

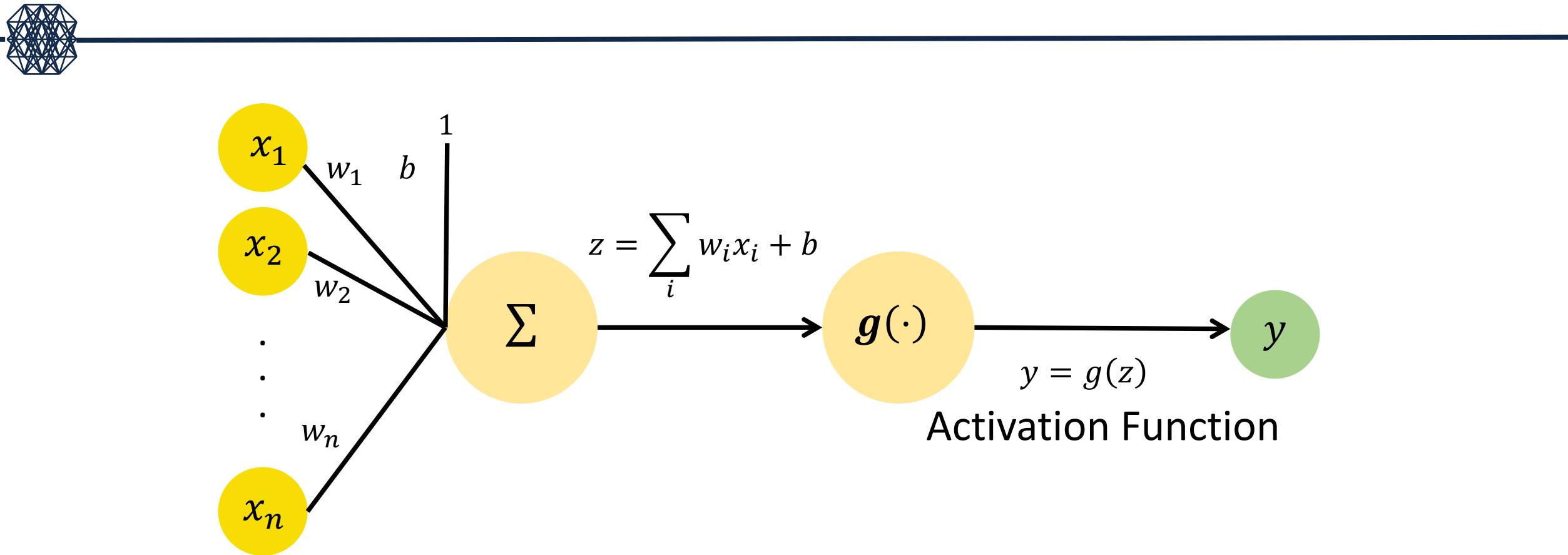


# Deep Learning for Healthcare

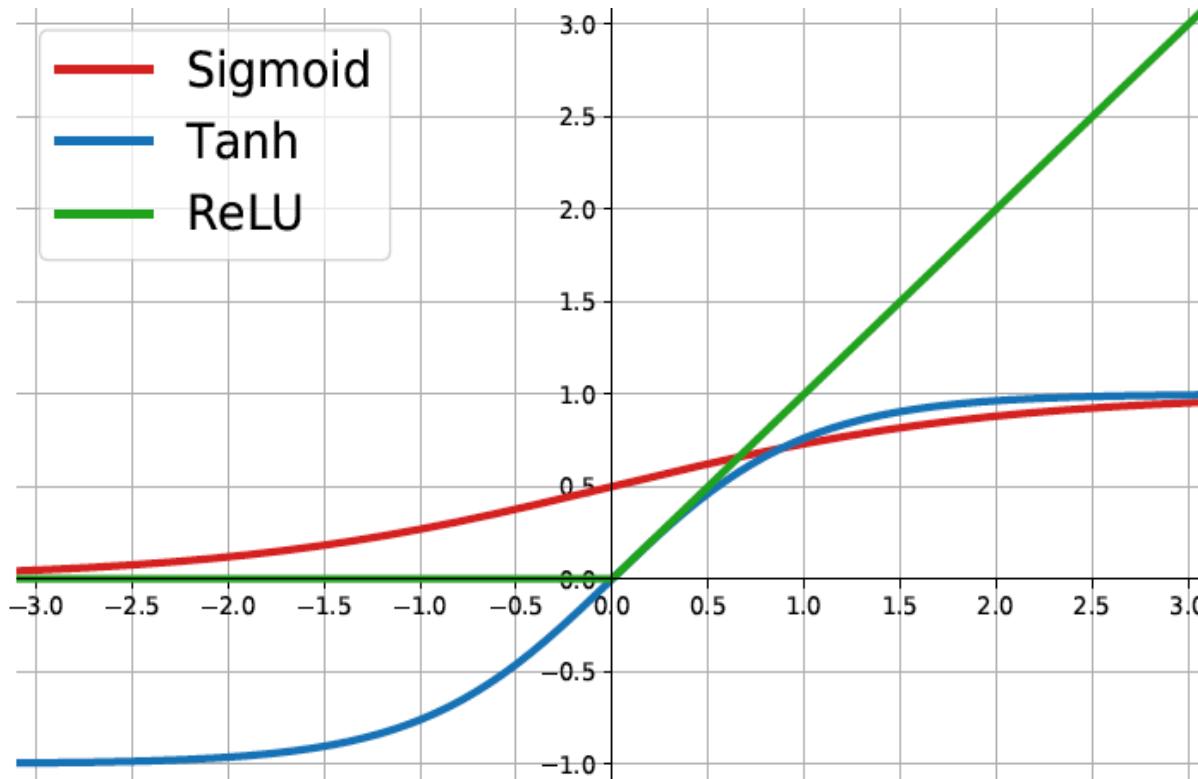
Deep Neural  
Networks

*Prof. Jimeng Sun*

# A Single Neuron

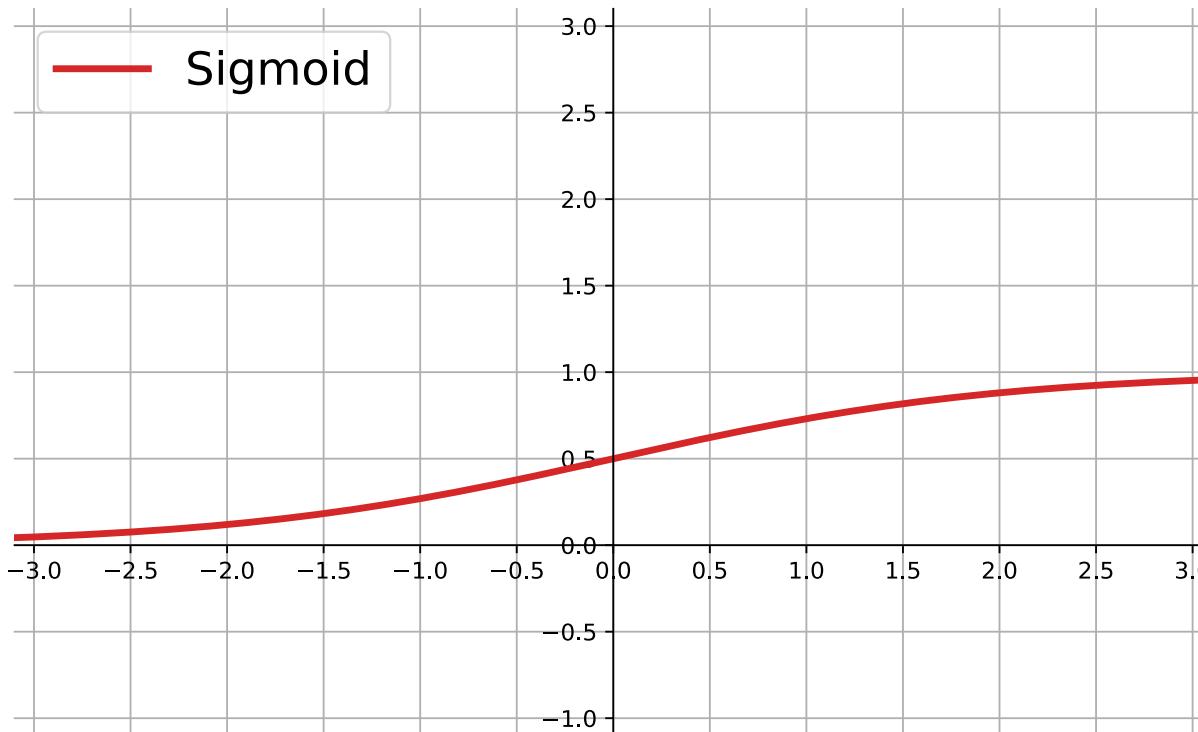
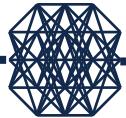


# Activation Functions $g(x)$



- Activation function describes the nonlinear transformation
- Specified by users
- Not learned from data
- Popular ones include
  - Sigmoid
  - Tanh
  - ReLU

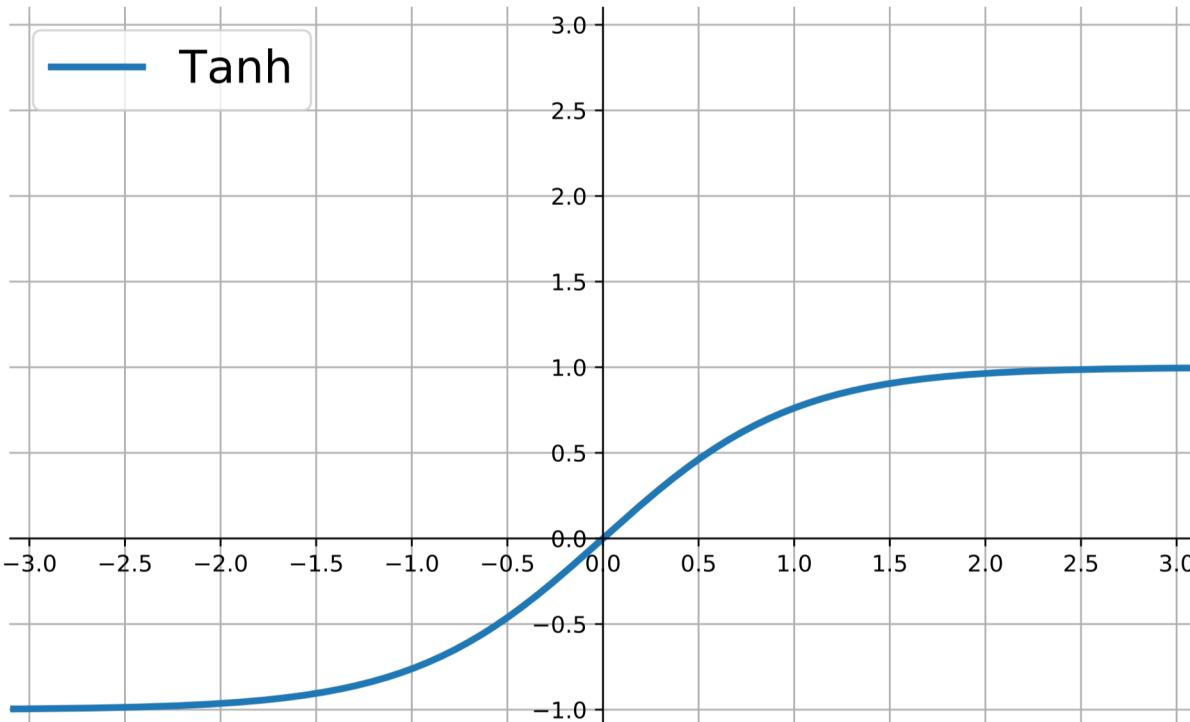
# Sigmoid Function



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

- output **real values** bounded in  $(0,1)$
- has a natural interpretation of probability of an event
- **vanishing gradient** problem

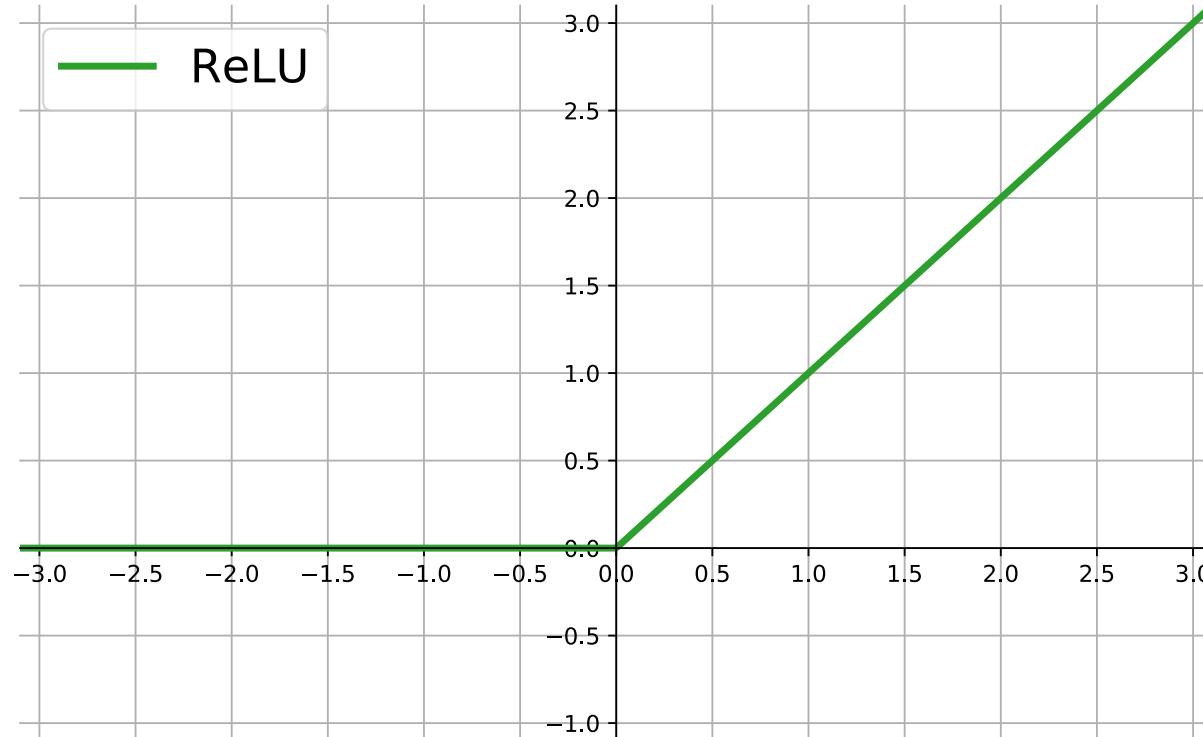
# Tanh Function



$$g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{2}{1 + e^{(-2x)}} - 1$$

- outputs are **zero-centered** and **bounded** in  $(-1,1)$
- scaled and shifted Sigmoid
- **stronger** gradient but still has **vanishing gradient** problem

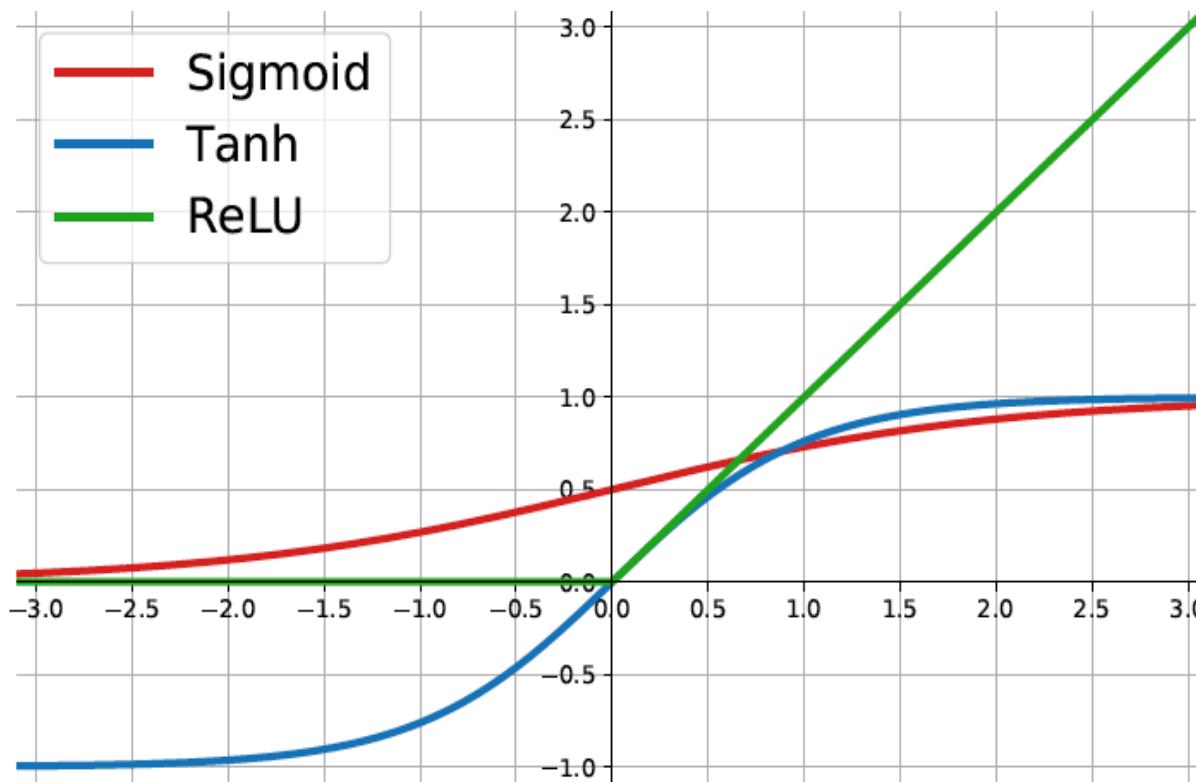
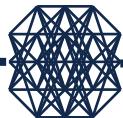
# Rectified Linear Unit (ReLU)



$$g(x) = \max(0, x)$$

- outputs are in  $(0, \infty)$ , thus not bounded
- half rectified: activation threshold at 0
- No vanishing gradient problem

