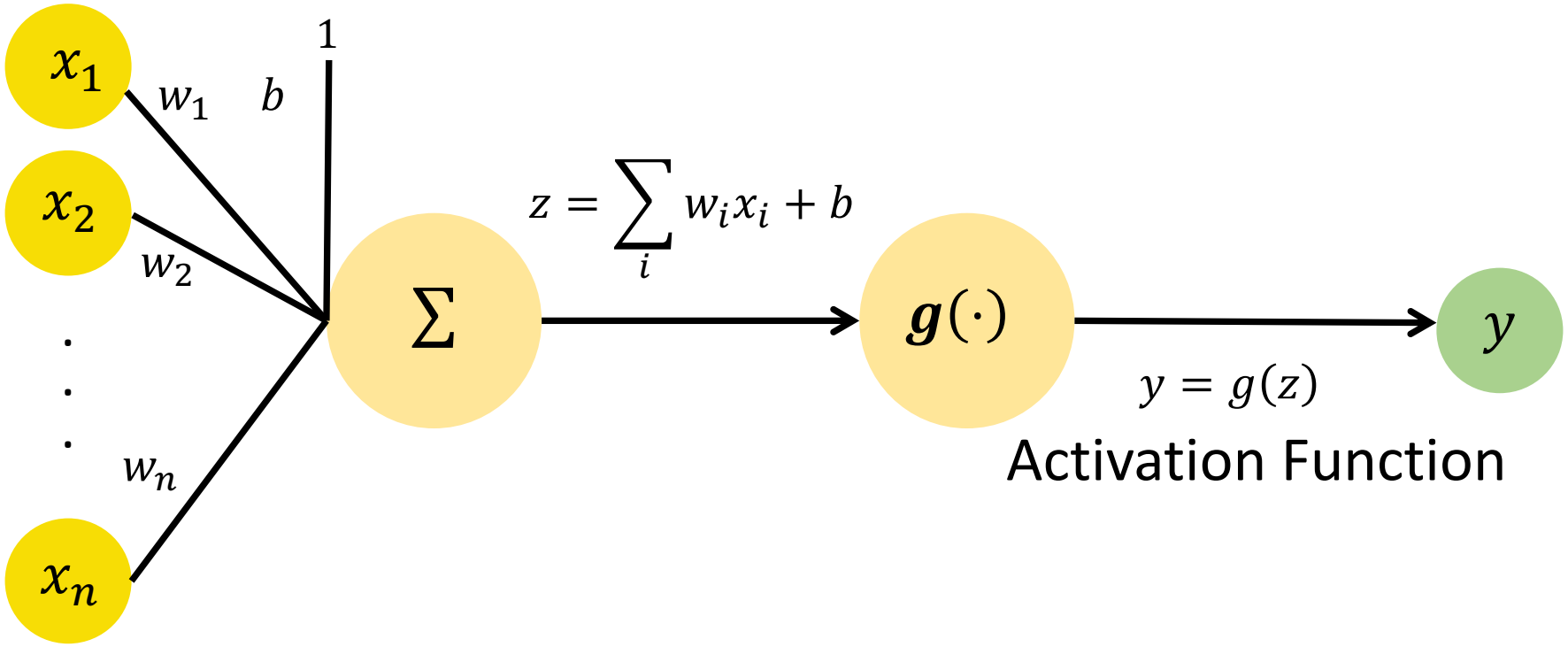
Deep Neural Networks

Prof. Jimeng Sun

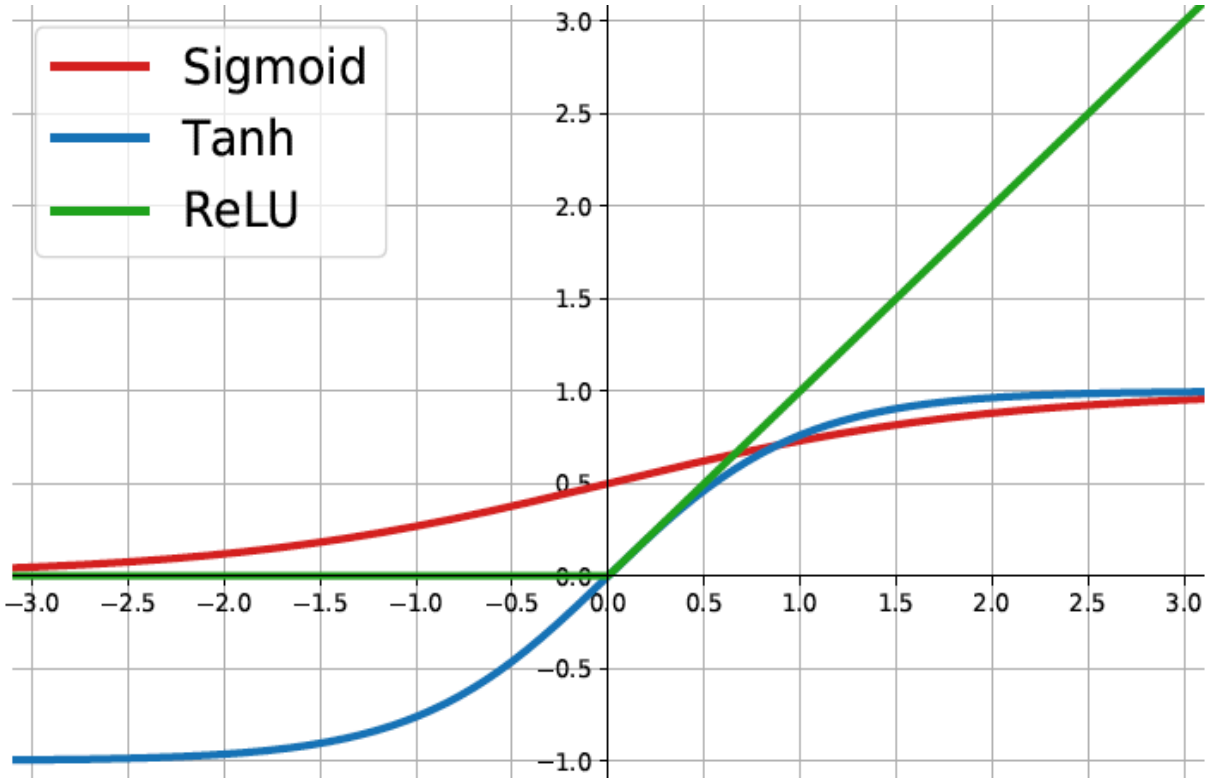
ILLINOIS

A Single Neuron



ILLINOIS

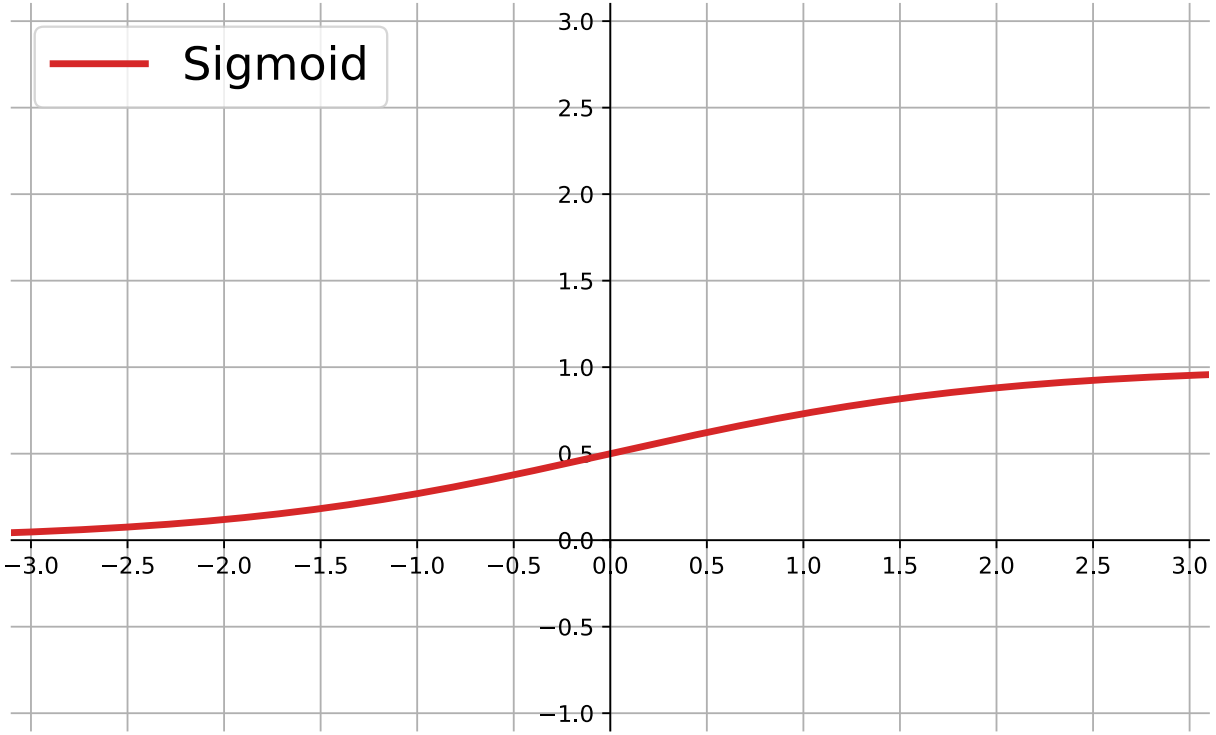
Activation Functions g(x)



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

ILLINOIS

Sigmoid Function

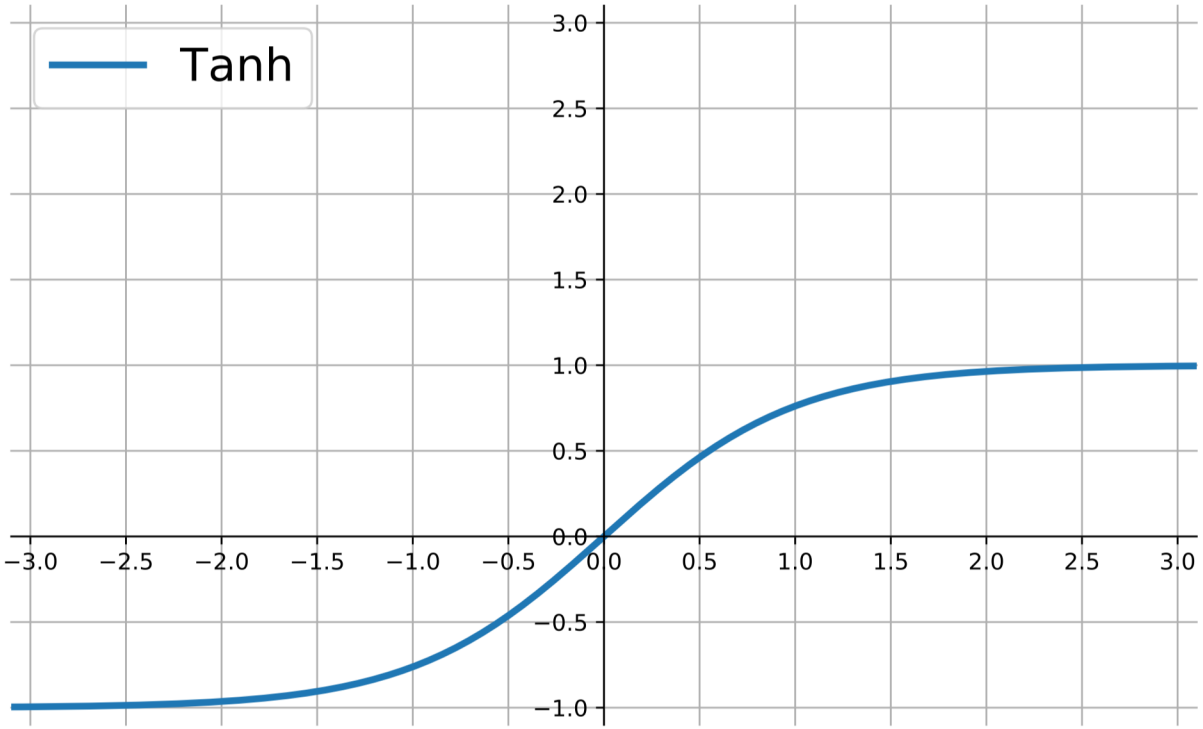


$$g(x) = \frac{1}{1 + e^{-x}}$$

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

ILLINOIS

Tanh Function

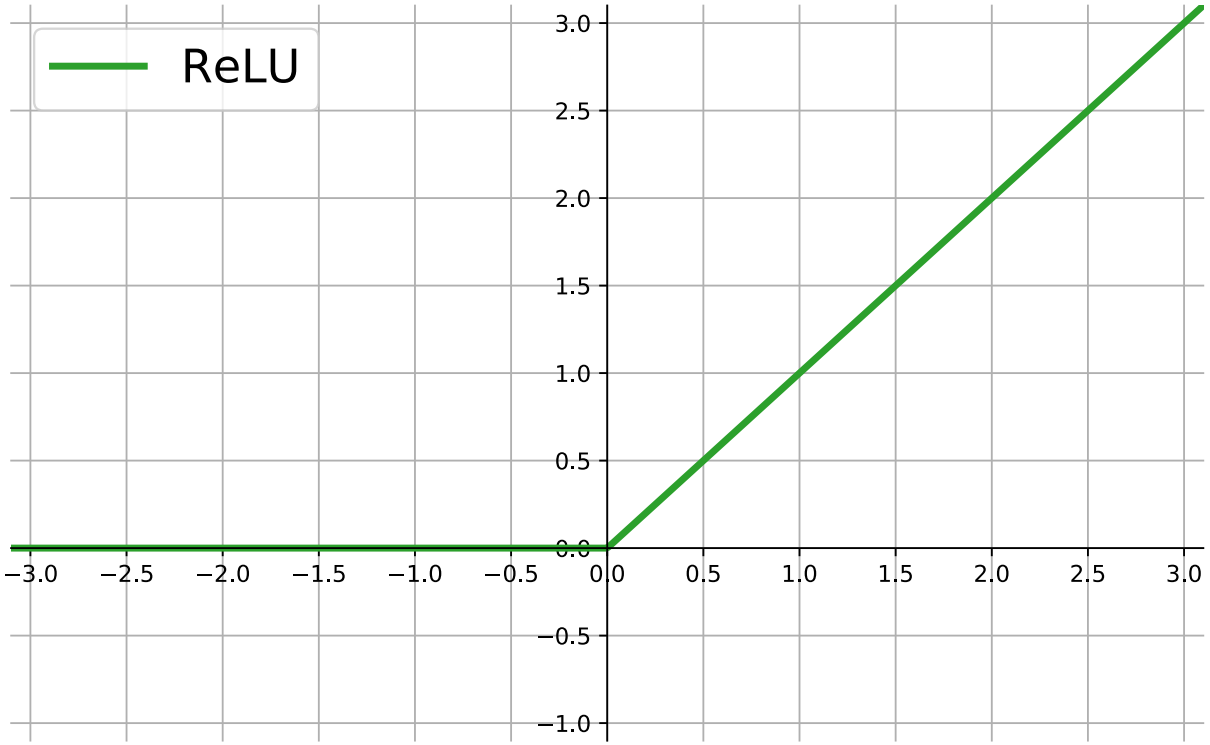


$$g(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{2}{1 + e^{(-2x)}} - 1$$

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

ILLINOIS

Rectified Linear Unit (ReLU)

