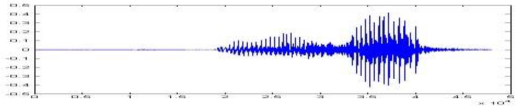
Neural Network
(Basic Ideas)

Hung-yi Lee

Learning ≈ Looking for a Function

- Speech Recognition

$f($

$)=$ “你好”

- Handwritten Recognition

$f($

$)=$ “2”

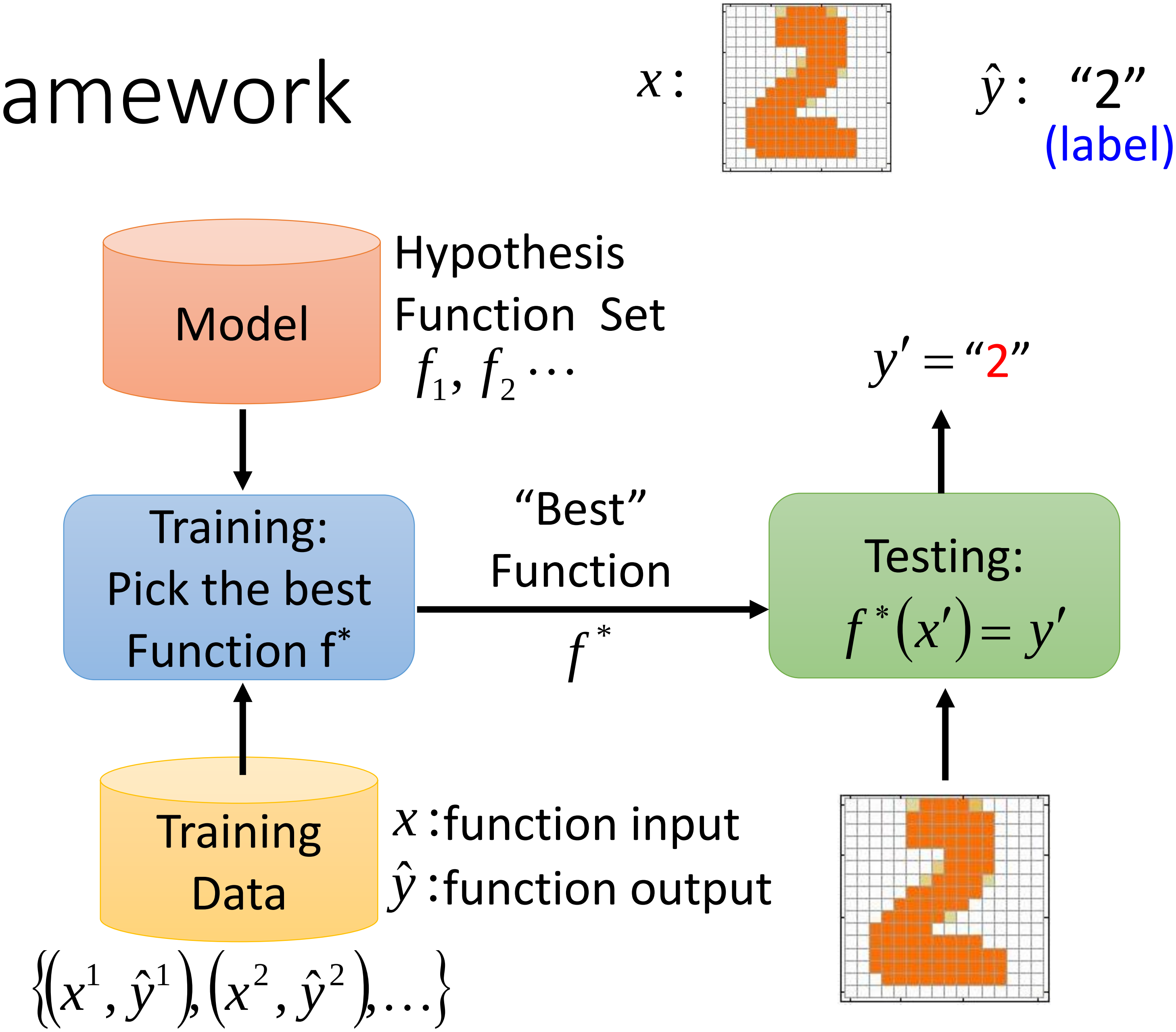
- Weather forecast

$f($ weather today $)=$ “sunny tomorrow”

- Play video games

$f($ Positions and number of enemies $)=$ “fire”

Framework



$x:$

$\hat{y}:$ “2”

(label)

Model

Hypothesis Function Set

$f_1, f_2 \cdots$

Training: Pick the best Function f*

“Best” Function

f^*

Testing:

$f^*(x') = y'$

$y' = $ “2”