# Activation Functions: Summary



Sigmoid

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

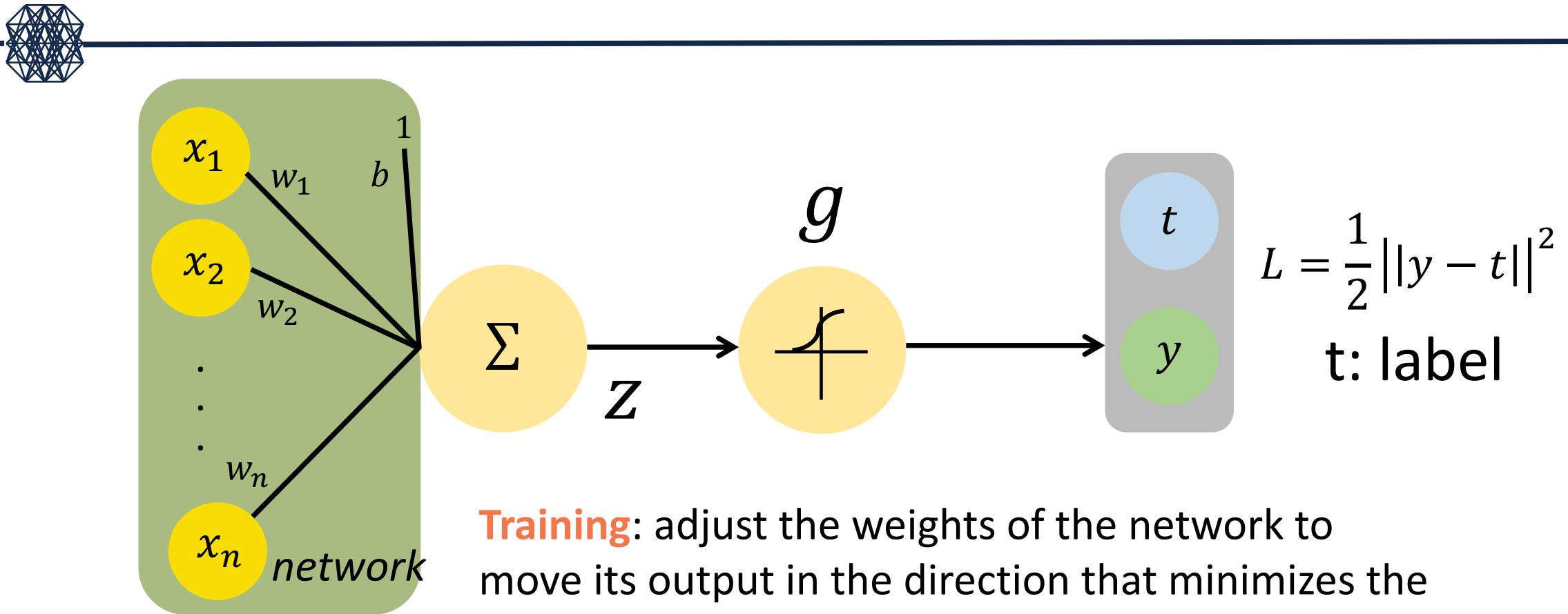
Tanh

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

ReLU

$$g(x) = \max(0, x)$$

# Train A Single Neuron



# Gradient Descent Method



$D$	
X	Y



$P(D|\theta)$

Repeat

$$\theta_{new} = \theta_{old} - g$$

$$g = \frac{dp(D|\theta)}{d\theta}$$

# Gradient Descent Method For Linear Regression



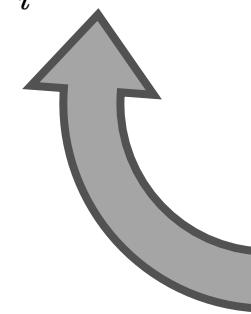
$D$	
X	Y



$$l(\beta) = c - \frac{1}{2\sigma^2} \sum_i^N (y^{(i)} - \beta^T \mathbf{x}^{(i)})^2$$

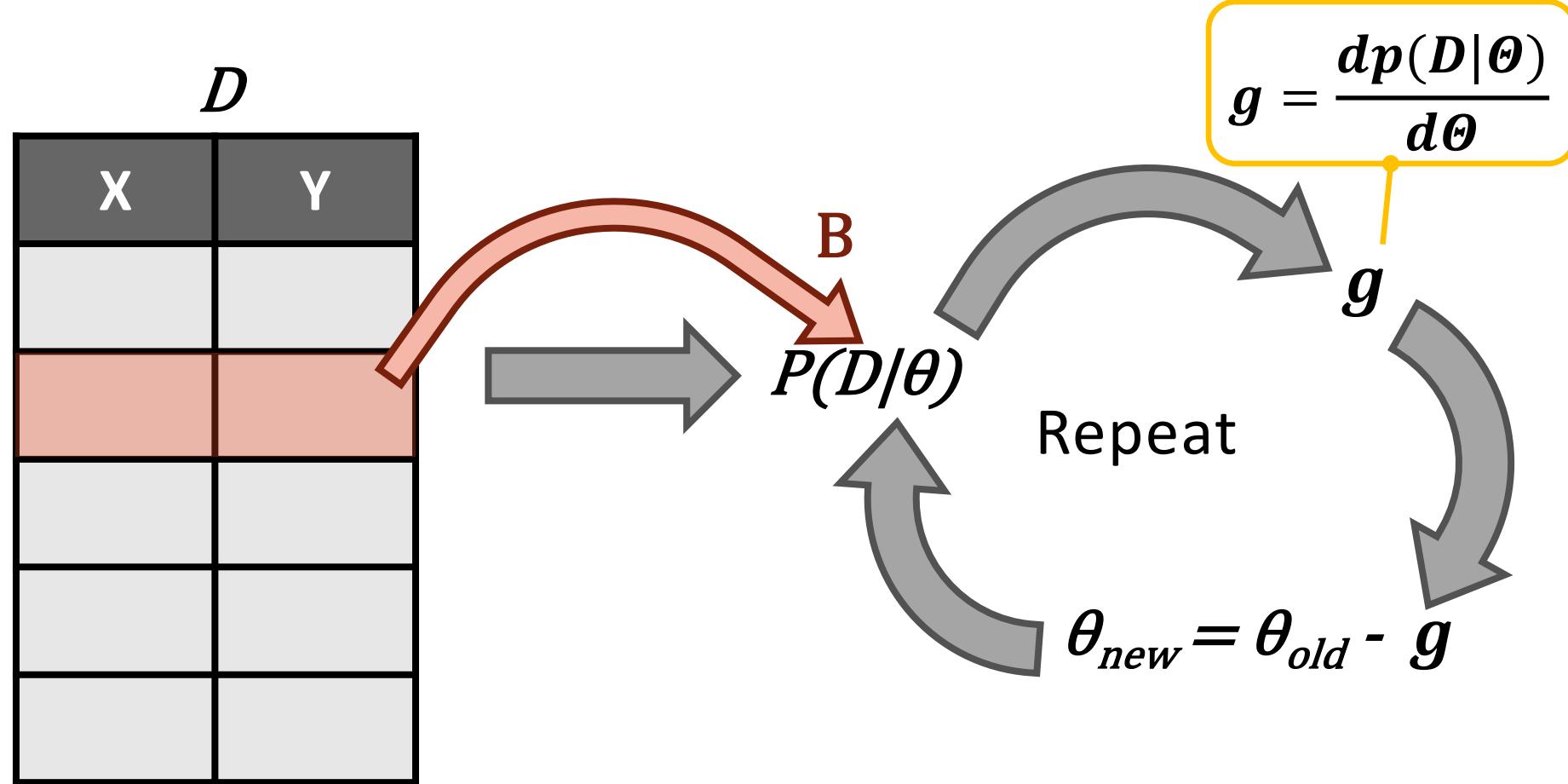
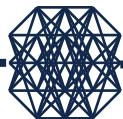
$$\frac{\partial l(\beta)}{\partial \beta_j} = -2 \sum_i (y^{(i)} - \beta^T \mathbf{x}^{(i)}) x_j^{(i)}$$

Repeat

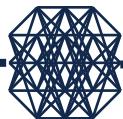


$$\beta_j = \beta_j - \eta 2 \sum_i (y^{(i)} - \beta^T \mathbf{x}^{(i)}) x_j^{(i)}$$

# Stochastic Gradient Descent (SGD) Method



# SGD For Linear Regression



$D$

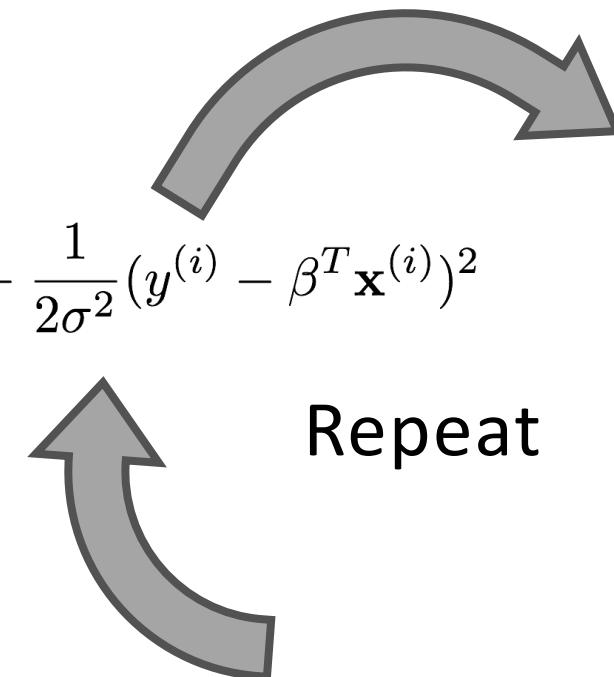
X	Y



$$l(\beta) = \text{const} - \frac{1}{2\sigma^2} (y^{(i)} - \beta^T \mathbf{x}^{(i)})^2$$

$$\frac{\partial l(\beta)}{\partial \beta_j} = -2(y^{(i)} - \beta^T \mathbf{x}^{(i)})x_j^{(i)}$$

Repeat



$$\beta_j = \beta_j - 2(y^{(i)} - \beta^T \mathbf{x}^{(i)})x_j^{(i)}$$

# SGD for neural networks



SGD (Training data  $(x, t)$ , learning rate  $\eta$ )

Initialize each  $w_i$  and  $b$  to some small random value

Until convergence

- Pick training example  $(x, t)$

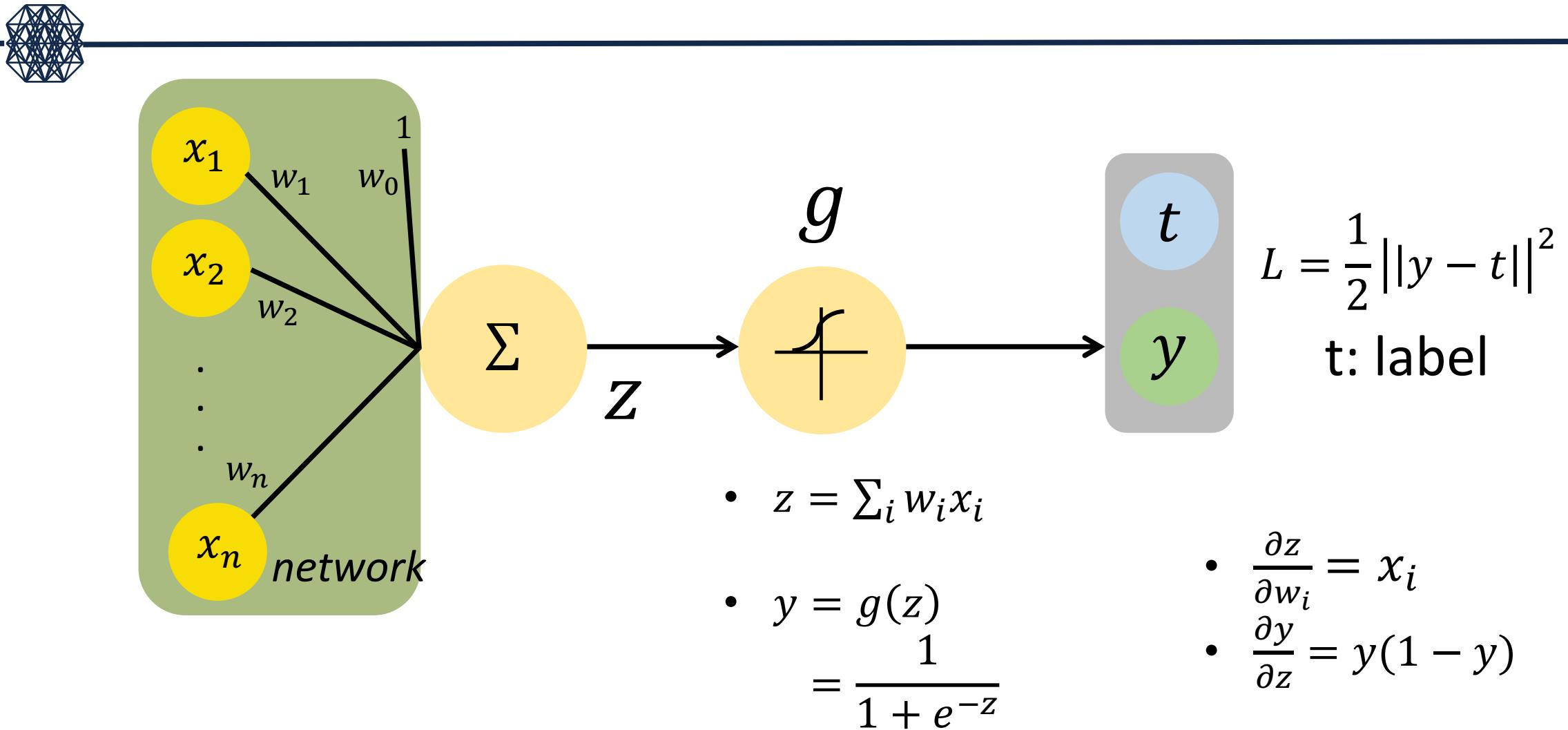
- compute gradient  $\nabla w$  and  $\frac{\partial L}{\partial b}$

- update weight vector  $w \leftarrow w - \eta \nabla w$

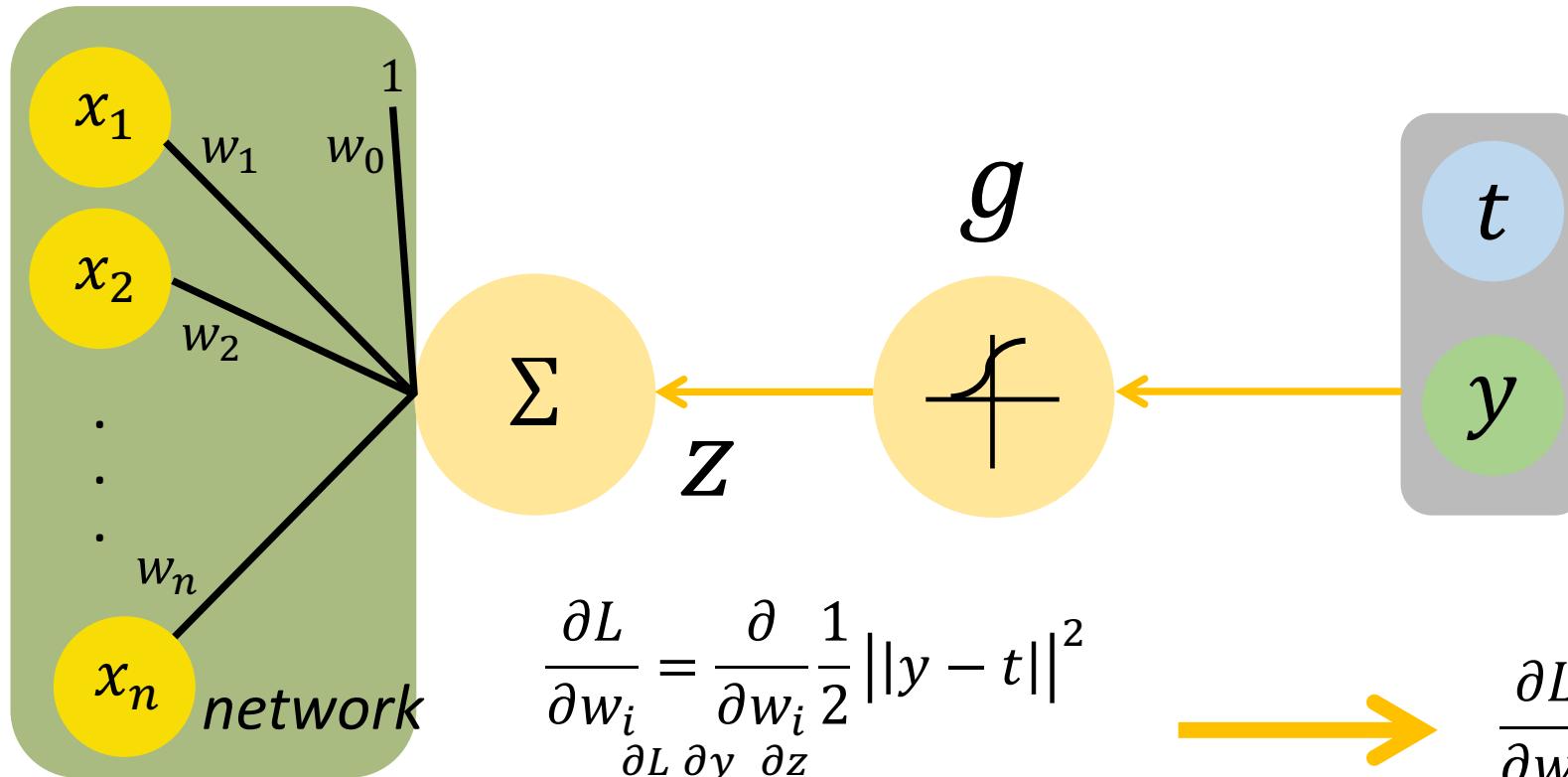
- update bias  $b \leftarrow b - \eta \frac{\partial L}{\partial b}$

$$\nabla w = \left[ \frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \dots, \frac{\partial L}{\partial w_n} \right]$$