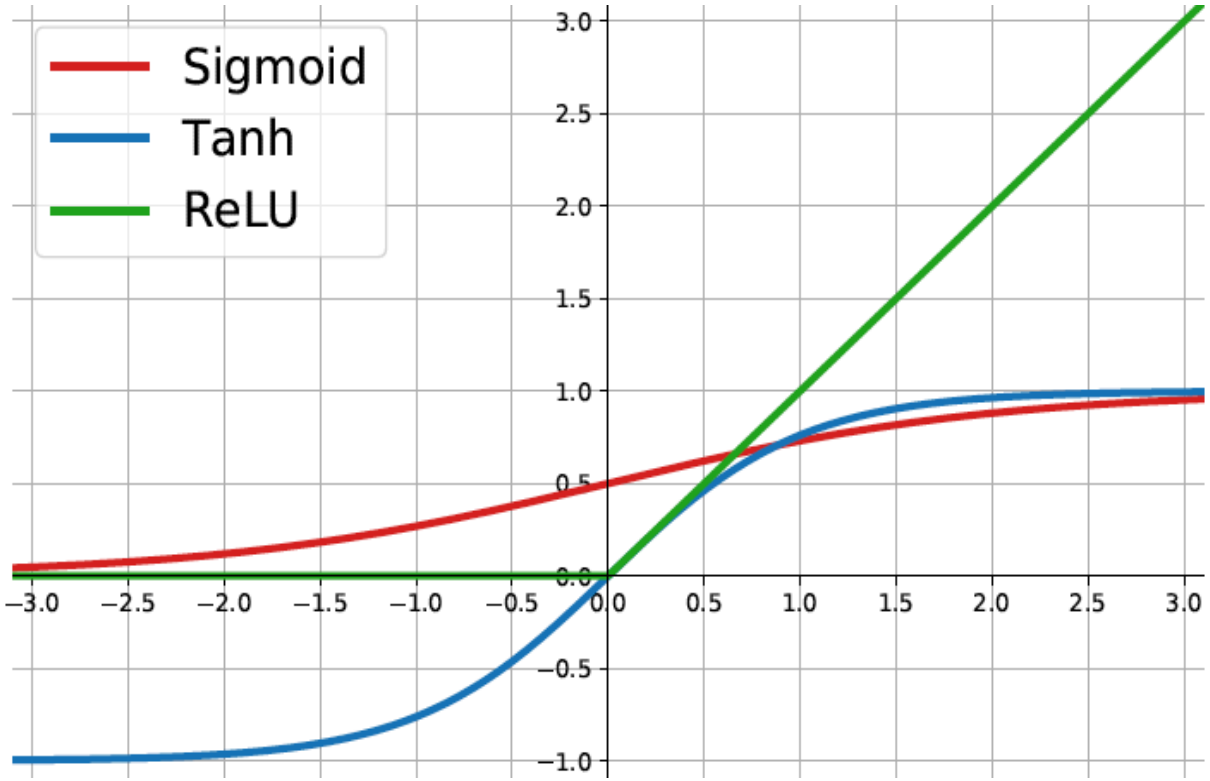


$$g(x) = \max(0, x)$$

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

ILLINOIS

Activation Functions: Summary



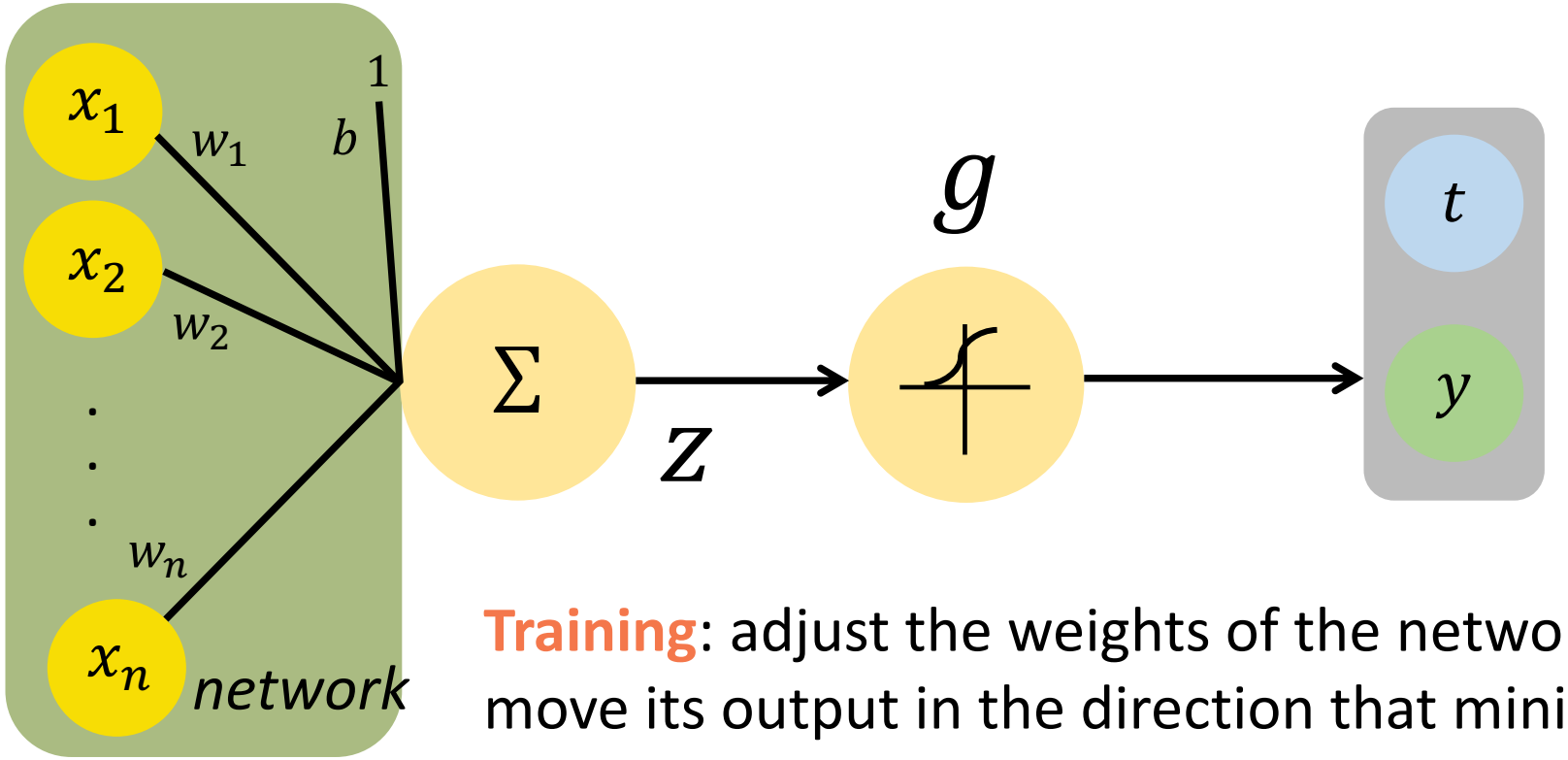
Sigmoid $g(x) = \frac{1}{1 + e^{-x}}$

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

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

ILLINOIS

Train A Single Neuron



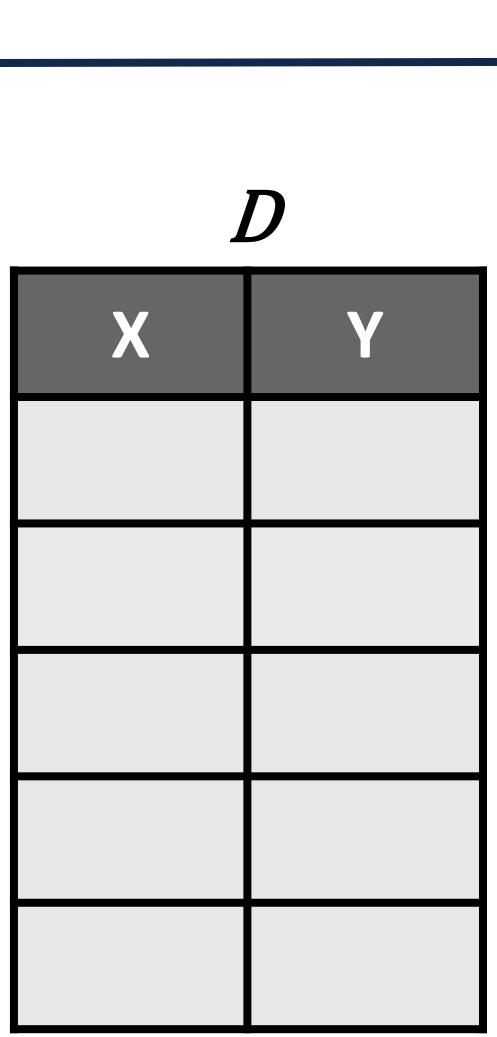
$$L = \frac{1}{2}||y - t||^2$$

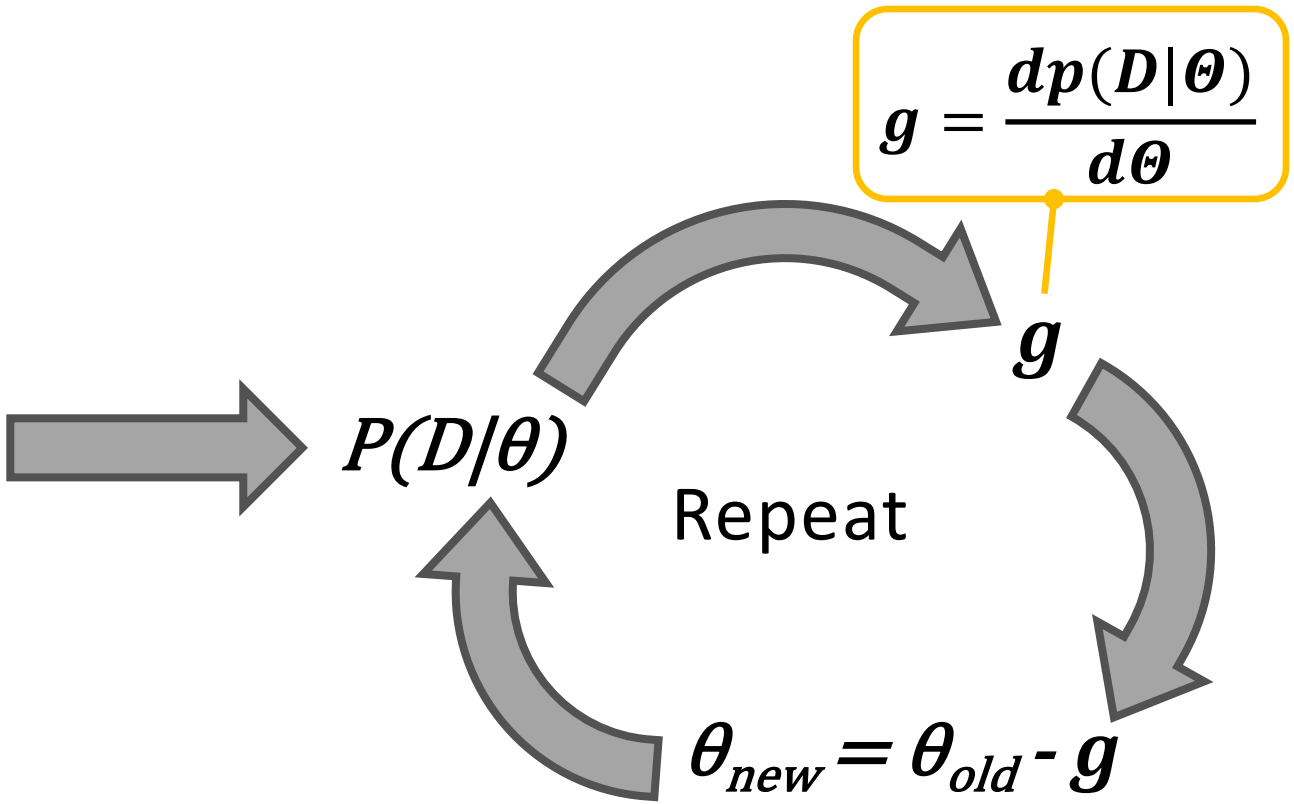
t: label

Training: adjust the weights of the network to move its output in the direction that minimizes the loss on training data.

ILLINOIS

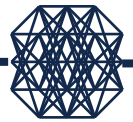
Gradient Descent Method





ILLINOIS

Gradient Descent Method For Linear Regression



D

X	Y

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

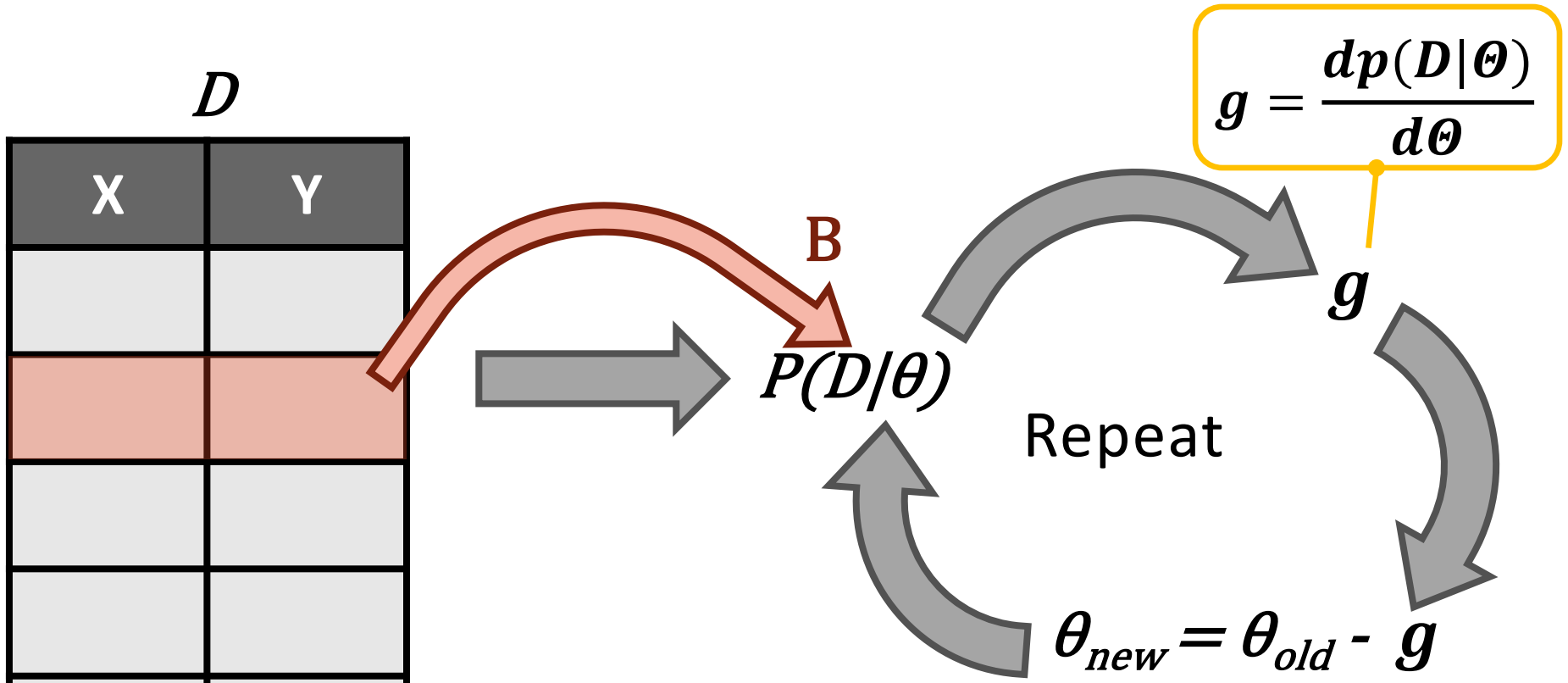
$$\frac{\partial l(\beta)}{\partial \beta_j} = -2\sum_{i}(y^{(i)} - \beta^T \mathbf{x}^{(i)})x_j^{(i)}$$

Repeat

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

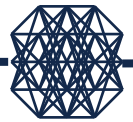
ILLINOIS

Stochastic Gradient Descent (SGD) Method



ILLINOIS

SGD For Linear Regression



D

X	Y

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

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

Repeat

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

ILLINOIS

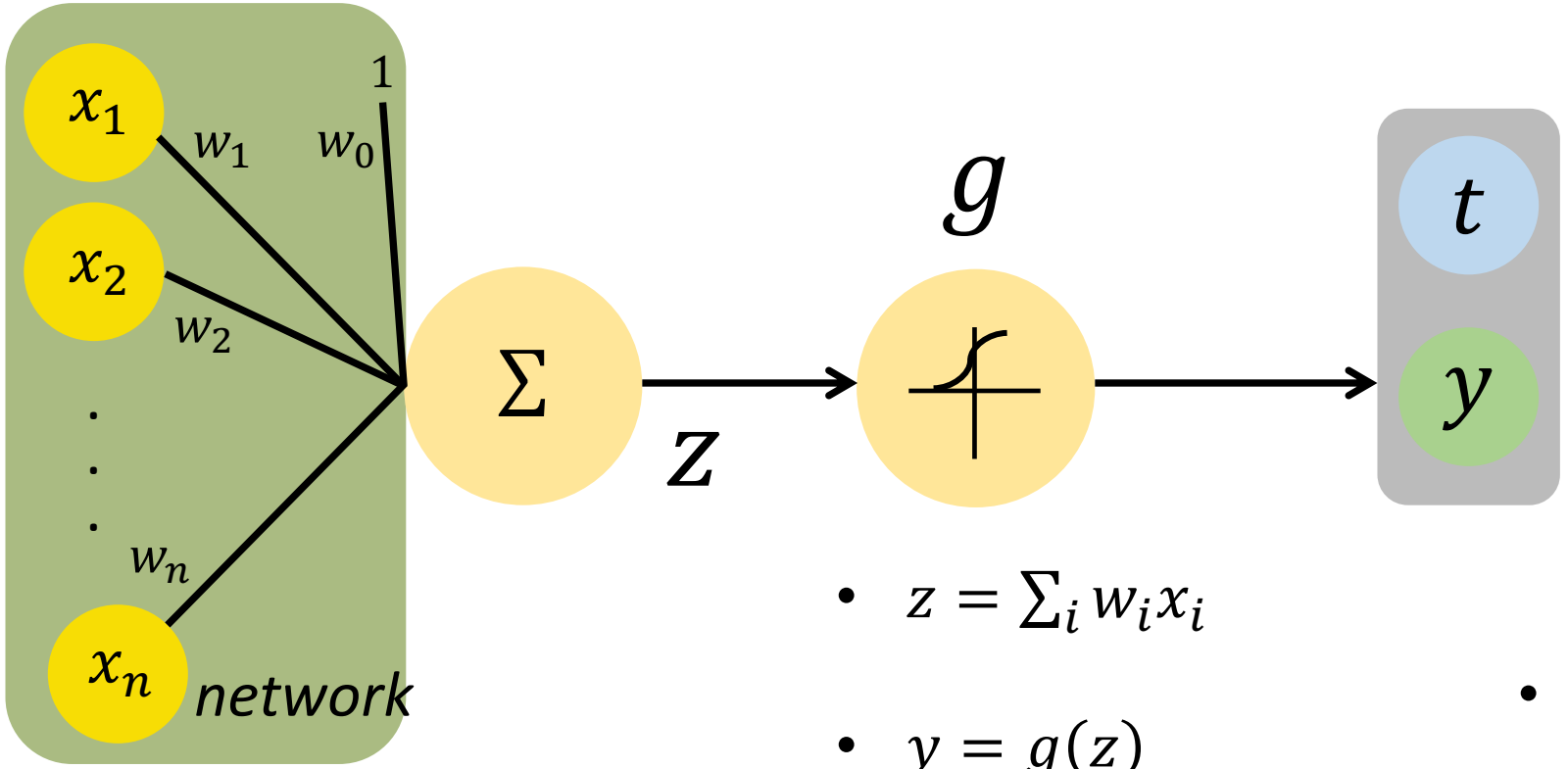
SGD for neural networks

SGD (Training data (x, t), learning rate η)

