

$$\begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ 0 & & & \sigma_n \end{bmatrix}$$

$$X = \sum_{i=1}^{\text{rank}(X)} \sigma_i u_i v_i^T = U \Sigma V^T$$

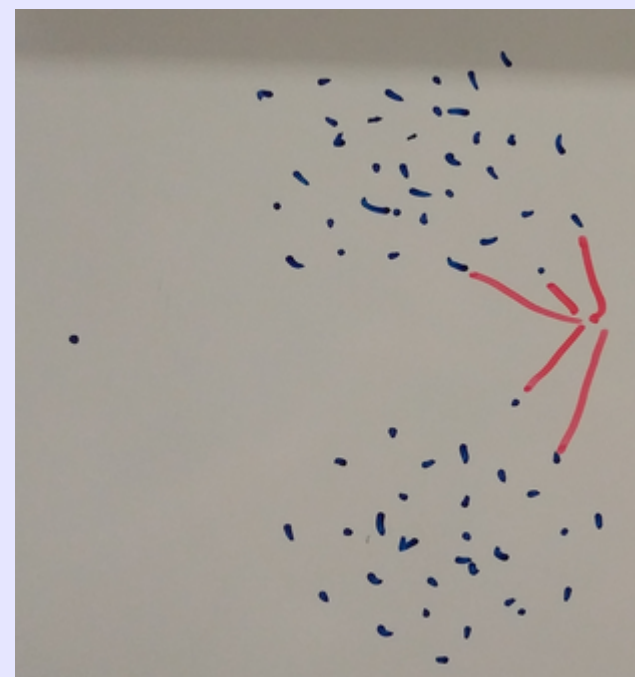
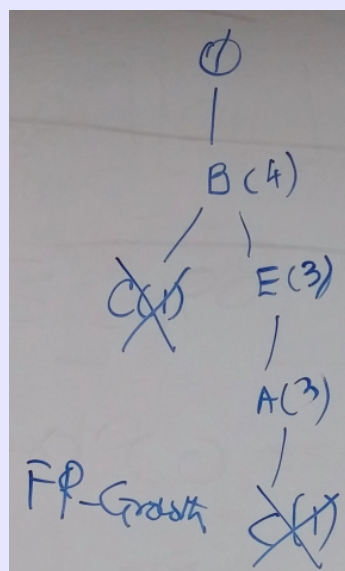
i^{th} singular value of X i^{th} left singular value of X (i^{th} column of U) i^{th} right singular vector of X (i^{th} column of V^T)

Captures the patterns among attributes
 Captures the patterns among the objects

$$\chi^2 = \sum \frac{(o - e)^2}{e}$$

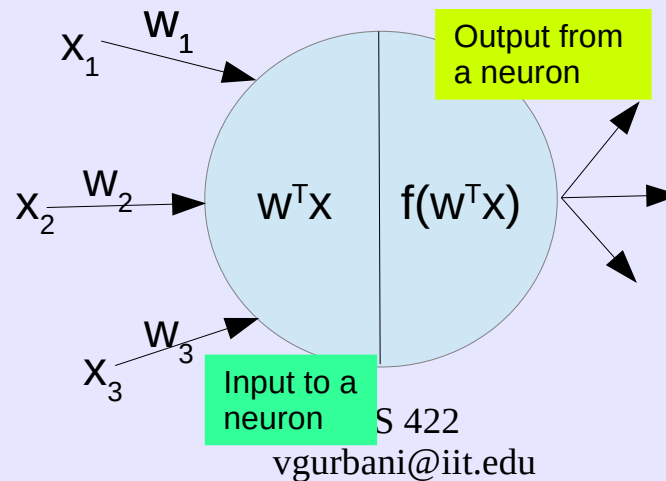
CS 422: Data Mining
 Vijay K. Gurbani, Ph.D.,
 Illinois Institute of Technology

Lecture: Introduction to Artificial Neural Networks

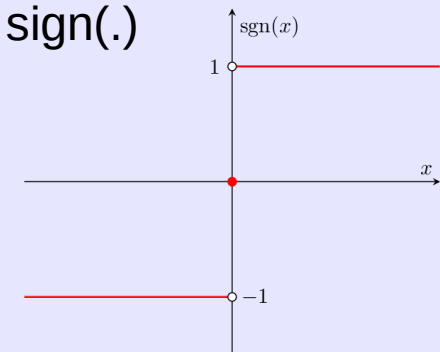


Introduction

- Hidden nodes learn *latent* representation (features useful for class boundaries).
- First hidden layer captures simpler features (since it receives the predictors as input).
- Subsequent hidden layers hone into specific patterns of the data to extract features.
- So, what does a hidden layer neuron do?



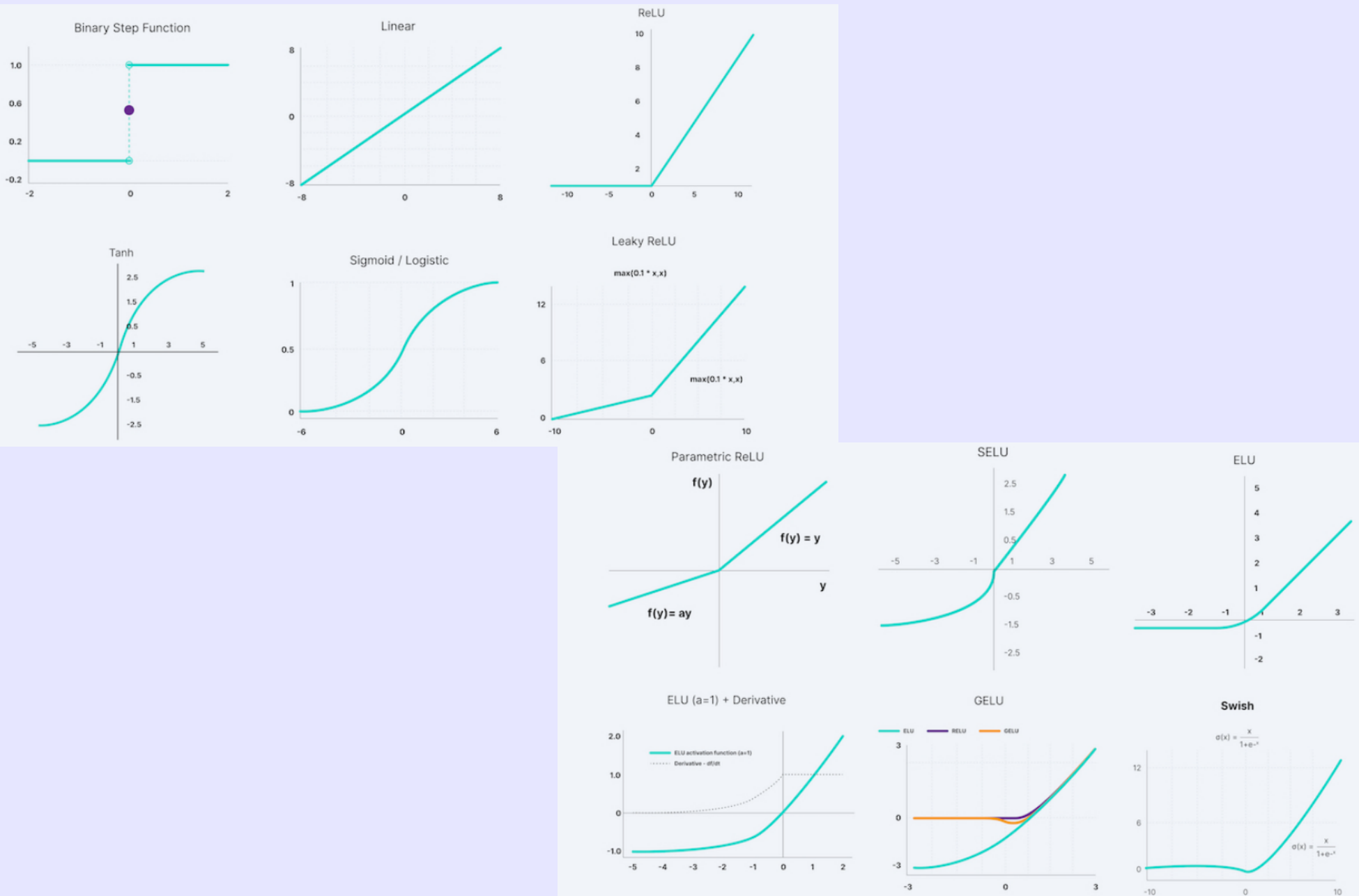
$f(.)$ is an activation function.
You have seen this before ...
... $\text{sign}(.)$