# Forward Computation For A Neuron



# Backward Computation For A Neuron

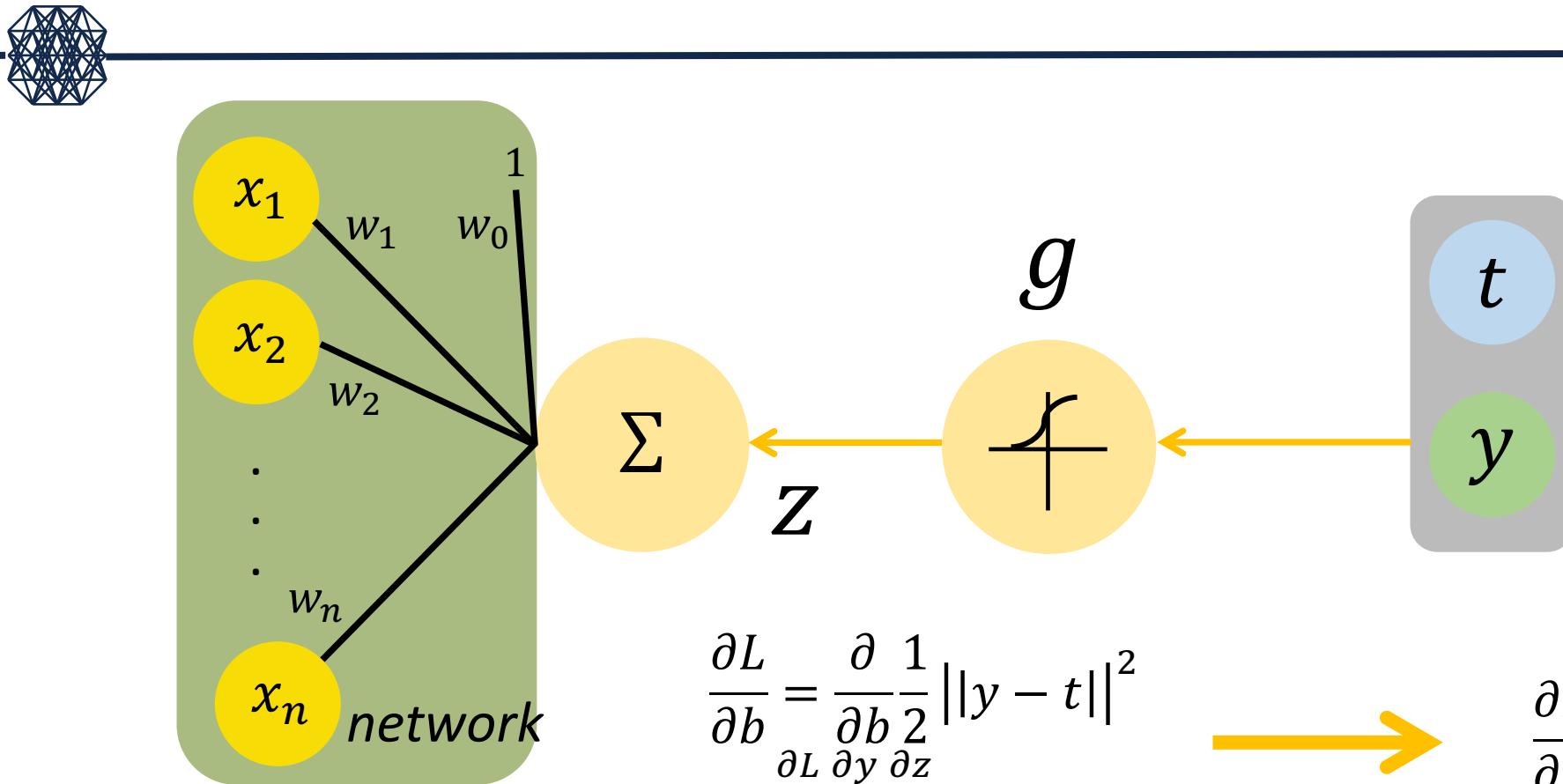


$$\begin{aligned}\frac{\partial L}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \|y - t\|^2 \\ &= \frac{\partial L}{\partial y} \frac{\partial y}{\partial z} \frac{\partial z}{\partial w_i} \\ &= (y - t)y(1 - y) \frac{\partial z}{\partial w_i}\end{aligned}$$



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

# Backward Computation For A Neuron



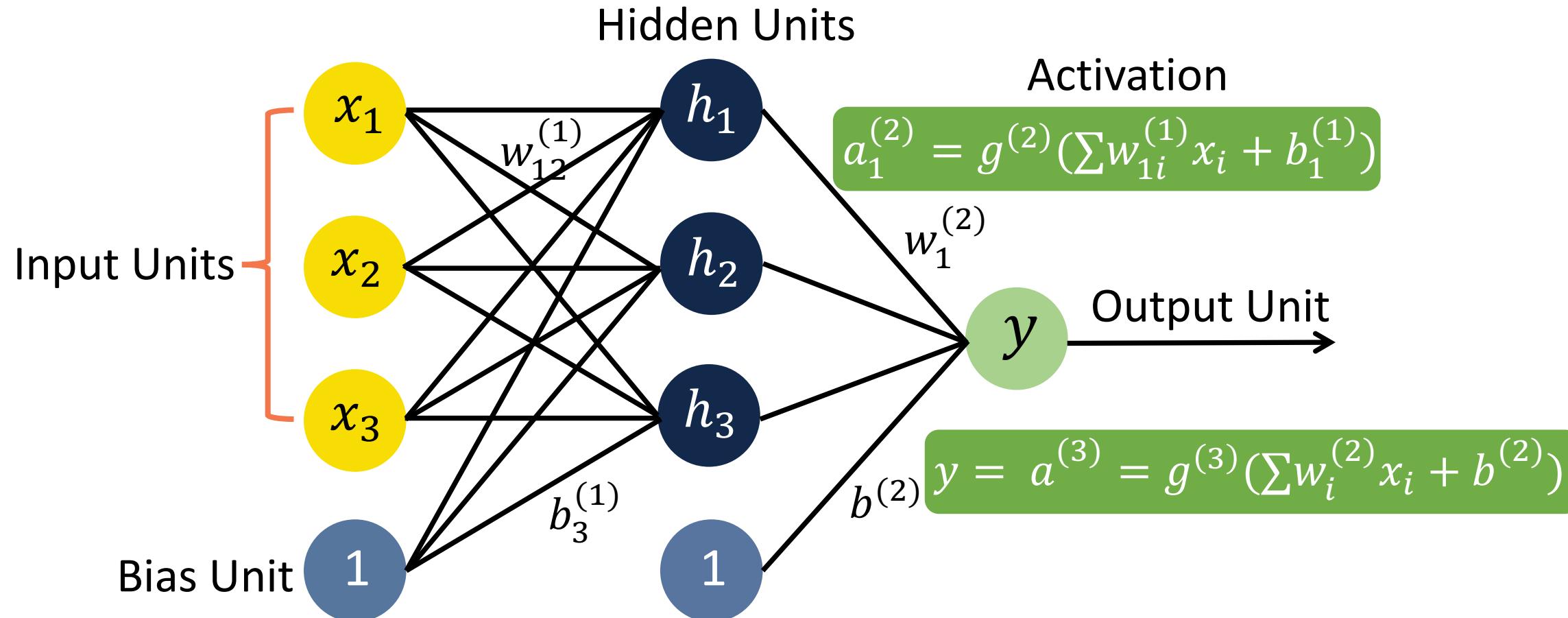
$$\begin{aligned}\frac{\partial L}{\partial b} &= \frac{\partial}{\partial b} \frac{1}{2} \|y - t\|^2 \\ &= \frac{\partial L}{\partial y} \frac{\partial y}{\partial z} \frac{\partial z}{\partial b} \\ &= (y - t)y(1 - y) \frac{\partial z}{\partial b}\end{aligned}$$



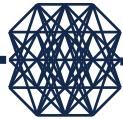
$$\frac{\partial L}{\partial b} = (y - t)y(1 - y)$$

# Multilayer Neural Networks

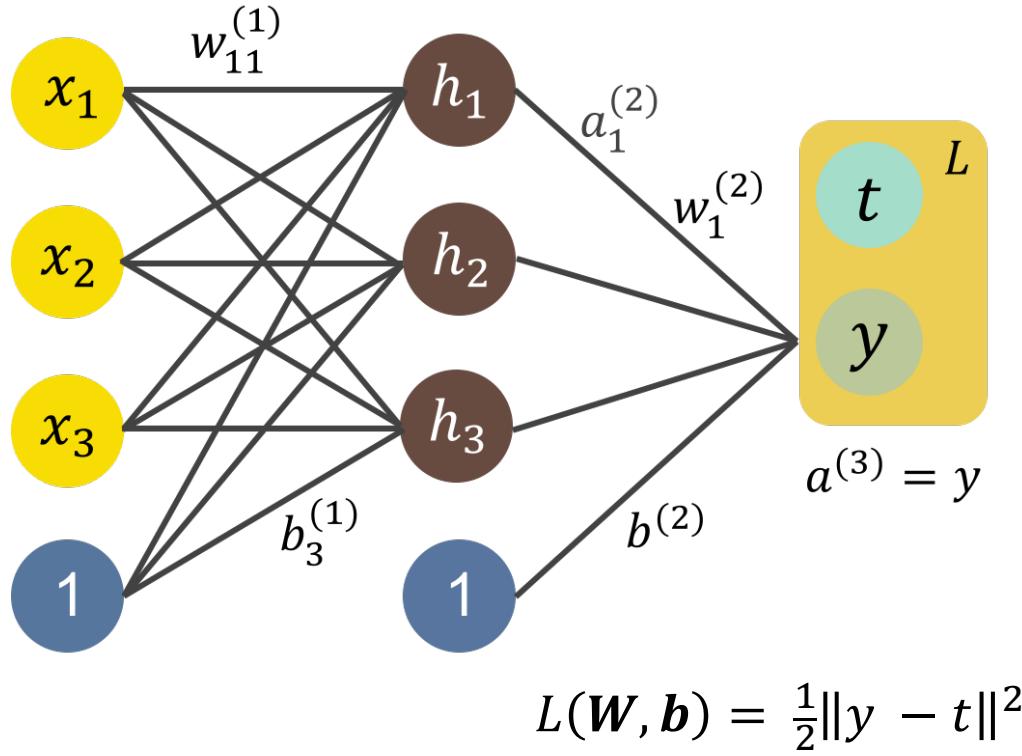
# Multilayer Neural Network



# Train A Multilayer Neural Network



Given training sample  $(x, t)$



SGD (Training data  $(x, t)$ , learning rate  $\eta$ )

Initialize each  $w_{ji}^{(l)}$  and  $b_j^{(l)}$  to small random value  
Until convergence

**DO**

For each  $(x, t)$  in training data, **DO**

- compute  $\frac{\partial L(\mathbf{W}, \mathbf{b})}{\partial w_{ji}^{(l)}}$  and  $\frac{\partial L(\mathbf{W}, \mathbf{b})}{\partial b_j^{(l)}}$

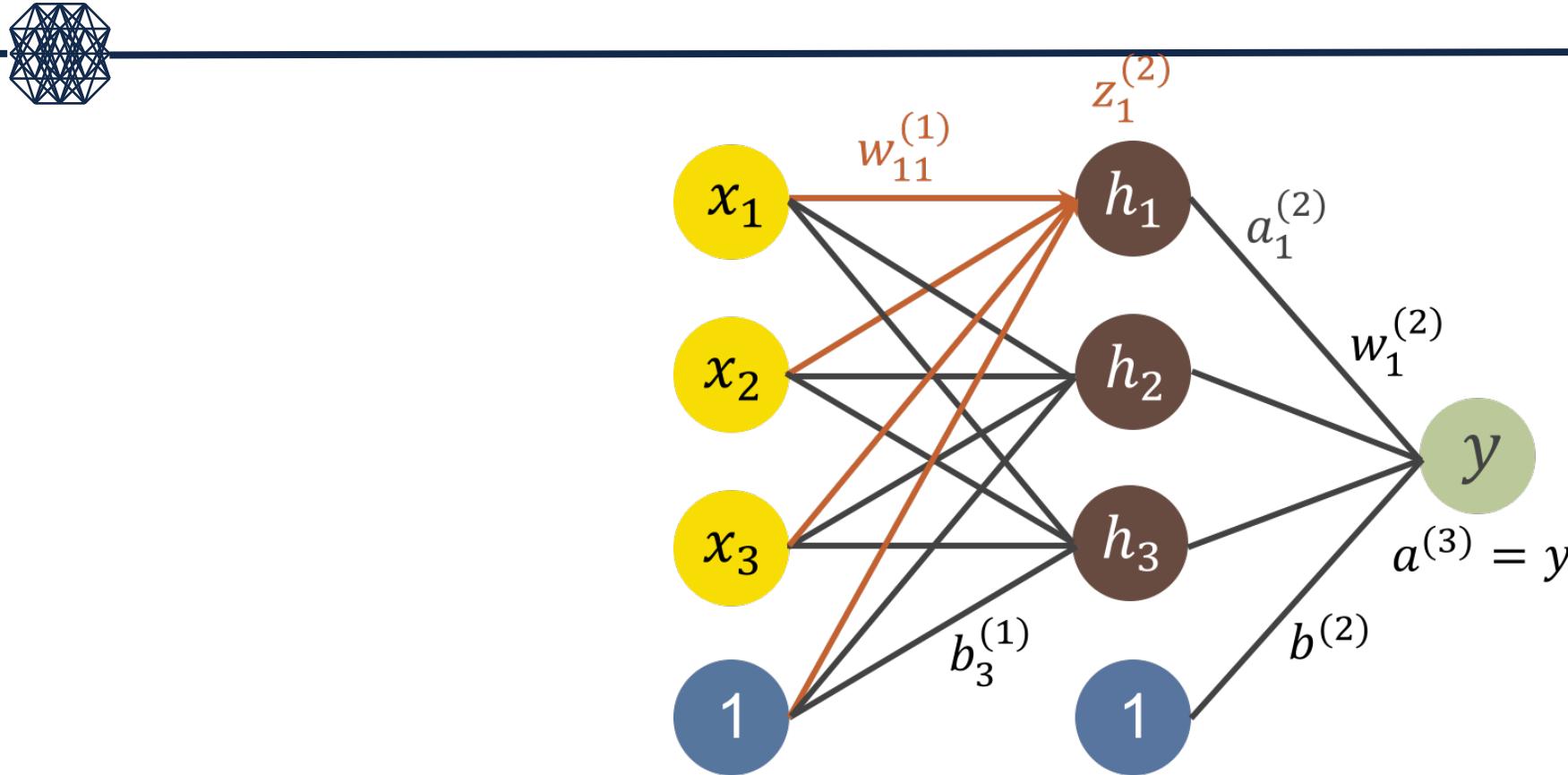
- For each  $w_{ji}^{(l)}$  and  $b_j^{(l)}$ :

$$w_{ji}^{(l)} = w_{ji}^{(l)} - \eta \frac{\partial L(\mathbf{W}, \mathbf{b})}{\partial w_{ji}^{(l)}}$$

$$b_j^{(l)} = b_j^{(l)} - \eta \frac{\partial L(\mathbf{W}, \mathbf{b})}{\partial b_j^{(l)}}$$

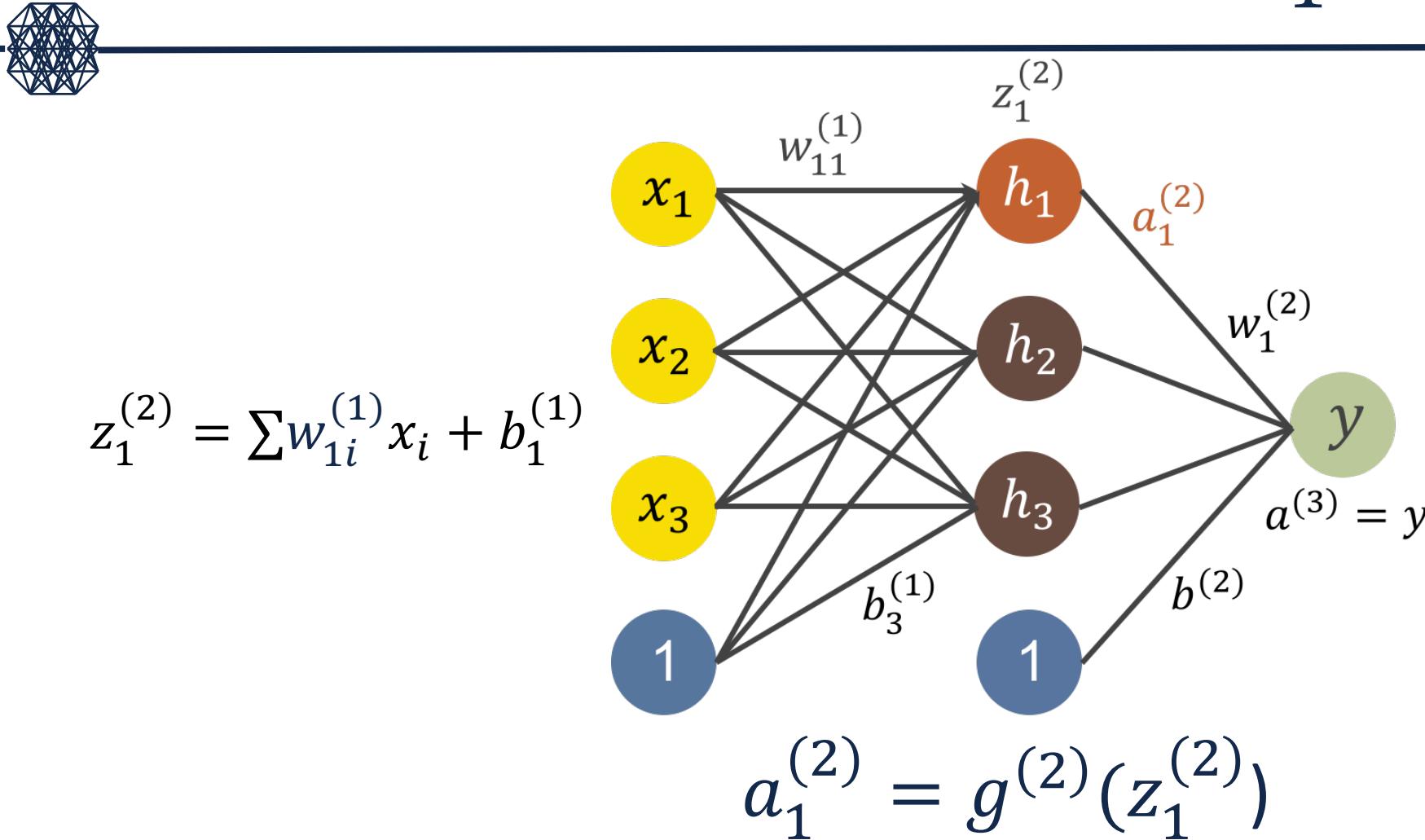
# Forward Computation:

$z_1^{(2)}$

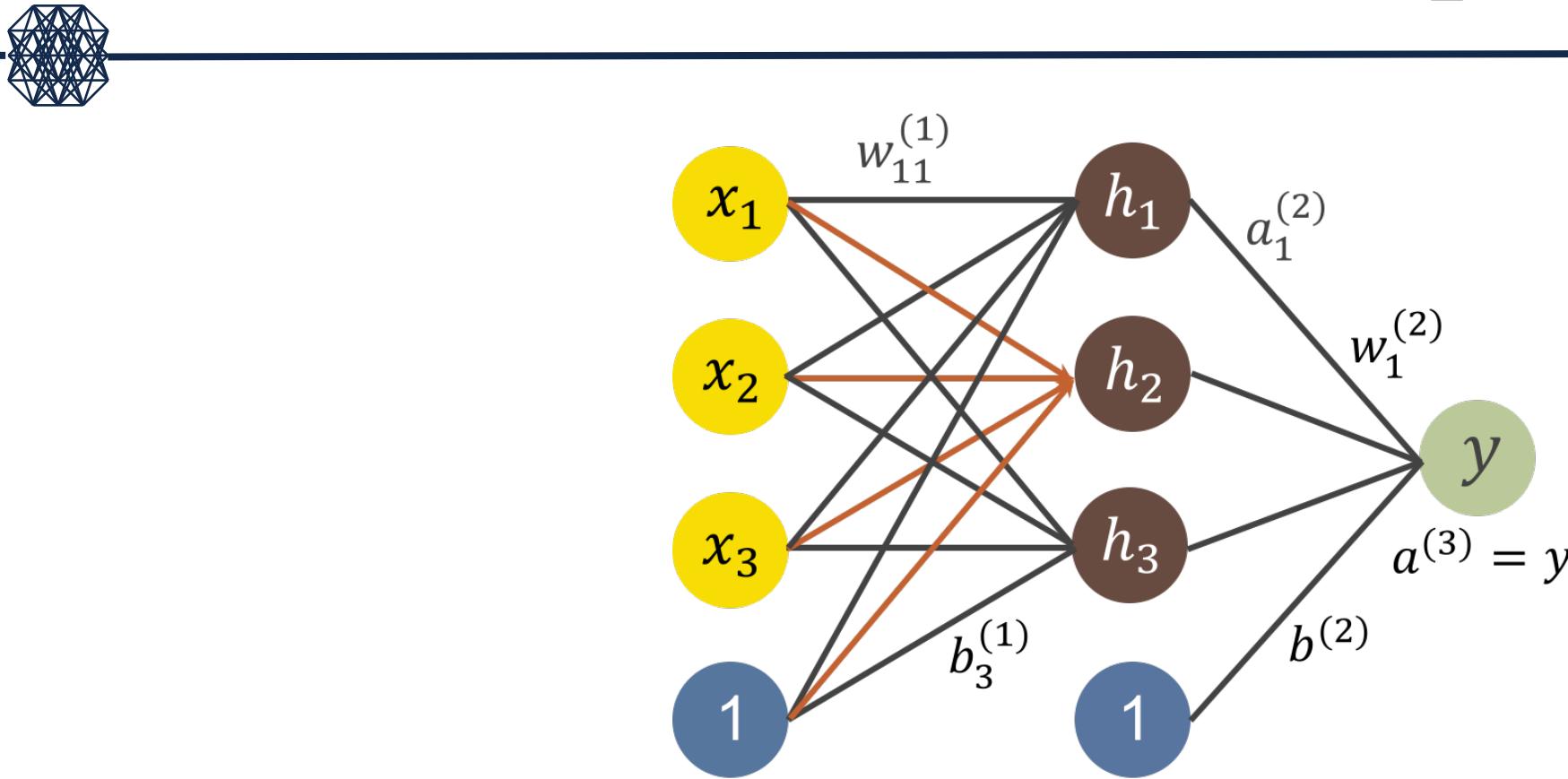


$$z_1^{(2)} = \sum w_{1i}^{(1)} x_i + b_1^{(1)}$$

# Forward Computation: $a_1^{(2)}$

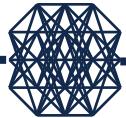


# Forward Computation: $z_2^{(2)}$

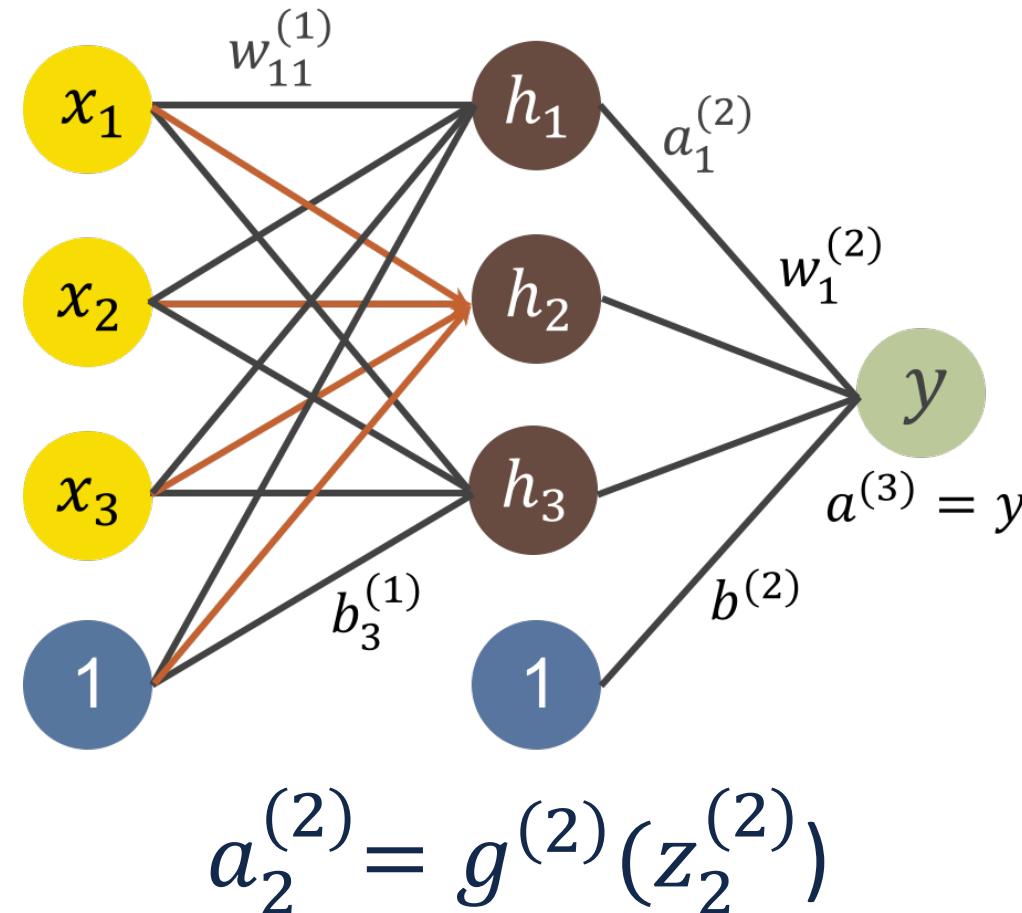


$$z_2^{(2)} = \sum w_{2i}^{(1)} x_i + b_2^{(1)}$$

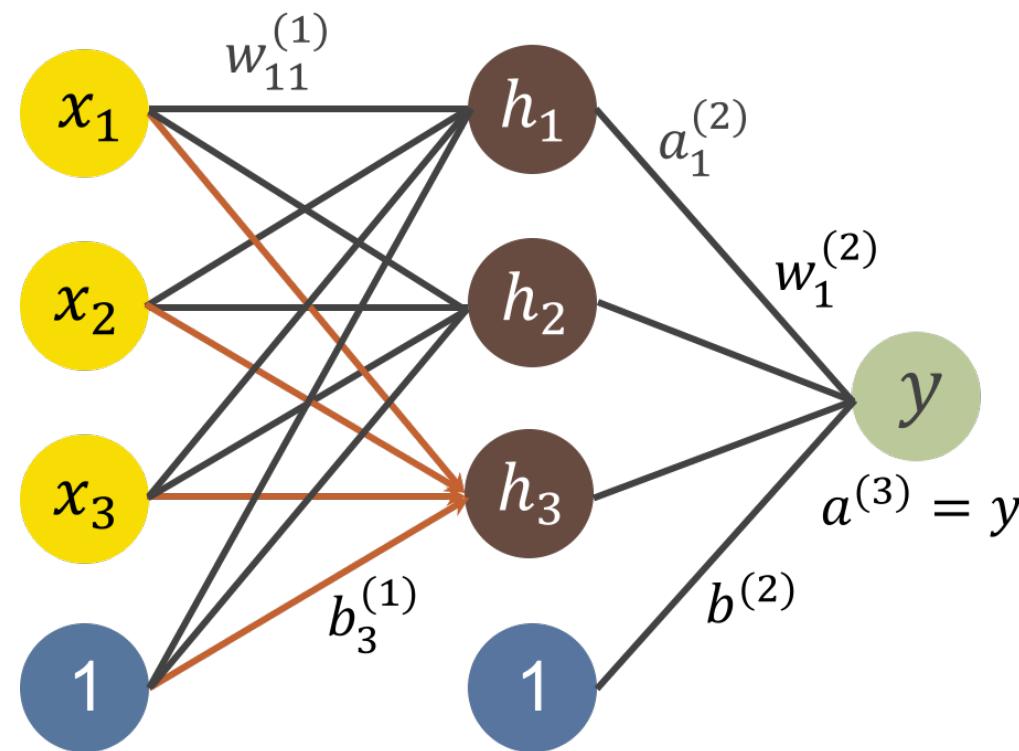
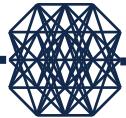
# Forward Computation: $a_2^{(2)}$



$$z_2^{(2)} = \sum w_{2i}^{(1)} x_i + b_2^{(1)}$$

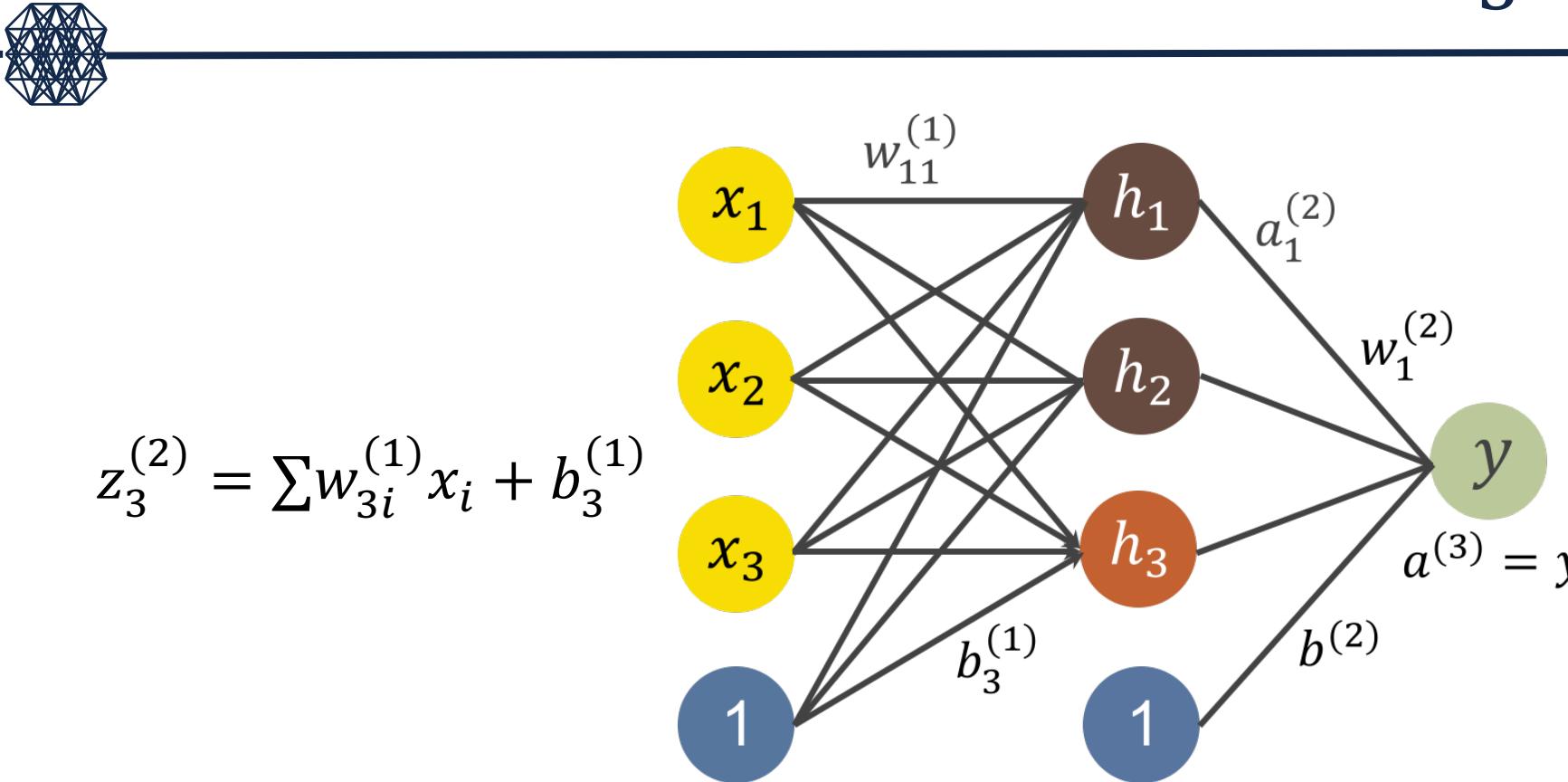


# Forward Computation: $z_3^{(2)}$



$$z_3^{(2)} = \sum w_{3i}^{(1)} x_i + b_3^{(1)}$$

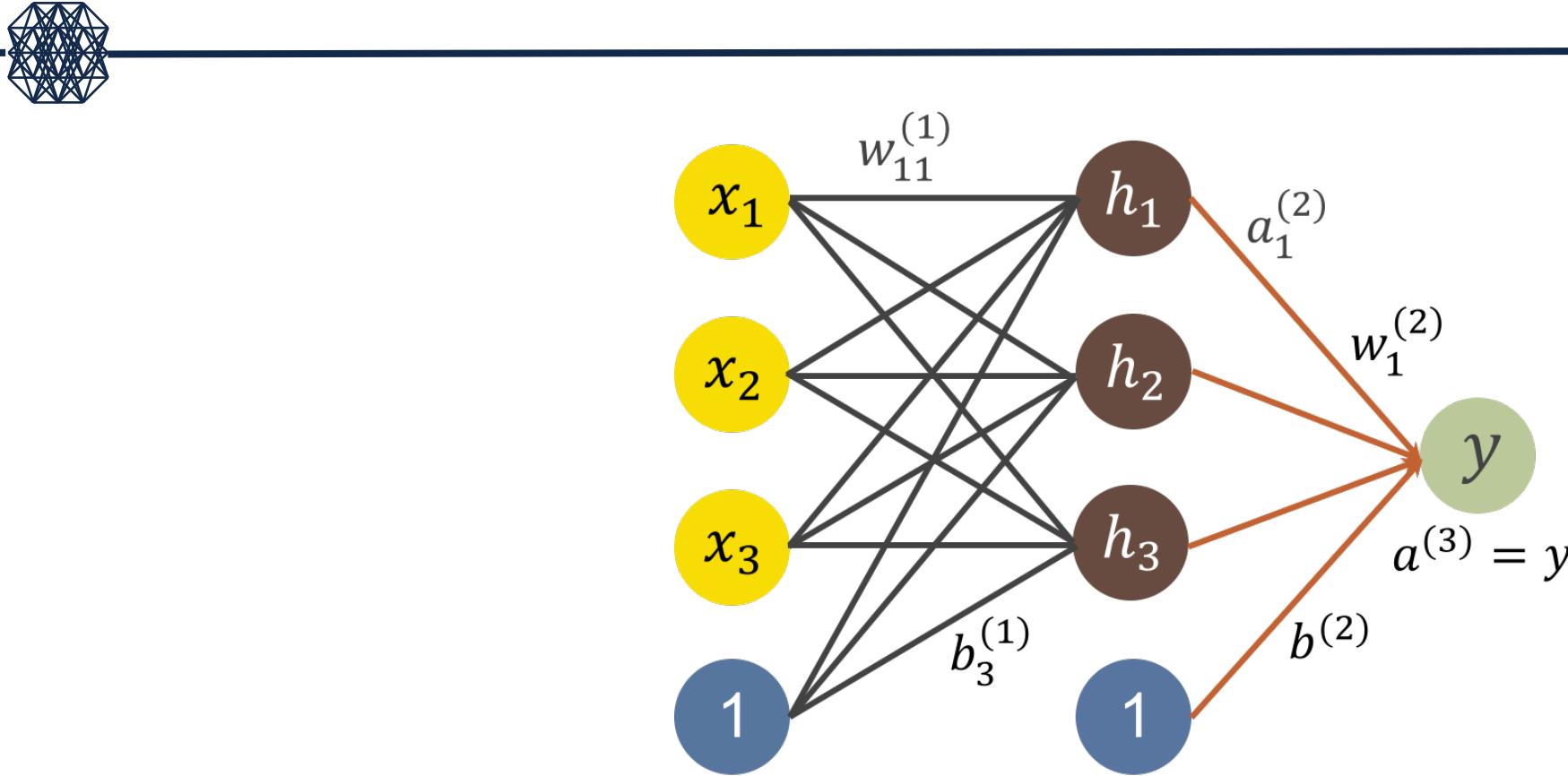
# Forward Computation: $a_3^{(2)}$



$$z_3^{(2)} = \sum w_{3i}^{(1)} x_i + b_3^{(1)}$$

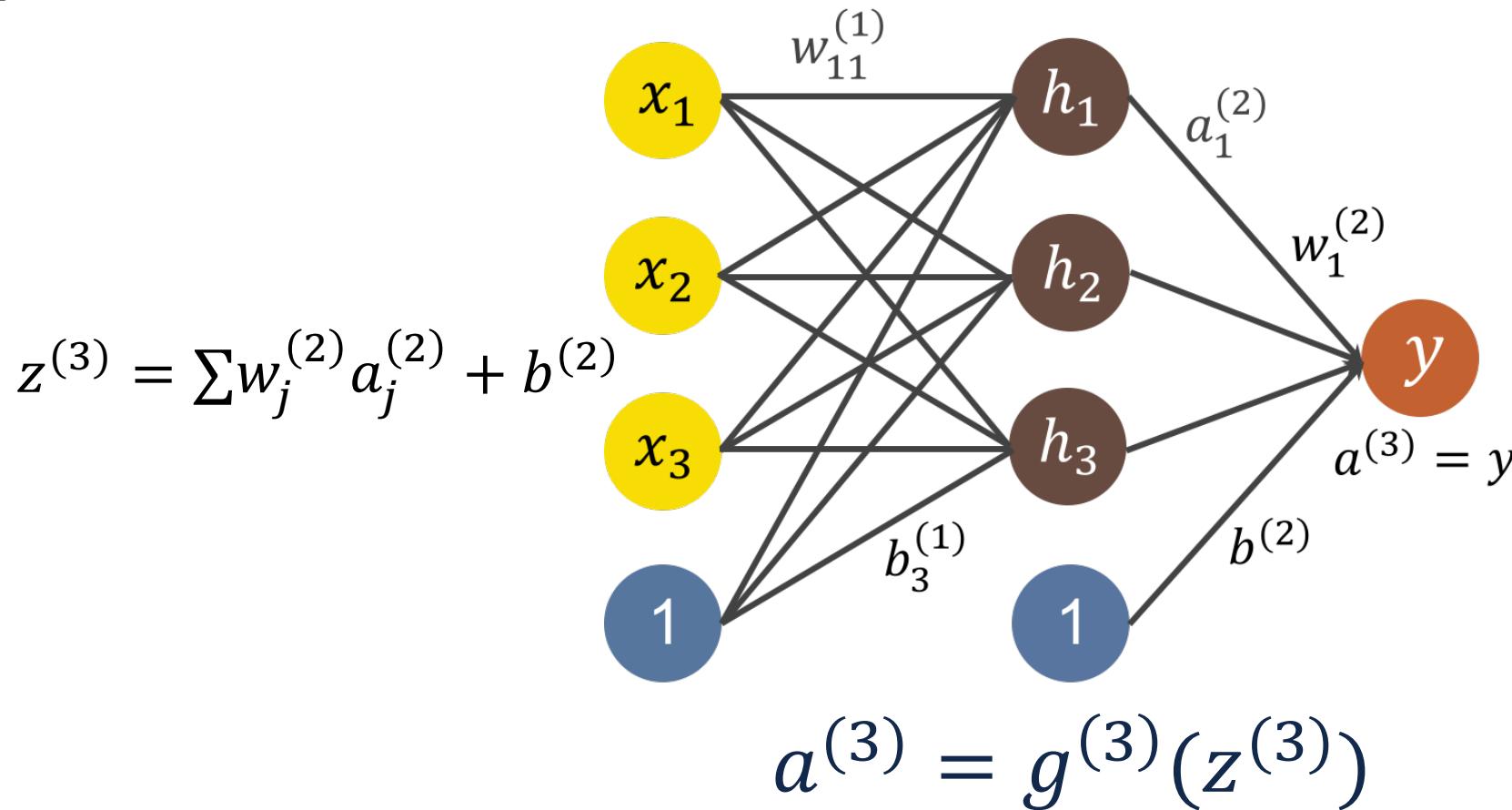
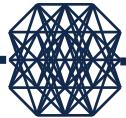
$$a_3^{(2)} = g^{(2)}(z_3^{(2)})$$