Initialize each w_i and b to some small random value
Until convergence
- Pick training example (x, t)
  - compute gradient $\boldsymbol{\nabla w}$ and $\frac{\partial L}{\partial b}$
  - update weight vector $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \boldsymbol{\nabla w}$
  - update bias $b \leftarrow b - \eta \frac{\partial L}{\partial b}$

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

Forward Computation For A Neuron



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

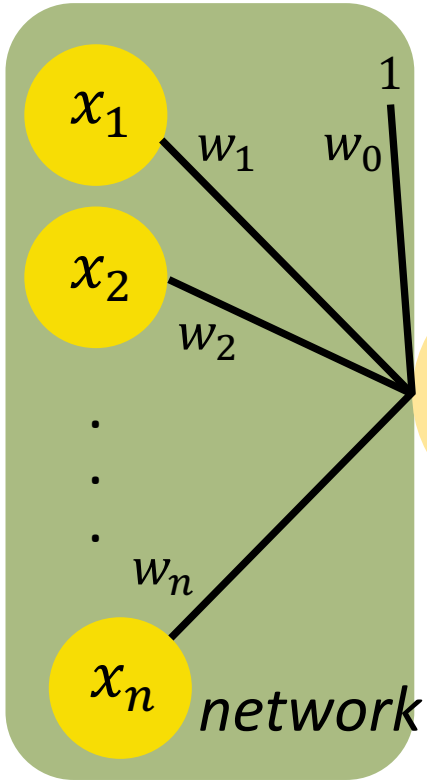
t: label

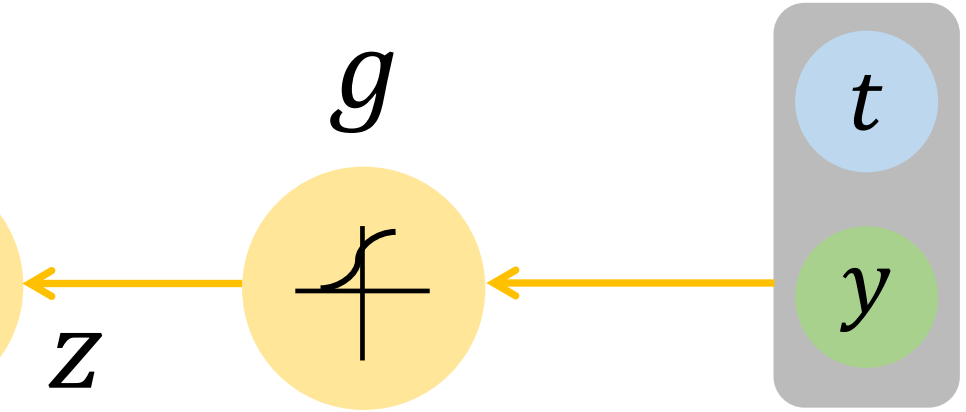
- $z = \sum_i w_i x_i$
- $y = g(z)$
  $= \dfrac{1}{1 + e^{-z}}$

- $\dfrac{\partial z}{\partial w_i} = x_i$
- $\dfrac{\partial y}{\partial z} = y(1 - y)$

ILLINOIS

Backward Computation For A Neuron





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

t: label

$$\frac{\partial L}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}||y - t||^2$$

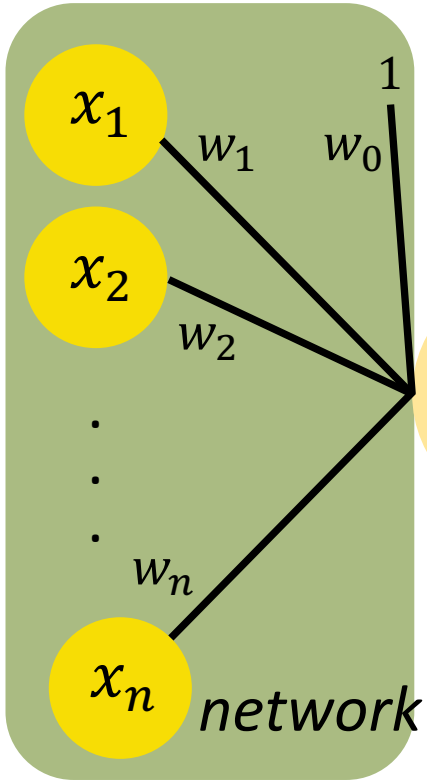
$$= \frac{\partial L}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial w_i}$$

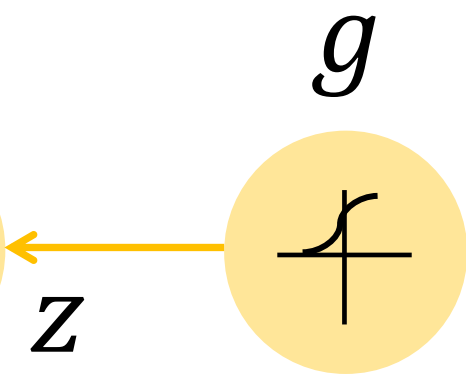
$$= (y - t)y(1 - y)\frac{\partial z}{\partial w_i}$$

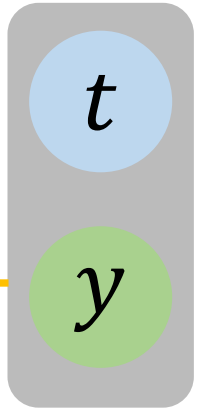
$$\frac{\partial L}{\partial w_i} = (y - t)y(1 - y)x_i$$

ILLINOIS

Backward Computation For A Neuron







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

t: label

$$\frac{\partial L}{\partial b} = \frac{\partial}{\partial b}\frac{1}{2}||y - t||^2$$

$$= \frac{\partial L}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial b}$$

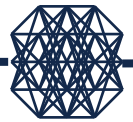
$$= (y - t)y(1 - y)\frac{\partial z}{\partial b}$$

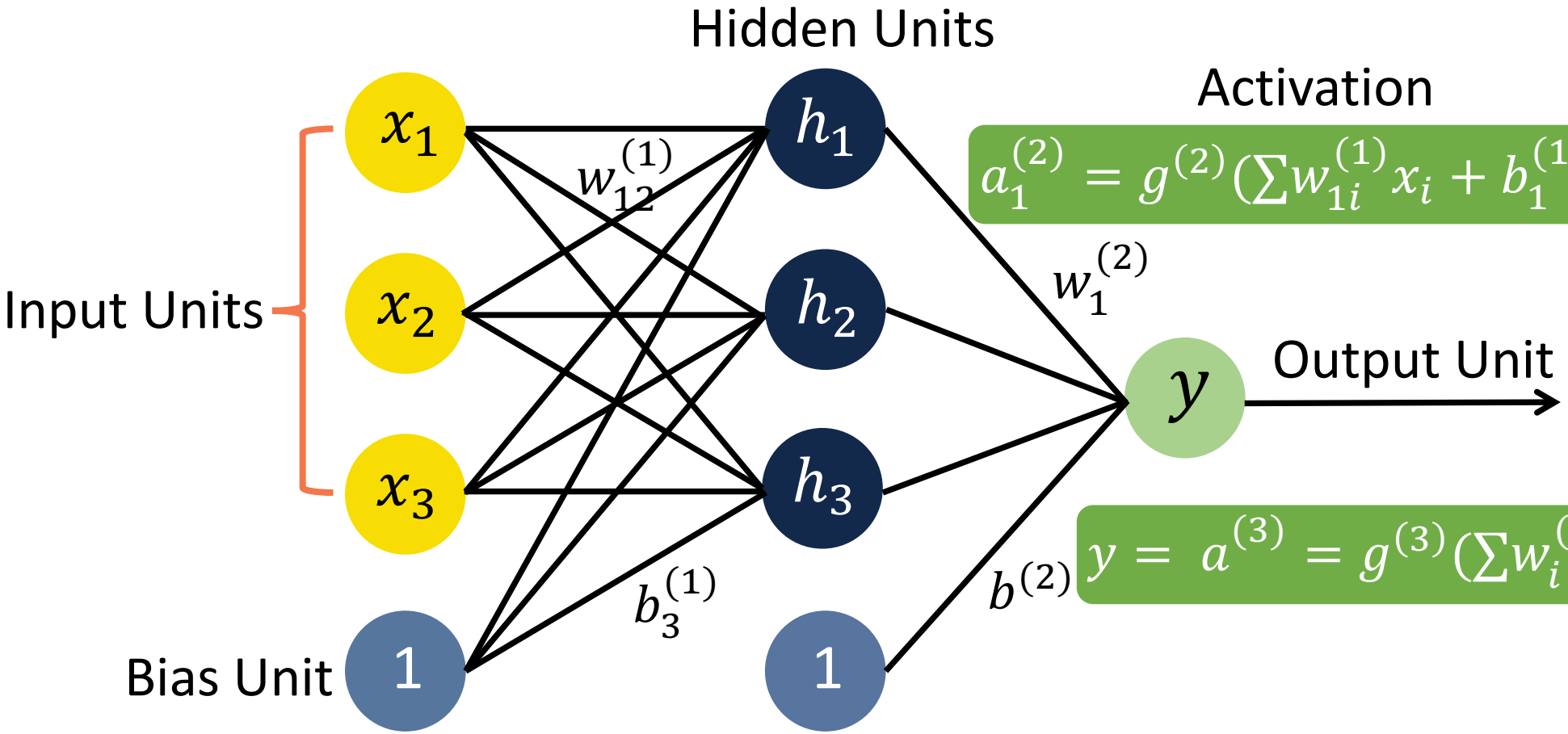
$$\rightarrow \quad \frac{\partial L}{\partial b} = (y - t)y(1 - y)$$

ILLINOIS

Multilayer Neural Networks

Multilayer Neural Network

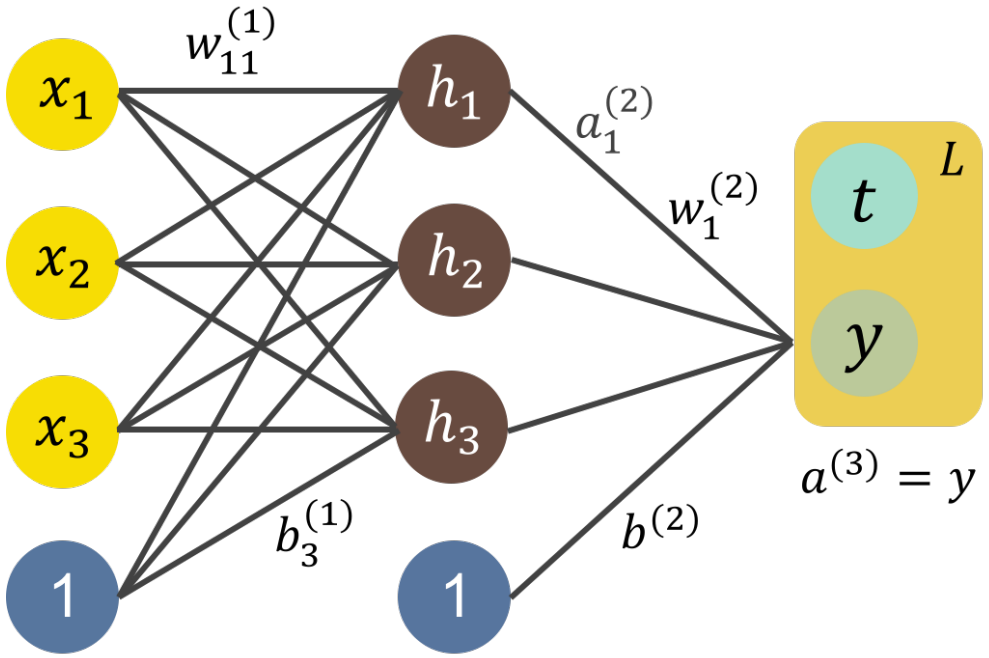




ILLINOIS

Train A Multilayer Neural Network

Given training sample (x, t)



$$L(\boldsymbol{W}, \boldsymbol{b}) = \frac{1}{2}\|y - t\|^2$$

SGD (Training data (x, t), learning rate η)

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

Until convergence

DO

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

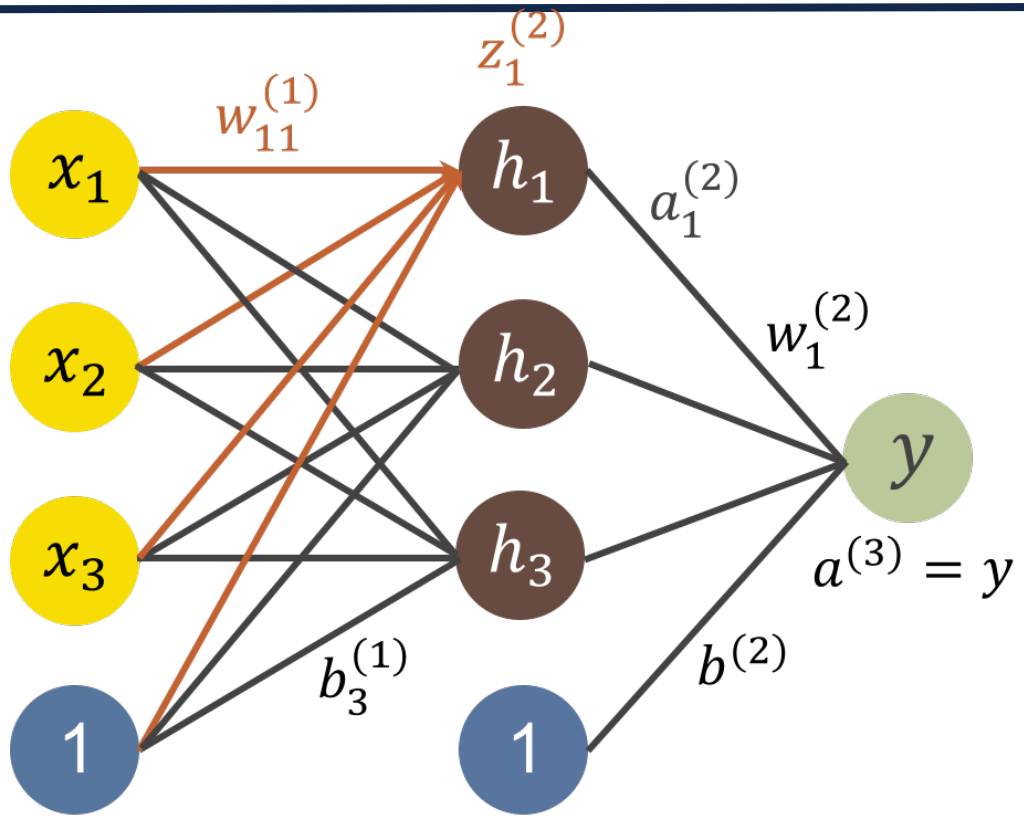
- compute $\frac{\partial L(\boldsymbol{W},\boldsymbol{b})}{\partial w_{ji}^{(l)}}$ and $\frac{\partial L(\boldsymbol{W},\boldsymbol{b})}{\partial b_j^{(l)}}$
- For each $w_{ji}^{(l)}$ and $b_j^{(l)}$:

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

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

ILLINOIS

Forward Computation: $z_1^{(2)}$

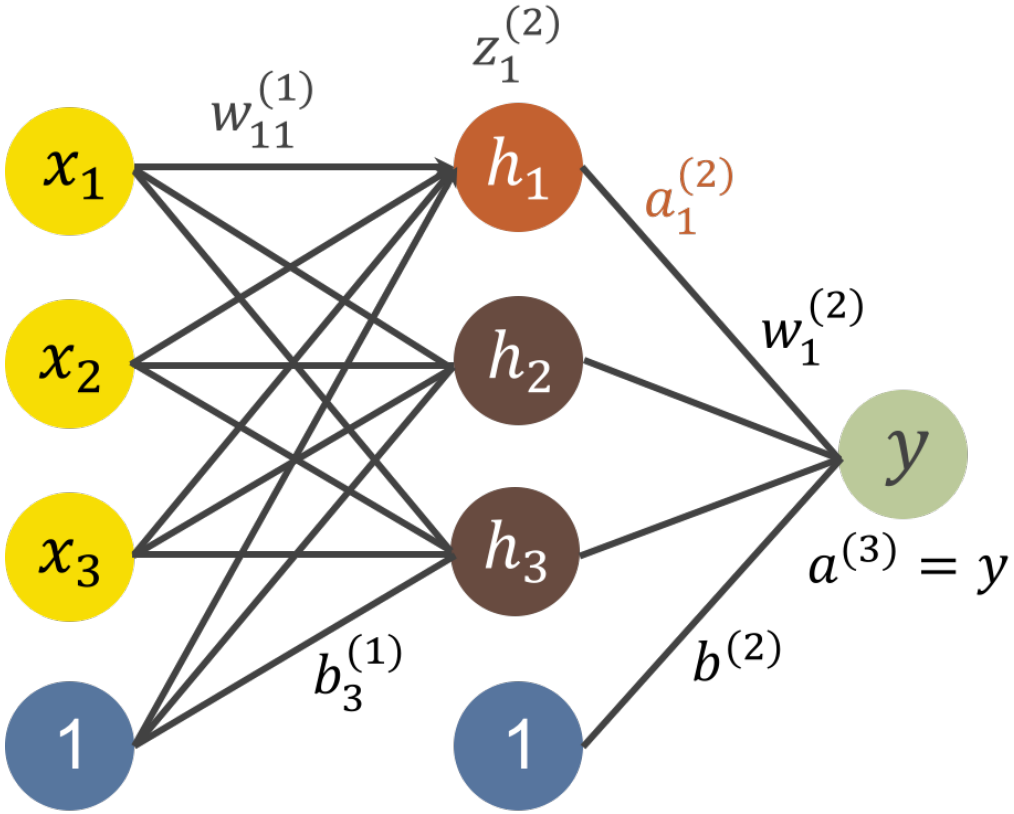


$$z_1^{(2)} = \sum w_{1i}^{(1)} x_i + b_1^{(1)}$$

ILLINOIS

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

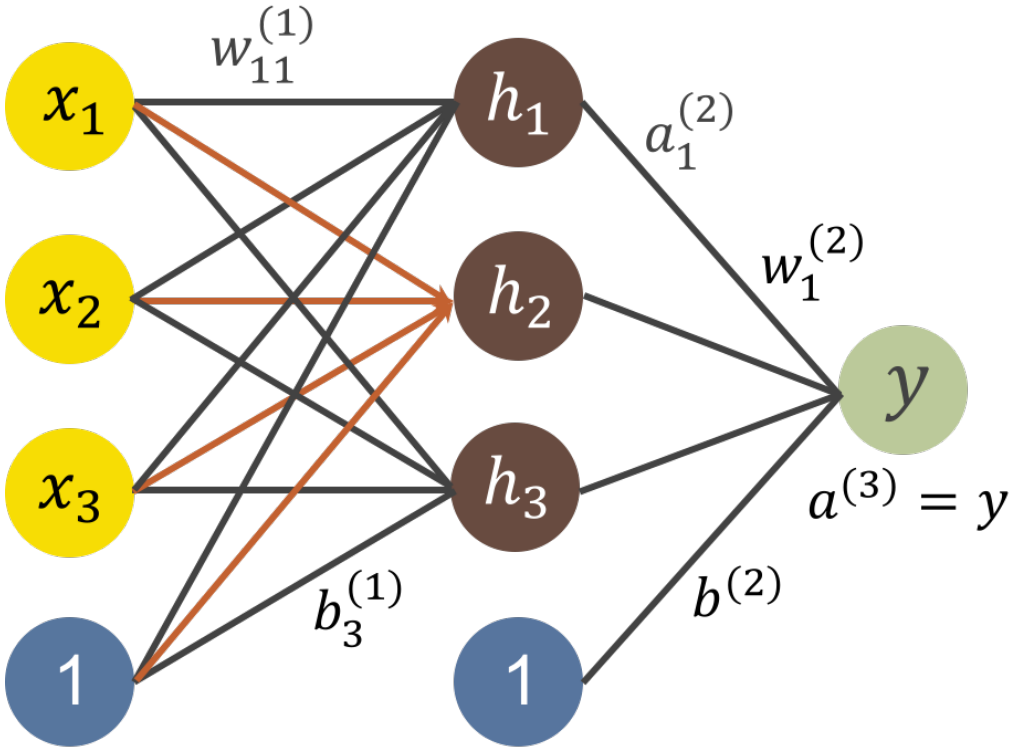
$$z_1^{(2)} = \sum w_{1i}^{(1)} x_i + b_1^{(1)}$$



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

ILLINOIS

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

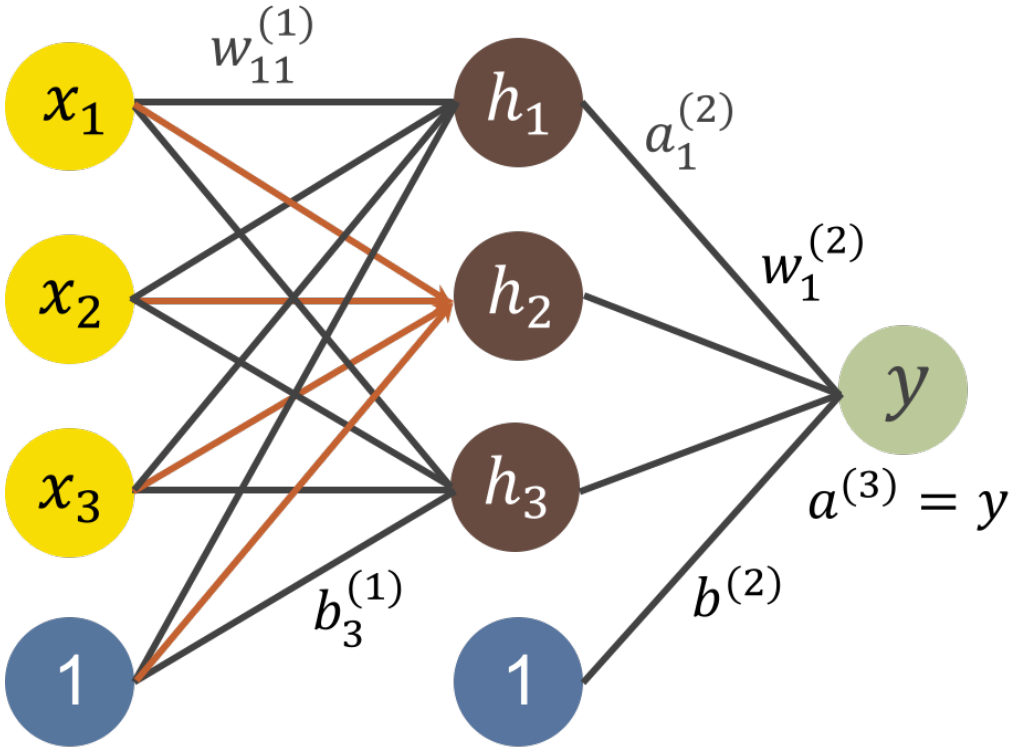


$$z_2^{(2)} = \sum w_{2i}^{(1)} x_i + b_2^{(1)}$$

ILLINOIS

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

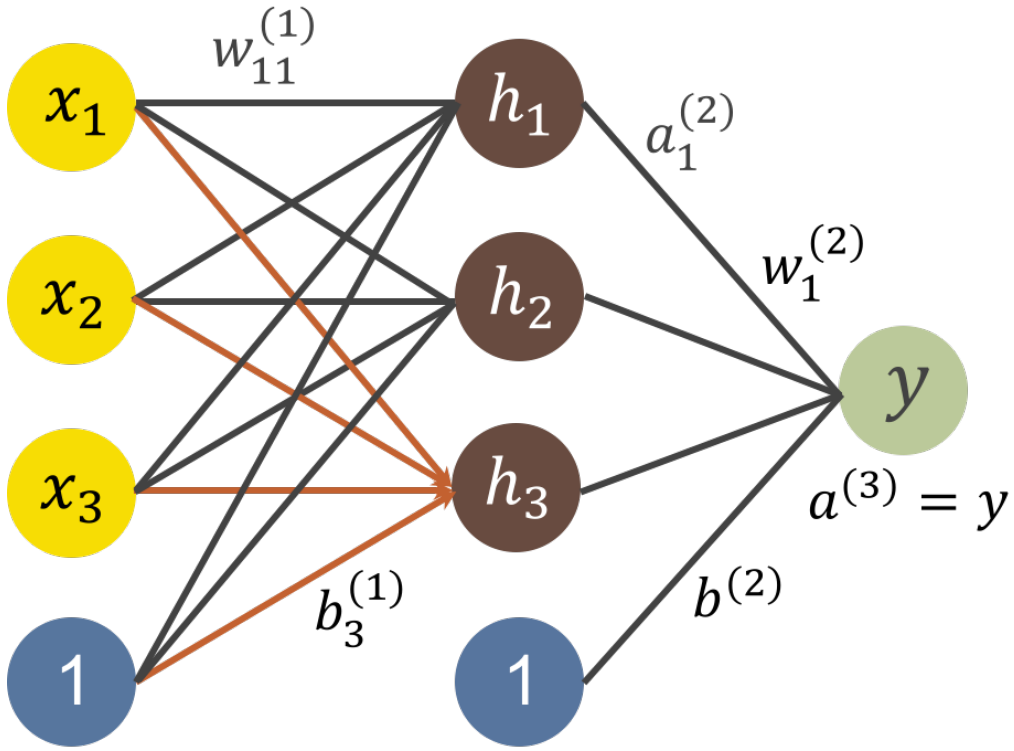
$$z_2^{(2)} = \sum w_{2i}^{(1)} x_i + b_2^{(1)}$$



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

ILLINOIS

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

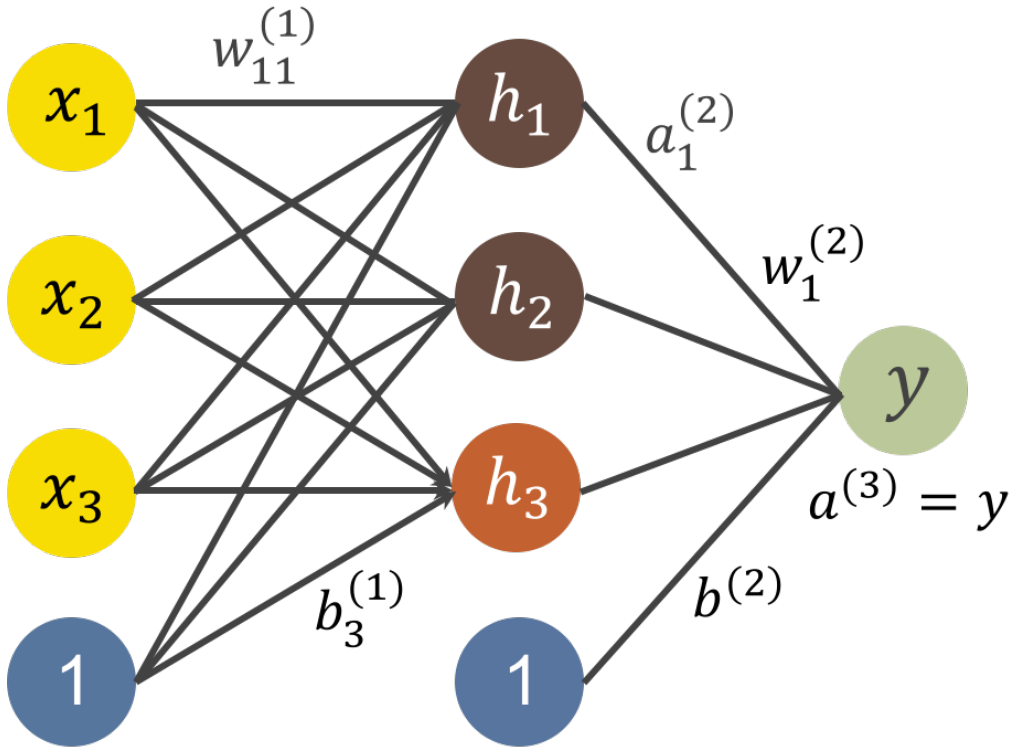


$$z_3^{(2)} = \sum w_{3i}^{(1)} x_i + b_3^{(1)}$$

ILLINOIS

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

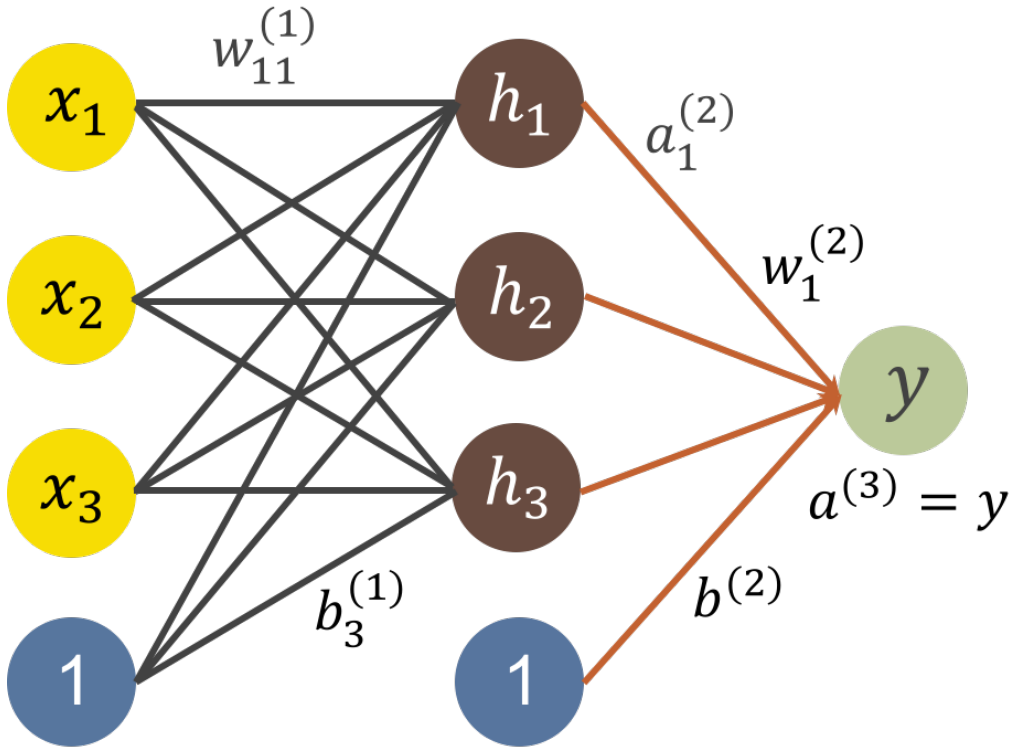
$$z_3^{(2)} = \sum w_{3i}^{(1)} x_i + b_3^{(1)}$$



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

ILLINOIS

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

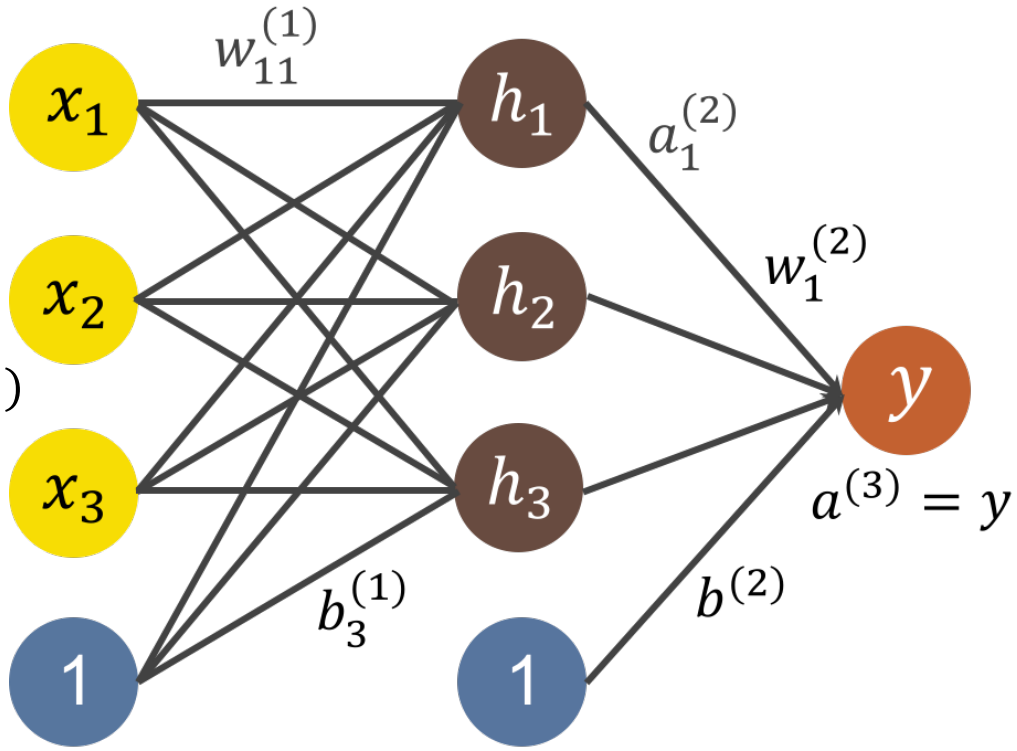


$$z^{(3)} = \sum w_j^{(2)} a_j^{(2)} + b^{(2)}$$

ILLINOIS

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

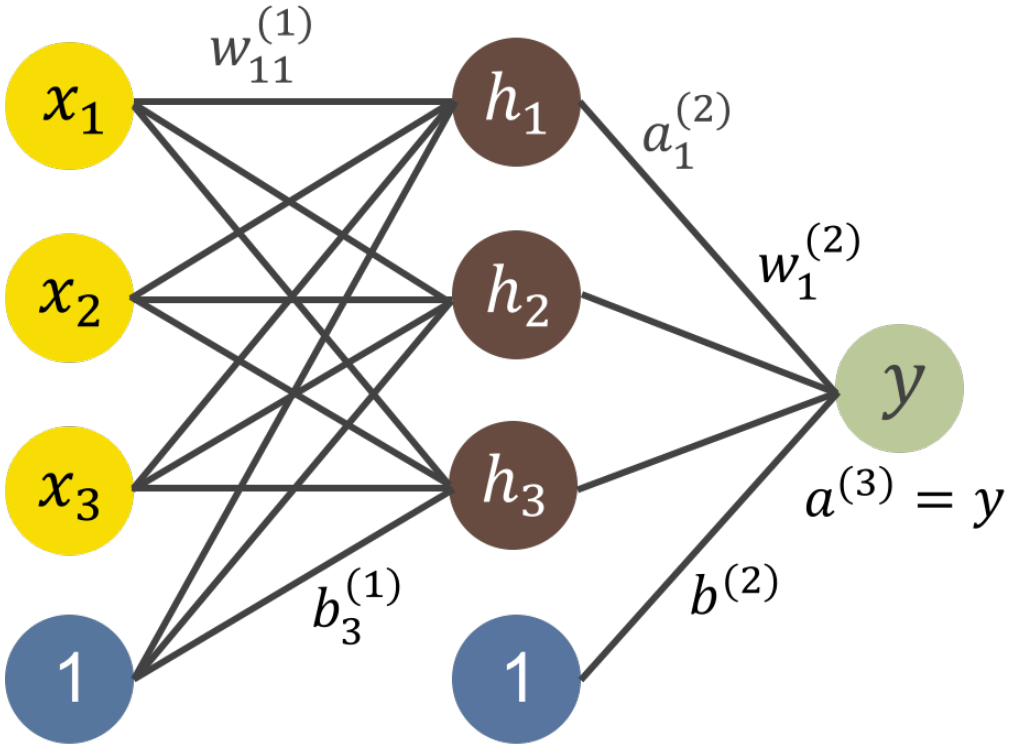
$$z^{(3)} = \sum w_j^{(2)} a_j^{(2)} + b^{(2)}$$



