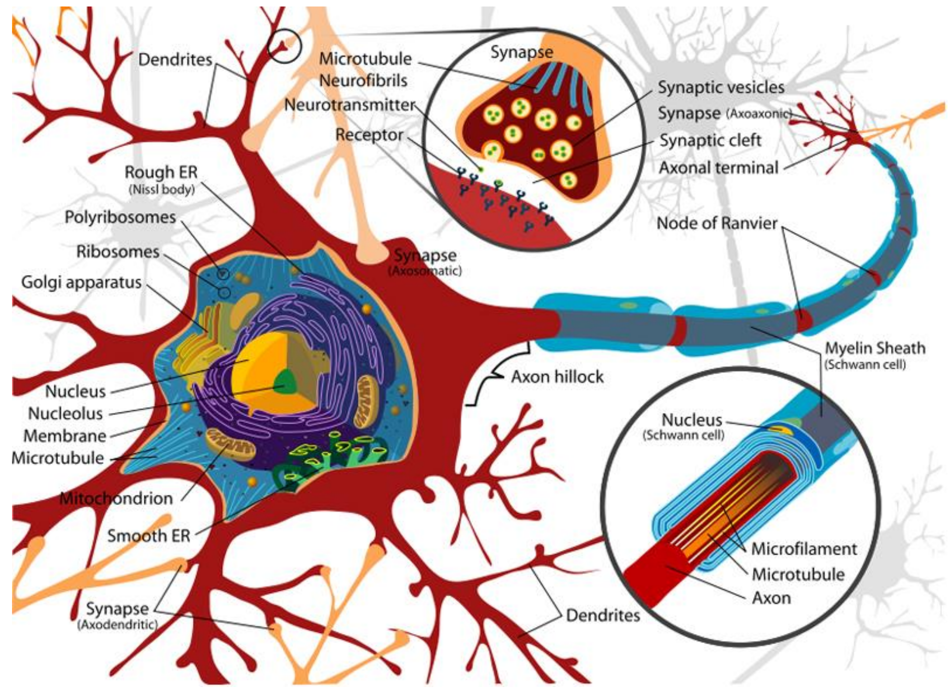
Korea University

AGENDA

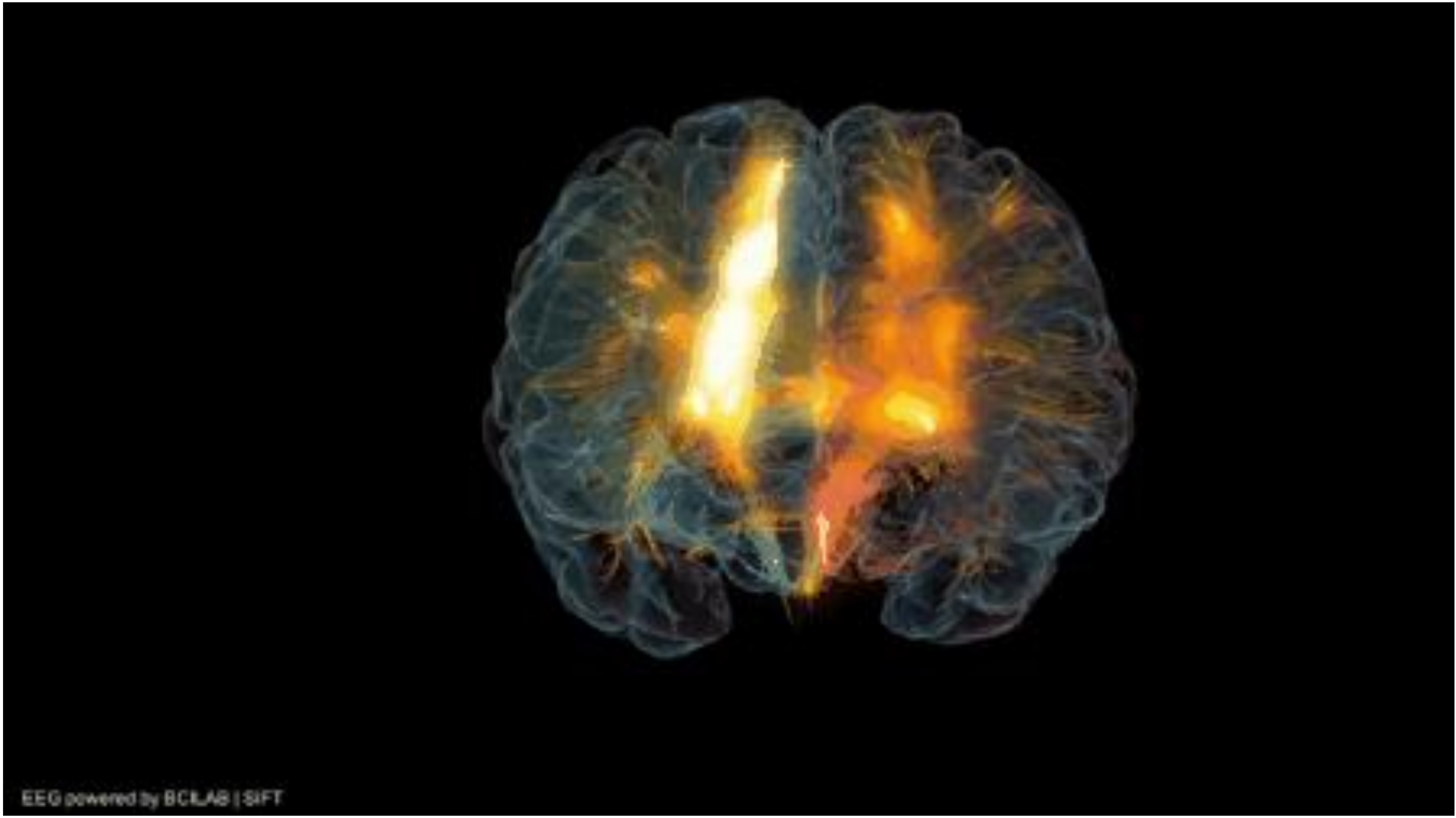
Brain Structure

- 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





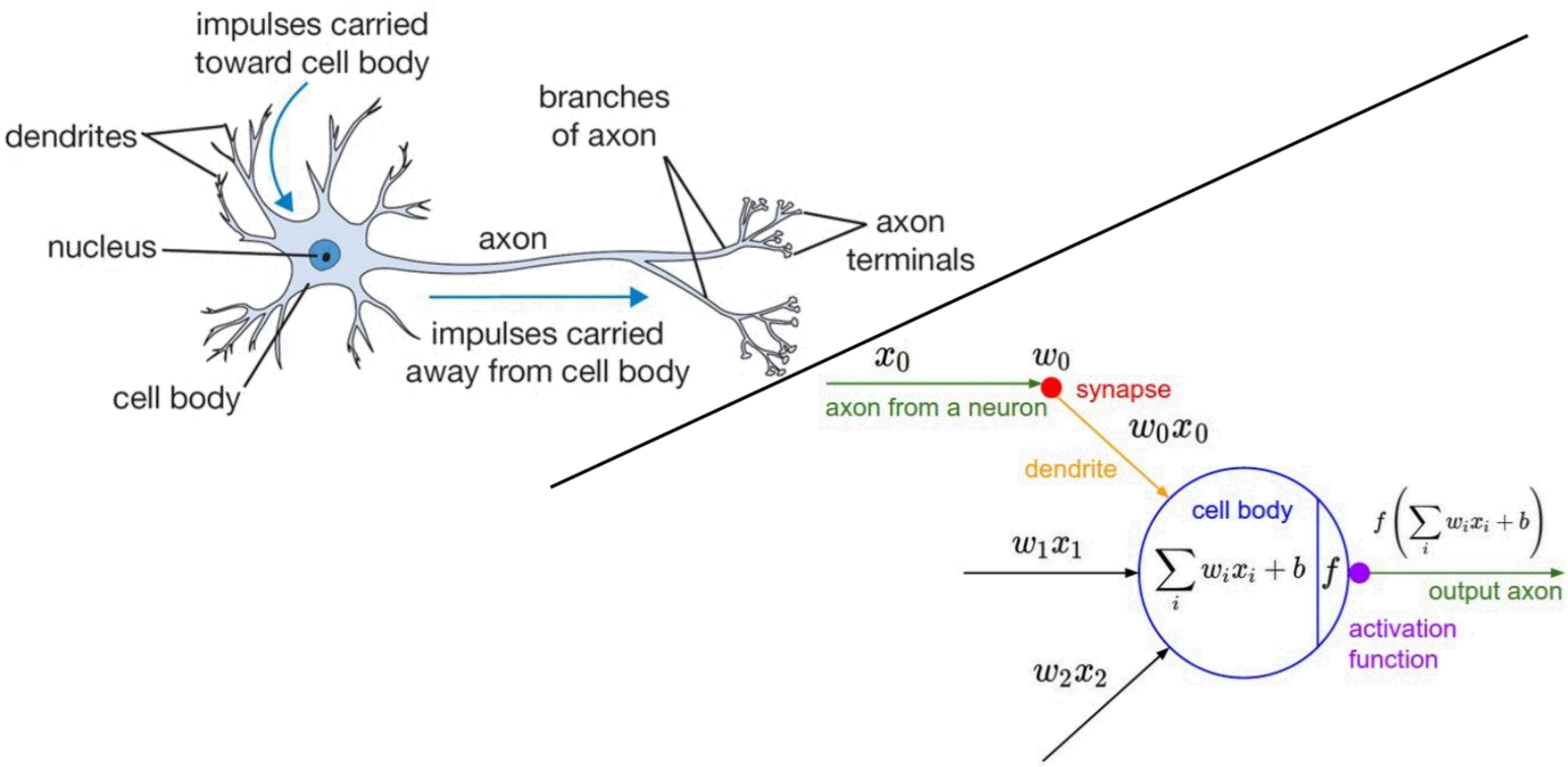
Neuron Firing Off in Real-Time



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

Perceptron

- Imitate a single neuron

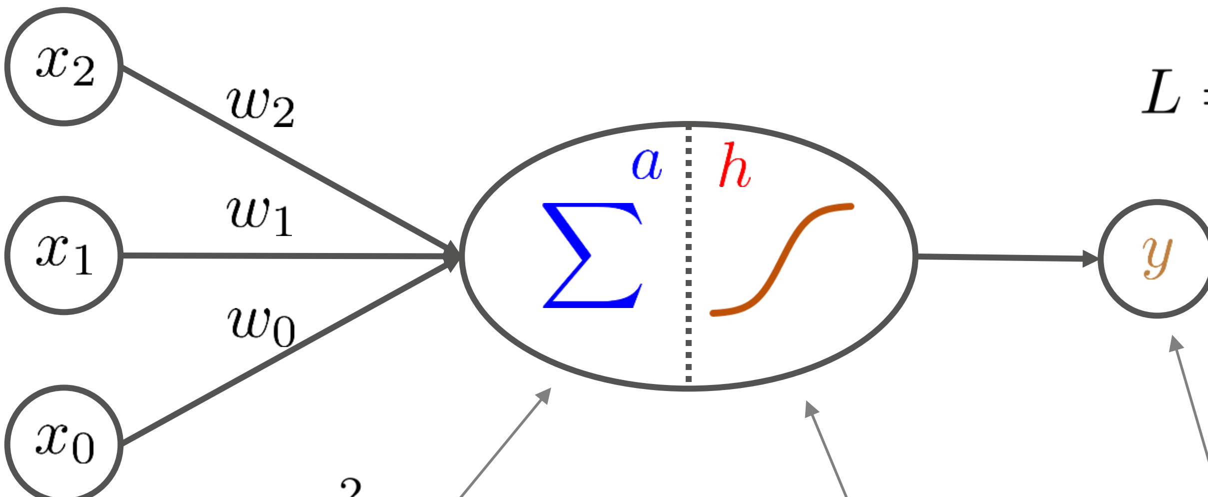


Perceptron

- Perceptron
  - ✓ An organism with only 1 neuron

Define the loss function as the squared difference between the desired value (t) and the predicted value (y)