# Forward Computation: $z^{(3)}$

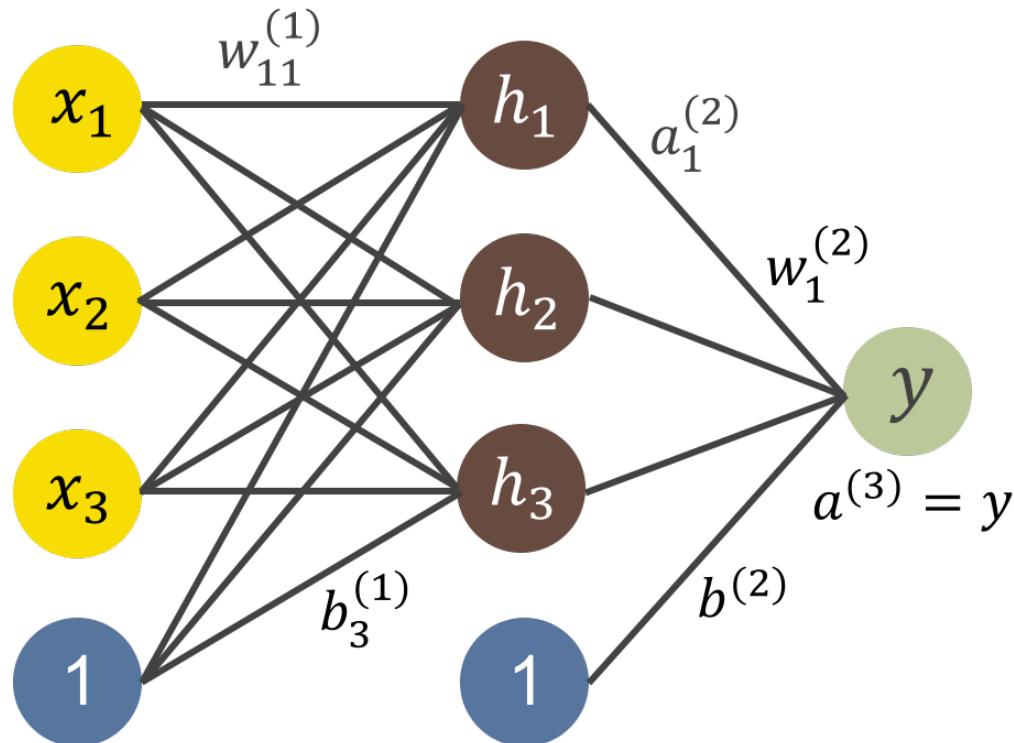


$$z^{(3)} = \sum w_j^{(2)} a_j^{(2)} + b^{(2)}$$

# Forward Computation: $a^{(3)}$



# Forward Computation



$$z_1^{(2)} = \sum w_{1i}^{(1)} x_i + b_1^{(1)}$$

$$z_2^{(2)} = \sum w_{2i}^{(1)} x_i + b_2^{(1)}$$

$$z_3^{(2)} = \sum w_{3i}^{(1)} x_i + b_3^{(1)}$$

$$a_1^{(2)} = g^{(2)}(z_1^{(2)})$$

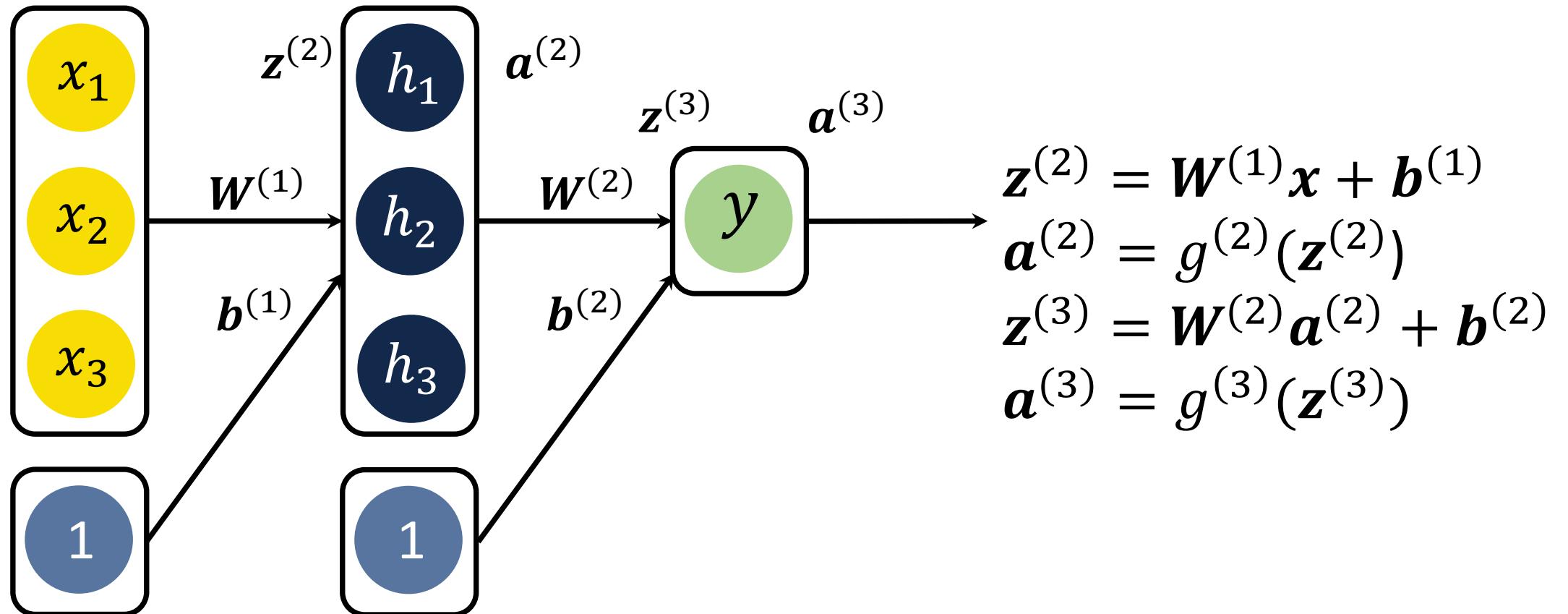
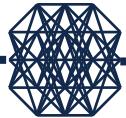
$$a_2^{(2)} = g^{(2)}(z_2^{(2)})$$

$$a_3^{(2)} = g^{(2)}(z_3^{(2)})$$

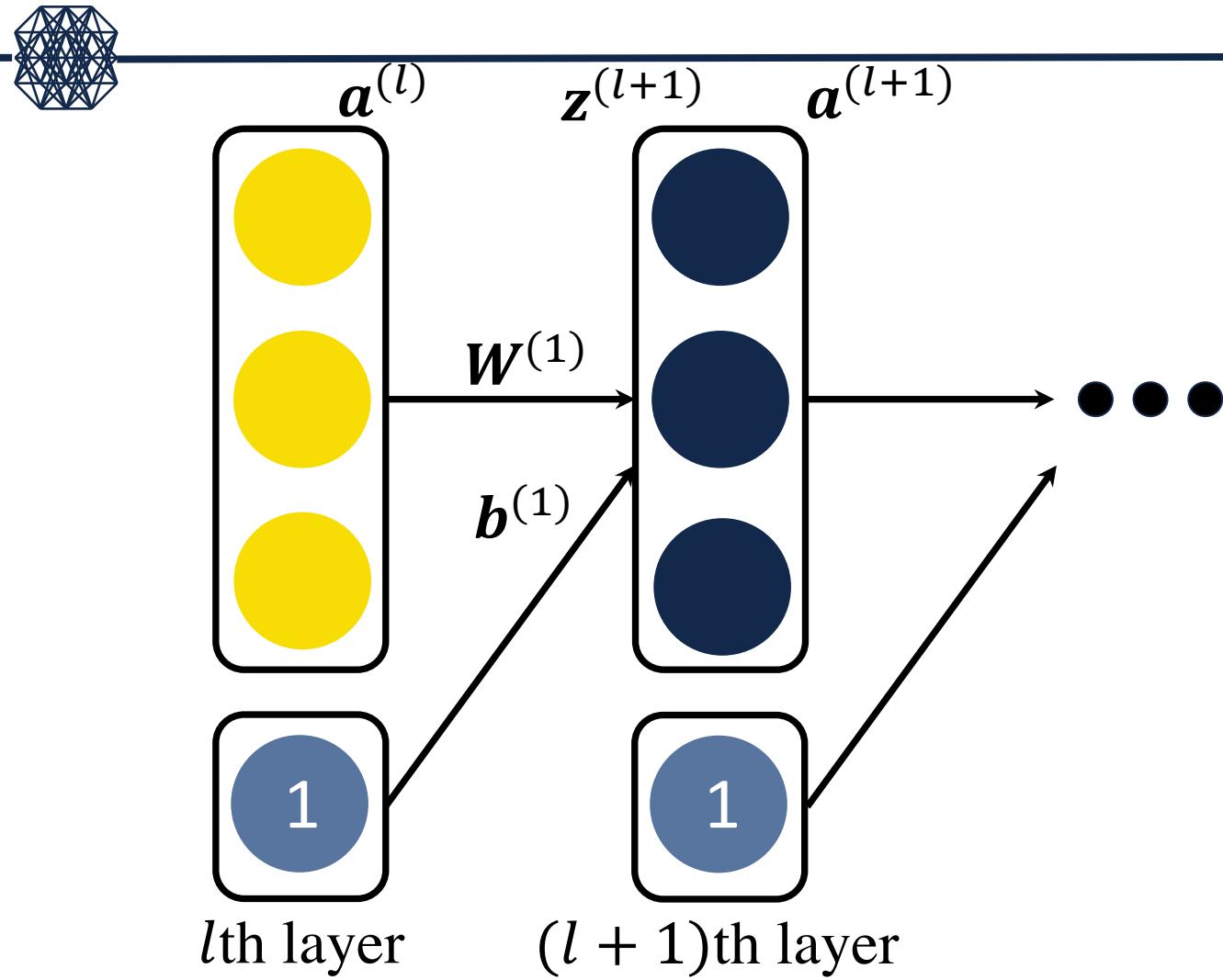
$$z^{(3)} = \sum w_j^{(2)} a_j^{(2)} + b^{(2)}$$

$$a^{(3)} = g^{(3)}(z^{(3)})$$

# Forward Computation: Vector Form

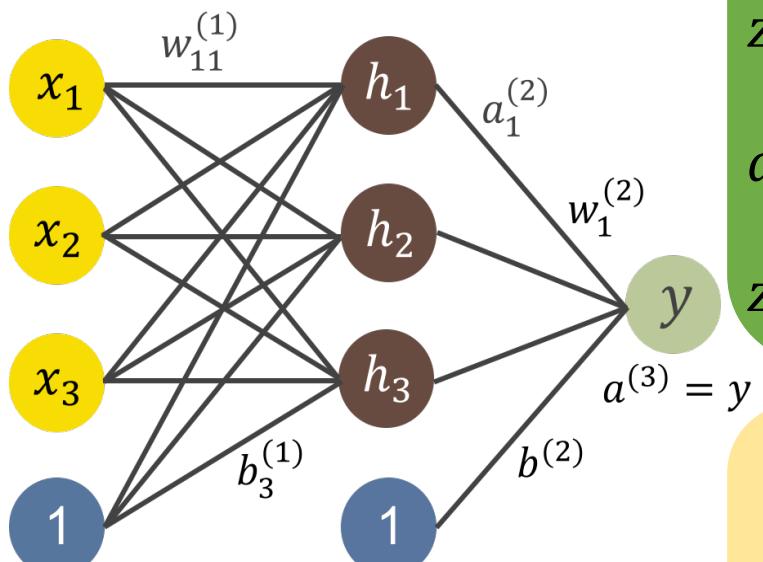


# Forward Computation: General Form



$$\begin{aligned}\mathbf{z}^{(l+1)} &= \mathbf{W}^{(l)} \mathbf{a}^{(l)} + \mathbf{b}^{(l)} \\ \mathbf{a}^{(l+1)} &= g^{(l+1)}(\mathbf{z}^{(l+1)})\end{aligned}$$

# Forward Computation: Summary



Forward flow

$\mathbf{z}^{(2)}, \mathbf{a}^{(2)}, \mathbf{z}^{(3)}, \mathbf{a}^{(3)}$

Forward Computation of the Neural Network

$$\begin{aligned} z_1^{(2)} &= \sum w_{1i}^{(1)} x_i + b_1^{(1)}; & z_2^{(2)} &= \sum w_{2i}^{(1)} x_i + b_2^{(1)}; & z_3^{(2)} &= \sum w_{3i}^{(1)} x_i + b_3^{(1)} \\ a_1^{(2)} &= g^{(2)}(z_1^{(2)}); & a_2^{(2)} &= g^{(2)}(z_2^{(2)}); & a_3^{(2)} &= g^{(2)}(z_3^{(2)}) \\ z^{(3)} &= \sum w_j^{(2)} a_j^{(2)} + b^{(2)}; & a^{(3)} &= g^{(3)}(z^{(3)}) \end{aligned}$$

Vector Form

$$\begin{aligned} \mathbf{z}^{(2)} &= \mathbf{W}^{(1)} \mathbf{x} + \mathbf{b}^{(1)} \\ \mathbf{a}^{(2)} &= g^{(2)}(\mathbf{z}^{(2)}) \\ \mathbf{z}^{(3)} &= \mathbf{W}^{(2)} \mathbf{a}^{(2)} + \mathbf{b}^{(2)} \\ \mathbf{a}^{(3)} &= g^{(3)}(\mathbf{z}^{(3)}) \end{aligned}$$



More General Form

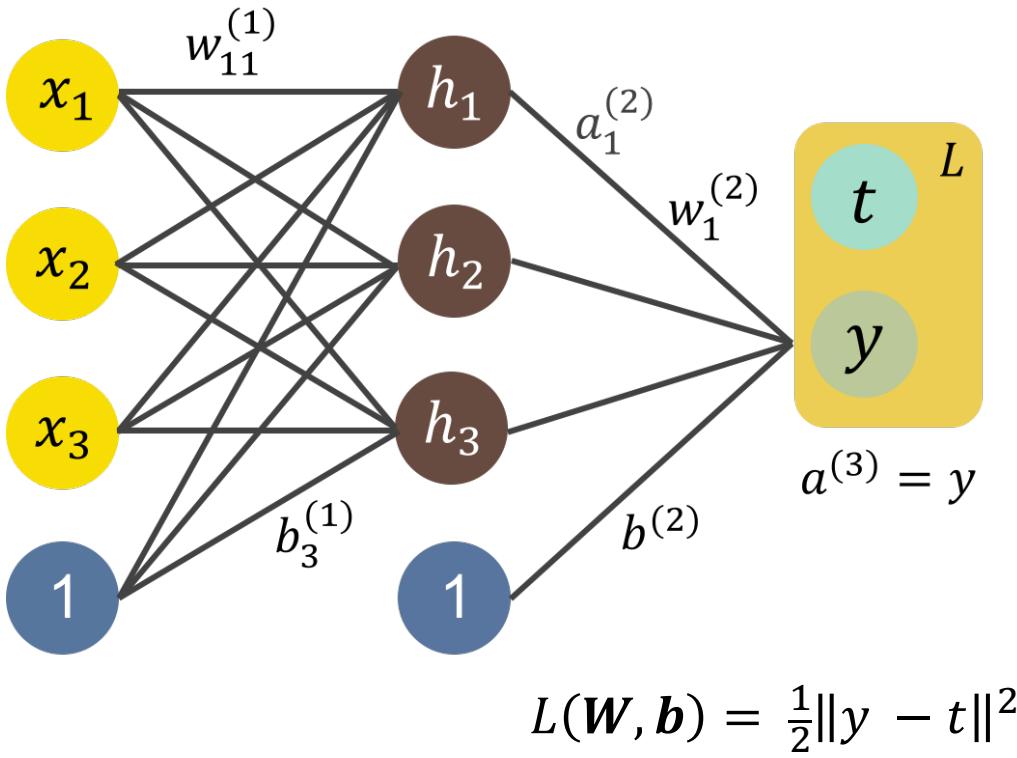
$$\begin{aligned} \mathbf{z}^{(l+1)} &= \mathbf{W}^{(l)} \mathbf{a}^{(l)} + \mathbf{b}^{(l)} \\ \mathbf{a}^{(l+1)} &= g^{(l+1)}(\mathbf{z}^{(l+1)}) \end{aligned}$$

# Backpropagation Algorithm

# Recap: SGD For Neural Networks



Given training sample  $(x, t)$



SGD (Training data  $(x, t)$ , learning rate  $\eta$ )

Initialize each  $w_{ji}^{(l)}$  and  $b_j^{(l)}$  to small random value  
Until convergence

**DO**

For each  $(x, t)$  in training data, **DO**

- compute  $\frac{\partial L(\mathbf{W}, \mathbf{b})}{\partial w_{ji}^{(l)}}$  and  $\frac{\partial L(\mathbf{W}, \mathbf{b})}{\partial b_j^{(l)}}$