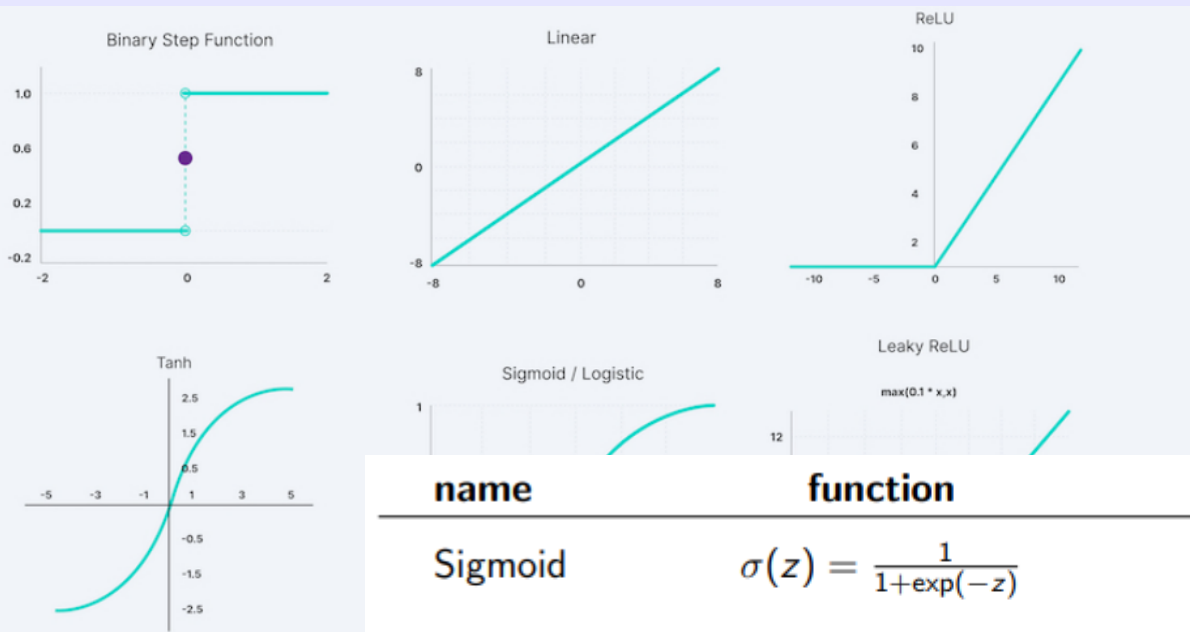
Activation functions

- This activation function is important, as it provides non-linearity to an ANN and allows it to create non-linear class boundaries.
- Different type of activation functions:

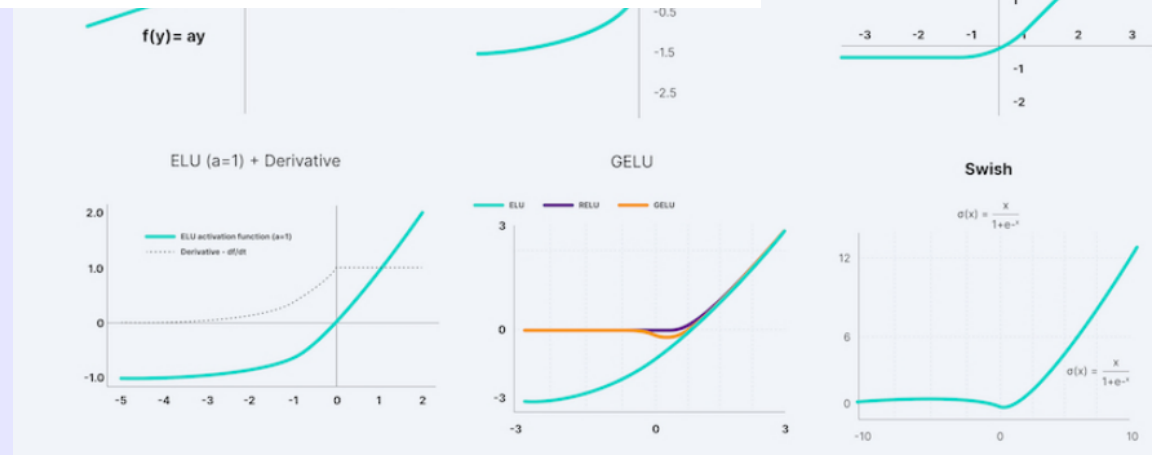
Activation functions



Activation functions



name	function	derivative
Sigmoid	$\sigma(z) = \frac{1}{1+\exp(-z)}$	$\sigma(z) \cdot (1 - \sigma(z))$
ReLU	$\text{ReLU}(z) = \max(0, z)$	$\begin{cases} 1, & \text{if } z > 0 \\ 0, & \text{if } z \leq 0 \end{cases}$

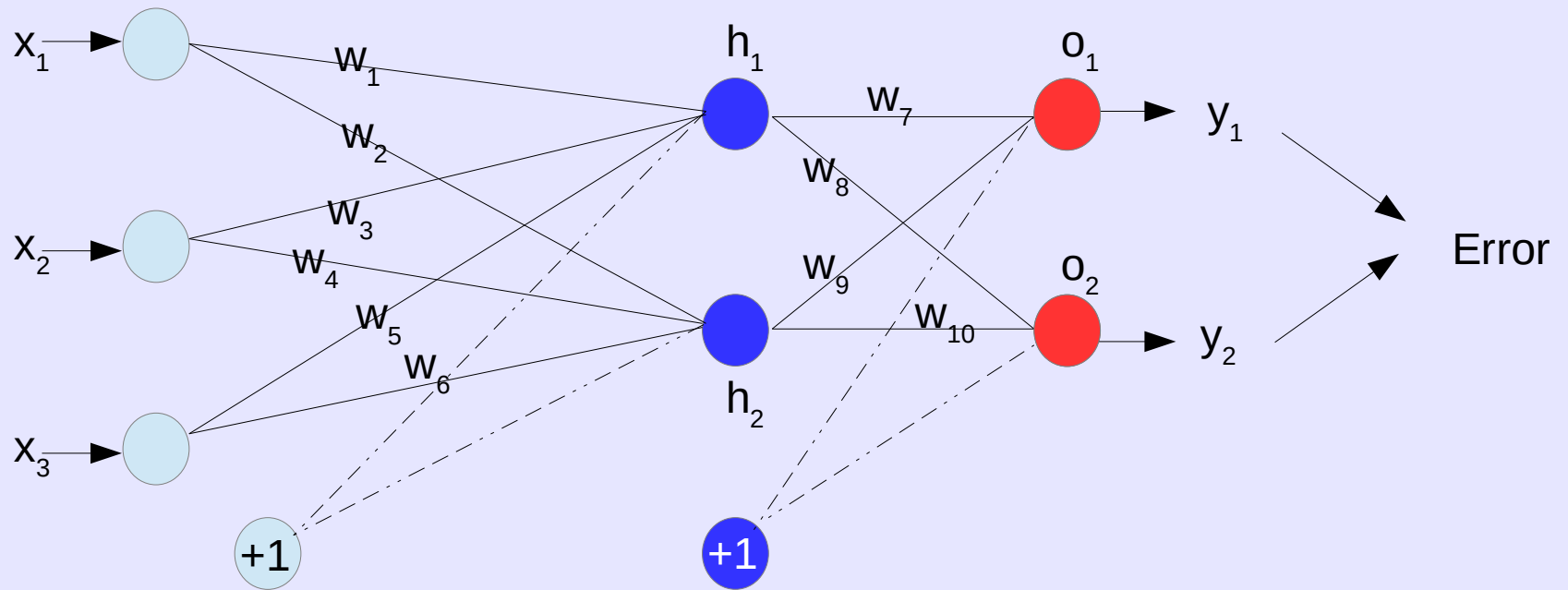


Introduction

- How do you choose an activation function?
 - Match the activation function at the output layer based on the type of prediction problem:
 - Regression: Linear activation function.
 - Binary classification: Sigmoid / Logistic activation function.
 - Multiclass classification: Softmax activation function.
 - For the hidden layers:
 - Start with relu activation function and move to others if results are sub-optimal.