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



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

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

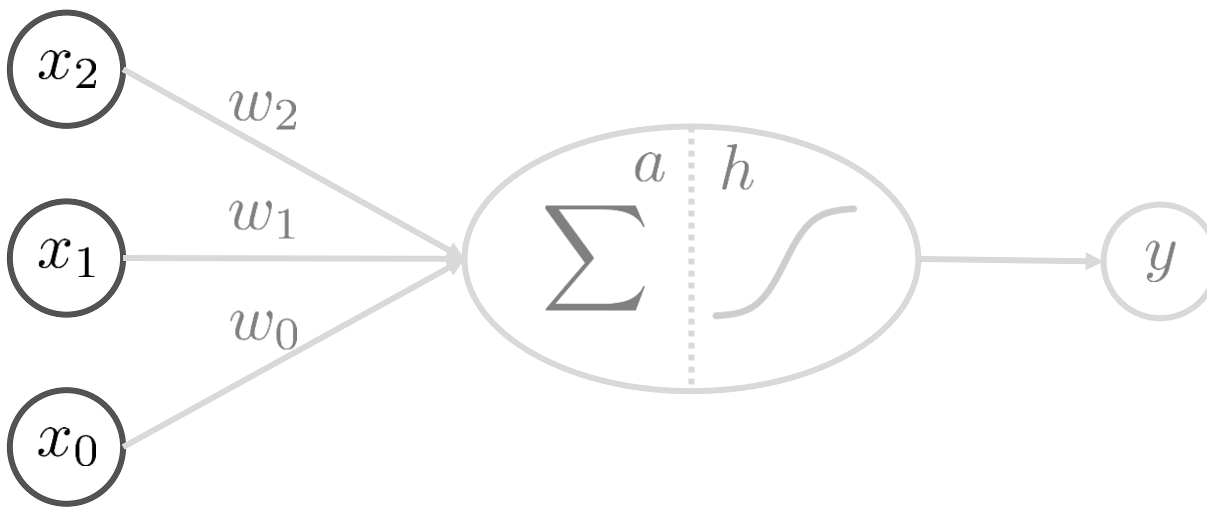
$$y = h$$

Take information from different sources

Decide the amount of information to be delivered (activation)

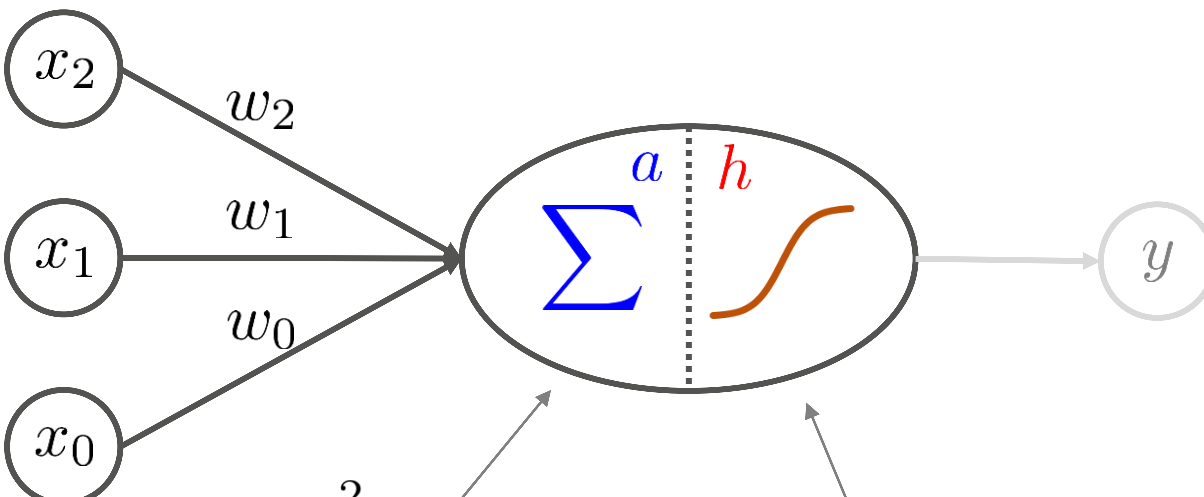
Perceptron

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



Perceptron

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



$$a = \sum_{i=0}^{2} w_i x_i \qquad h = \frac{1}{1 + exp(-a)}$$

여러 변수들의 정보를
나름대로 취합해서

얼마만큼 다음 단계로
전달할지 결정한다
(활성화)

Perceptron

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

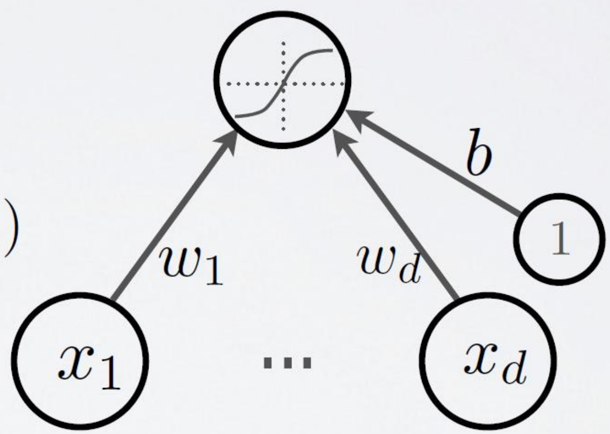
- Neuron pre-activation (or input activation):

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

- 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

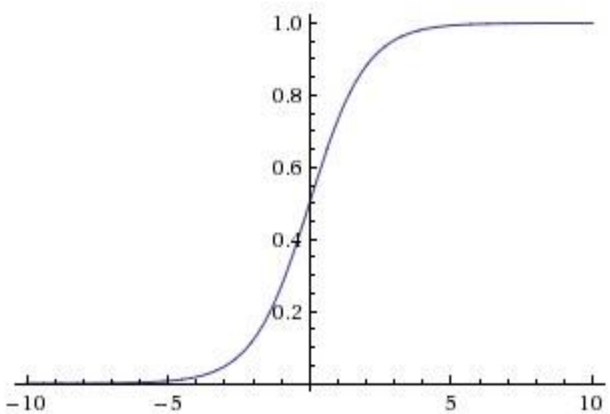


http://info.usherbrooke.ca/hlarochelle/cours/ift725_A2013/contenu.html

Perceptron

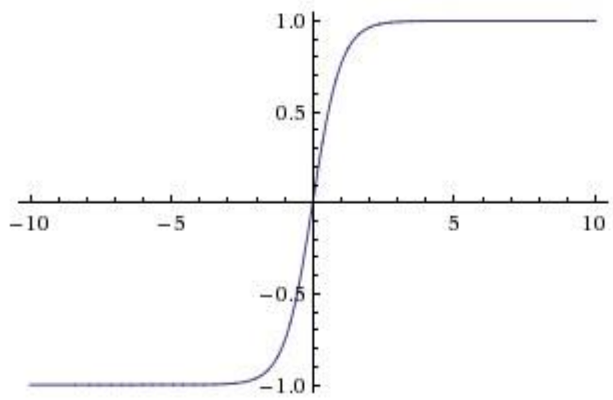
- Representative activation functions
  - ✓ Sigmoid: the most commonly used activation, [0, 1] range, learning speed is relatively slow
  - ✓ Tanh: Similar to sigmoid but [-1, 1] range, learning speed is relatively fast
  - ✓ ReLU (Rectified linear unit): very fast learning speed, easy to compute (without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

[Sigmoid]



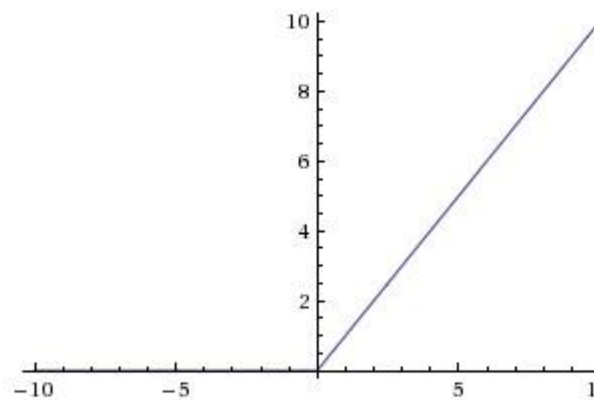
$$g(a) = \mathrm{sigm}(a) = \frac{1}{1+\exp(-a)}$$

[Tanh]



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

[ReLU]



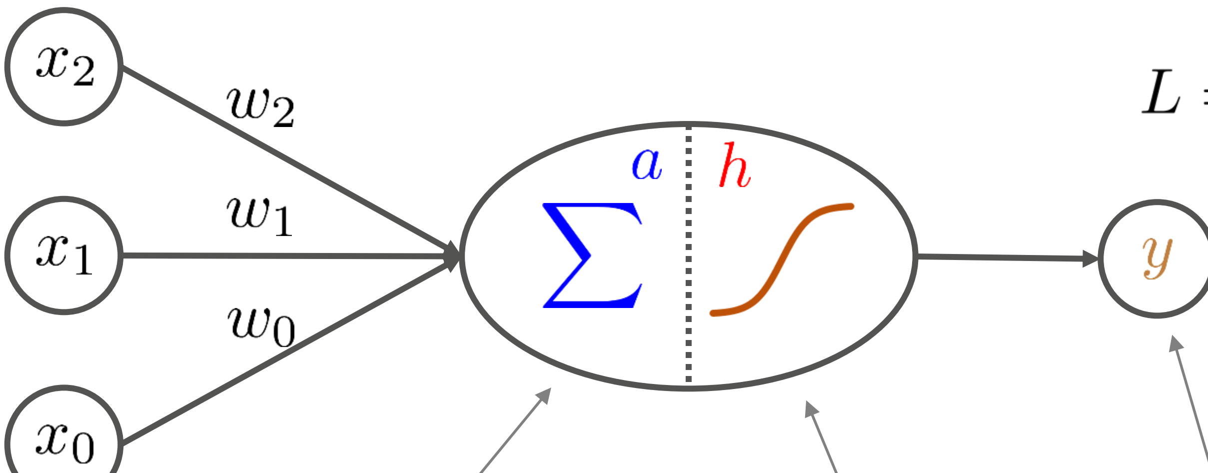
$$g(a) = \mathrm{reclin}(a) = \max(0, a)$$