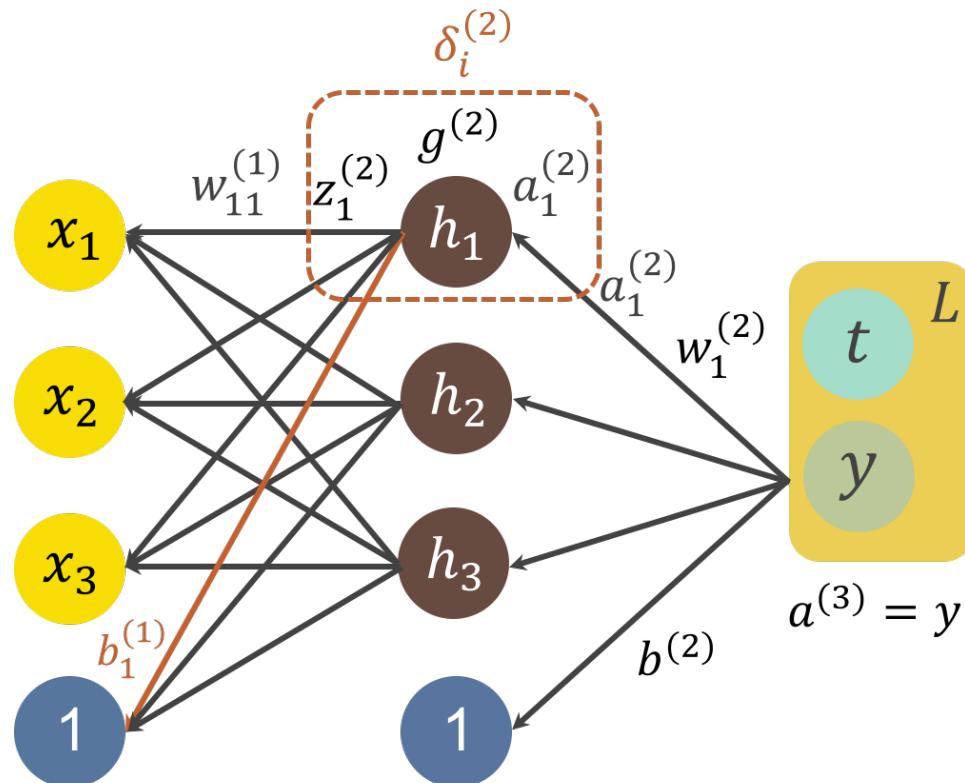
- For each  $w_{ji}^{(l)}$  and  $b_j^{(l)}$ :

$$w_{ji}^{(l)} = w_{ji}^{(l)} - \eta \frac{\partial L(\mathbf{W}, \mathbf{b})}{\partial w_{ji}^{(l)}}$$

$$b_j^{(l)} = b_j^{(l)} - \eta \frac{\partial L(\mathbf{W}, \mathbf{b})}{\partial b_j^{(l)}}$$

# Backward Propagation:

$$\frac{\partial L(\mathbf{W}, \mathbf{b})}{\partial w_{ji}^{(l)}}$$



**Claim:**  $\frac{\partial L}{\partial w_{ji}^{(l)}} = a_i^{(l-1)} \delta_j^{(l)}$

**Derivation :**

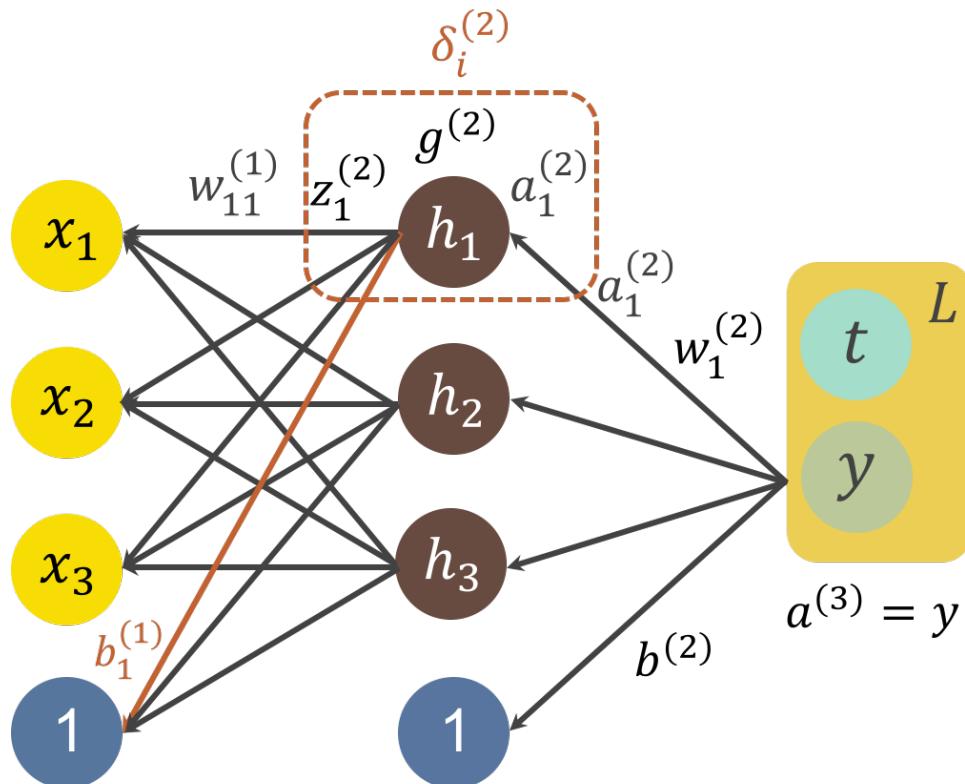
$$\begin{aligned}\frac{\partial L}{\partial w_{ji}^{(l)}} &= \frac{\partial L}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial w_{ji}^{(l)}} \\ &= \delta_j^{(l)} \frac{\partial (\sum_s w_{js}^{(l)} a_s^{(l-1)} + b_s^{(l)})}{\partial w_{ji}^{(l)}} \\ &= \delta_j^{(l)} a_i^{(l-1)}\end{aligned}$$

Define  $\delta_j^{(l)} = \frac{\partial L}{\partial z_j^{(l)}}$  to measure how much that node  $j$  of  $l$ th layer was responsible for the final error

Here we use  $s$  for general index, only one  $s$  matches the  $i$  in  $\partial w_{ji}^{(l)}$ .

# Backward Propagation:

$$\frac{\partial L(\mathbf{W}, \mathbf{b})}{\partial b_j^{(l)}}$$



**Claim :**  $\frac{\partial L}{\partial b_j^{(l)}} = \delta_j^{(l)}$

**Derivation :**

$$\begin{aligned}\frac{\partial L}{\partial b_j^{(l)}} &= \frac{\partial L}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial b_j^{(l)}} \\ &= \delta_j^{(l)} \frac{\partial (\sum_s w_{js}^{(l)} a_s^{(l-1)} + b_s^{(l)})}{\partial b_j^{(l)}} \\ &= \delta_j^{(l)}\end{aligned}$$



# Backward Propagation: $\frac{\partial L(W, b)}{\partial w_{ji}^{(l)}}$ and $\frac{\partial L(W, b)}{\partial b_j^{(l)}}$

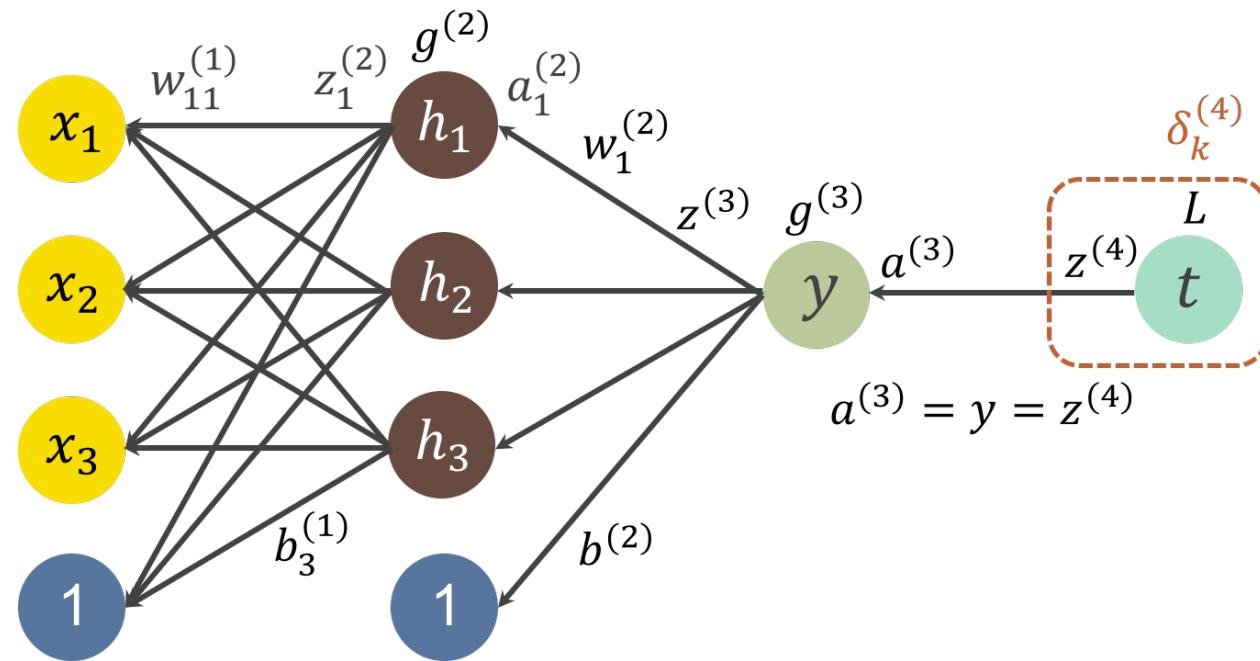
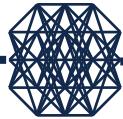
$$\frac{\partial L}{\partial w_{ji}^{(l)}} = a_i^{(l-1)} \delta_j^{(l)}$$

$$\frac{\partial L}{\partial b_j^{(l)}} = \delta_j^{(l)}$$



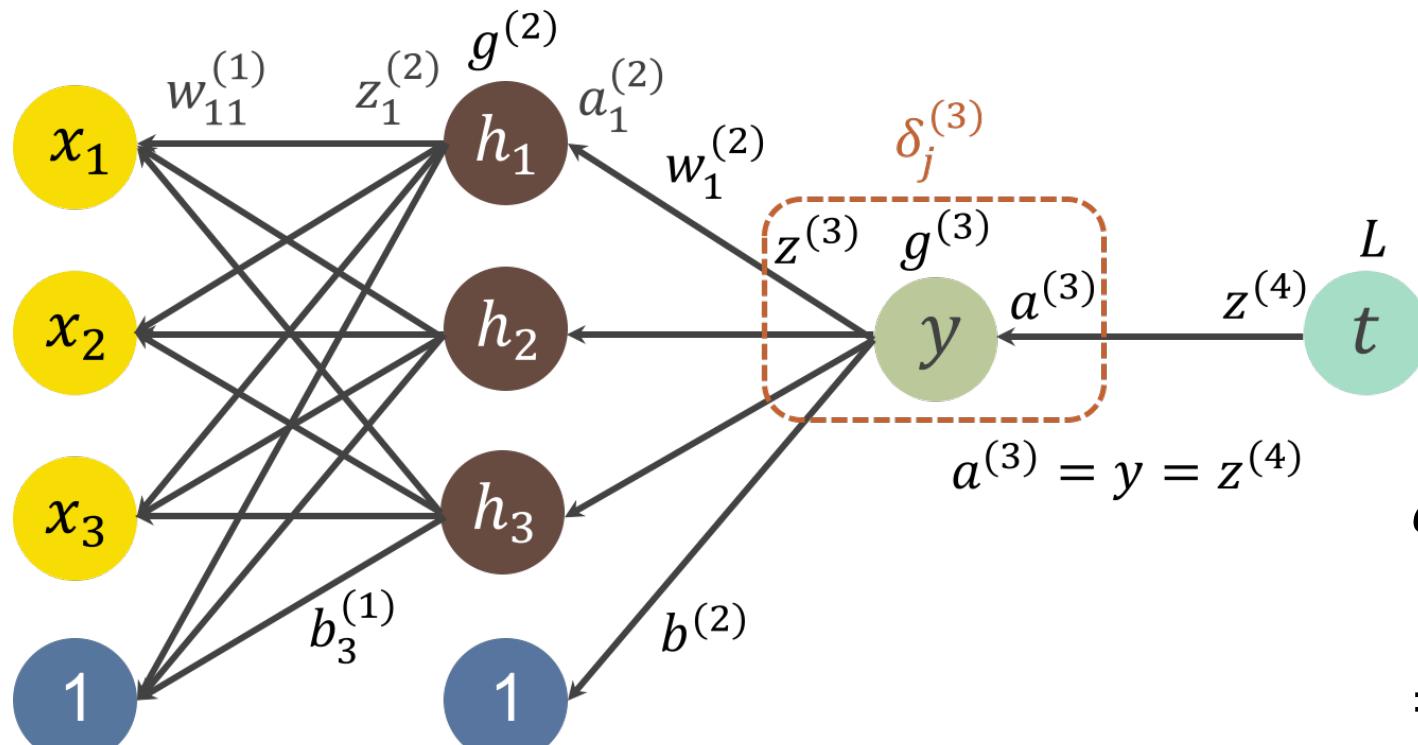
To compute the partial derivatives, we need all  $\delta_j^{(l)}$  which can be computed from **OUTPUT** to **INPUT** layer in a backward fashion

# Backward Propagation: Initialize $\delta_k^{(4)}$



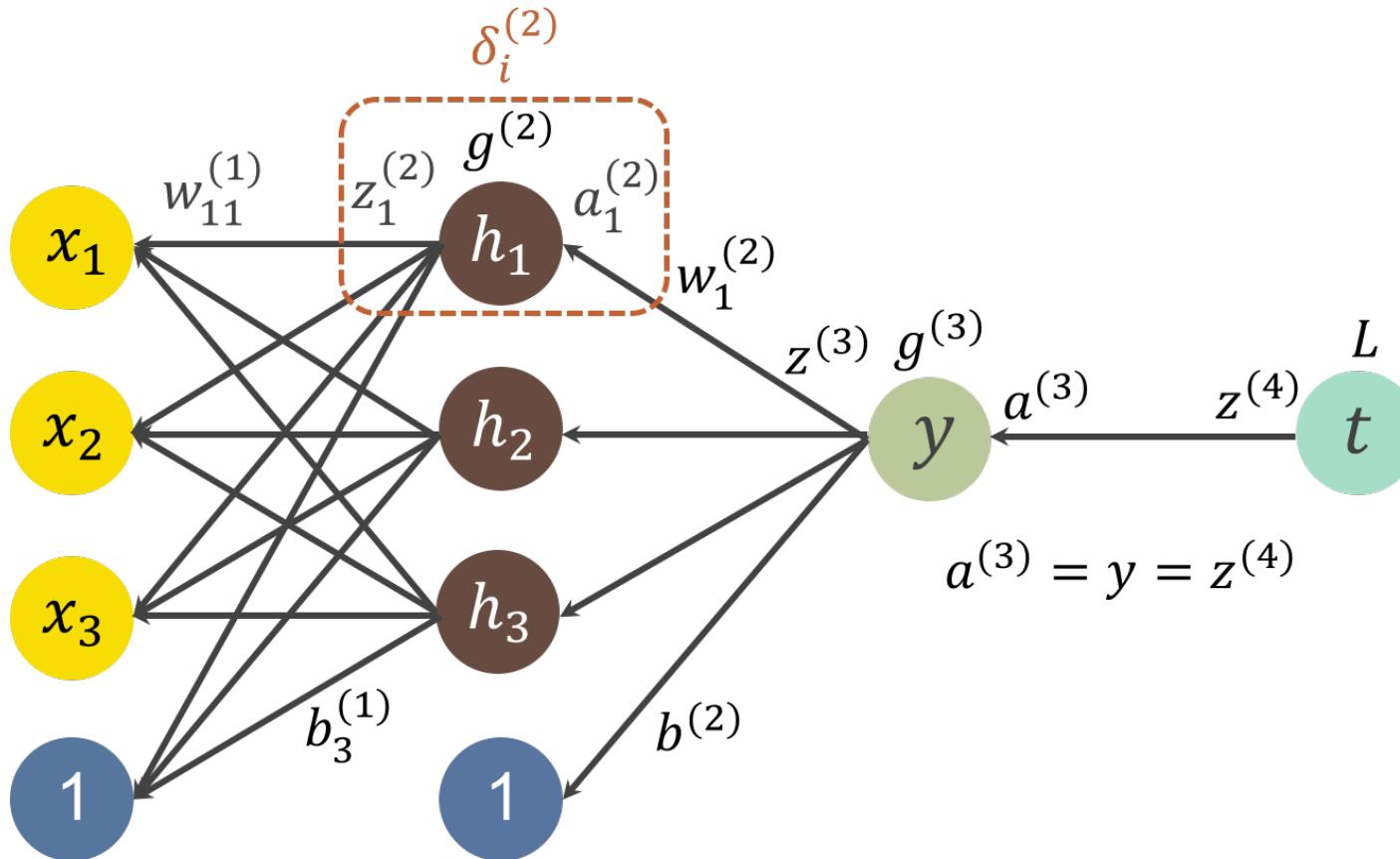
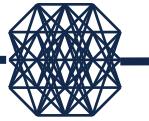
$$\begin{aligned}\delta_k^{(4)} &= \frac{\partial L}{\partial z_k^{(4)}} = \frac{\partial \frac{1}{2} \|y - t\|^2}{\partial y} \\ &= -(t - y)\end{aligned}$$

# Backward Propagation: $\delta_j^{(3)}$



$$\begin{aligned}\delta_j^{(3)} &= \frac{\partial L}{\partial z_j^{(3)}} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z_j^{(3)}} \\ &= \delta^{(4)}(g^{(3)})'(z_j^{(3)}) \\ &= -(t - y)(g^{(3)})'(z_j^{(3)})\end{aligned}$$