Feed-Forward Neural Network

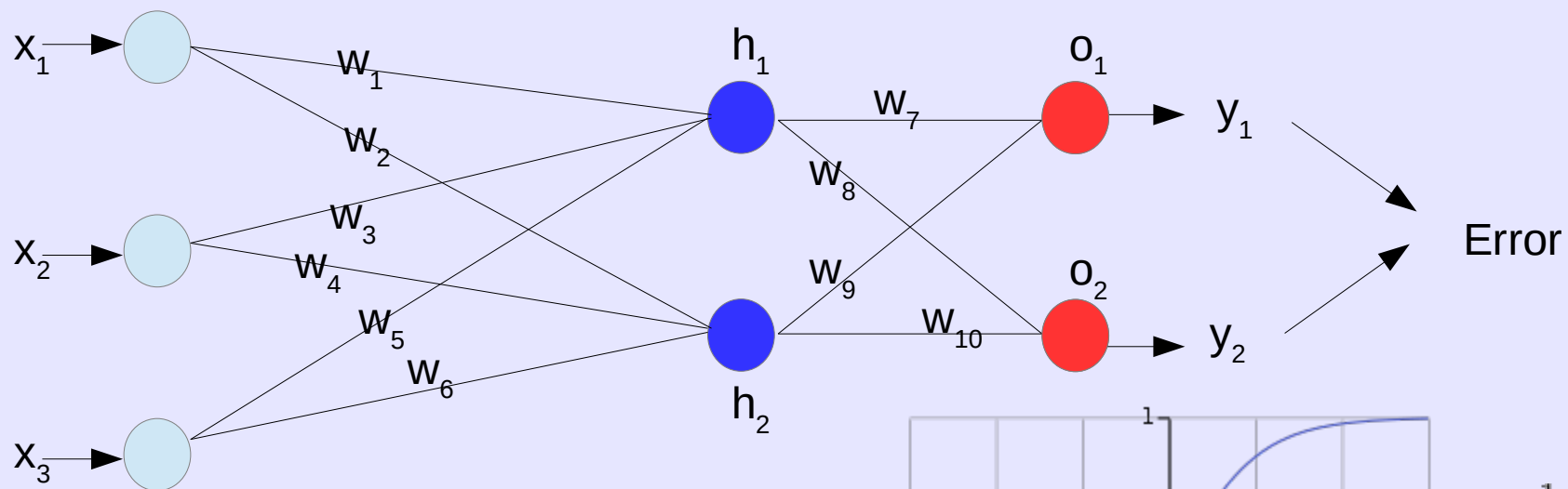
- Let's consider the simple one hidden layer neural network to understand how it learns.



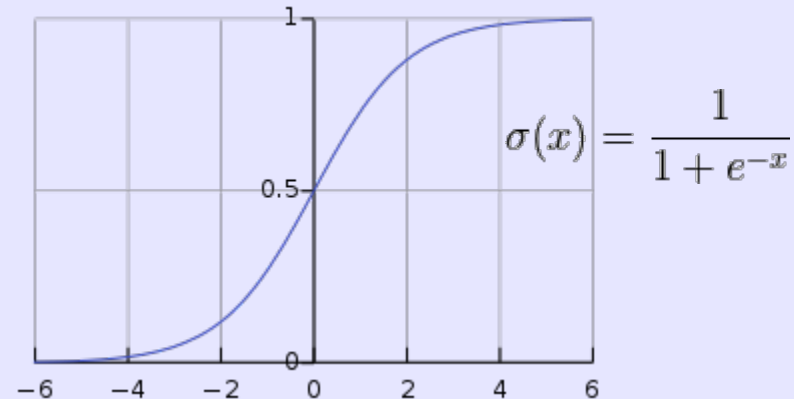
- Three inputs, plus bias, one hidden layer with two neurons, plus bias, and two output neurons.

Feed-Forward Neural Network

- Let's assume bias = 0, for simplification.

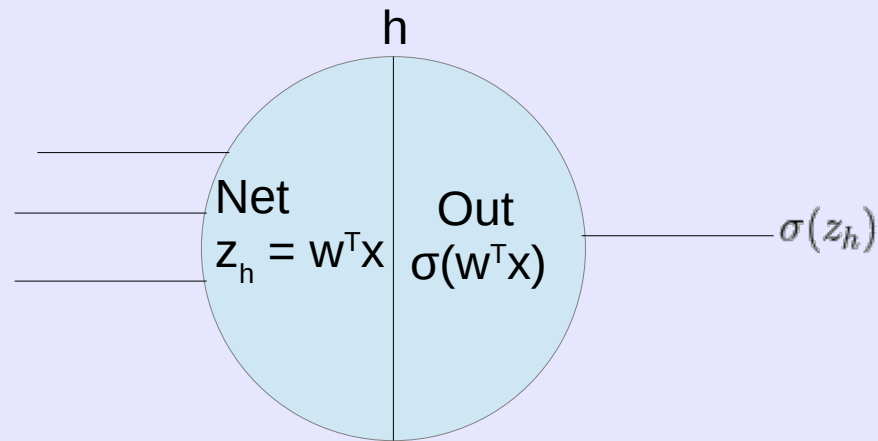


- The activation function at the hidden layers and at the output layer is the sigmoid:



- Loss: $\frac{1}{2} \sum_{i=1}^n (\hat{y}_i - y_i)^2$

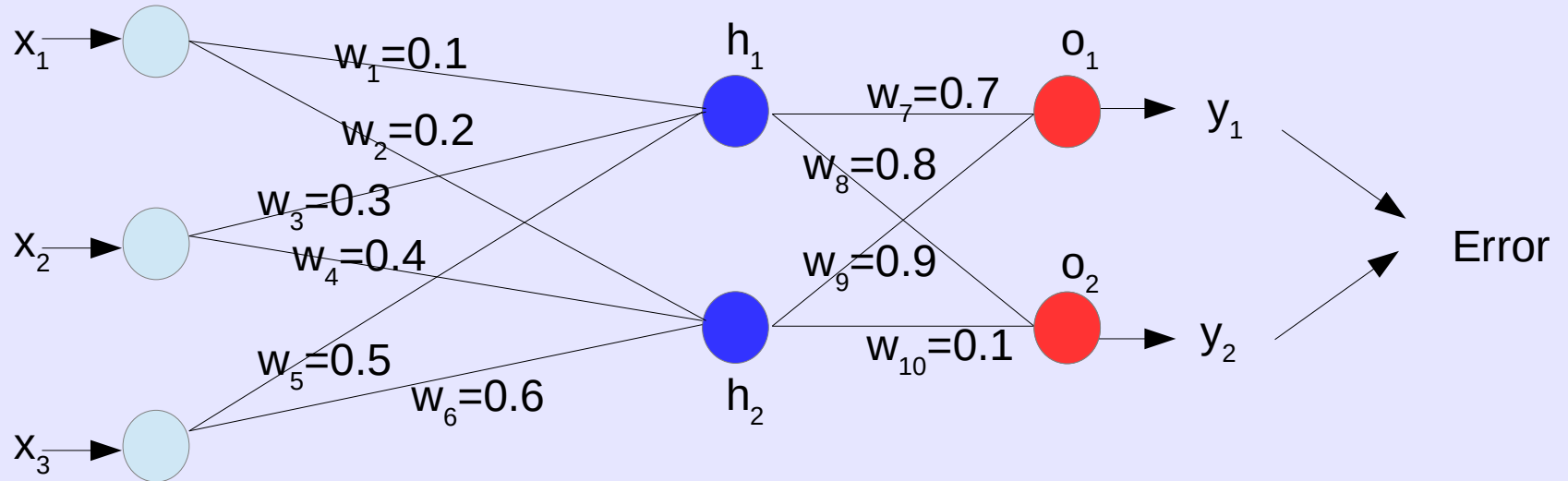
Feed-Forward Neural Network



For a neuron h , the network input to the neuron is: $z_h = w^T x = \sum_{i=1}^n w_i x_i = w_1 x_1 + \dots + w_n x_n$

For a neuron h , the output from the neuron is: $\sigma(w^T x) = \sigma(z_h)$, where σ is the activation function.

Feed-Forward Neural Network



Training example: $X = [1, 4, 5]$, $y = [0.1, 0.05]$; weights initialized as shown.