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

ILLINOIS

Forward Computation



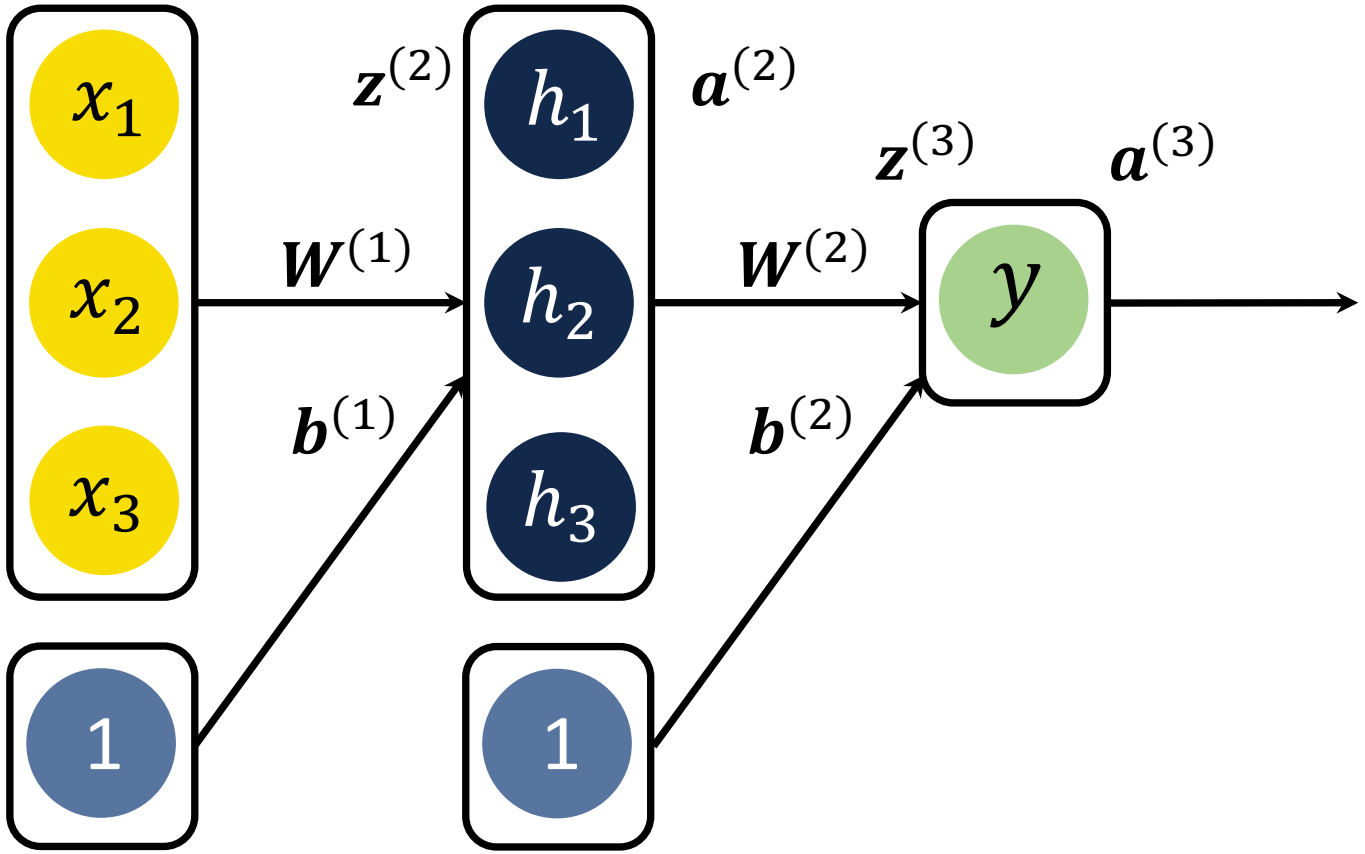
$$z_1^{(2)} = \sum w_{1i}^{(1)} x_i + b_1^{(1)}$$
$$z_2^{(2)} = \sum w_{2i}^{(1)} x_i + b_2^{(1)}$$
$$z_3^{(2)} = \sum w_{3i}^{(1)} x_i + b_3^{(1)}$$

$$a_1^{(2)} = g^{(2)}(z_1^{(2)})$$
$$a_2^{(2)} = g^{(2)}(z_2^{(2)})$$
$$a_3^{(2)} = g^{(2)}(z_3^{(2)})$$

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

ILLINOIS

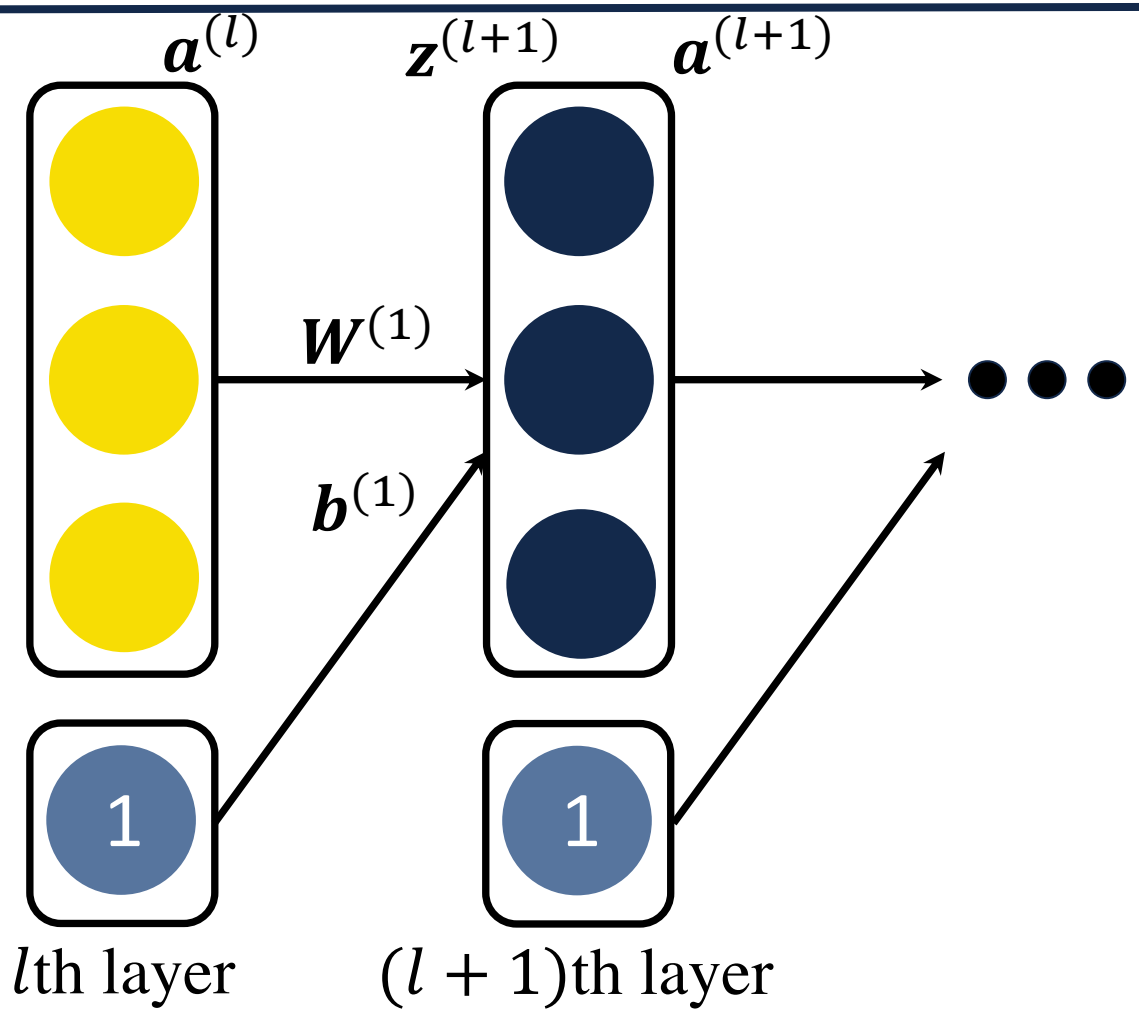
Forward Computation: Vector Form



$$z^{(2)} = W^{(1)}x + b^{(1)}$$
$$a^{(2)} = g^{(2)}(z^{(2)})$$
$$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$$
$$a^{(3)} = g^{(3)}(z^{(3)})$$

ILLINOIS

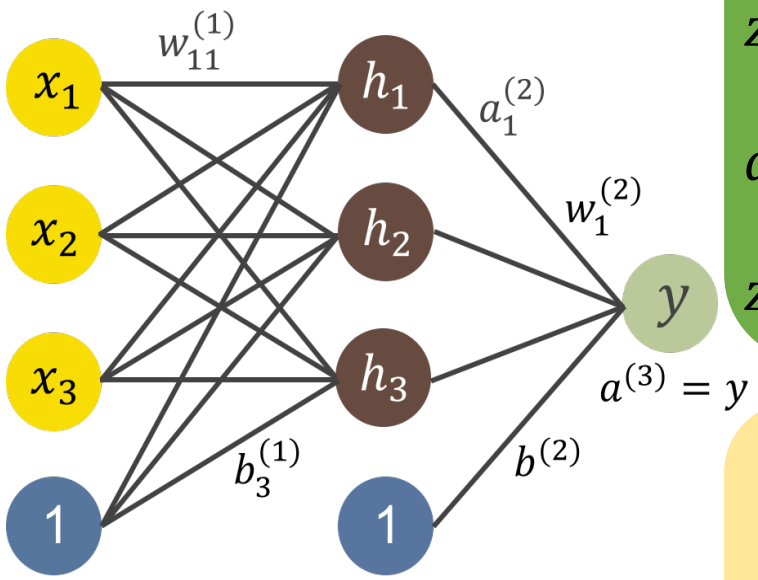
Forward Computation: General Form



$$z^{(l+1)} = W^{(l)} a^{(l)} + b^{(l)}$$
$$a^{(l+1)} = g^{(l+1)}(z^{(l+1)})$$

ILLINOIS

Forward Computation: Summary



Forward Computation of the Neural Network

$$z_1^{(2)} = \sum w_{1i}^{(1)} x_i + b_1^{(1)};\ z_2^{(2)} = \sum w_{2i}^{(1)} x_i + b_2^{(1)};\ z_3^{(2)} = \sum w_{3i}^{(1)} x_i + b_3^{(1)}$$

$$a_1^{(2)} = g^{(2)}(z_1^{(2)});\ a_2^{(2)} = g^{(2)}(z_2^{(2)});\ a_3^{(2)} = g^{(2)}(z_3^{(2)})$$

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

Forward flow

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

Vector Form

$$\boldsymbol{z}^{(2)} = \boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}$$
$$\boldsymbol{a}^{(2)} = g^{(2)}(\boldsymbol{z}^{(2)})$$
$$\boldsymbol{z}^{(3)} = \boldsymbol{W}^{(2)}\boldsymbol{a}^{(2)} + \boldsymbol{b}^{(2)}$$
$$a^{(3)} = g^{(3)}(\boldsymbol{z}^{(3)})$$

More General Form

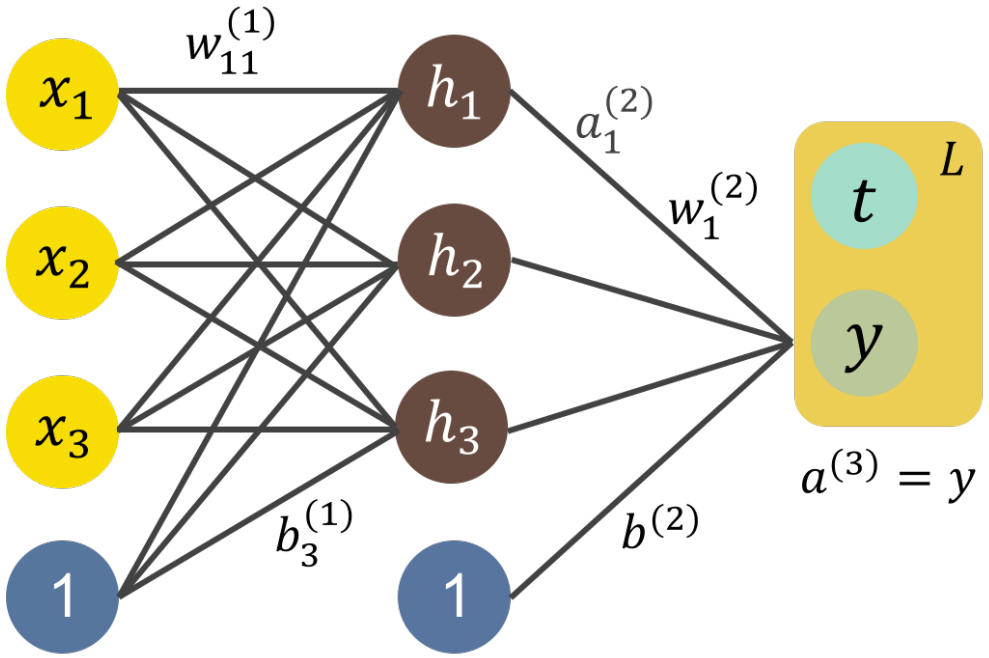
$$\boldsymbol{z}^{(l+1)} = \boldsymbol{W}^{(l)}\boldsymbol{a}^{(l)} + \boldsymbol{b}^{(l)}$$
$$\boldsymbol{a}^{(l+1)} = g^{(l+1)}(\boldsymbol{z}^{(l+1)})$$

ILLINOIS

Backpropagation Algorithm

Recap: SGD For Neural Networks

Given training sample (x, t)



$$L(\boldsymbol{W}, \boldsymbol{b}) = \frac{1}{2}\|y - t\|^2$$

SGD (Training data (x, t), learning rate η)

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

Until convergence

DO

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

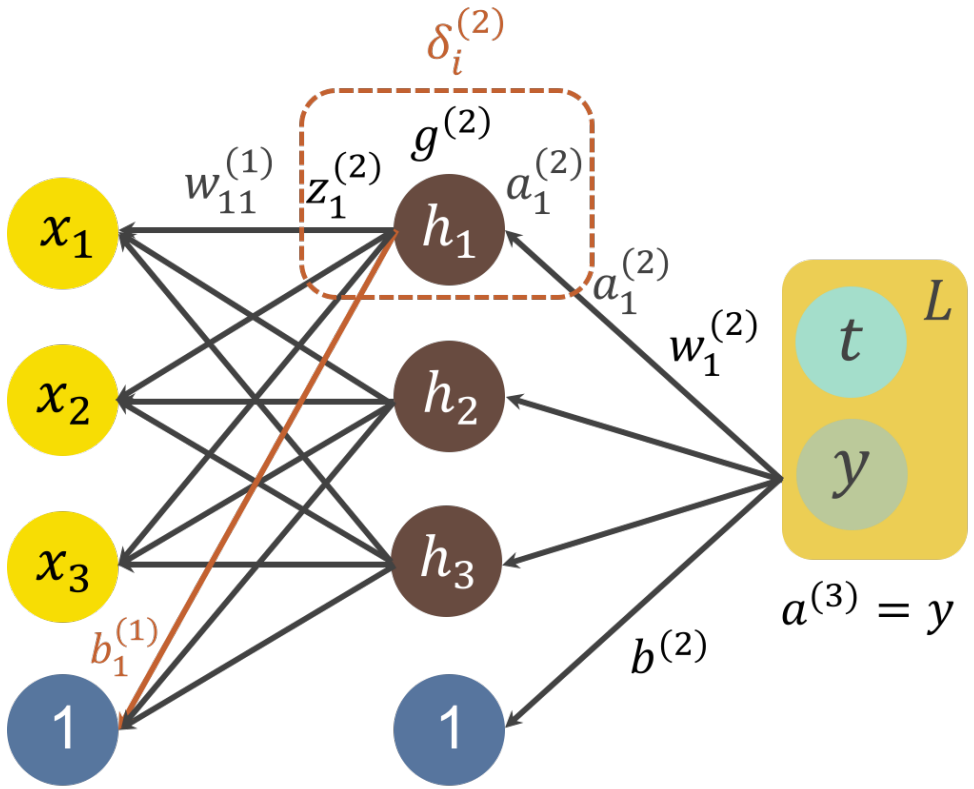
- compute $\frac{\partial L(\boldsymbol{W},\boldsymbol{b})}{\partial w_{ji}^{(l)}}$ and $\frac{\partial L(\boldsymbol{W},\boldsymbol{b})}{\partial b_j^{(l)}}$
- For each $w_{ji}^{(l)}$ and $b_j^{(l)}$:

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

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

ILLINOIS

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



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

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

Derivation :

$$\frac{\partial L}{\partial w_{ji}^{(l)}} = \frac{\partial L}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial w_{ji}^{(l)}}$$

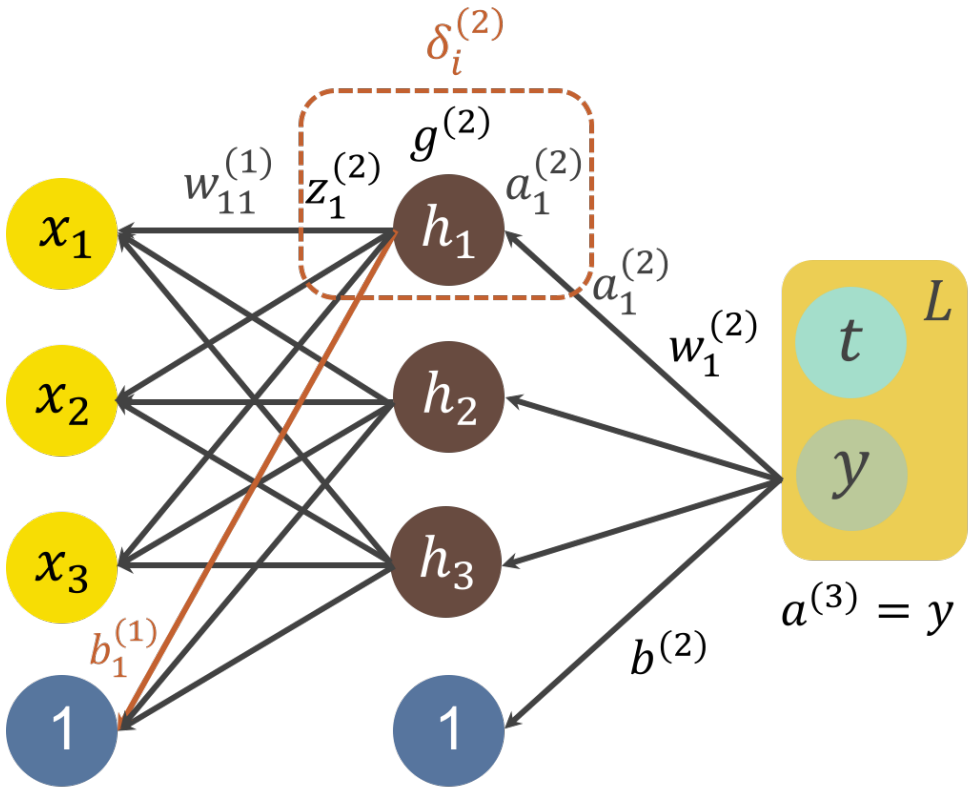
$$= \delta_j^{(l)} \frac{\partial (\sum_s w_{js}^{(l)} a_s^{(l-1)} + b_s^{(l)})}{\partial w_{ji}^{(l)}}$$

$$= \delta_j^{(l)} a_i^{(l-1)}$$

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

ILLINOIS

Backward Propagation: $\frac{\partial L(\boldsymbol{W}, \boldsymbol{b})}{\partial b_j^{(l)}}$



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

Derivation :

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

$$= \delta_j^{(l)} \frac{\partial (\sum_s w_{js}^{(l)} a_s^{(l-1)} + b_s^{(l)})}{\partial b_j^{(l)}}$$

$$= \delta_j^{(l)}$$

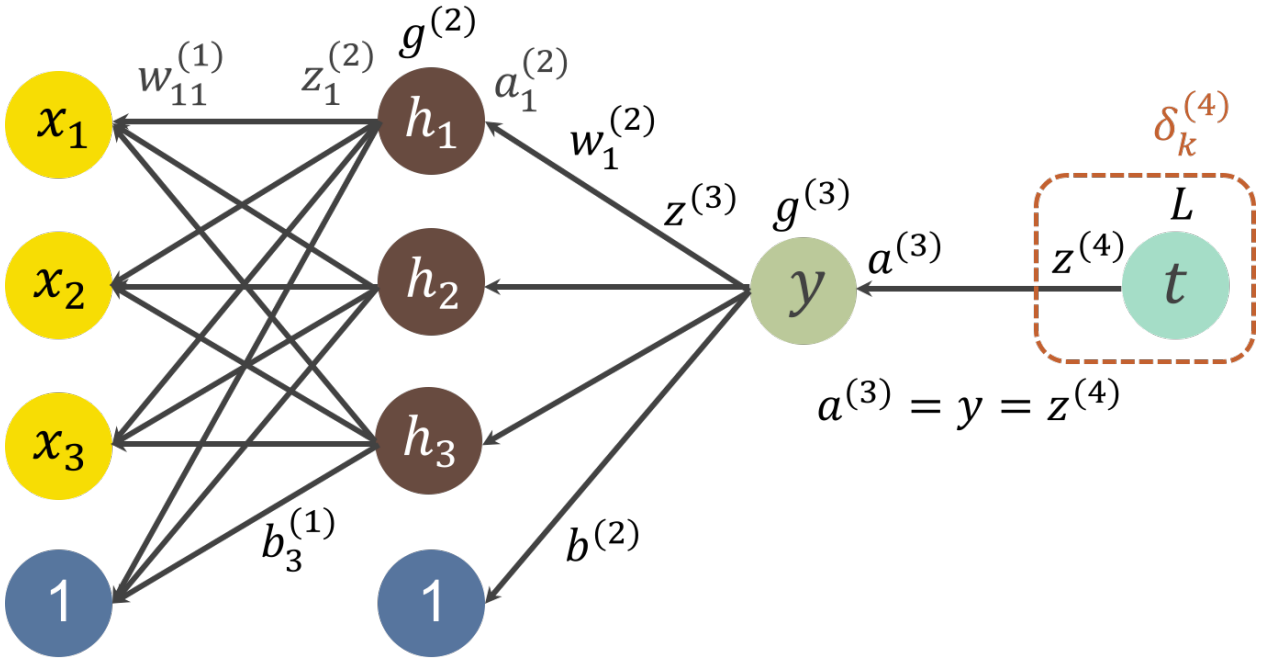
ILLINOIS

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

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

→ 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 $\delta_k^{(4)}$

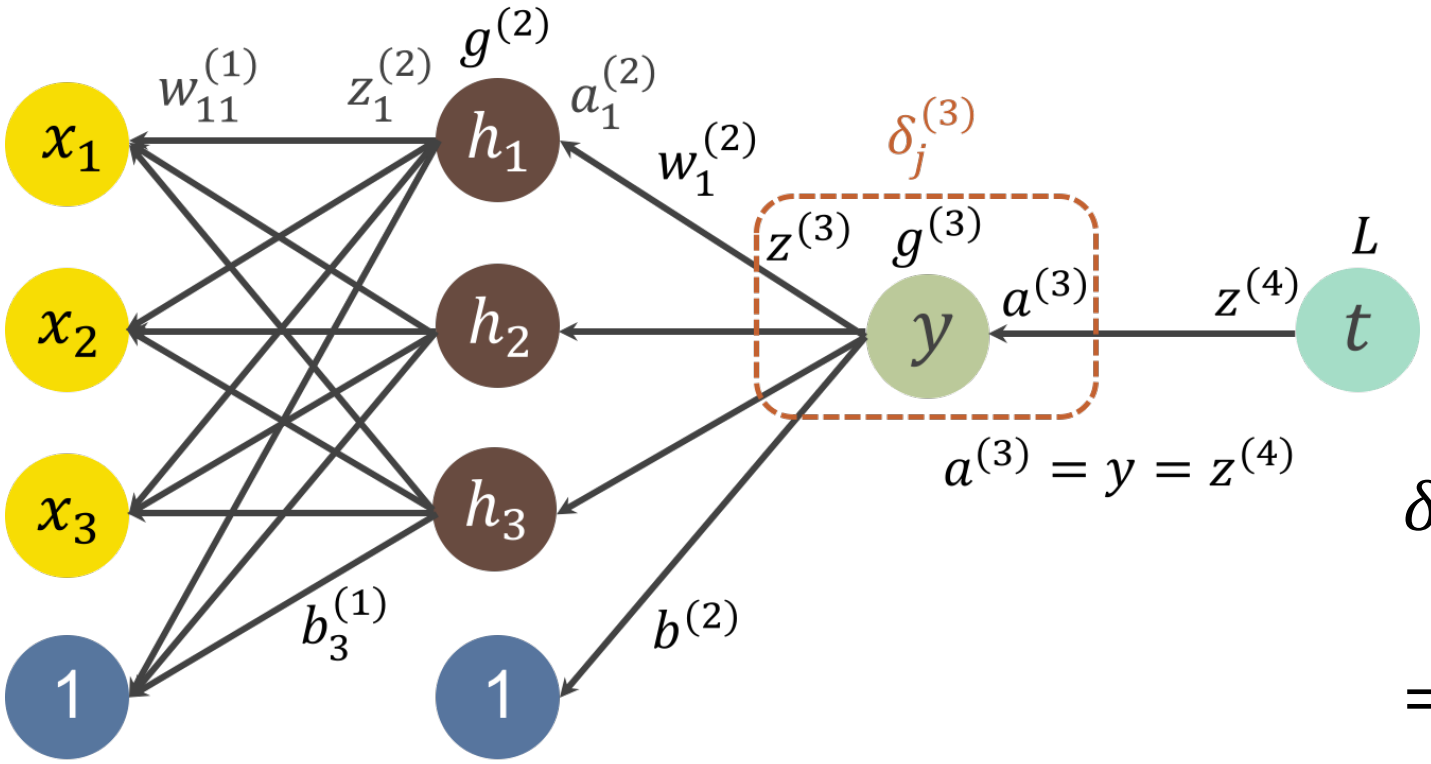


$$\delta_k^{(4)} = \frac{\partial L}{\partial z_k^{(4)}} = \frac{\partial \frac{1}{2}\|y - t\|^2}{\partial y}$$

$$= -(\mathrm{t} - \mathrm{y})$$

ILLINOIS

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



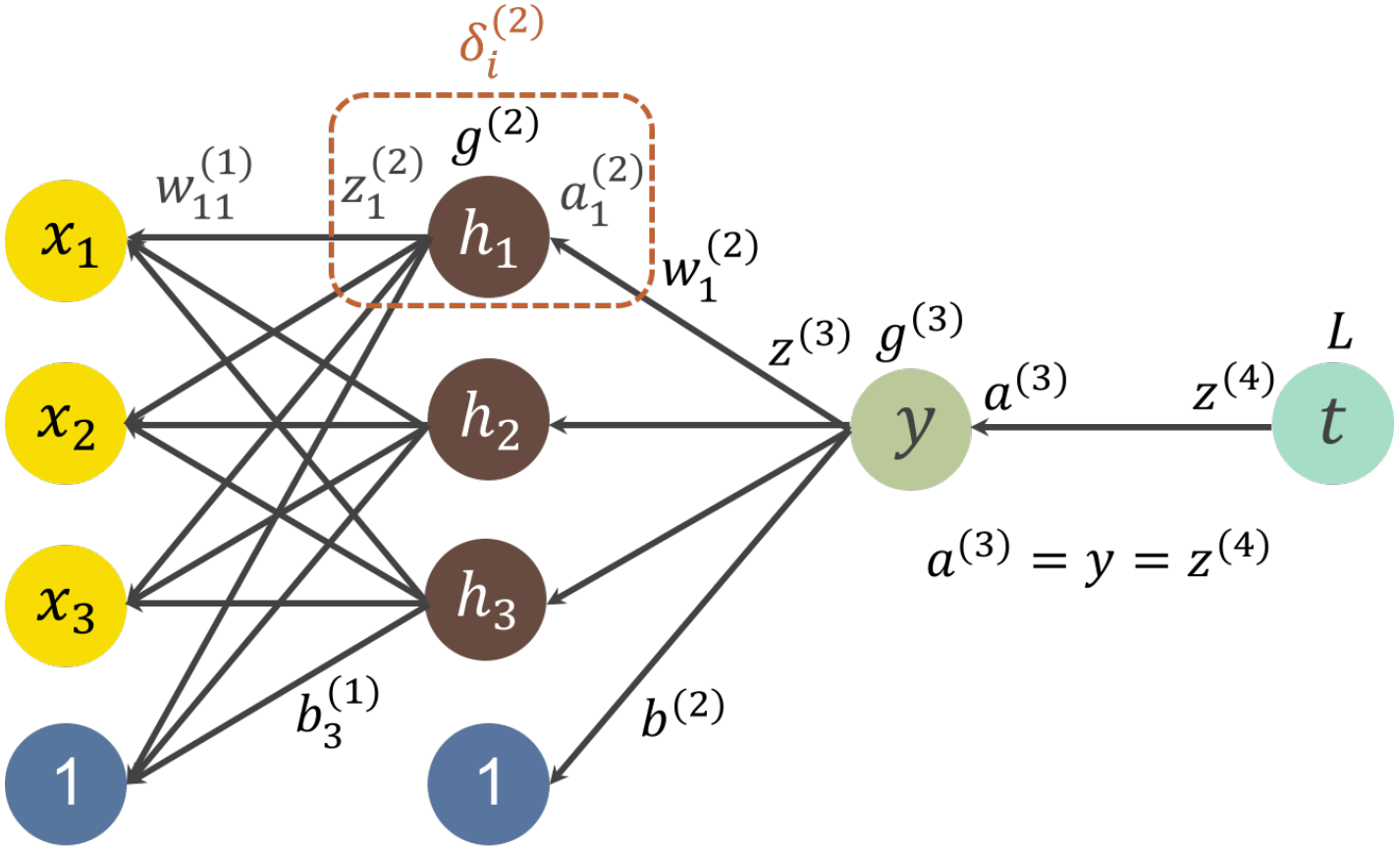
$$\delta_j^{(3)} = \frac{\partial L}{\partial z_j^{(3)}} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z_j^{(3)}}$$

$$= \delta^{(4)}(g^{(3)})'(z_j^{(3)})$$

$$= -(\mathrm{t} - \mathrm{y})(g^{(3)})'(z_j^{(3)})$$

ILLINOIS

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



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

$$\delta_i^{(2)} = \frac{\partial L}{\partial z_i^{(2)}} = \frac{\partial L}{\partial z_j^{(3)}} \frac{\partial z_j^{(3)}}{\partial z_i^{(2)}}$$

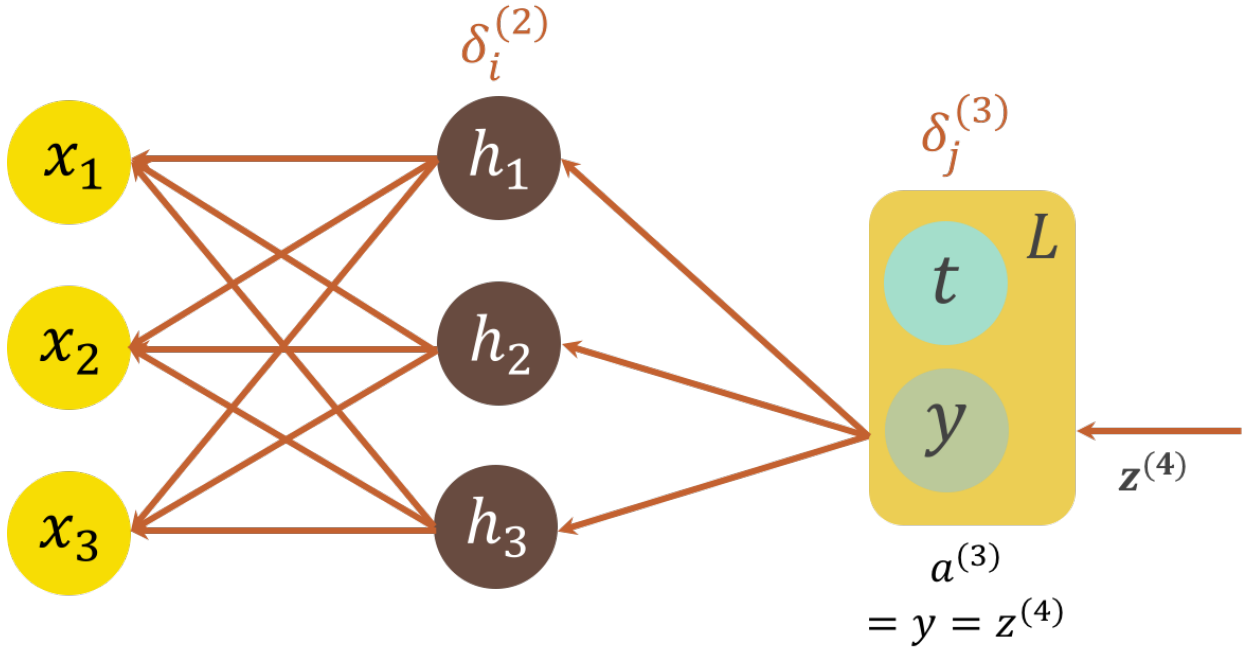
$$= \delta^{(3)} \frac{\partial \sum_{i=1}^{3} w_i^{(2)} a_i^{(2)} + b^{(2)}}{\partial a_i^{(2)}} \frac{\partial a_i^{(2)}}{\partial z_i^{(2)}}$$

$$= \delta^{(3)} w_i^{(2)} \frac{\partial a_i^{(2)}}{\partial z_i^{(2)}}$$

$$= \delta^{(3)} w_i^{(2)} (g^{(2)})'(z_i^{(2)})$$

ILLINOIS

Backward Propagation: Summary



$$\delta_k^{(4)} = -(\mathrm{t} - \mathrm{y})$$

$$\delta_j^{(3)} = \delta^{(4)}(g^{(3)})'(z_j^{(3)}) = -(\mathrm{t} - \mathrm{y})(g^{(3)})'(z_j^{(3)})$$