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

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



$$a = \sum_{i=0}^{2} w_i x_i \qquad h = \frac{1}{1 + exp(-a)} \qquad y = h$$

여러 변수들의 정보를
나름대로 취합해서

얼마만큼 다음 단계로
전달할지 결정한다
(활성화)

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)

$$L = -\sum_i t_i \log p_i$$

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

고려대학교 KOREA UNIVERSITY

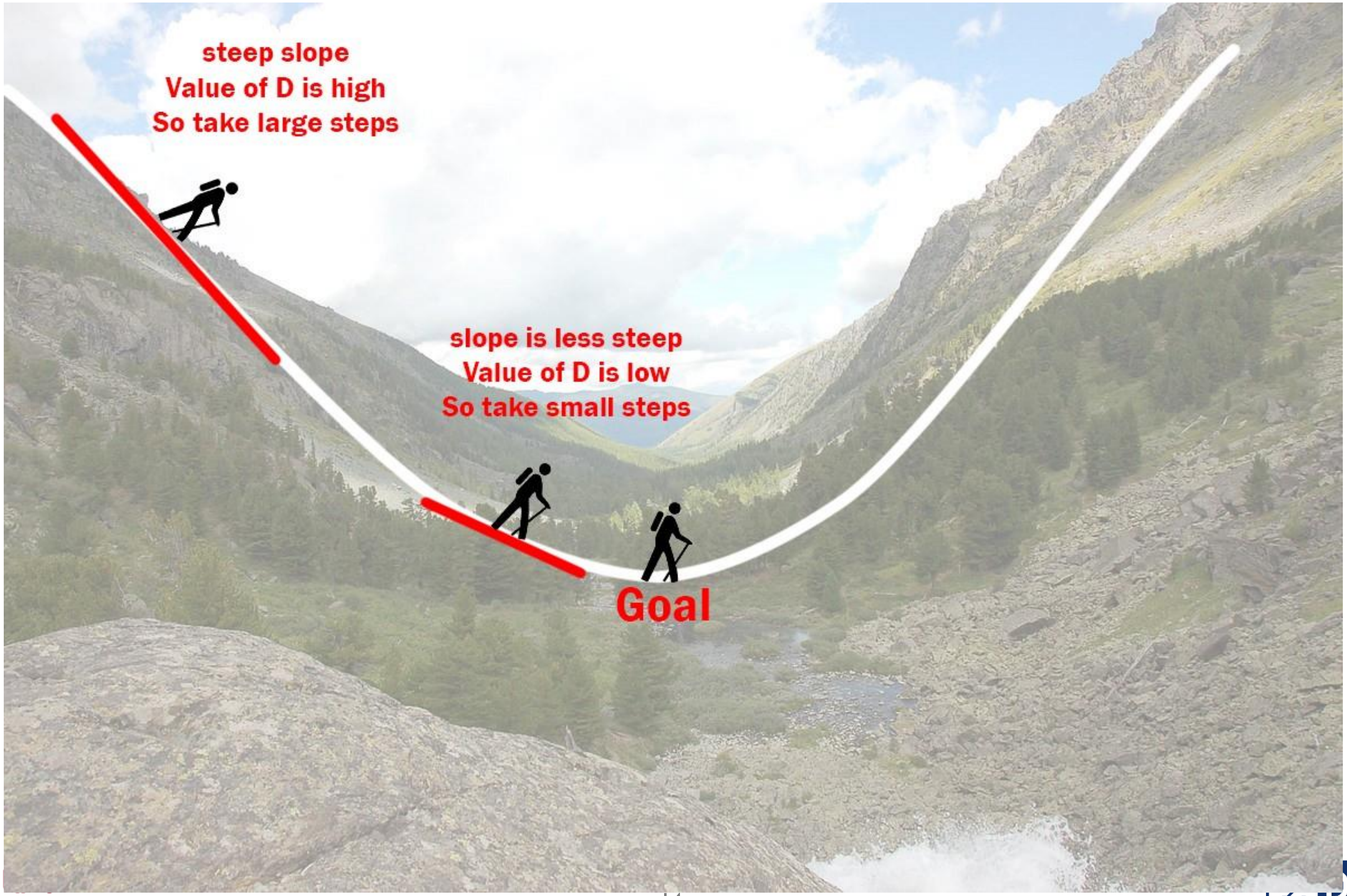
DSBA Data Science & Business Analytics

Learning: Gradient Descent

- Gradient Descent

Learning: Gradient Descent

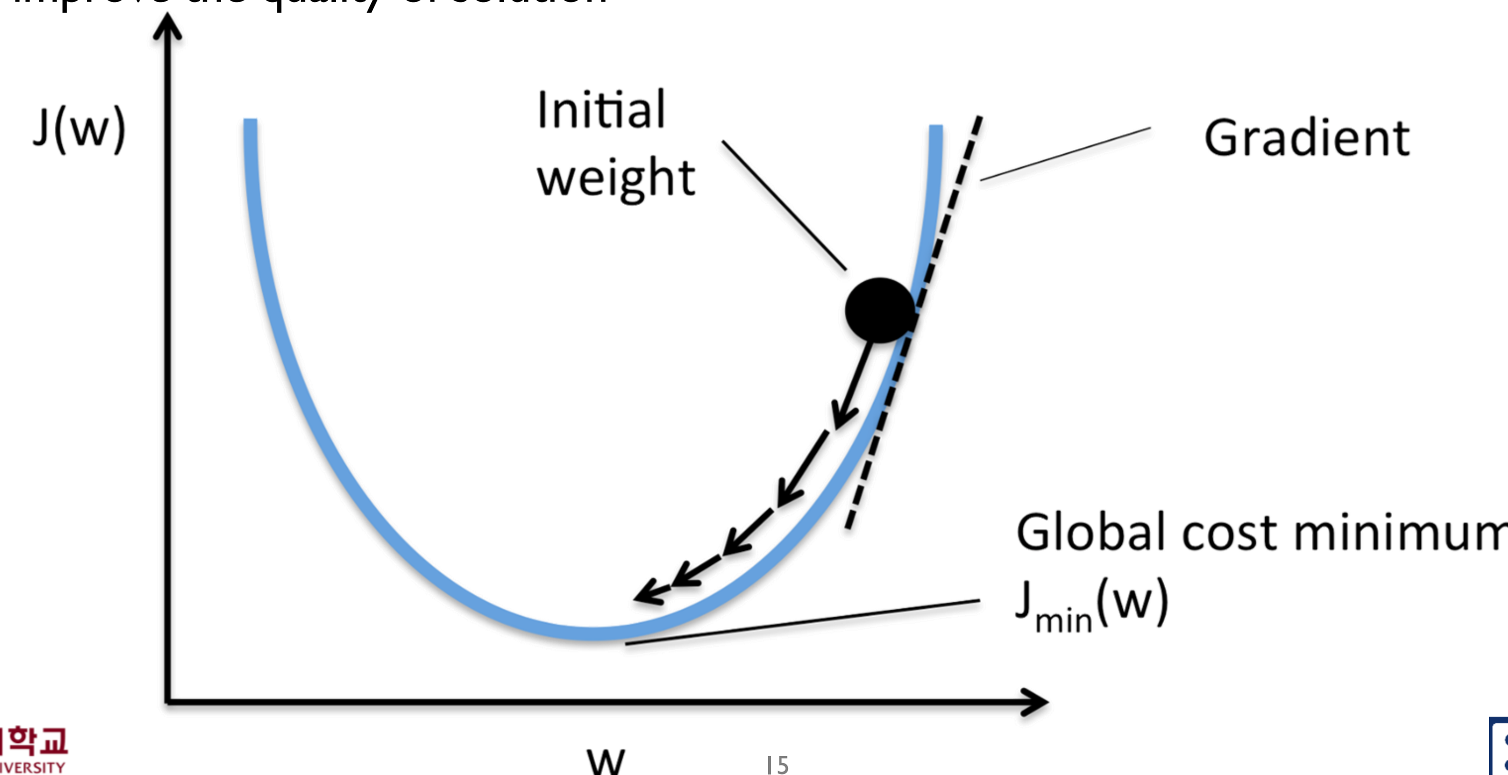
- Gradient Descent



https://towardsdatascience.com/linear-regression-using-gradient-descent-97a6c8700931