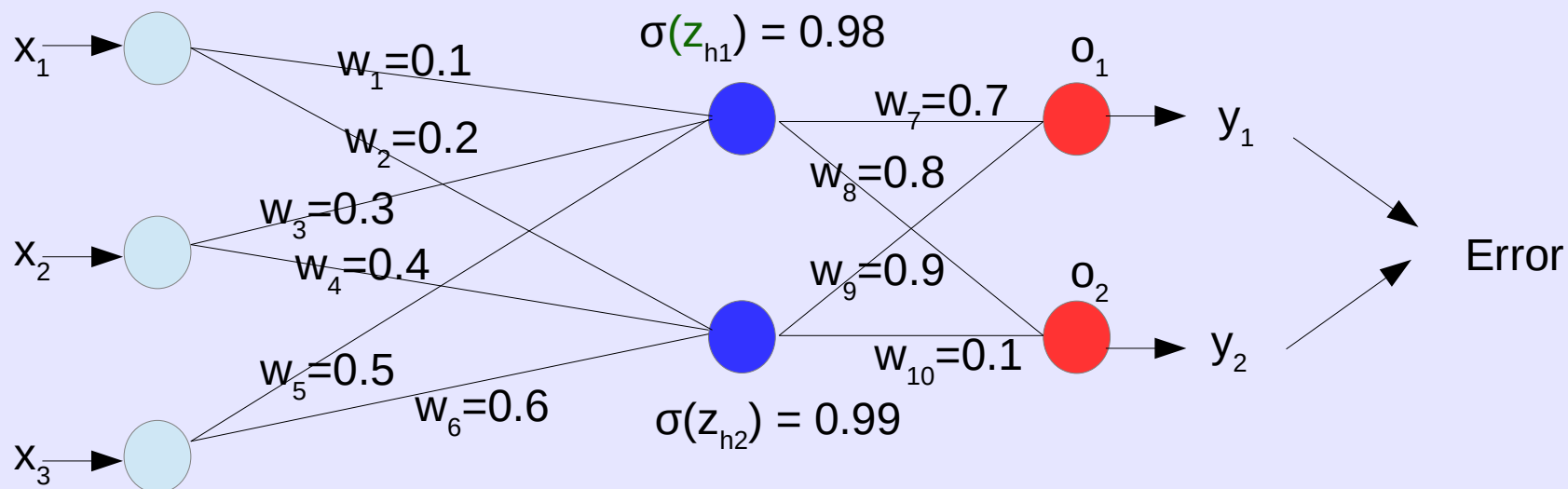
$$z_{h1} = (x_1 * w_1 + x_2 * w_3 + x_3 * w_5) = (1 * 0.1) + (4 * 0.3) + (5 * 0.5) = 3.8$$

$$\sigma(z_{h1}) = \sigma(3.8) = 0.98$$

$$z_{h2} = (x_1 * w_2 + x_2 * w_4 + x_3 * w_6) = (1 * 0.2) + (4 * 0.4) + (5 * 0.6) = 4.8$$

$$\sigma(z_{h2}) = \sigma(4.8) = 0.99$$

Feed-Forward Neural Network



Training example: $X = [1, 4, 5]$, $y = [0.1, 0.05]$; weights initialized as shown.

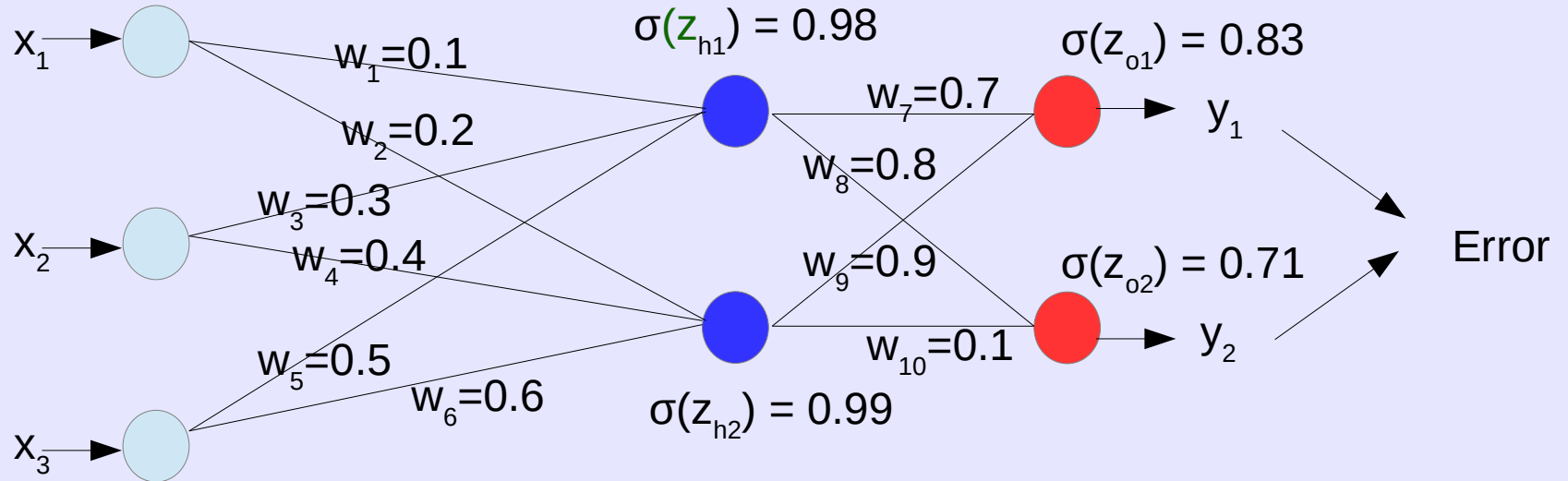
$$z_{o1} = (\sigma(z_{h1}) * w_7 + \sigma(z_{h2}) * w_9) = (0.98 * 0.7) + (0.99 * 0.9) = 1.58$$

$$\sigma(z_{o1}) = \sigma(1.58) = 0.83$$

$$z_{o2} = (\sigma(z_{h1}) * w_8 + \sigma(z_{h2}) * w_{10}) = (0.98 * 0.8) + (0.99 * 0.1) = 0.88$$

$$\sigma(z_{o2}) = \sigma(0.88) = 0.71$$

Feed-Forward Neural Network

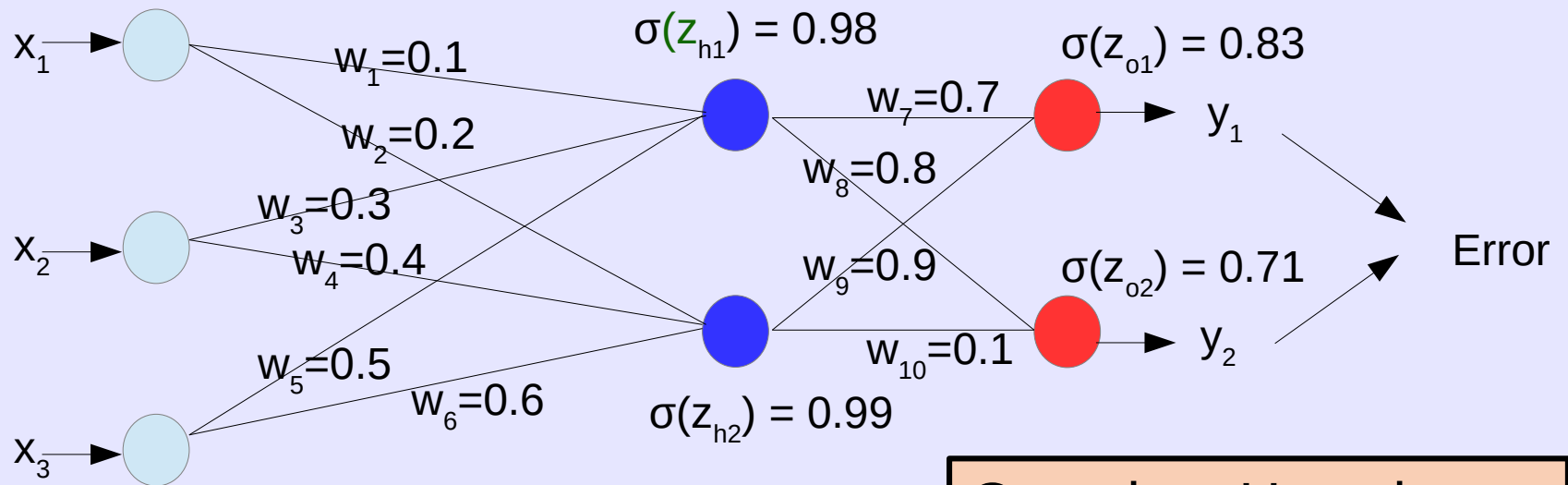


Training example: $X = [1, 4, 5]$, $y = [0.10, 0.05]$; weights initialized as shown.

$$z_{o1} = (\sigma(z_{h1}) * w_7 + \sigma(z_{h2}) * w_9) = (0.98 * 0.7) + (0.99 * 0.9) = 1.58$$
$$\sigma(z_{o1}) = \sigma(1.58) = 0.83$$

$$z_{o2} = (\sigma(z_{h1}) * w_8 + \sigma(z_{h2}) * w_{10}) = (0.98 * 0.8) + (0.99 * 0.1) = 0.88$$
$$\sigma(z_{o2}) = \sigma(0.88) = 0.71$$