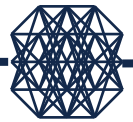
$$\delta_i^{(2)} = (g^{(2)})'(z_i^{(2)})w_i^{(2)}\delta^{(3)}$$

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

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

ILLINOIS

Backpropagation Algorithm



The FORWARD pass

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

The BACKWARD pass

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

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

ILLINOIS

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

A comparison of models for predicting early hospital readmissions

Joseph Futoma [a], Jonathan Morris [b], Joseph Lucas [a,c,*]

[a] Dept. of Statistical Science, Duke University, Box 90251, Durham, NC 27708, USA
[b] Quintiles, 4820 Emperor Blvd., Durham, NC 27703, USA
[c] 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

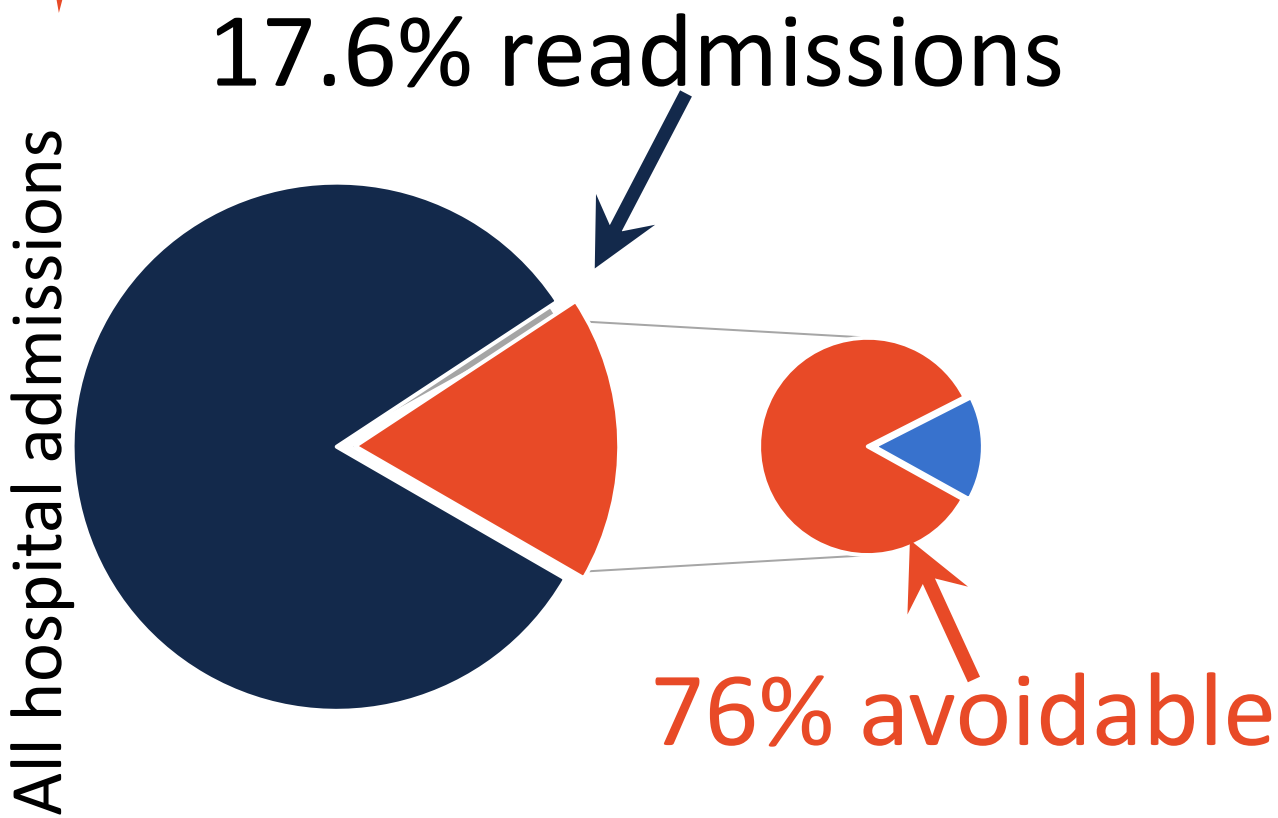
Keywords:
Electronic Health Records
Early readmission
Penalized methods
Random forest
Deep learning
Predictive models

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.

ILLINOIS

Hospital Readmission is a big problem



$15 billion in Medicare spending on readmission

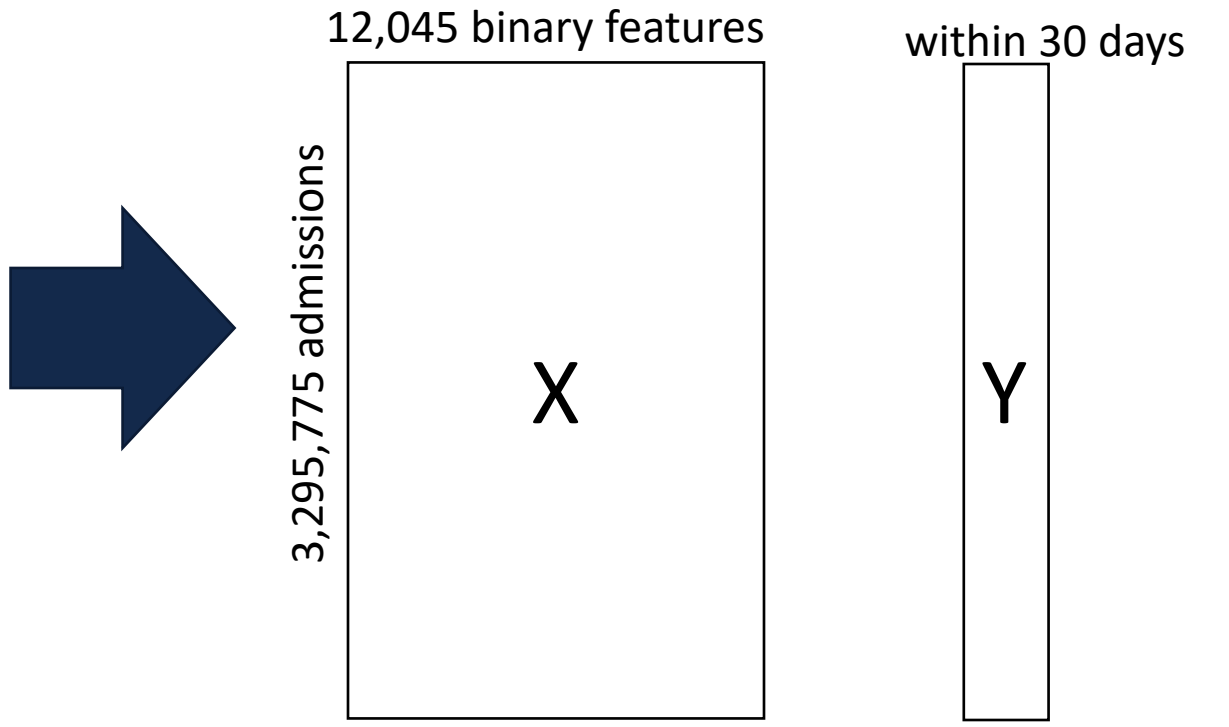
Over 2000 hospitals received
$280 million penalty

ILLINOIS

New Zealand National Minimum Dataset

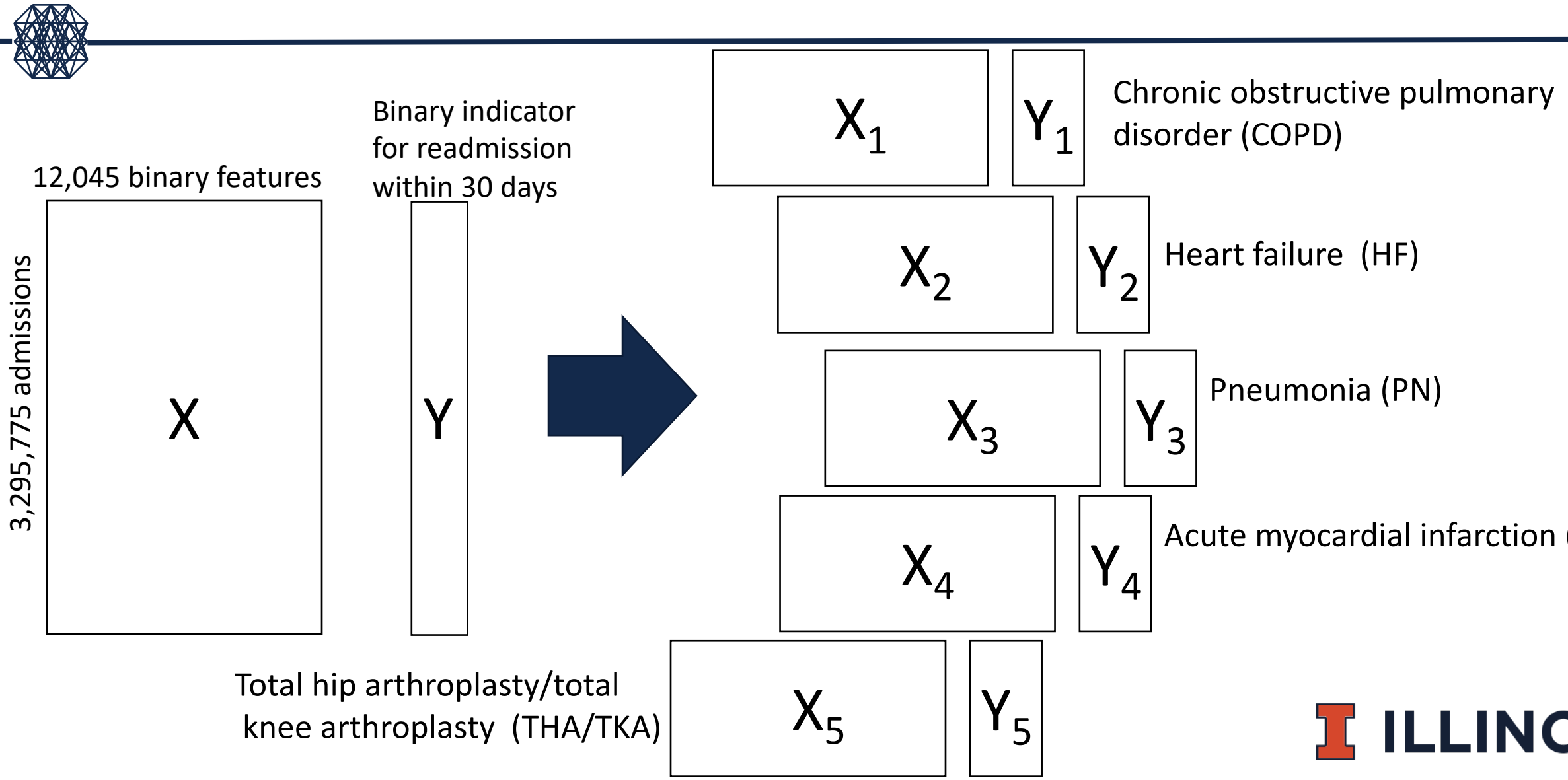
Full dataset, post-processing.

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)



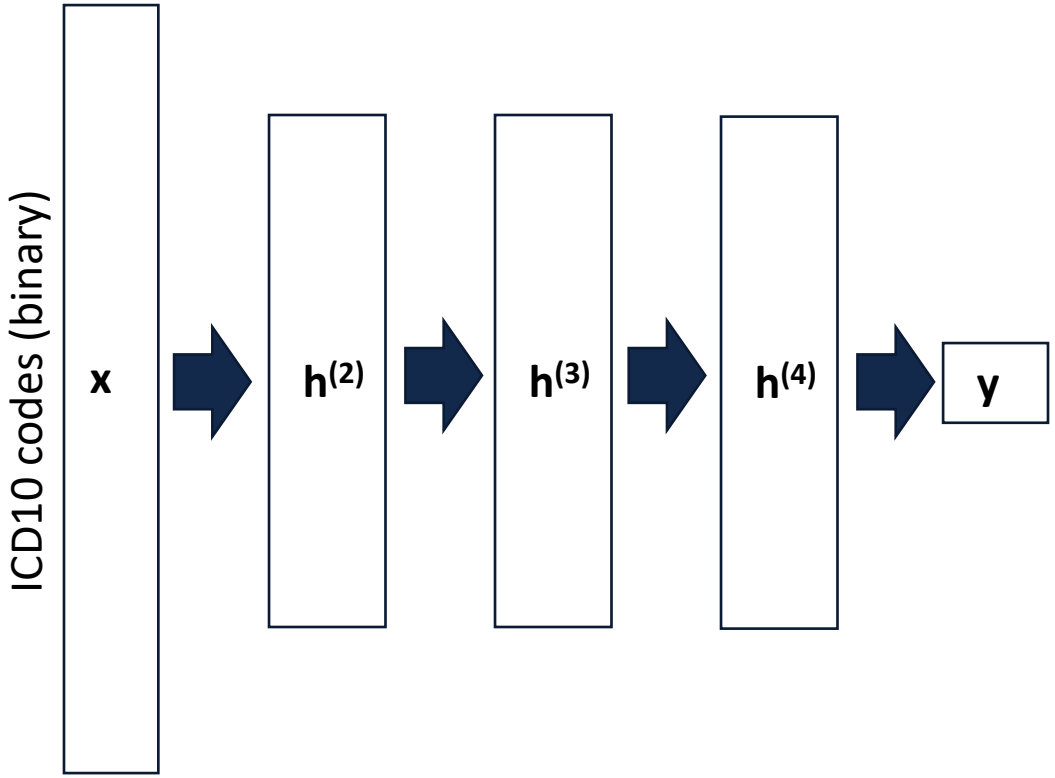
ILLINOIS

Cohort construction



ILLINOIS

DNN model



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

ILLINOIS

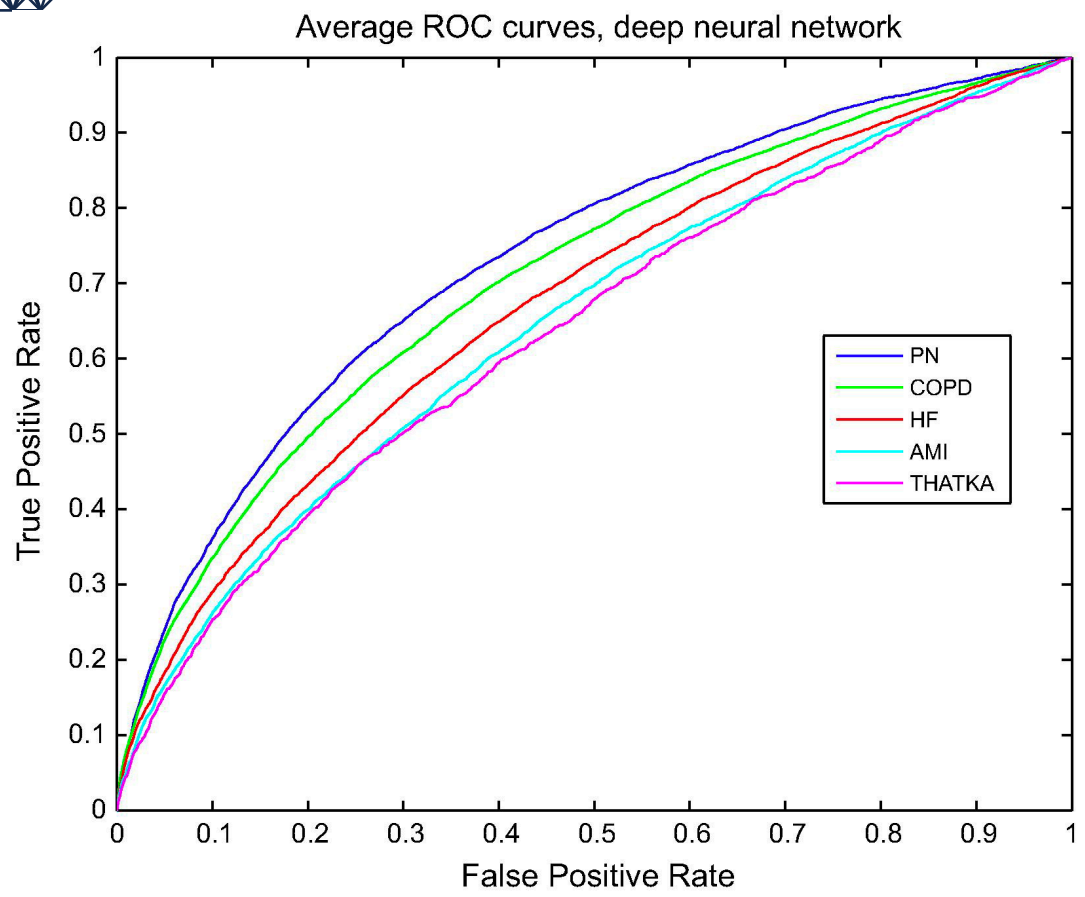
Comparison between logistic regression and DNN