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

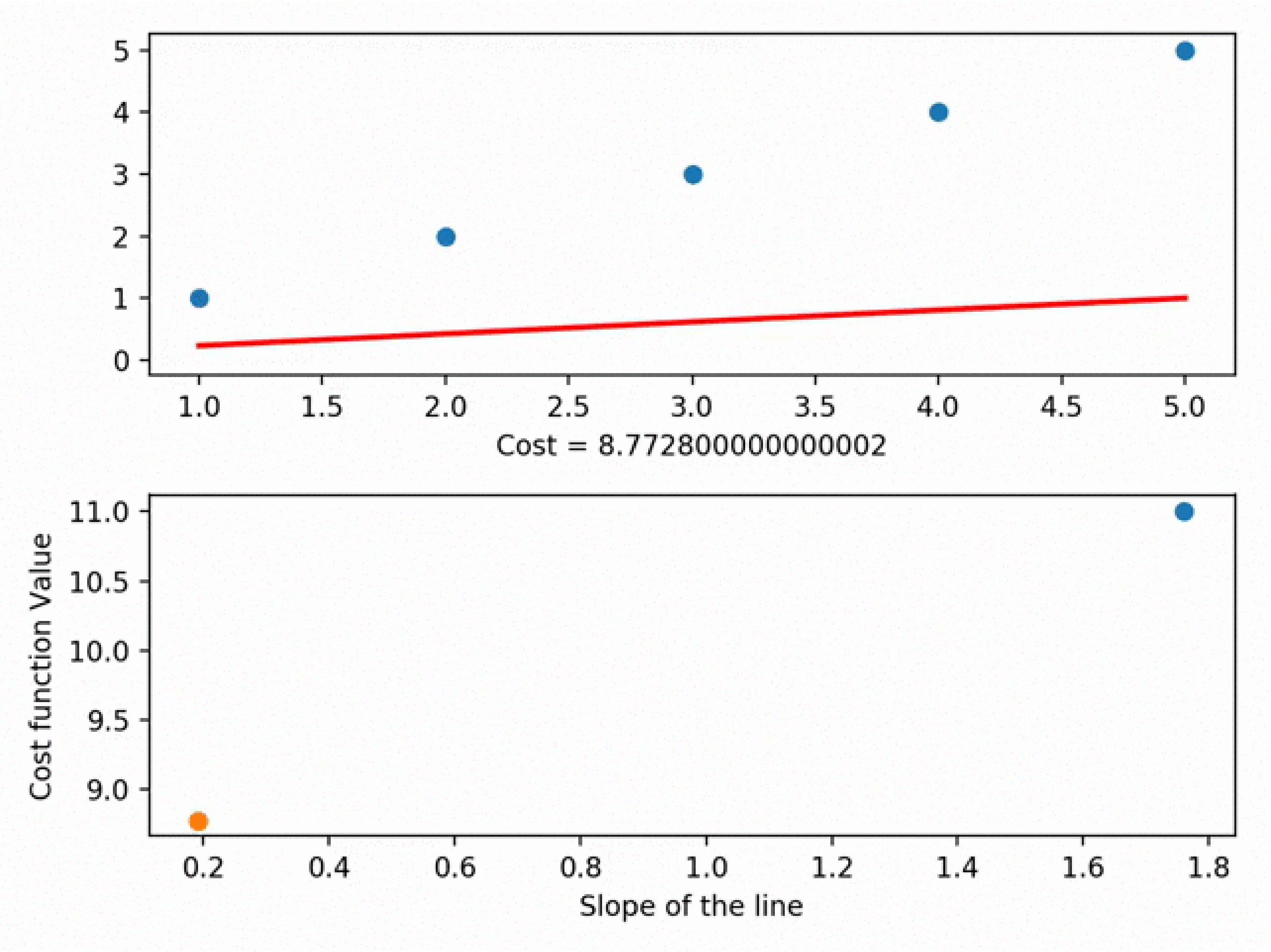


Learning: Gradient Descent

- 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

- Illustrative example



Learning: Gradient Descent

- Theoretical Background
  - ✓ Taylor expansion

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

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

어느 방향으로 갈 것인가?

$$w_{new} = w_{old} - \alpha f'(w), \quad \text{where } 0 < \alpha < 1.$$

얼마만큼 갈 것인가?

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

$$f(w_{new}) = f(w_{old} - \alpha f'(w_{old})) \cong f(w_{old}) - \alpha \left| f'(w) \right|^2 < f(w_{old})$$

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

Learning: Gradient Descent

- Use chain rule

$$\frac{\partial L}{\partial y} = y - t \qquad \frac{\partial y}{\partial h} = 1$$

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

$$\frac{\partial a}{\partial w_i} = x_i$$

- Gradients for w and x

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

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

Learning: Gradient Descent

- Weight update by Gradient Descent

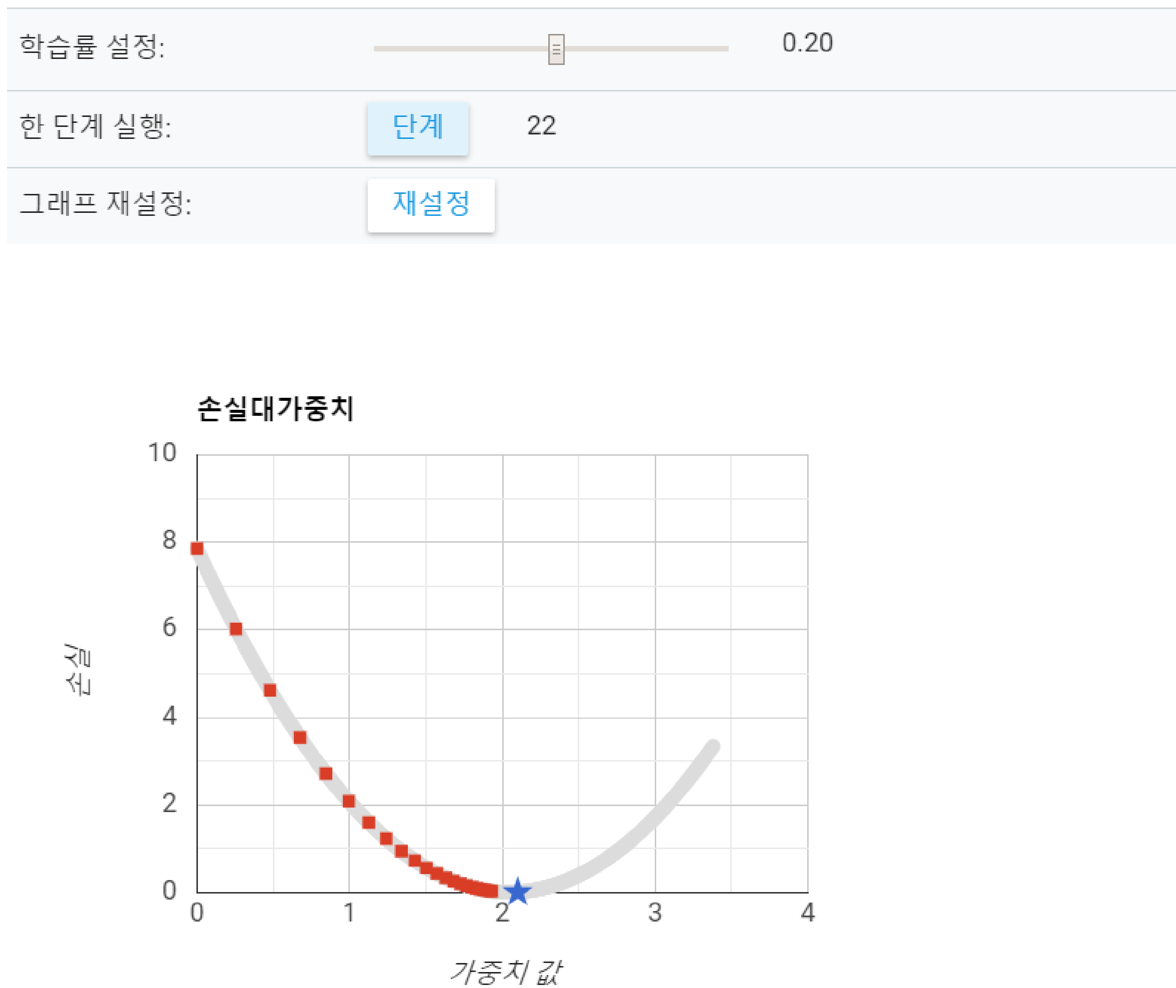
$$w_i^{new} = w_i^{old} - \alpha \times (y - t) \cdot 1 \cdot h(1 - h) \cdot x_i$$

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

Learning: Gradient Descent

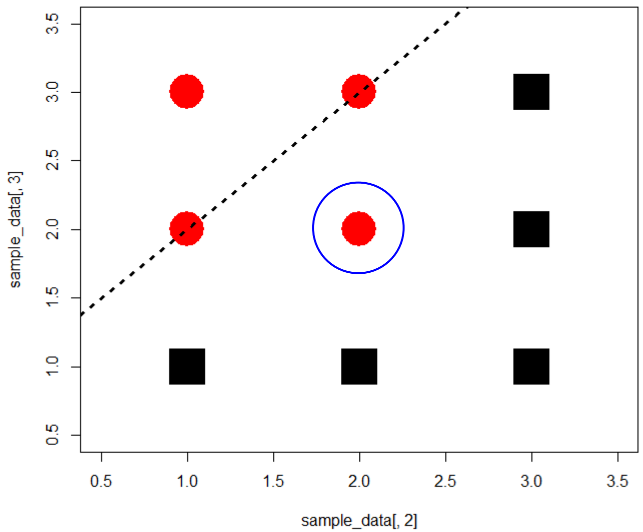
- The Effect of learning rate α

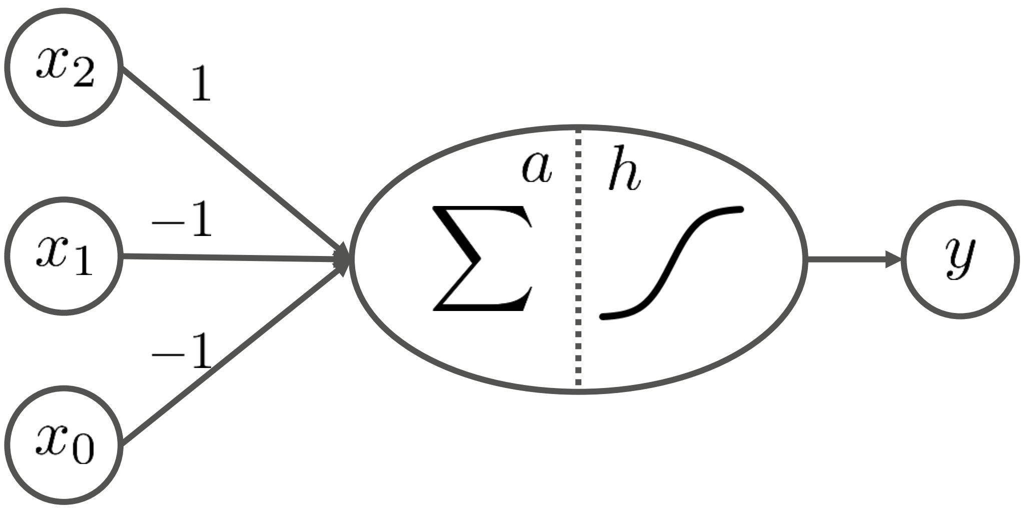


https://developers.google.com/machine-learning/crash-course/fitter/graph

Training Perceptron: Example 1 (alpha = 1)

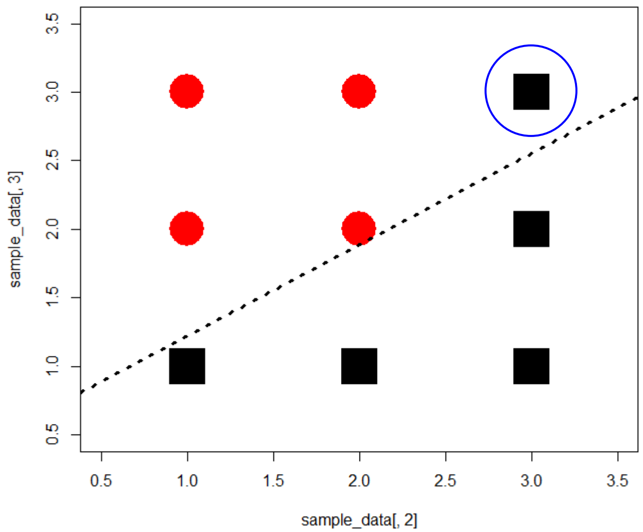
- Initialize and select the first training example (x_1=2, x_2=2, t=1)