Feed-Forward Neural Network



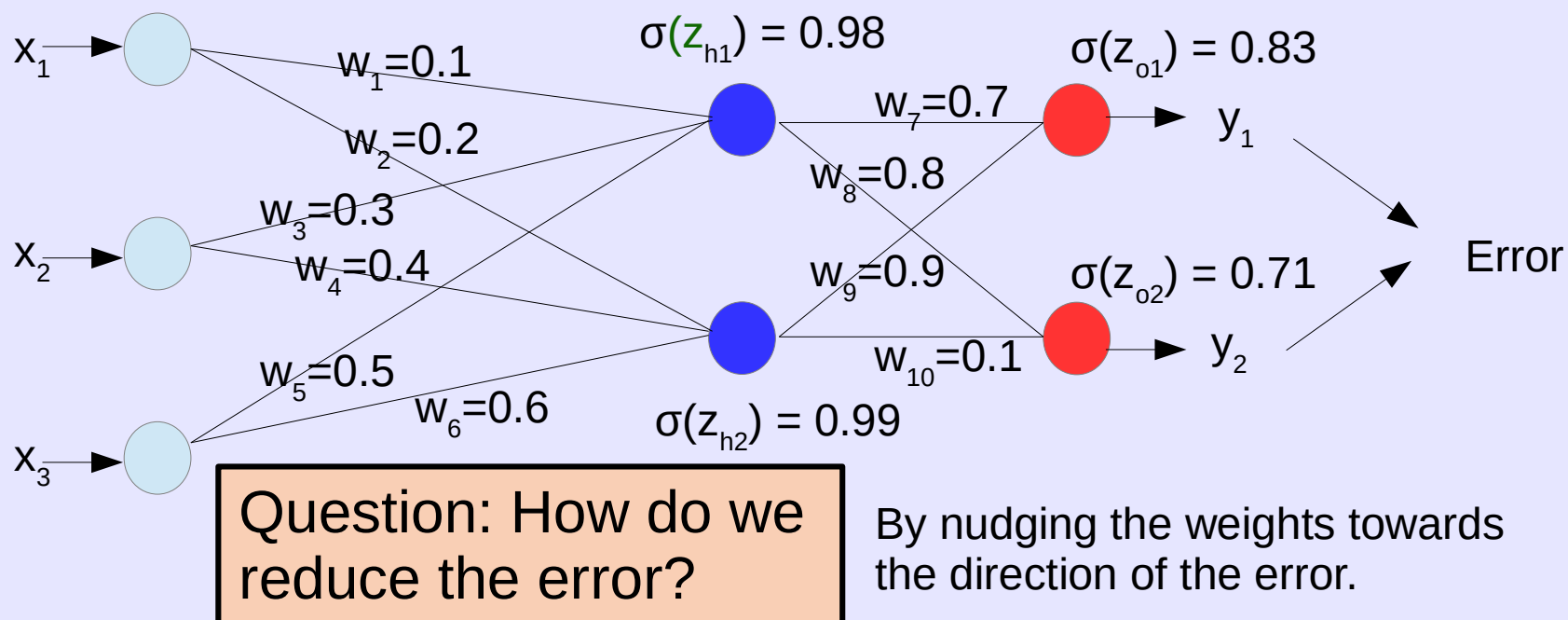
Error:

$$\frac{1}{2}[(\sigma(z_{o1}) - y_1)^2 + (\sigma(z_{o2}) - y_2)^2] =$$

$$\frac{1}{2}[(0.83 - 0.1)^2 + (0.71 - 0.05)^2] = 0.48$$

Question: How do we reduce the error?

Feed-Forward Neural Network



But how?

- Using the chain rule from calculus to derive a **gradient** vector. Recall:

$$\nabla f(x) = \left[\frac{\partial f(x_1, \dots, x_n)}{\partial(x_1)}, \frac{\partial f(x_1, \dots, x_n)}{\partial(x_2)}, \dots, \frac{\partial f(x_1, \dots, x_n)}{\partial(x_n)} \right]$$

- Here, the gradient will be of the error function with respect to each weight (and bias):

$$E(w, b) = \frac{1}{2} \sum_{i=1}^n \text{Loss}(y_i, \hat{y}_i)^2$$

- Typical choice of a loss function is the squared loss function discussed previously.

- Use the gradients to explore the minima of the error function: Gradient Descent algorithm.

Backpropagation

- Chain Rule:

$$y = f(g(x)),$$

What is $\frac{dy}{dx}$?

Backpropagation

- Chain Rule:

$$y = f(g(x)),$$

What is $\frac{dy}{dx}$?

Let $u = g(x)$, then
 $y = f(u)$, and

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

Backpropagation

- Chain Rule:

$$y = f(g(x)),$$

What is $\frac{dy}{dx}$?

Let $u = g(x)$, then
 $y = f(u)$, and

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

$$y = f(g(h(x)))$$

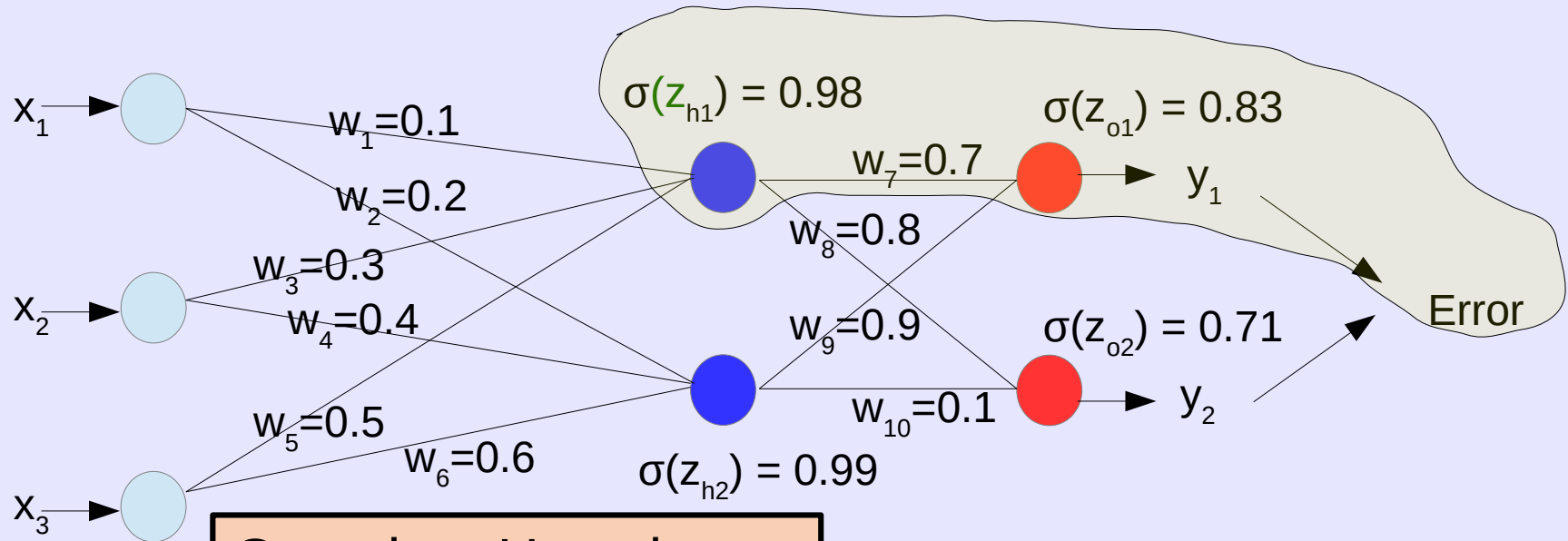
Let $u = h(x)$

$$v = g(u)$$

$$y = f(v)$$

$$\frac{dy}{dx} = \frac{dy}{dv} \cdot \frac{dv}{du} \cdot \frac{du}{dx}$$

Backpropagation



Question: How do we reduce the error?

By nudging the weights towards the direction of the error.

But how?