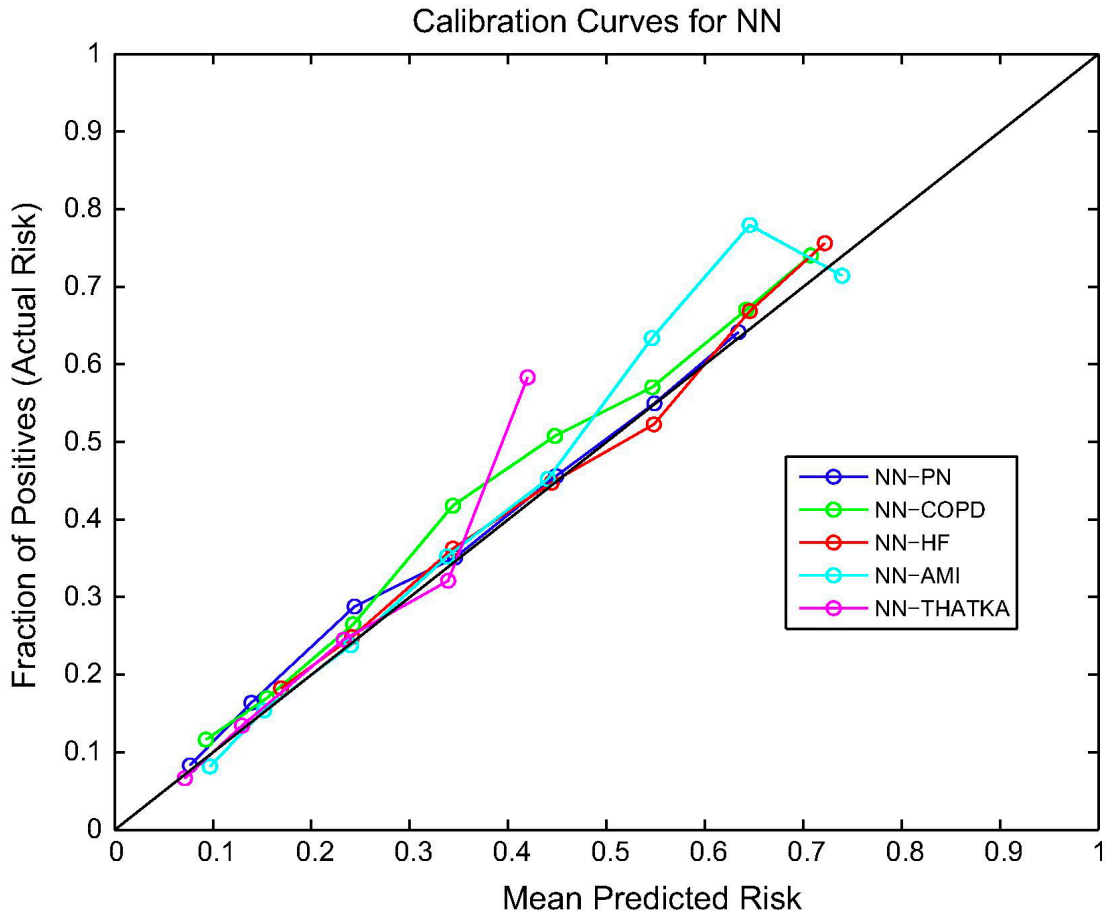
Condition	Size	Readmission rate (%)	Logistic regression	DNN
PN	40,442	27.9	0.715 (0.005)	**0.734 (0.004)**
COPD	31,457	20.4	0.703 (0.003)	**0.711 (0.003)**
HF	25,941	19.0	0.654 (0.005)	**0.676 (0.005)**
AMI	29,060	29.5	**0.633 (0.006)**	**0.649 (0.007)**
THA/TKA	23,128	8.7	**0.629 (0.005)**	**0.638 (0.006)**

- DNN consistently had better AUC than logistic regression

ILLINOIS

Result





- DNN achieves reasonable AUC and great calibration

ILLINOIS

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

ILLINOIS

2012

MERCK
Be well

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

Prizes

Evaluation

Visualization-Prospect

Submission-Instructions

Winners

Help enable the development of safe, effective medicines.

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

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.

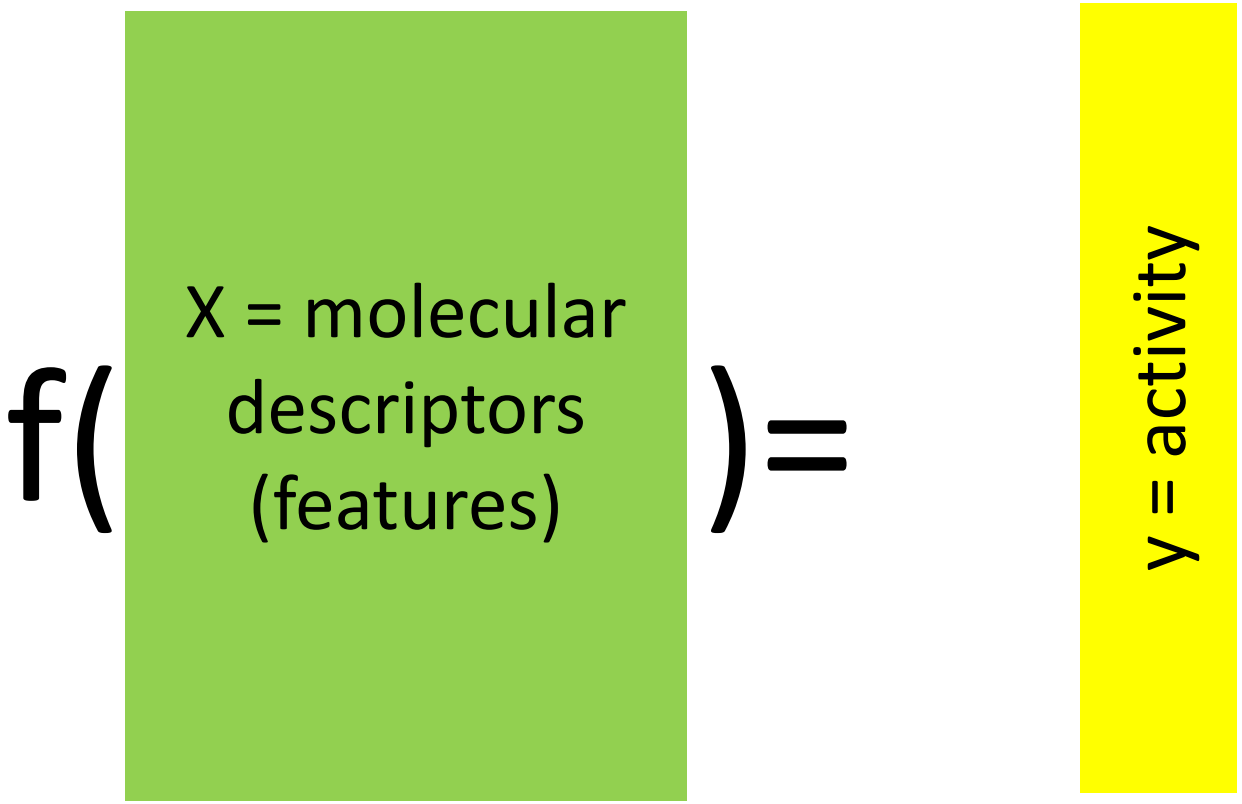
In addition to the prediction competition, Merck is also hosting a visualization challenge with a $2,000 prize for the most insightful and elegant graphical representations of the data.

Prizes total **$40,000**.

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.

ILLINOIS

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.

ILLINOIS

Datasets & tasks

15 tasks/datasets (Kaggle competition)

+ **15** additional datasets

Largest dataset has **318,795** molecules

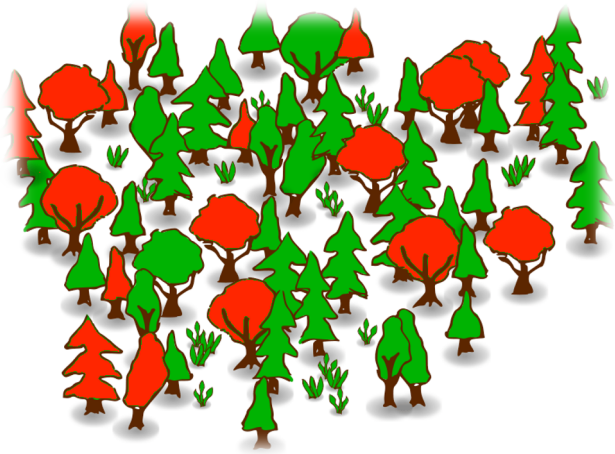
and **12,508** descriptors/features

data set	type	description	number of molecules	number of unique AP, DP descriptors
		Kaggle Data Sets		
3A4	ADME	CYP P450 3A4 inhibition −log(IC50) M	50000	9491
CB1	target	binding to cannabinoid receptor 1 −log(IC50) M	11640	5877
DPP4	target	inhibition of dipeptidyl peptidase 4 −log(IC50) M	8327	5203
HIVINT	target	inhibition of HIV integrase in a cell based assay −log(IC50) M	2421	4306
HIVPROT	target	inhibition of HIV protease −log(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(IC50) M	13482	5803
		n 1 receptor −log(K_i) M	7135	4730
		n 2 receptor −log(K_i) M	14875	5790
		coprotein log(BA/AB)	8603	5135
		tein binding log(bound/unbound)	11622	5470
		lity) at 2 mg/kg	7821	5698
		A4 inhibitions log(IC50 without NADPH/IC50 with	5559	5945
		nhibition −log(IC50) M	6924	5552
		Additional Data Sets		
2C8	ADME	CYP P450 2C8 inhibition −log(IC50) M	29958	8217
2C9	ADME	CYP P450 2C9 inhibition −log(IC50) M	189670	11730
		(IC50) M	50000	9729
		tor −log(IC50) M	2763	5242
		g(IC50) M	17469	6200
		l	50000	8959
		log(clearance) μL/min·mg	23292	6782
		(IC50) M	12843	6596
		50) M	9536	6136
		itions log(solubility) mol/L	89531	9541
HERG	ADME	inhibition of hERG channel −log(IC50) M	50000	9388
HERG (full data set)	ADME	inhibition of hERG ion channel −log(IC50) M	318795	12508
NAV	ADME	inhibition of Nav1.5 ion channel −log(IC50) M	50000	8302
PAPP	ADME	apparent passive permeability in PK1 cells log(permeability) cm/s	30938	7713
PXR	ADME	induction of 3A4 by pregnane X receptor; percentage relative to rifampicin	50000	9282

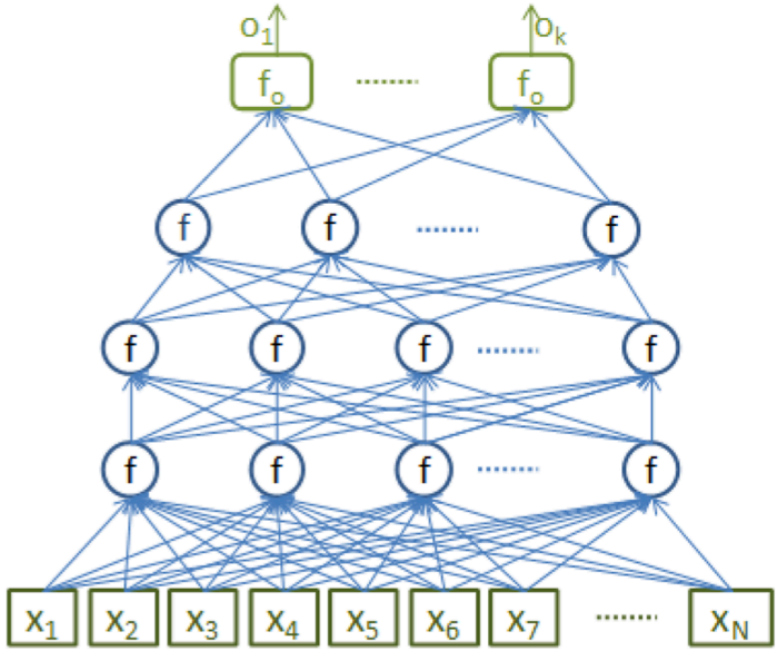
ILLINOIS

Methods

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

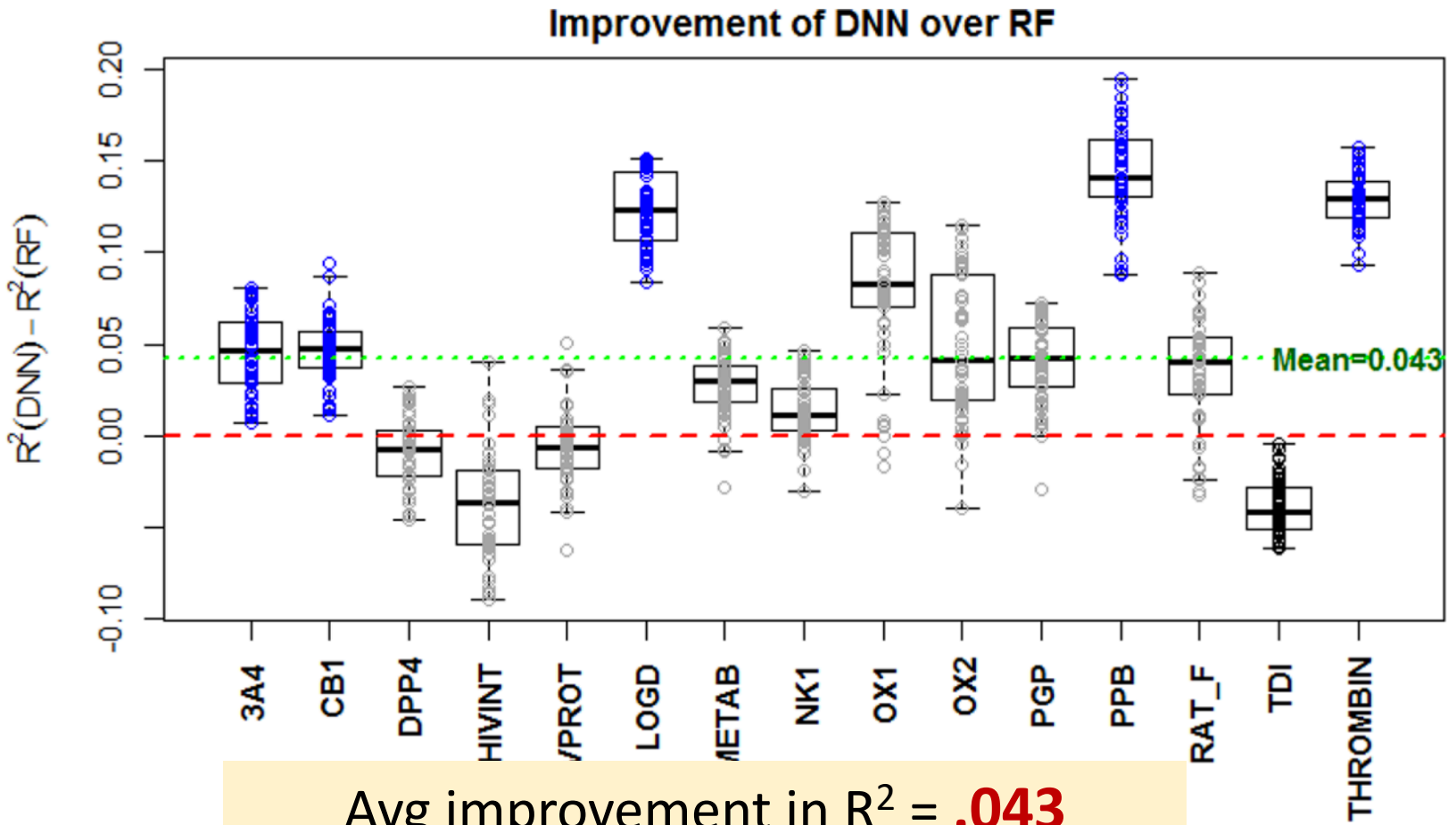


- Fully connected neural networks with 1 or 2 hidden layers



ILLINOIS

Results on Kaggle competition data



Avg improvement in R^2 = **.043**

RF = .423

DNN = .496

ILLINOIS

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	**0.255**
2C9BIG	0.279	**0.363**
2D6	0.130	**0.195**
A-II	0.805	**0.812**
BACE	0.629	**0.644**
CAV	0.399	**0.463**
CLINT	0.393	**0.554**
ERK2	**0.257**	0.198
FACTORXIA	0.241	**0.244**
FASSIF	**0.294**	0.271
HERG	0.305	**0.352**
HERGfull	0.294	**0.367**
NAV	0.277	**0.347**
PAPP	0.621	**0.678**
PXR	0.333	**0.416**
mean	0.361	***0.411***

ILLINOIS