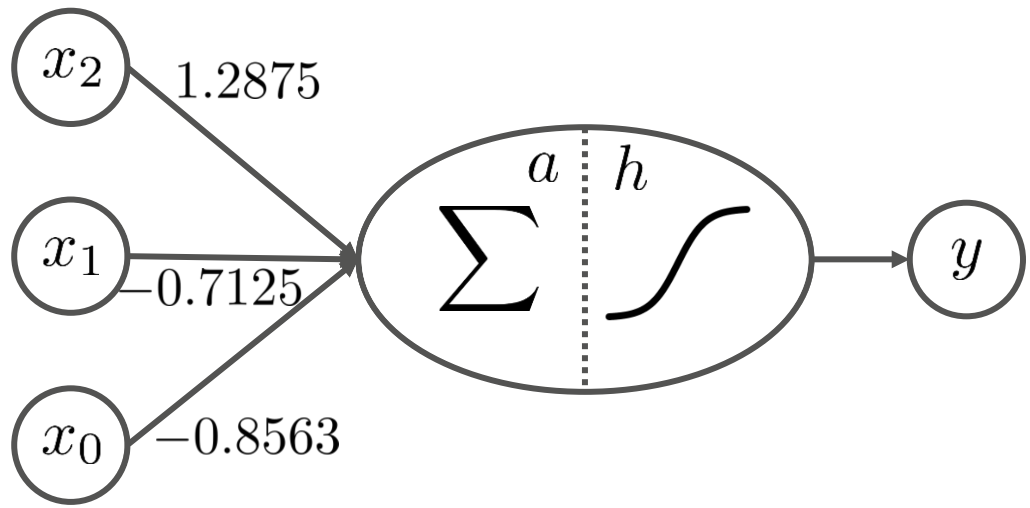




Training Perceptron: Example 1 (alpha = 1)

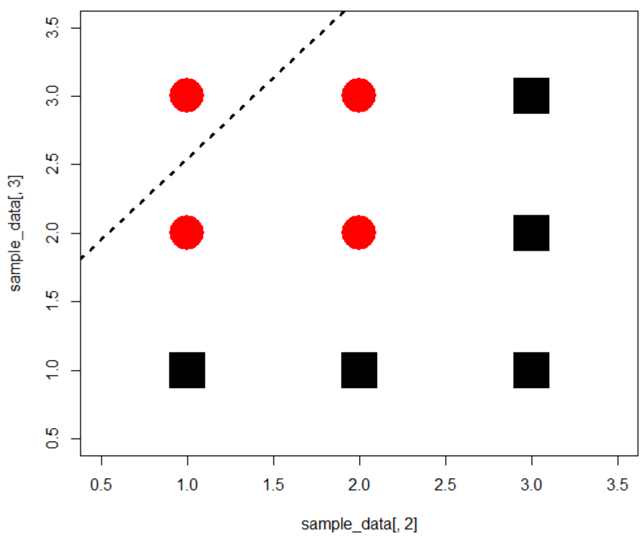
- Training result and selection of the second training example (x_1=3, x_2=3, t=0)

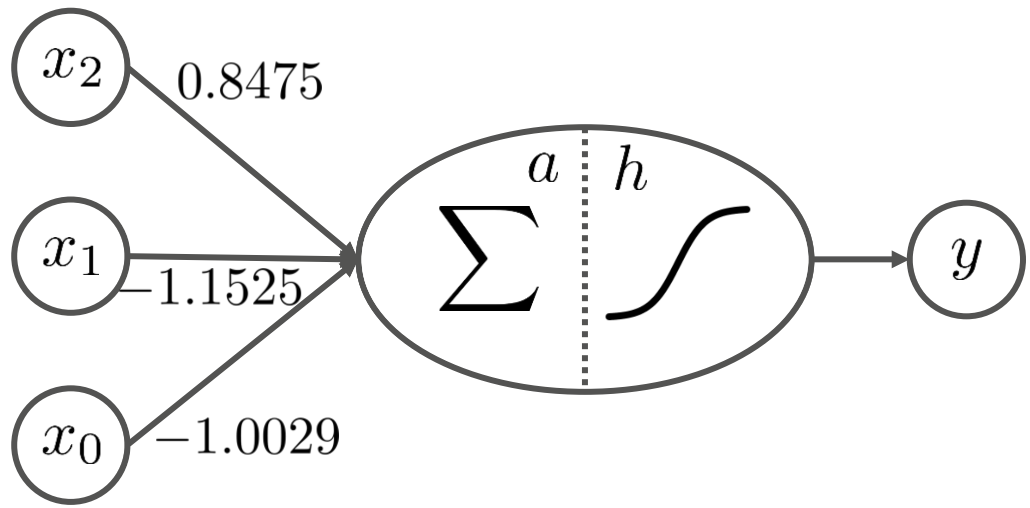




Training Perceptron: Example 1 (alpha = 1)

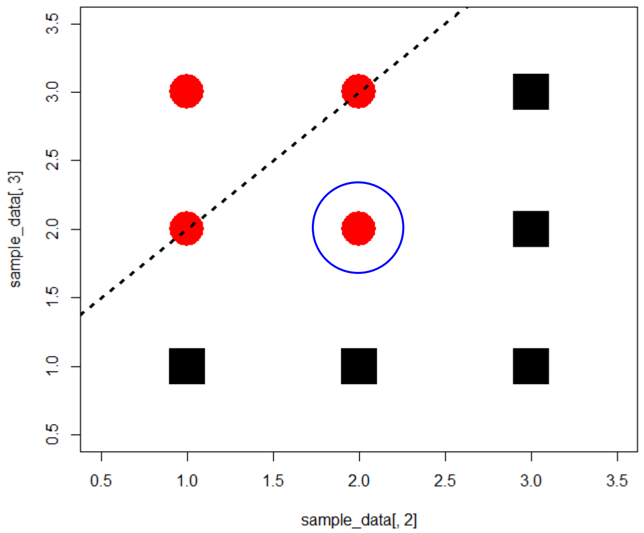
- Training result

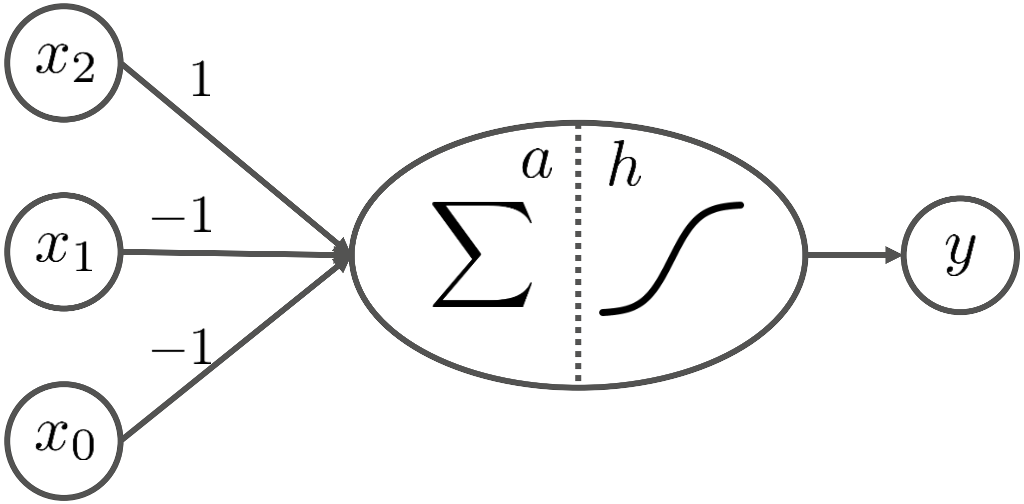




Training Perceptron: Example 1 (alpha = 0.5)

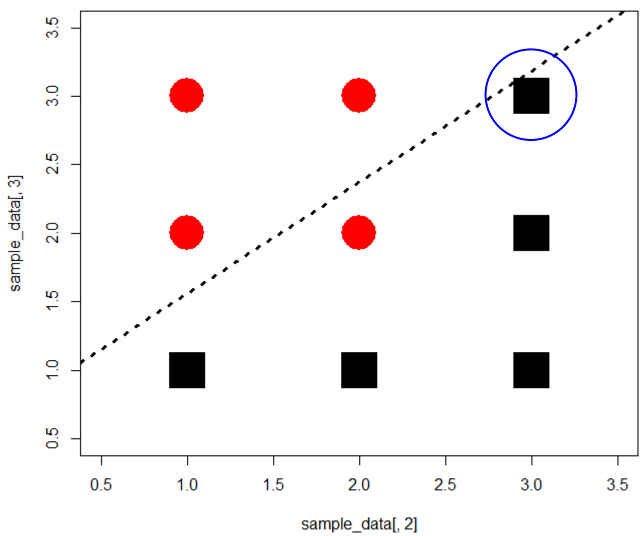
- Initialize and select the first training example (x_1=2, x_2=2, t=1)

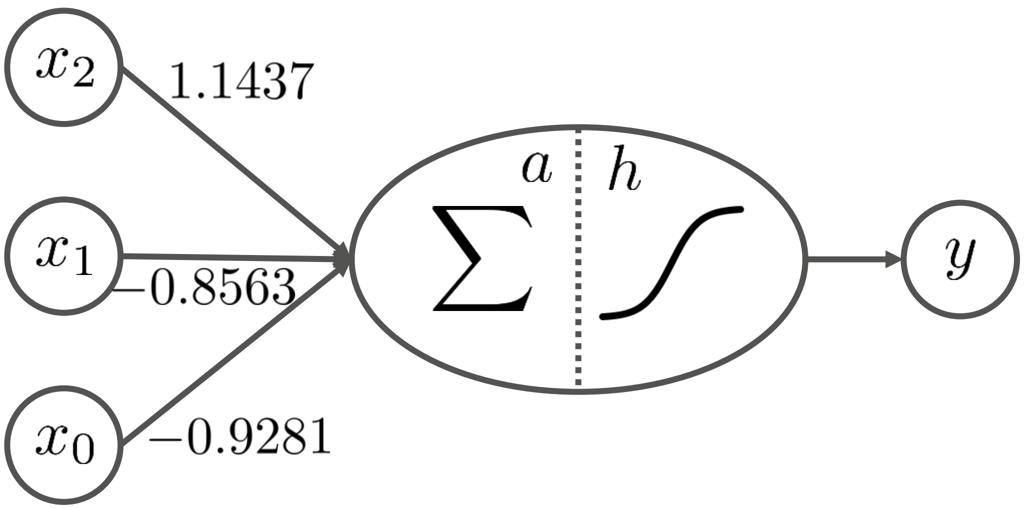




Training Perceptron: Example 1 (alpha = 0.5)