- Using the chain rule from calculus to derive a **gradient** vector. Recall:

$$\nabla f(x) = \left[\frac{\partial f(x_1, \dots, x_n)}{\partial(x_1)}, \frac{\partial f(x_1, \dots, x_n)}{\partial(x_2)}, \dots, \frac{\partial f(x_1, \dots, x_n)}{\partial(x_n)} \right]$$

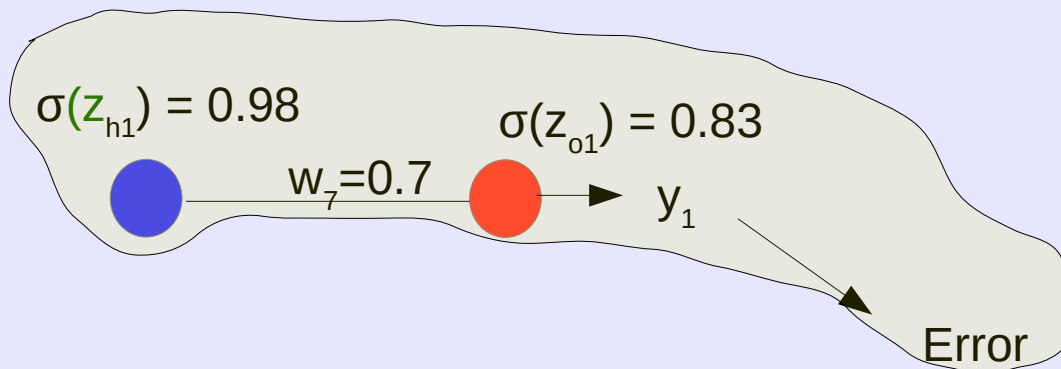
- Here, the gradient will be of the error function with respect to each weight (and bias):

$$E(w, b) = \frac{1}{2} \sum_{i=1}^n \text{Loss}(y_i, \hat{y}_i)^2$$

- Typical choice of a loss function is the squared loss function discussed previously.

- Use the gradients to explore the minima of the error function: Gradient Descent algorithm.

Backpropagation



$$y = [0.10, 0.05]$$

Activation function is Sigmoid, $\sigma(x) = \frac{1}{1 + e^{-x}}$

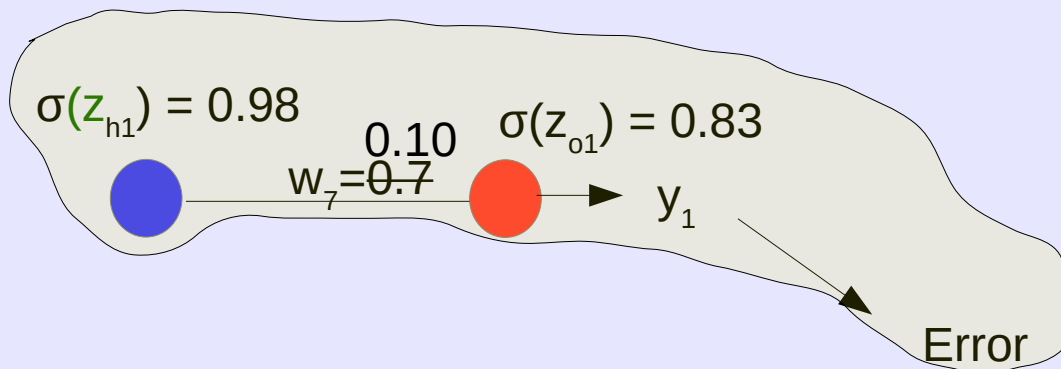
$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2} = \sigma(x)(1 - \sigma(x))$$

$$\sigma(Z_{o1}) = 0.83, y = 0.10, \sigma(Z_{h1}) = 0.98$$

Our error is 0.48 See slide 20 We need to nudge the error to 0.00

$$\begin{aligned} E(w, b) &= \frac{1}{2} \sum_{i=1}^n \text{Loss}(y_i, \hat{y}_i)^2 \\ &= \frac{1}{2} [(\sigma(Z_{o1}) - y_1)^2 + (\sigma(Z_{o2}) - y_2)^2] \\ \frac{\partial E}{\partial \sigma(Z_{o1})} &= \sigma(Z_{o1}) - y_1 \end{aligned}$$

Backpropagation



$$y = [0.10, 0.05]$$

Activation function is Sigmoid, $\sigma(x) = \frac{1}{1 + e^{-x}}$

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

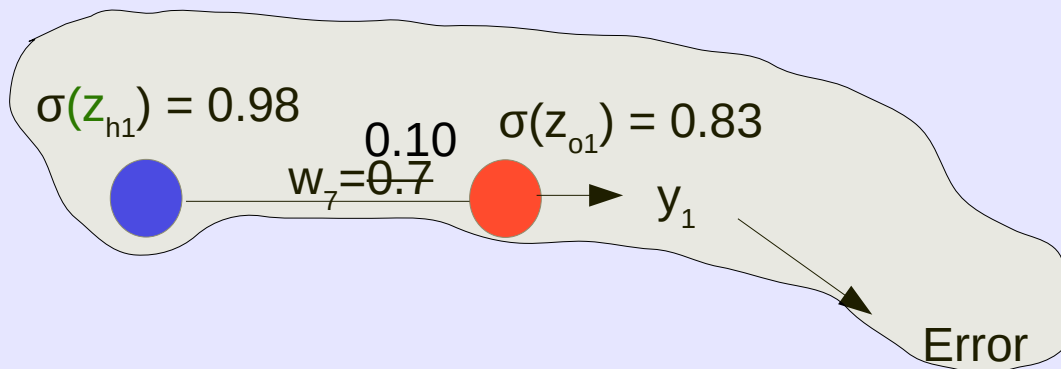
$$\sigma(Z_{o1}) = 0.83, y = 0.10, \sigma(Z_{h1}) = 0.98$$

Our error is 0.48 See slide 20 We need to nudge the error to 0.00

$$\begin{aligned} E(w, b) &= \frac{1}{2} \sum_{i=1}^n \text{Loss}(y_i, \hat{y}_i)^2 \\ &= \frac{1}{2} [(\sigma(Z_{o1}) - y_1)^2 + (\sigma(Z_{o2}) - y_2)^2] \\ \frac{\partial E}{\partial \sigma(Z_{o1})} &= \sigma(Z_{o1}) - y_1 \end{aligned}$$

$$\begin{aligned} \frac{\partial E}{\partial w_7} &= \frac{\partial E}{\partial \sigma(Z_{o1})} \cdot \frac{\partial \sigma(Z_{o1})}{\partial Z_{o1}} \cdot \frac{\partial Z_{o1}}{\partial w_7} \\ &= [(\sigma(Z_{o1}) - y_1)] \cdot [\sigma(Z_{o1})(1 - \sigma(Z_{o1}))] \cdot [\sigma(Z_{h1})] \\ &= [(0.83 - 0.1)] \cdot [(0.83 - (1 - 0.83))] \cdot [0.98] \\ &= 0.10 \end{aligned}$$

Backpropagation



$$y = [0.10, 0.05]$$

Activation function is Sigmoid, $\sigma(x) = \frac{1}{1 + e^{-x}}$

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

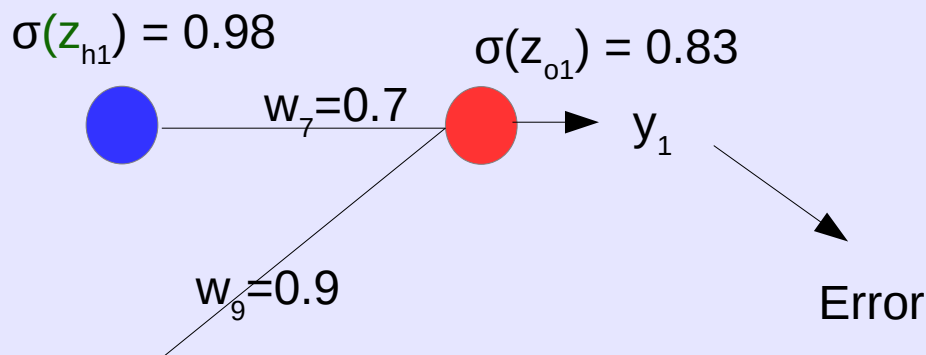
$$\sigma(Z_{o1}) = 0.83, y = 0.10, \sigma(Z_{h1}) = 0.98$$

Our error is 0.48 See slide 20 We need to nudge the error to 0.00

$$\begin{aligned} E(w, b) &= \frac{1}{2} \sum_{i=1}^n \text{Loss}(y_i, \hat{y}_i)^2 \\ &= \frac{1}{2} [(\sigma(Z_{o1}) - y_1)^2 + (\sigma(Z_{o2}) - y_2)^2] \\ \frac{\partial E}{\partial \sigma(Z_{o1})} &= \sigma(Z_{o1}) - y_1 \end{aligned}$$