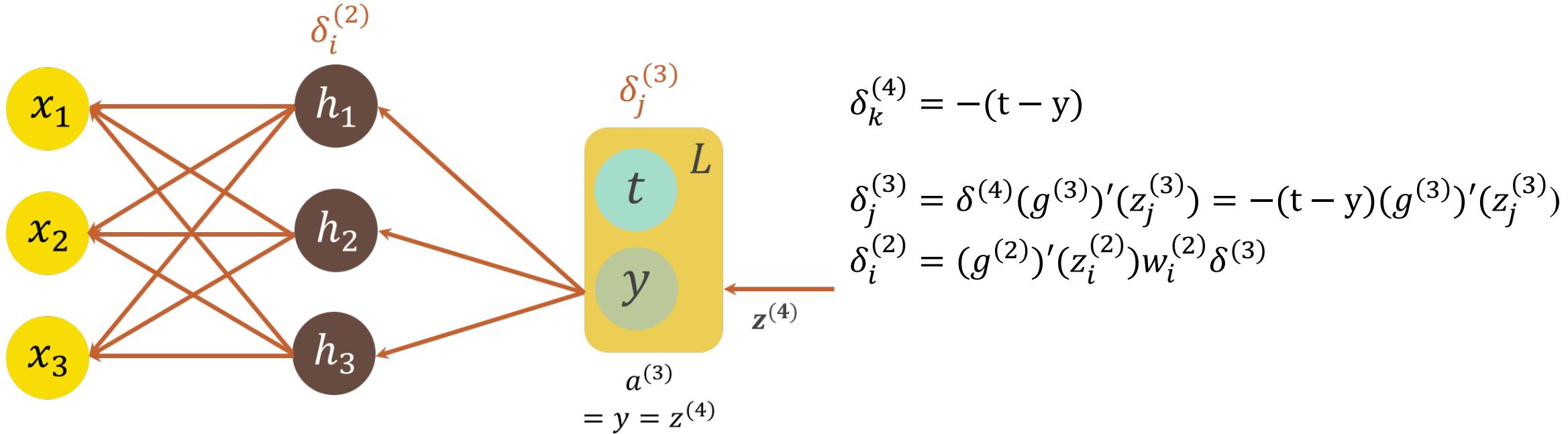
# Backward Propagation: $\delta_i^{(2)}$



$$a^{(3)} = y = z^{(4)}$$

$$\begin{aligned}\delta_i^{(2)} &= \frac{\partial L}{\partial z_i^{(2)}} = \frac{\partial L}{\partial z_j^{(3)}} \frac{\partial z_j^{(3)}}{\partial z_i^{(2)}} \\ &= \delta^{(3)} \frac{\partial \sum_{i=1}^3 w_i^{(2)} a_i^{(2)} + b^{(2)}}{\partial a_i^{(2)}} \frac{\partial a_i^{(2)}}{\partial z_i^{(2)}} \\ &= \delta^{(3)} w_i^{(2)} \frac{\partial a_i^{(2)}}{\partial z_i^{(2)}} \\ &= \delta^{(3)} w_i^{(2)} (g^{(2)})'(z_i^{(2)})\end{aligned}$$

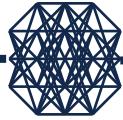
# Backward Propagation: Summary



In general we have the following relation between  $\delta_j^{(l)}$  and  $\delta_k^{(l+1)}$ :

$$\delta_j^{(l)} = (g^{(l)})'(z_j^{(l)}) \sum_{k=1}^{d^{(l)}} w_{kj}^{(l)} \delta_k^{(l+1)}$$

# Backpropagation Algorithm



## The FORWARD pass



Starting with the input  $x$ , go **forward** to output layer, compute and store the variables  $\mathbf{z}^{(2)}, \mathbf{a}^{(2)}, \mathbf{z}^{(3)}, \mathbf{a}^{(3)}, \dots, \mathbf{z}^{(L)}, \mathbf{a}^{(L)}, \mathbf{z}^{(L+1)}$

## The BACKWARD pass

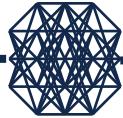


Initialize  $\delta^{(L+1)} = -(t - y) = -(t - z^{(L+1)})$  and compute the derivatives at the output layer:  $\frac{\partial L}{\partial w_k^{(L+1)}} = -(t - y)a_j^{(L)}$

Compute  $\delta_j^{(l)} = (g^{(l)})'(z_j^{(l)}) \sum_{k=1}^d w_{kj}^{(l)} \delta_k^{(l+1)}$  and also compute the derivatives at layer  $l$ :  $\frac{\partial L}{\partial w_{ji}^{(l)}} = a_i^{(l-1)} \delta_j^{(l)}$

# DNN Application: readmission prediction

# Hospital Readmission Prediction



Journal of Biomedical Informatics 56 (2015) 229–238



Contents lists available at ScienceDirect

Journal of Biomedical Informatics

journal homepage: [www.elsevier.com/locate/yjbin](http://www.elsevier.com/locate/yjbin)



## A comparison of models for predicting early hospital readmissions



CrossMark

Joseph Futoma <sup>a</sup>, Jonathan Morris <sup>b</sup>, Joseph Lucas <sup>a,c,\*</sup>

<sup>a</sup> Dept. of Statistical Science, Duke University, Box 90251, Durham, NC 27708, USA

<sup>b</sup> Quintiles, 4820 Emperor Blvd., Durham, NC 27703, USA

<sup>c</sup> Dept. of Electrical and Computer Engineering, Duke University, Box 90291, Durham, NC 27708, USA

---

### ARTICLE INFO

*Article history:*

Received 26 November 2014

Revised 13 May 2015

Accepted 24 May 2015

Available online 1 June 2015

---

*Keywords:*

Electronic Health Records

Early readmission

Penalized methods

Random forest

Deep learning

Predictive models

---

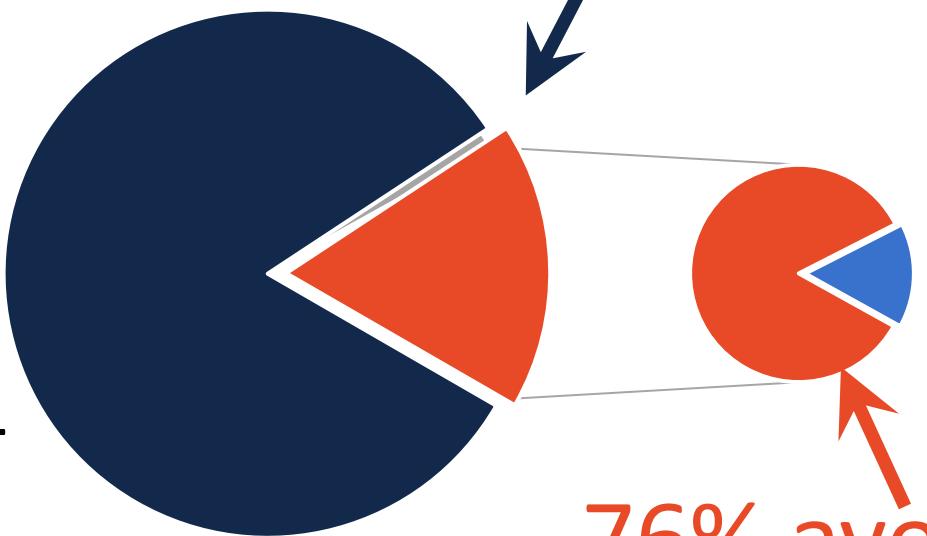
### ABSTRACT

Risk sharing arrangements between hospitals and payers together with penalties imposed by the Centers for Medicare and Medicaid (CMS) are driving an interest in decreasing early readmissions. There are a number of published risk models predicting 30 day readmissions for particular patient populations, however they often exhibit poor predictive performance and would be unsuitable for use in a clinical setting. In this work we describe and compare several predictive models, some of which have never been applied to this task and which outperform the regression methods that are typically applied in the healthcare literature. In addition, we apply methods from deep learning to the five conditions CMS is using to penalize hospitals, and offer a simple framework for determining which conditions are most cost effective to target.

© 2015 The Authors. Published by Elsevier Inc. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

# Hospital Readmission is a big problem

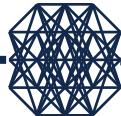
All hospital admissions



\$15 billion in Medicare spending  
on readmission

Over 2000 hospitals received  
**\$280 million** penalty

# New Zealand National Minimum Dataset



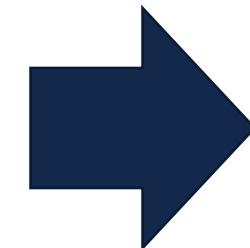
Full dataset, post-processing.

## *Characteristic*

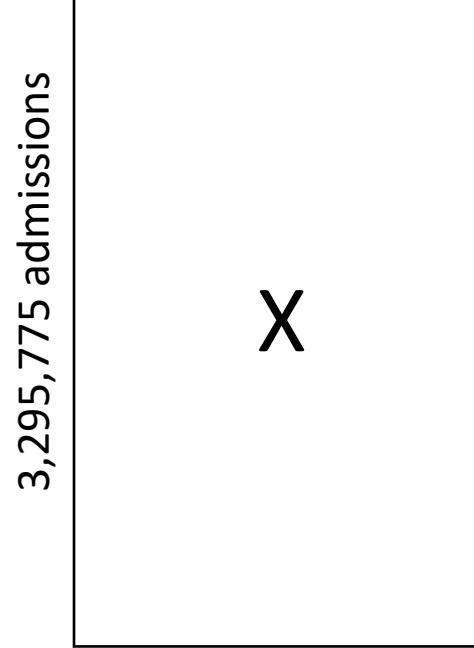
Total number of admissions	3,295,775
Number of unique individuals	1,328,384
Percent readmission within 30 days	19.0
Number of unique procedures (ICD-10 AM)	3599
Number of unique diagnoses (ICD-10 AM)	8446
Number of ICD-10 AM codes per visit, mean (SD)	5.1 (3.8)
Number of unique diagnosis related groups (DRGs)	815

## *Variables used in prediction*

Age (years), mean (SD)	41.2 (14.1)
Male (%)	38.8
White/Islander/Asian/Hispanic/African (%)	62.6/26.9/7.1/ 0.2/0.5
Public facility (%)	93.9
Transfer (%)	5.6
Length of Stay (days), mean (2.5%, 25%, 50%, 75%, 97.5% quantiles)	2.9 (0, 0, 1, 3, 16)
Number of admissions in past 365 days, mean (2.5%, 25%, 50%, 75%, 97.5% quantiles)	3.7 (0, 0, 0, 1, 25)



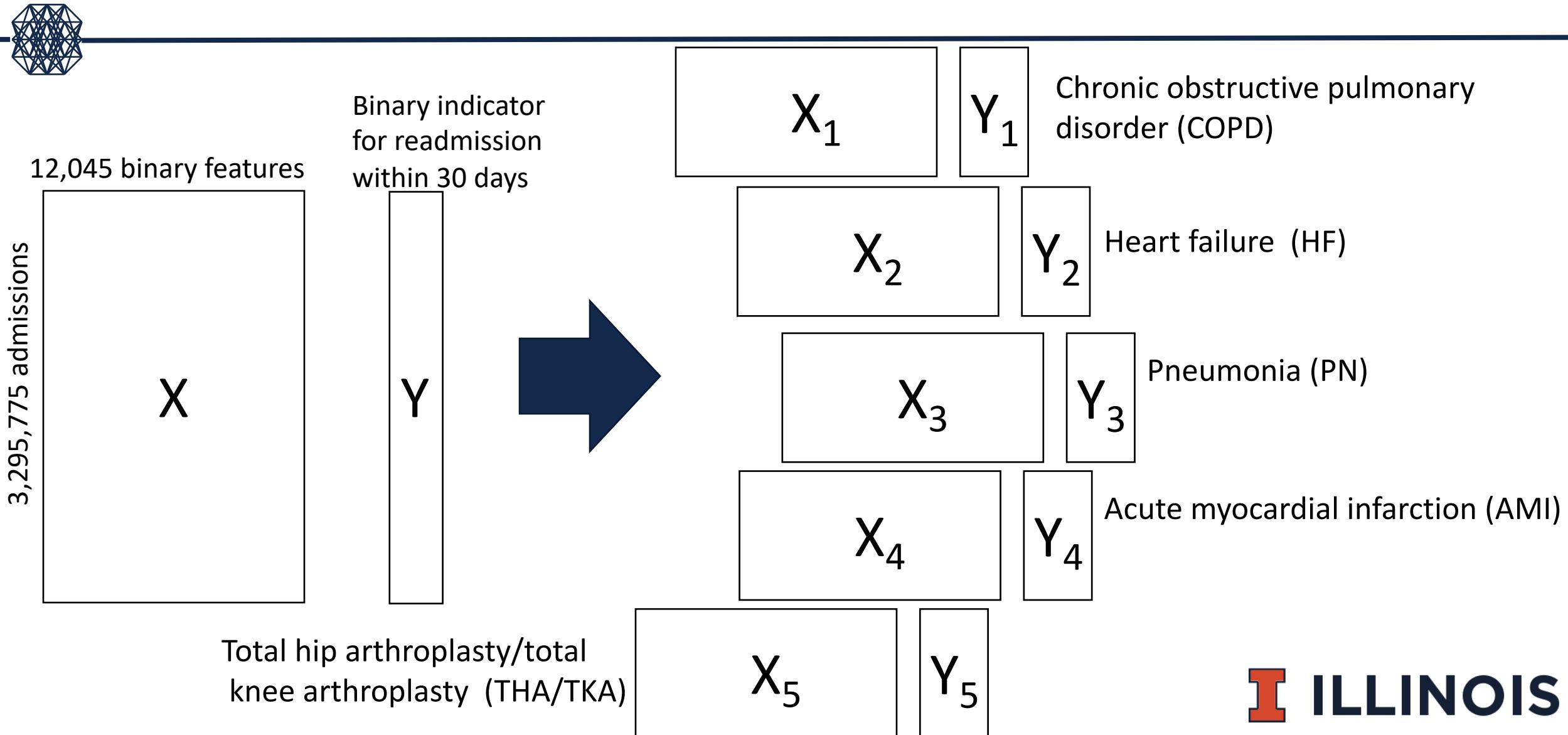
12,045 binary features  
3,295,775 admissions



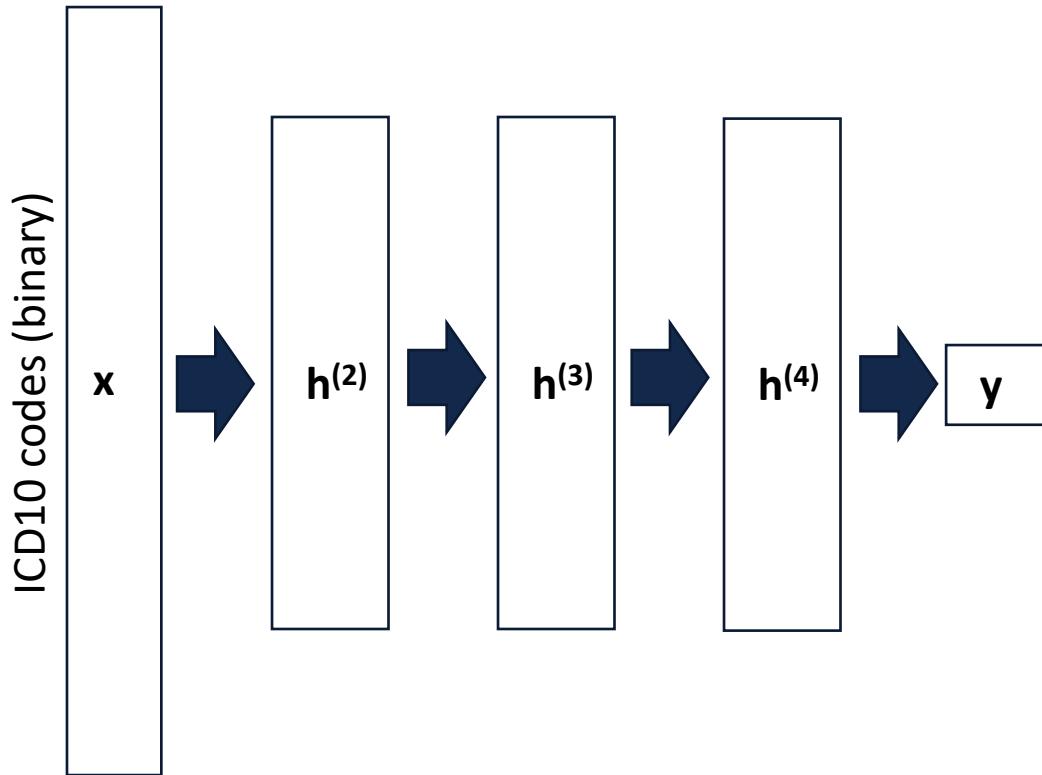
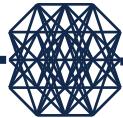
Binary indicator  
for readmission  
within 30 days



# Cohort construction



# DNN model



- $x$  is binary vector corresponding to ICD10 codes
- $\mathbf{h}^{(2)} = g(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$
- $\mathbf{h}^{(3)} = g(\mathbf{W}^{(2)}\mathbf{h}^{(2)} + \mathbf{b}^{(2)})$
- $\mathbf{h}^{(4)} = g(\mathbf{W}^{(3)}\mathbf{h}^{(3)} + \mathbf{b}^{(3)})$  where  $g()$  is sigmoid
- $\mathbf{y} = f(\mathbf{W}^{(4)}\mathbf{h}^{(4)} + \mathbf{b}^{(4)})$  where  $f()$  is softmax