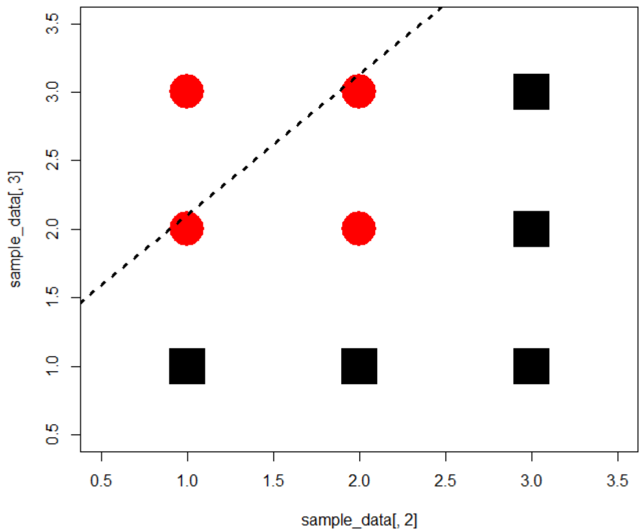
- Training result and selection of the second training example (x_1=3, x_2=3, t=0)

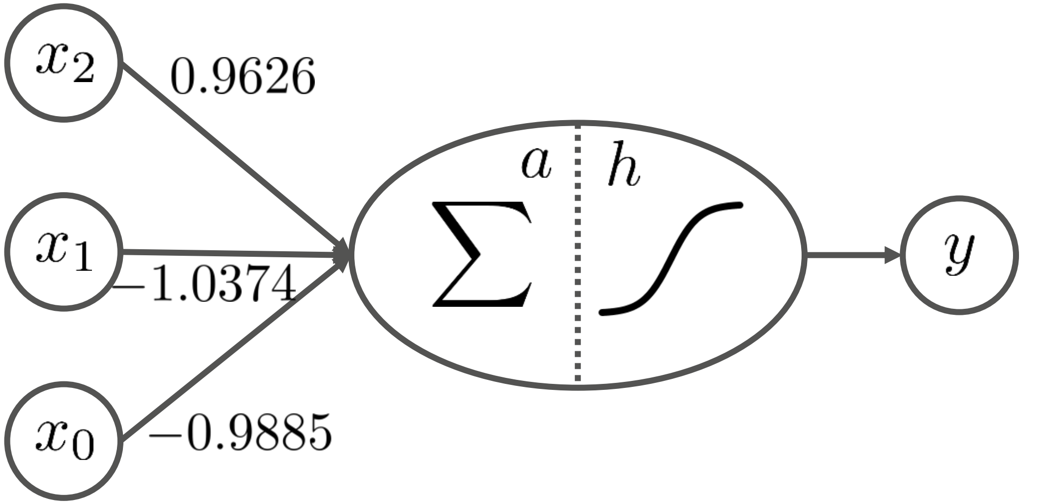




Training Perceptron: Example 1 (alpha = 0.5)