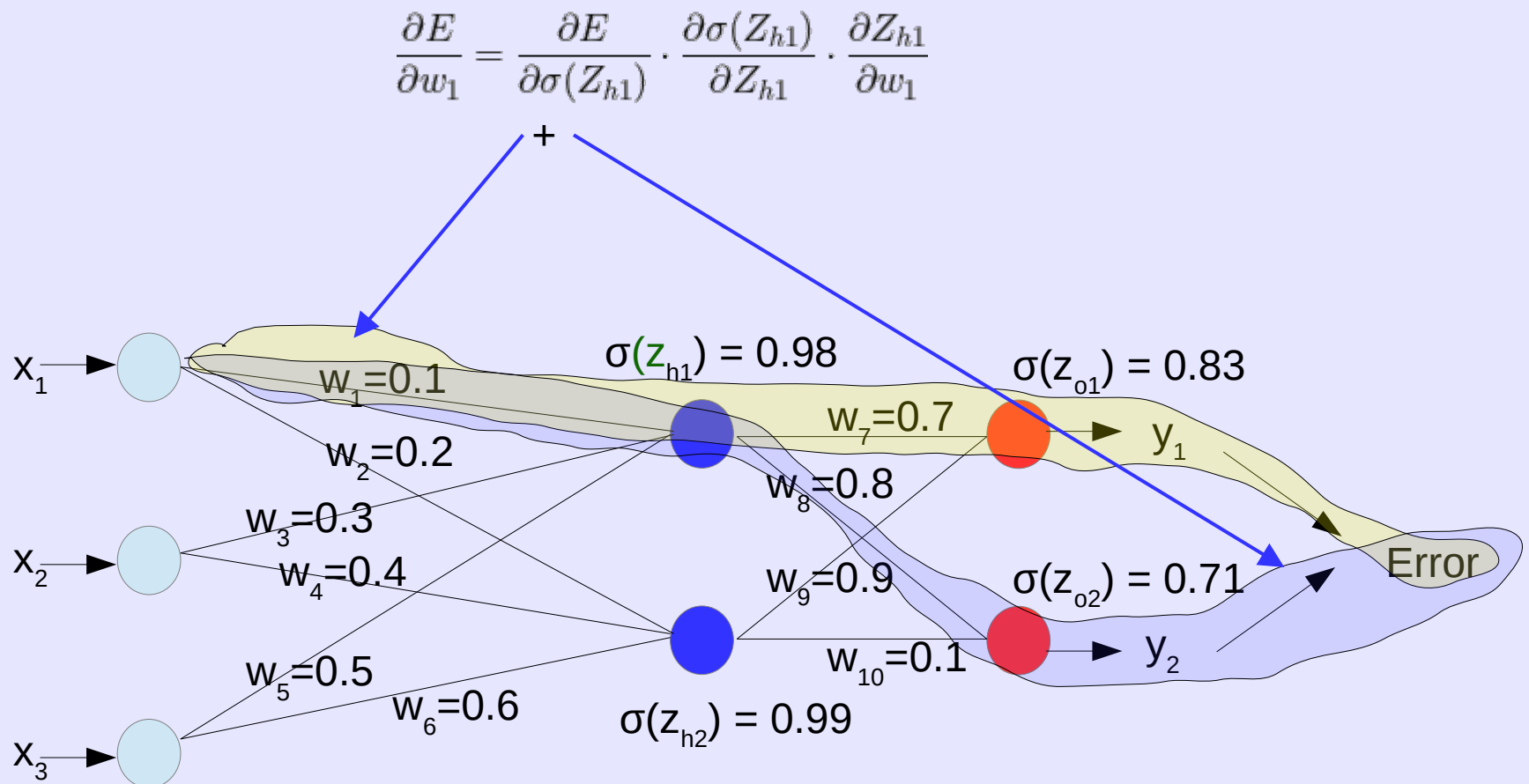
$$\begin{aligned} \frac{\partial E}{\partial w_7} &= \frac{\partial E}{\partial \sigma(Z_{o1})} \cdot \frac{\partial \sigma(Z_{o1})}{\partial Z_{o1}} \cdot \frac{\partial Z_{o1}}{\partial w_7} \\ &= [(\sigma(Z_{o1}) - y_1)] \cdot [\sigma(Z_{o1})(1 - \sigma(Z_{o1}))] \cdot [\sigma(Z_{h1})] \\ &= [(0.83 - 0.1)] \cdot [(0.83 - (1 - 0.83))] \cdot [0.98] \\ &= 0.10 \end{aligned}$$

$$\frac{\partial E}{\partial w_9} = \frac{\partial E}{\partial \sigma(Z_{o1})} \cdot \frac{\partial \sigma(Z_{o1})}{\partial Z_{o1}} \cdot \frac{\partial Z_{o1}}{\partial w_9} = 0.10 \quad (\text{Work it out!})$$



Backpropagation

Propagating errors through the hidden layers.



Backpropagation

- Result of backpropagation:
 - A gradient vector of weights used for updating the weights to get to the minimum gradient (Jacobian matrix).

$$\nabla E = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \frac{\partial E}{\partial w_3}, \dots, \frac{\partial E}{\partial w_n} \right]^T = [0.61, 0.87, 0.21, \dots, 0.99]^T$$

- Summary of backpropagation:
 - Feed forward computation
 - Backpropagation to the output layer
 - Backpropagation to the hidden layer(s)
 - Weight updates.

$$w_{new} = w_{old} - \eta \nabla E$$

Gradient descent



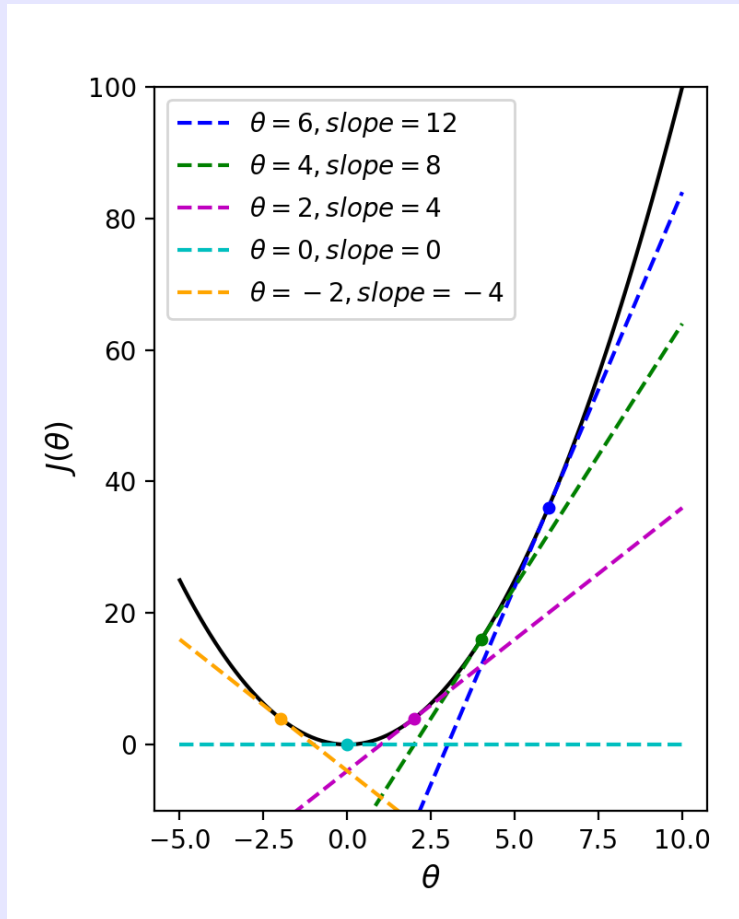
Gradient descent

- Relates two concepts:
 - Error (or cost) function minimization.
 - Minimizes the error function with respect to the weights. (Recall: $\nabla E = \frac{\partial E}{\partial w_i}$ for all $i \in \{1, 2, \dots, n\}$)
- A gradient descent algorithm iteratively goes through the training dataset, modifying weights during each pass (epoch) to minimize the cost.