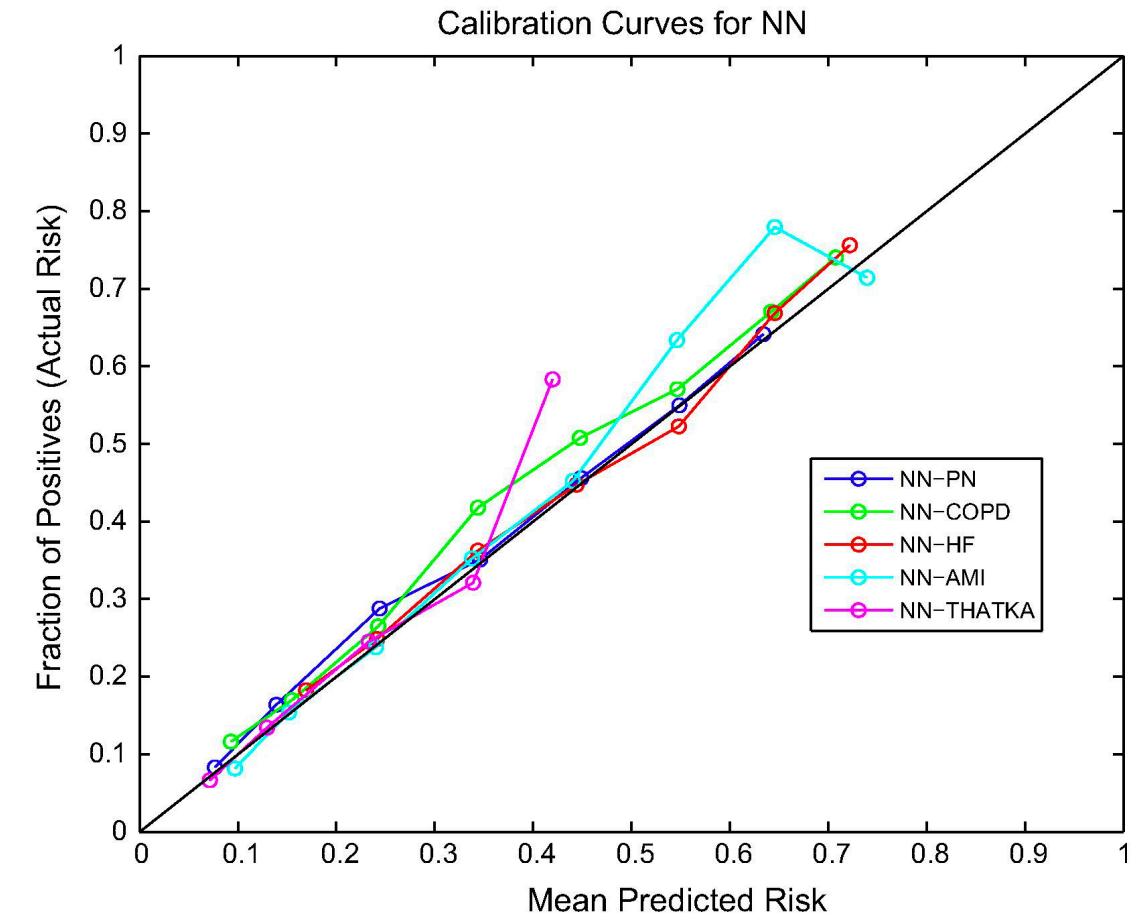
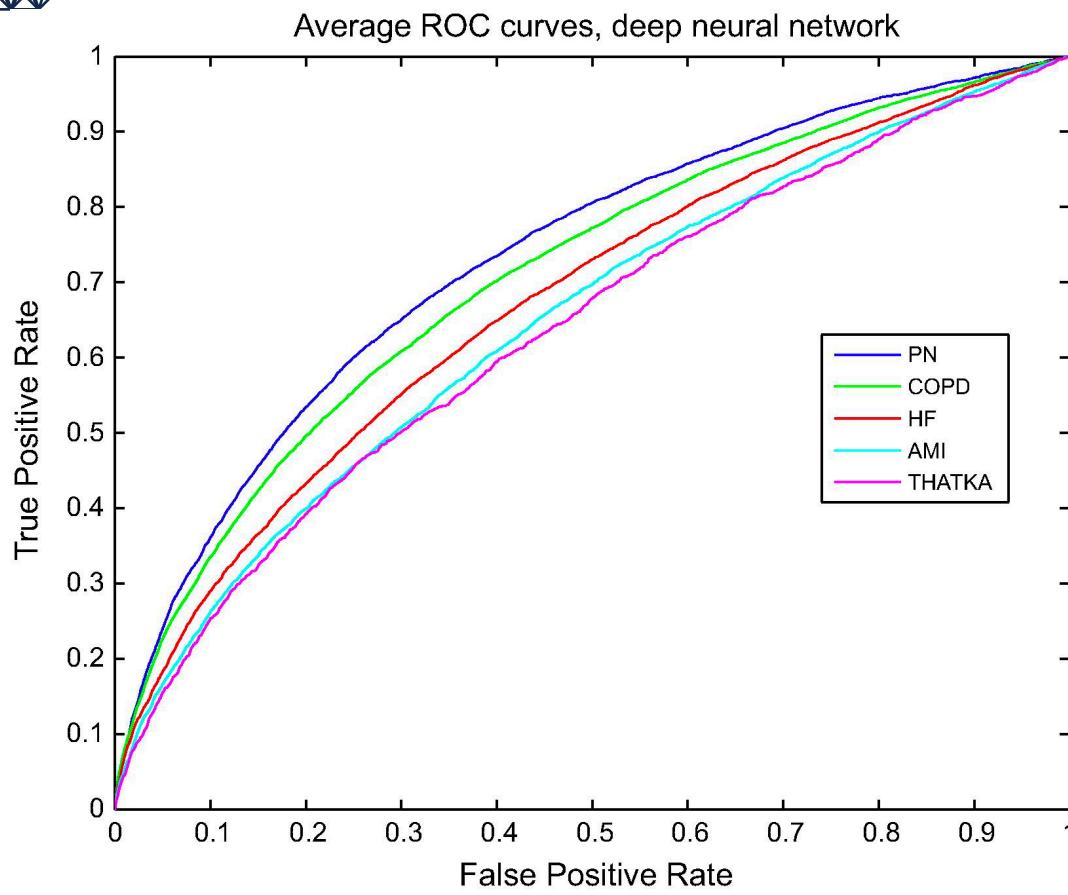
# Comparison between logistic regression and DNN



Condition	Size	Readmission rate (%)	Logistic regression	DNN
PN	40,442	27.9	0.715 (0.005)	<b>0.734 (0.004)</b>
COPD	31,457	20.4	0.703 (0.003)	<b>0.711 (0.003)</b>
HF	25,941	19.0	0.654 (0.005)	<b>0.676 (0.005)</b>
AMI	29,060	29.5	<b>0.633 (0.006)</b>	<b>0.649 (0.007)</b>
THA/TKA	23,128	8.7	<b>0.629 (0.005)</b>	<b>0.638 (0.006)</b>

- DNN consistently had better AUC than logistic regression

# Result



- DNN achieves reasonable AUC and great calibration

# Deep Neural Nets as a Method for Quantitative Structure-Activity Relationships.

Ma, Junshui, Robert P. Sheridan, Andy Liaw, George E. Dahl, and Vladimir Svetnik. 2015. Journal of Chemical Information and Modeling 55 (2): 263–74

2012



## Merck Molecular Activity Challenge

Help develop safe and effective medicines by predicting molecular activity.

\$40,000 · 236 teams · 7 years ago

[Overview](#)   [Data](#)   [Discussion](#)   [Leaderboard](#)   [Rules](#)

### Overview

#### Description

Help enable the development of safe, effective medicines.

#### Prizes

When [developing new medicines](#) it is important to identify molecules that are highly active toward their

intended targets but not toward other targets that might cause side effects. The objective of this competition is to identify the best statistical techniques for predicting biological activities of different molecules, both on- and off-target, given numerical descriptors generated from their chemical structures

#### Evaluation

The challenge is based on 15 molecular activity data sets, each for a biologically relevant target. Each row corresponds to a molecule and contains descriptors derived from that molecule's chemical structure.

#### Visualization- Prospect

In addition to the prediction competition, Merck is also hosting a [visualization challenge](#) with a \$2,000 prize for the most insightful and elegant graphical representations of the data.

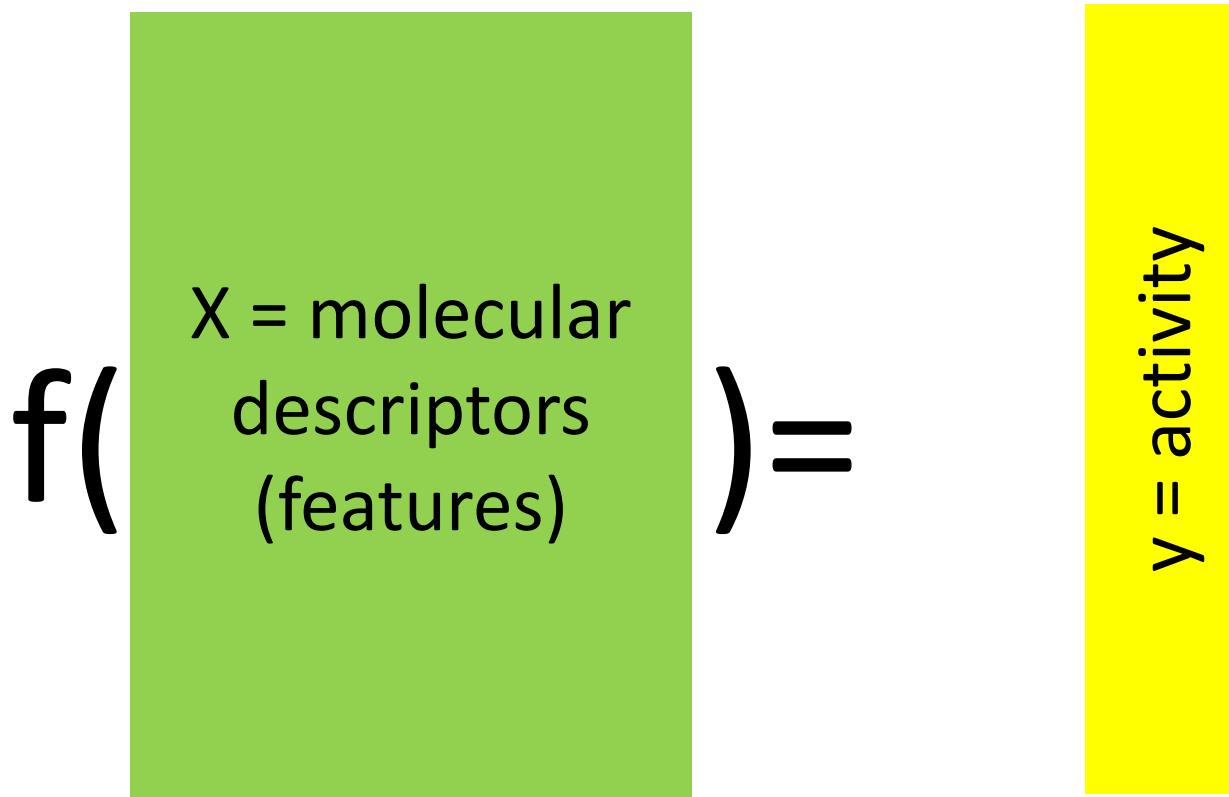
#### Submission- Instructions

Prizes total \$40,000.

#### Winners

Ma et al. 2015. "Deep Neural Nets as a Method for Quantitative Structure-Activity Relationships." *Journal of Chemical Information and Modeling* 55 (2): 263–74.

# Deep learning for Quantitative Structure-Activity Relationships (QSAR)



Ma et al. 2015. "Deep Neural Nets as a Method for Quantitative Structure-Activity Relationships." *Journal of Chemical Information and Modeling* 55 (2): 263–74.

# Datasets & tasks

**15** tasks/datasets (Kaggle competition)

+ **15** additional datasets

Largest dataset has **318,795** molecules

and **12,508** descriptors/features

data set	type	description		number of molecules	number of unique AP, DP descriptors
Kaggle Data Sets					
3A4	ADME	CYP P450 3A4 inhibition $-\log(\text{IC50})$ M		50000	9491
CB1	target	binding to cannabinoid receptor 1 $-\log(\text{IC50})$ M		11640	5877
DPP4	target	inhibition of dipeptidyl peptidase 4 $-\log(\text{IC50})$ M		8327	5203
HIVINT	target	inhibition of HIV integrase in a cell based assay $-\log(\text{IC50})$ M		2421	4306
HIVPROT	target	inhibition of HIV protease $-\log(\text{IC50})$ M		4311	6274
LOGD	ADME	logD measured by HPLC method		50000	8921
METAB	ADME	percent remaining after 30 min microsomal incubation		2092	4595
NK1	target	inhibition of neurokinin1 (substance P) receptor binding $-\log(\text{IC50})$ M		13482	5803
OX1	target	inhibition of pregnane X receptor $-\log(K_i)$ M		7135	4730
		inhibition of CYP2B6 receptor $-\log(K_i)$ M		14875	5790
		coprotein log(BA/AB)		8603	5135
		protein binding log(bound/unbound)		11622	5470
		ity) at 2 mg/kg		7821	5698
		A4 inhibitions $\log(\text{IC50}$ without NADPH/ $\text{IC50}$ with		5559	5945
		nhibition $-\log(\text{IC50})$ M		6924	5552
Additional Data Sets					
2C8	ADME	CYP P450 2C8 inhibition $-\log(\text{IC50})$ M		29958	8217
2C9	ADME	CYP P450 2C9 inhibition $-\log(\text{IC50})$ M		189670	11730
		( $\text{IC50}$ ) M		50000	9729
		tor $-\log(\text{IC50})$ M		2763	5242
		$g(\text{IC50})$ M		17469	6200
		1		50000	8959
		$\log(\text{clearance}) \mu\text{L}/\text{min}\cdot\text{mg}$		23292	6782
		( $\text{IC50}$ ) M		12843	6596
		$^{\circ}\text{C}$ M		9536	6136
		itions $\log(\text{solubility}) \text{ mol/L}$		89531	9541
HERG	ADME	inhibition of hERG channel $-\log(\text{IC50})$ M		50000	9388
HERG (full data set)	ADME	inhibition of hERG ion channel $-\log(\text{IC50})$ M		318795	12508
NAV	ADME	inhibition of Nav1.5 ion channel $-\log(\text{IC50})$ M		50000	8302
PAPP	ADME	apparent passive permeability in PK1 cells $\log(\text{permeability}) \text{ cm/s}$		30938	7713
PXR	ADME	induction of 3A4 by pregnane X receptor; percentage relative to rifampicin		50000	9282

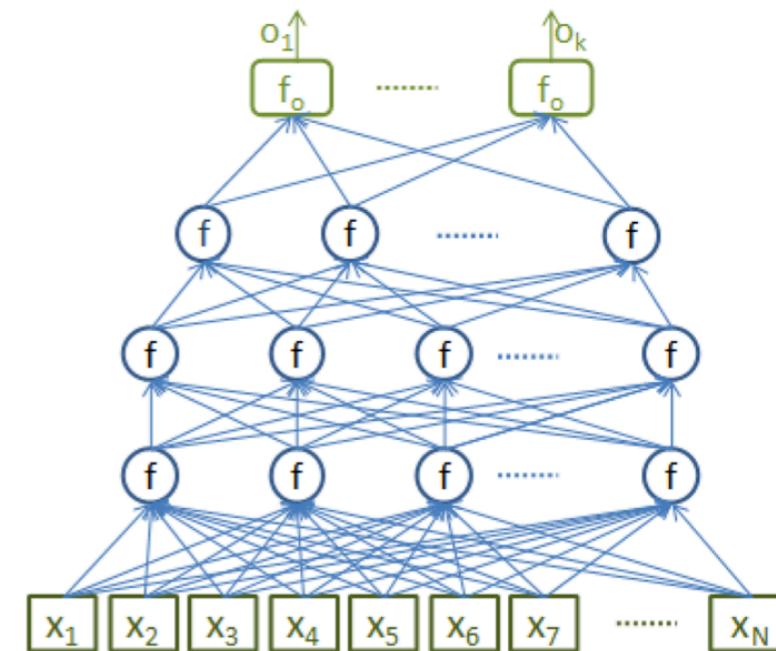
# Methods



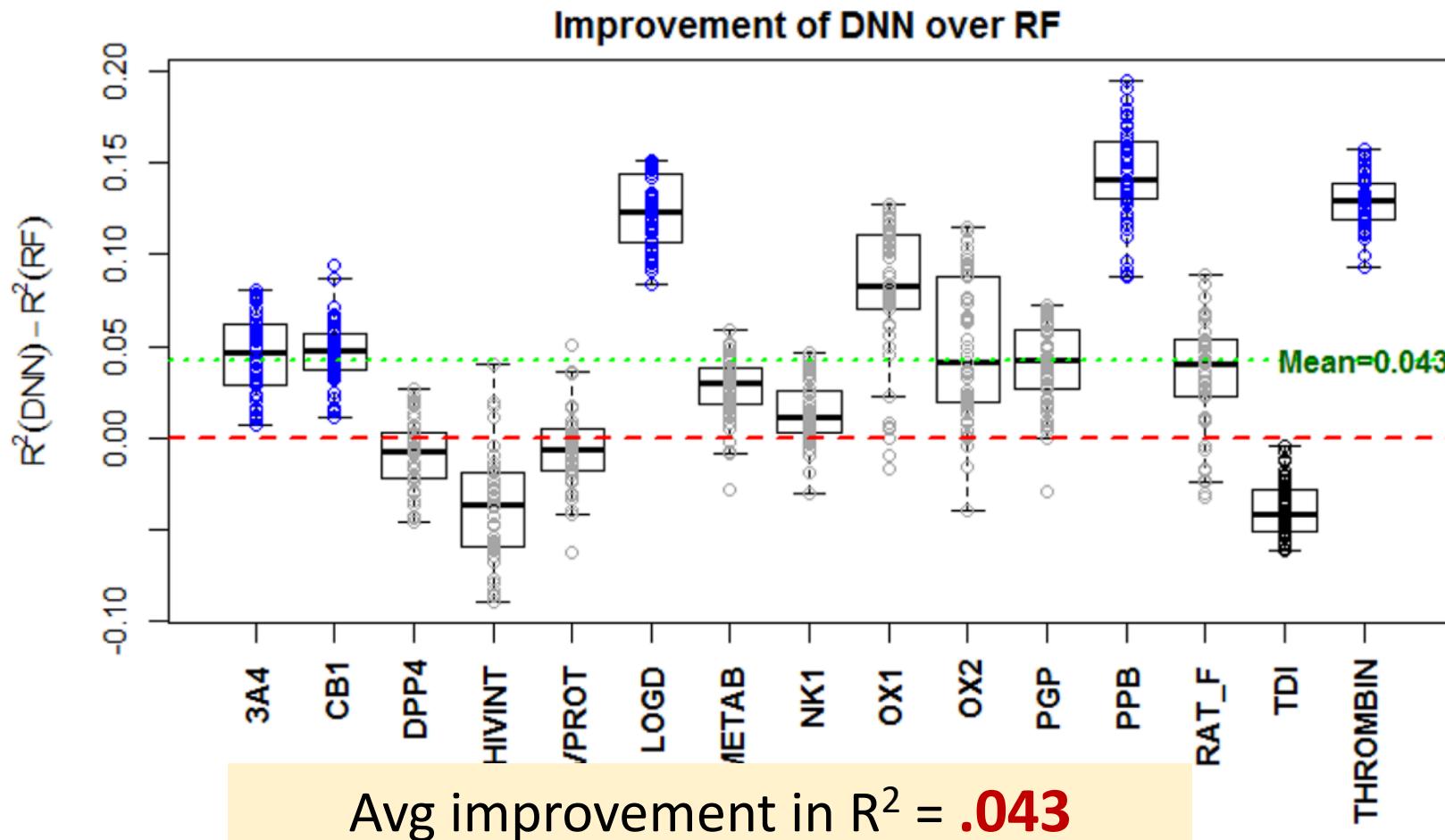
- Previous state of the art
  - Random forest (RF)



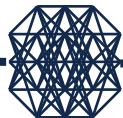
- Fully connected neural networks with 1 or 2 hidden layers



# Results on Kaggle competition data



# Results on additional data



Avg improvement **13.9%**

RF = .361

DNN = .411

**Table 3. Comparing RF with DNN Trained Using Recommended Parameter Settings on 15 Additional Datasets**

data set	random forest ( $R^2$ )	individual DNN ( $R^2$ )
2C8	0.158	<b>0.255</b>
2C9BIG	0.279	<b>0.363</b>
2D6	0.130	<b>0.195</b>
A-II	0.805	<b>0.812</b>
BACE	0.629	<b>0.644</b>
CAV	0.399	<b>0.463</b>
CLINT	0.393	<b>0.554</b>
ERK2	<b>0.257</b>	0.198
FACTORXIA	0.241	<b>0.244</b>
FASSIF	<b>0.294</b>	0.271
HERG	0.305	<b>0.352</b>
HERGfull	0.294	<b>0.367</b>
NAV	0.277	<b>0.347</b>
PAPP	0.621	<b>0.678</b>
PXR	0.333	<b>0.416</b>
<i>mean</i>	0.361	<b>0.411</b>