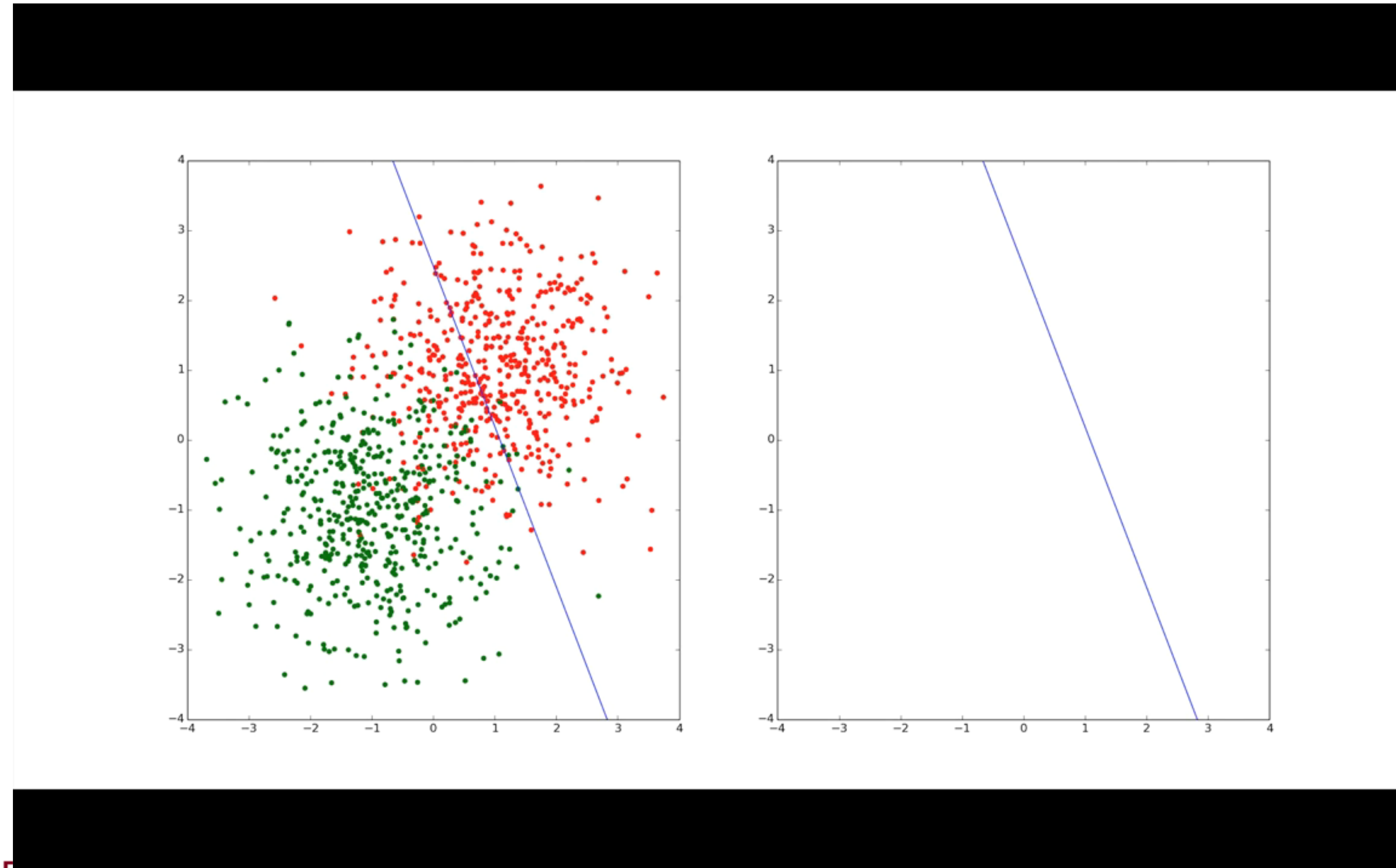
- Training result





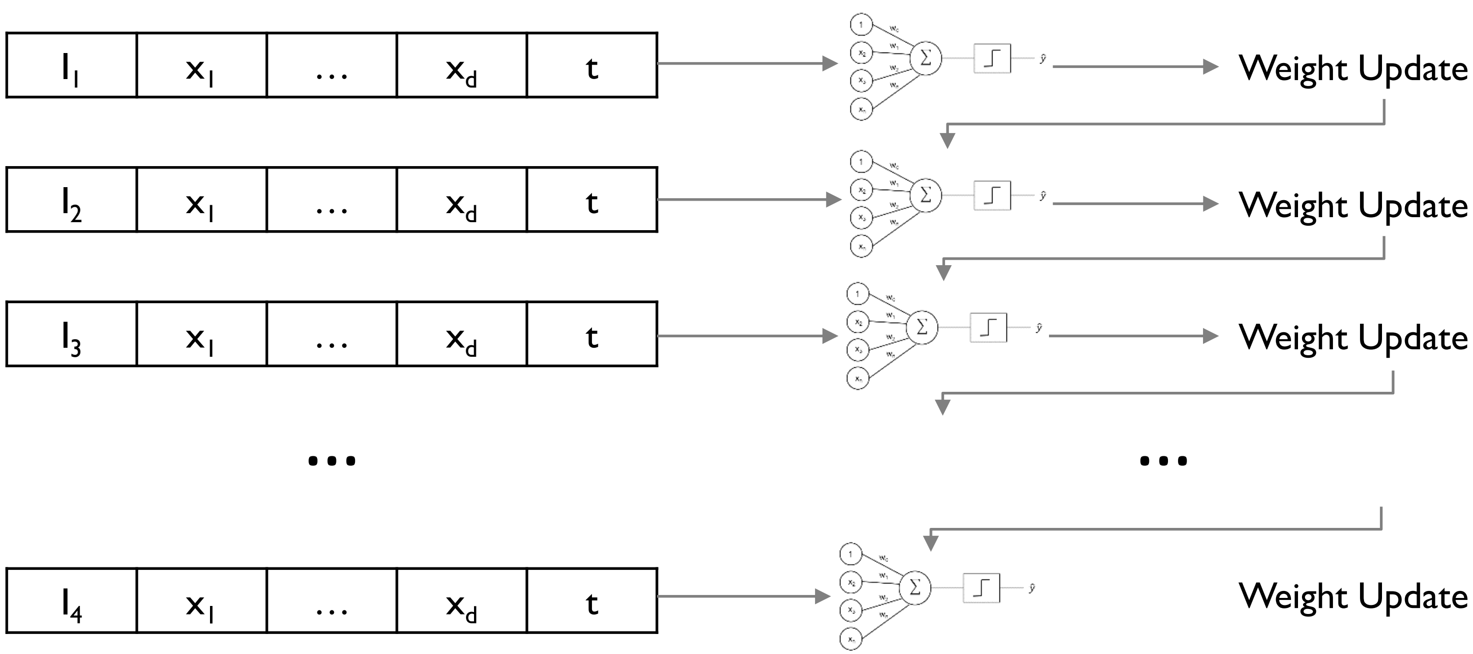
Perceptron

- Training Perceptron



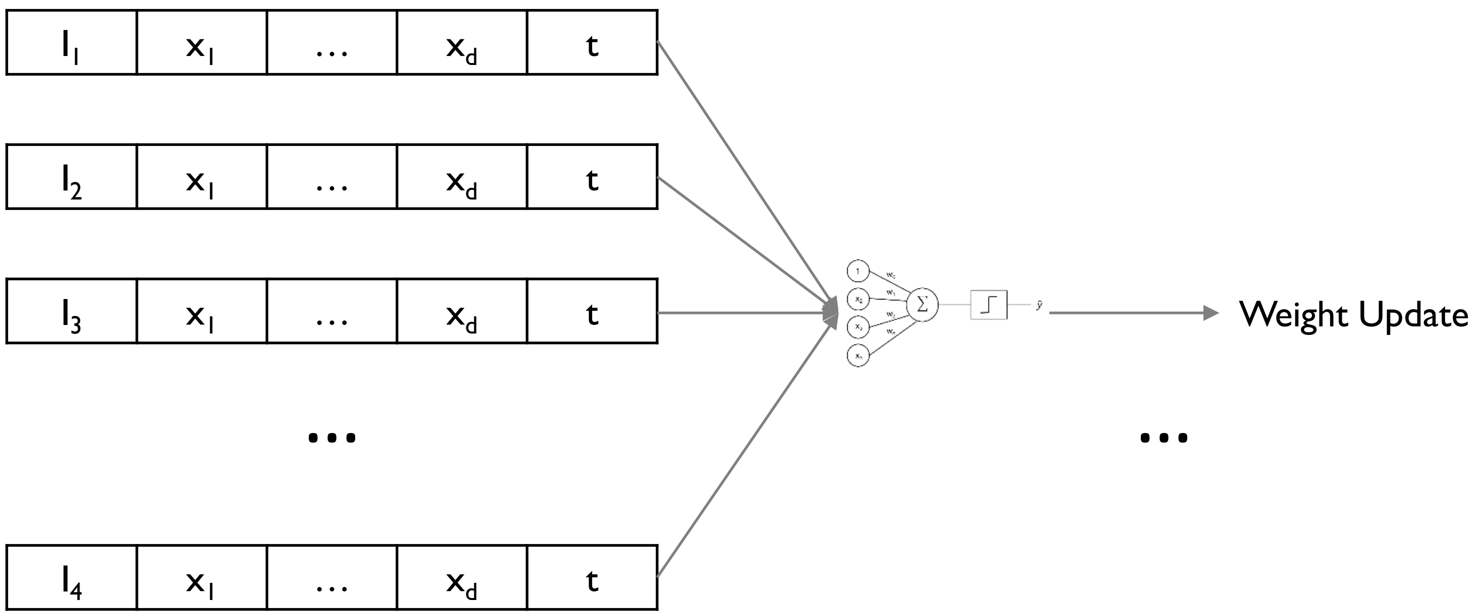
Issues on Gradient Descent

- 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



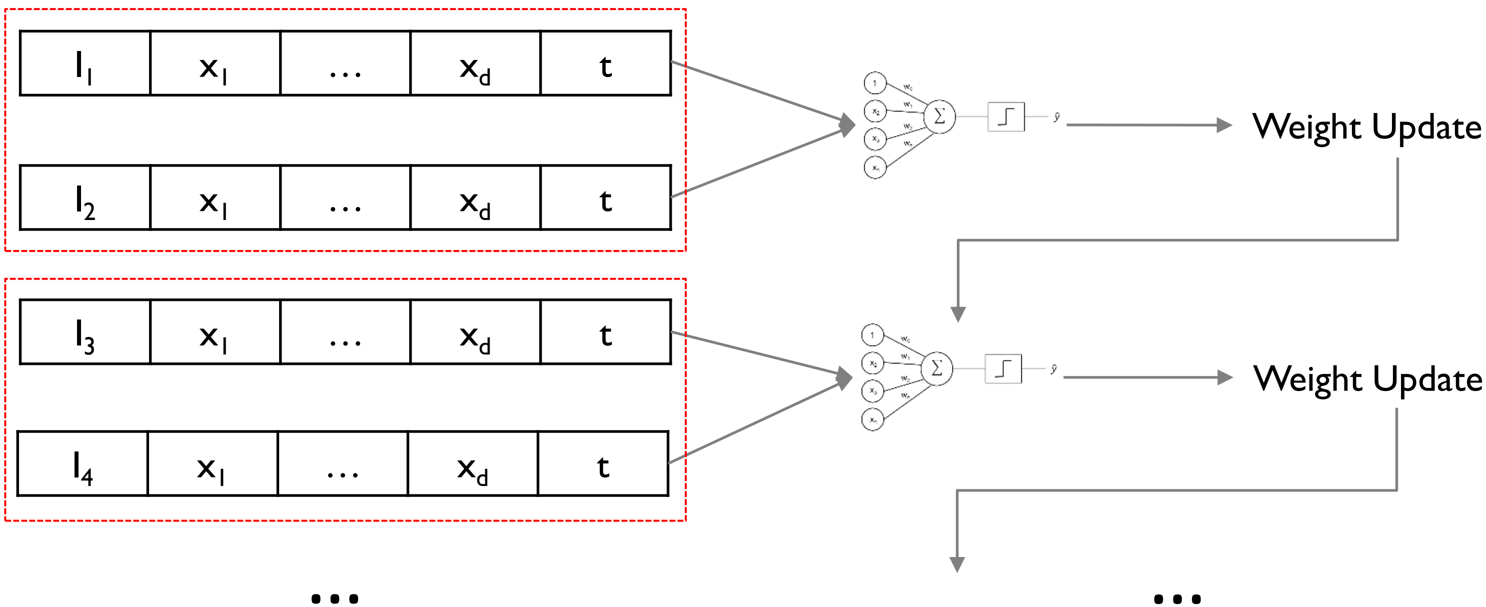
Issues on Gradient Descent

- 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



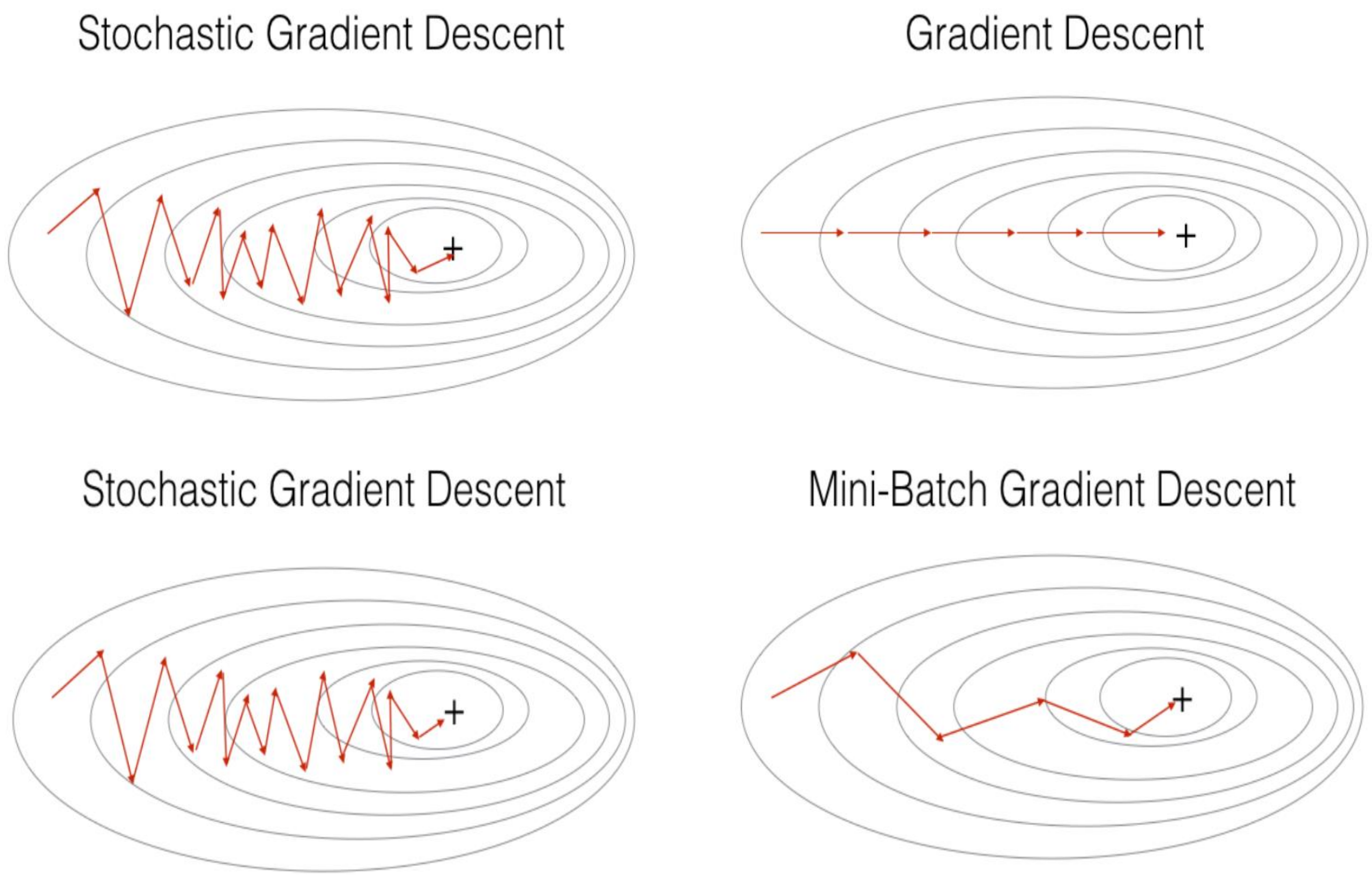
Issues on Gradient Descent

- 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

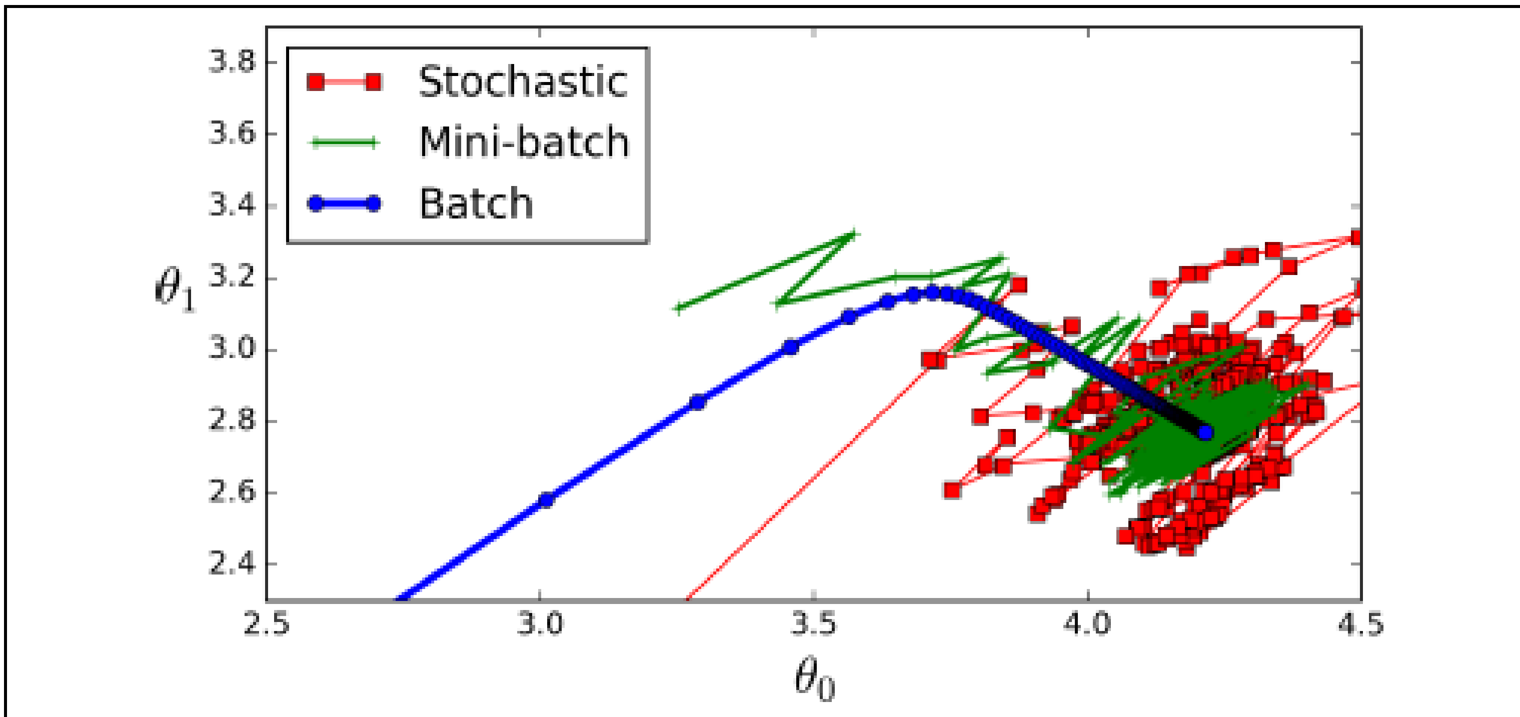
- Issue 1: How frequently should the weights be updated?