$$w_{new} = w_{old} - \eta \nabla E$$

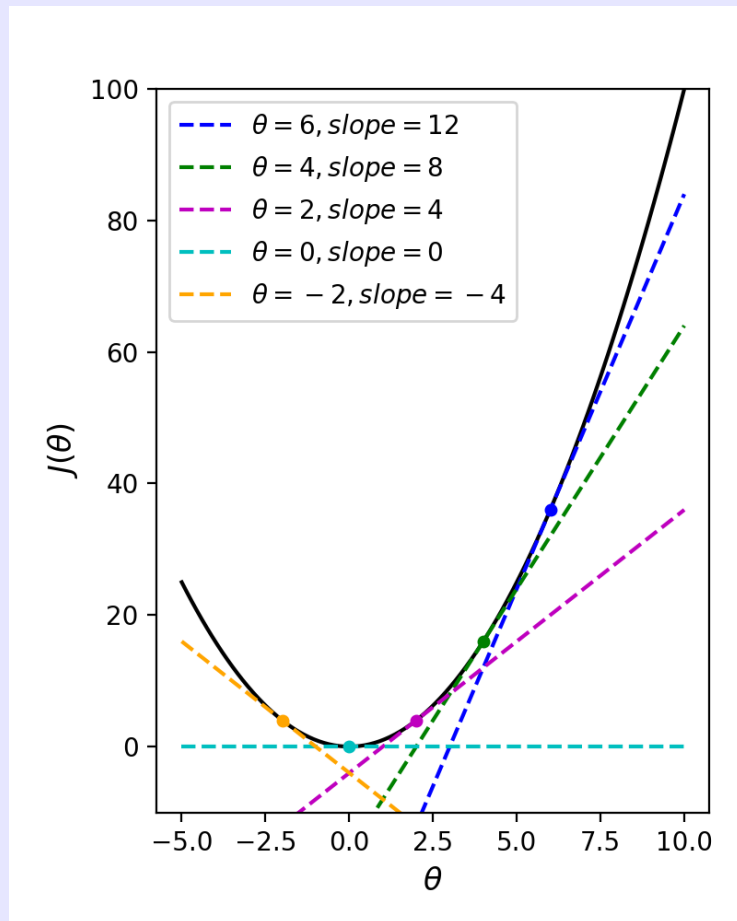
Gradient descent

Intuitive feel of the gradient.

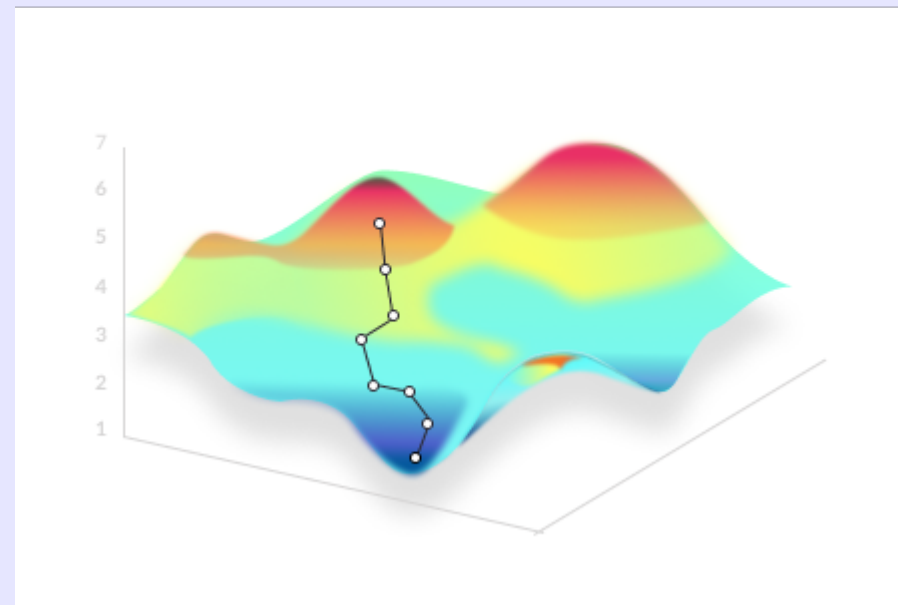


Gradient descent

Intuitive feel of the gradient.



Gradient descent in neural networks



Gradient descent

- Types of gradient descent algorithms:
 - Batch gradient descent
 - Calculate error for each observation, and at end of the training data calculate average error and update w .
 - Stochastic gradient descent
 - Calculate error for each observation, and update w .
 - Mini-batch gradient descent
 - Split data into small batches, calculate error for each observation in a batch, and at end of the batch calculate average error and update w .
 - Preferred method.

Gradient descent

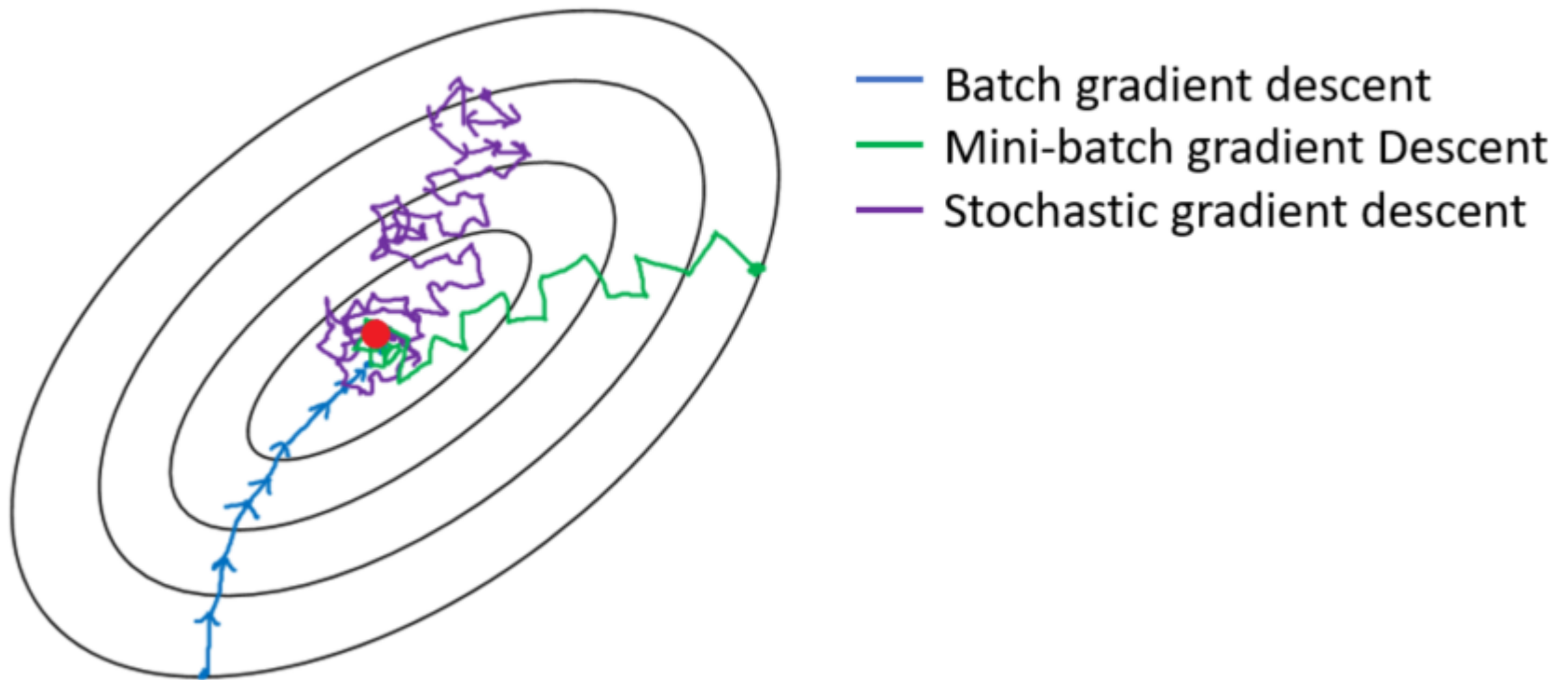
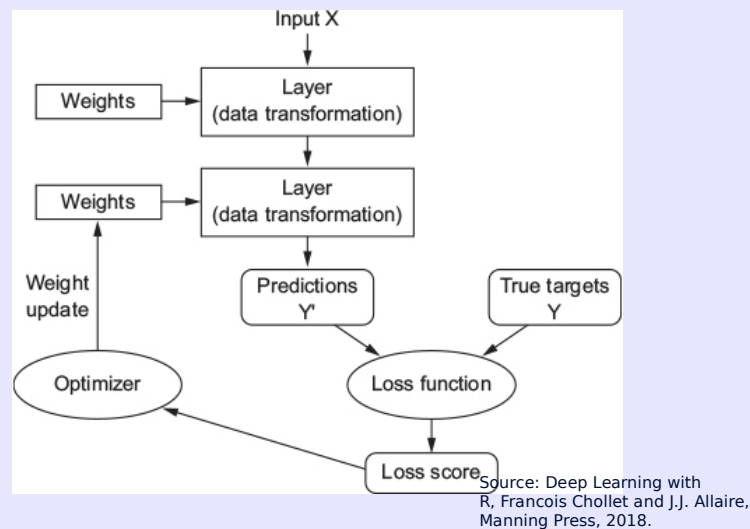


Image source: <https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3>

Summary

- The learning process in neural networks



Code

- `iris-nn.r`
- The Fashion MNIST dataset (`neural-main.r`)