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

Issues on Gradient Descent

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

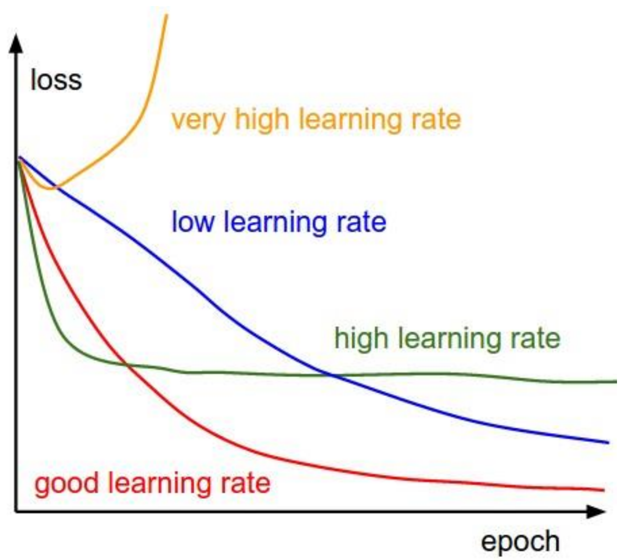


Issues on Gradient Descent

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

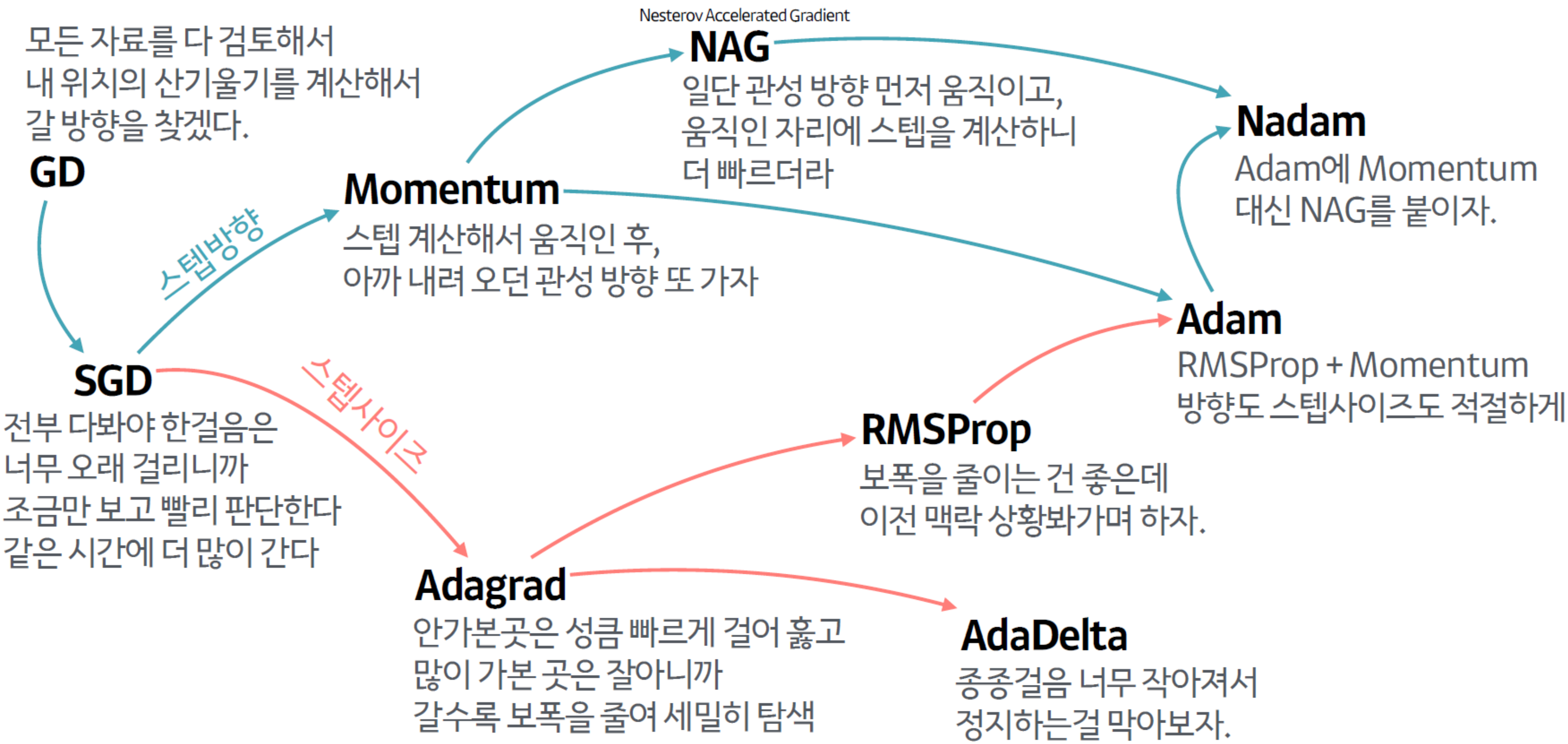
$$w_{new} = w_{old} - \alpha f'(w)$$

    - 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

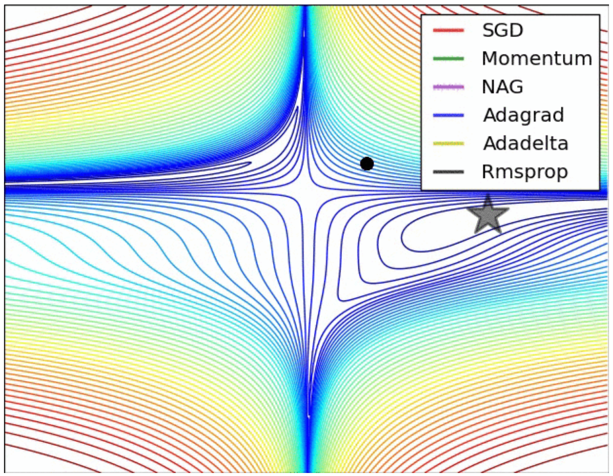
- Issue 2: How much should the weights be updated?



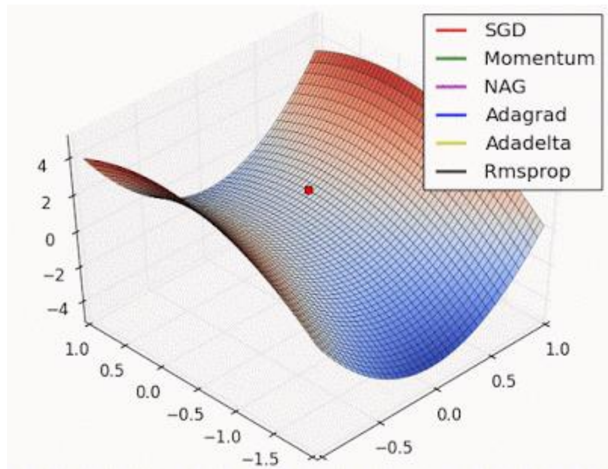
하용호. 자습해도 모르겠던 딥러닝, 머리속에 인스톨 시켜드립니다. (https://www.slideshare.net/yongho/ss-79607172)

Issues on Gradient Descent

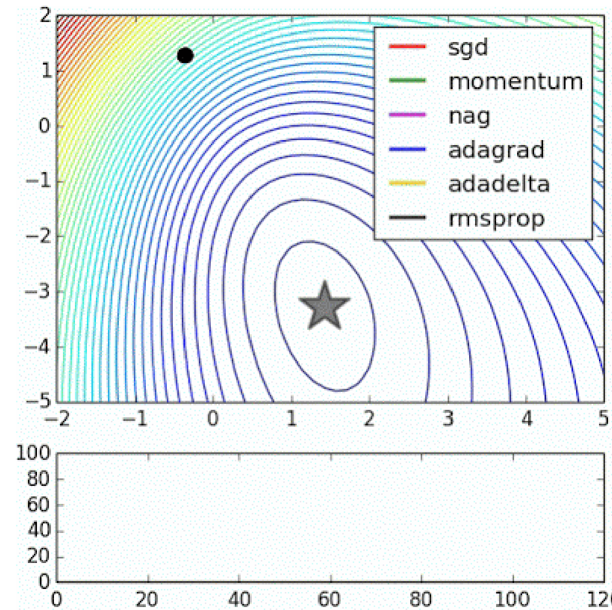
- Issue 2: How much should the weights be updated?



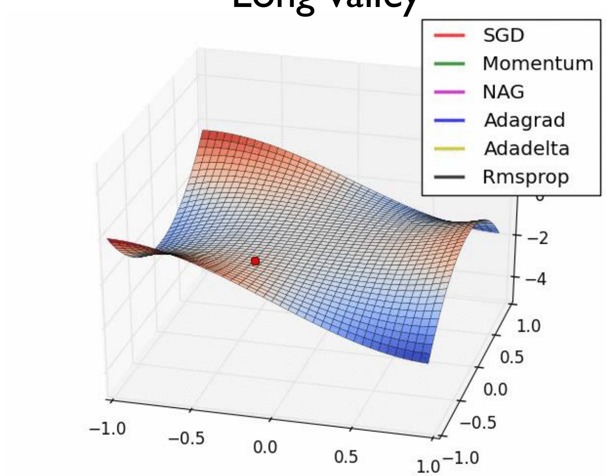
Beale's function



Long valley



Noisy moons



Saddle point

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


ANY
questions?