Training Data

x :function input

$\hat{y}$:function output

$\{(x^1, \hat{y}^1), (x^2, \hat{y}^2), \ldots\}$

Outline

1. What is the model (function hypothesis set)?

2. What is the "best" function?

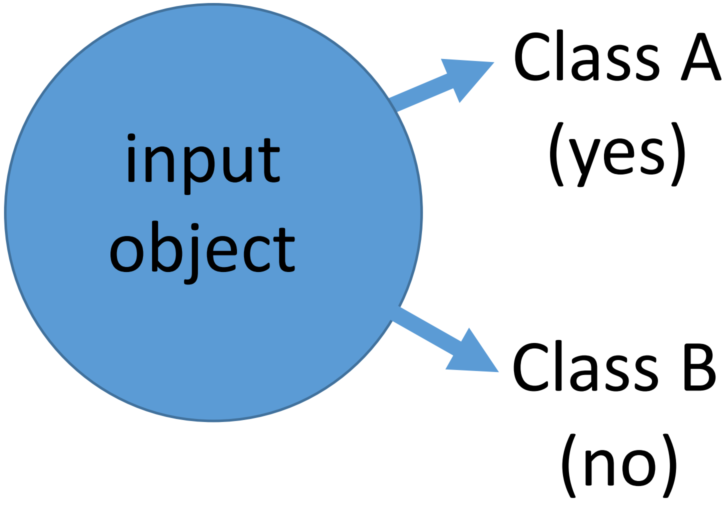
3. How to pick the "best" function?

Task Considered Today

- ***<u>Classification</u>***

Binary Classification

Only two classes



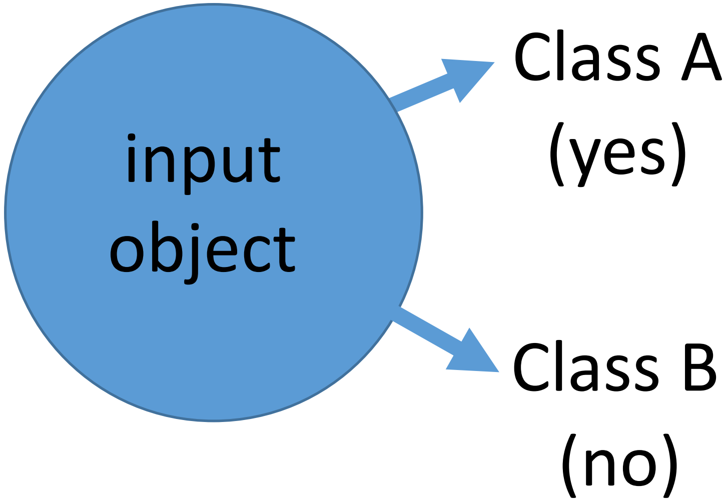
- **Spam filtering**
  - Is an e-mail spam or not?
- **Recommendation systems**
  - recommend the product to the customer or not?
- **Malware detection**
  - Is the software malicious or not?
- **Stock prediction**
  - Will the future value of a stock increase or not?

Task Considered Today

- ***Classification***

Binary Classification

Only two classes



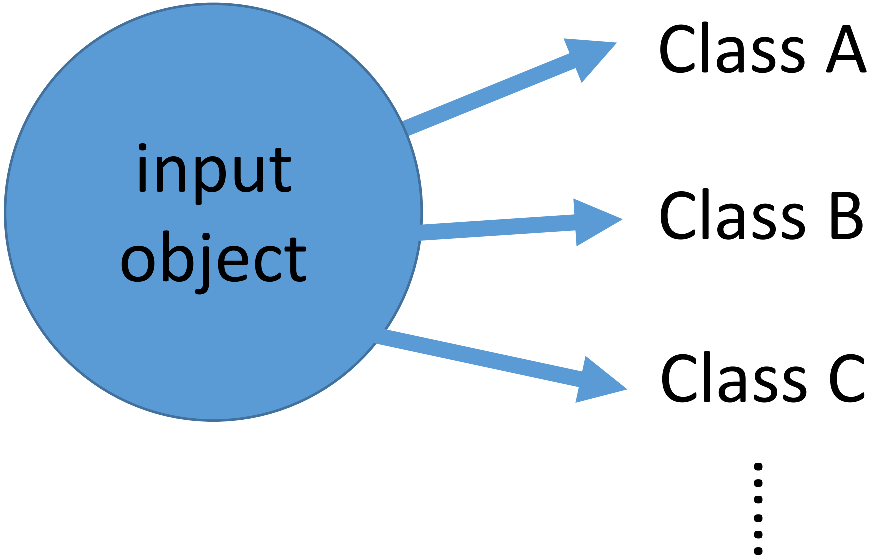
input object

Class A (yes)

Class B (no)

Multi-class Classification

More than two classes



input object

Class A

Class B

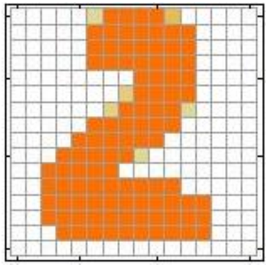
Class C

⋮

Multi-class Classification

- Handwriting Digit Classification

Input:



Class: “1”, “2”, …., “9”, “0”

10 classes

- Image Recognition

Input:

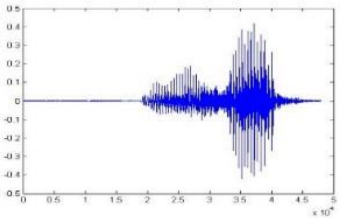
Class: “dog”, “cat”, “book”, ….

Thousands of classes

Multi-class Classification

- ***Real*** speech recognition is not multi-class classification

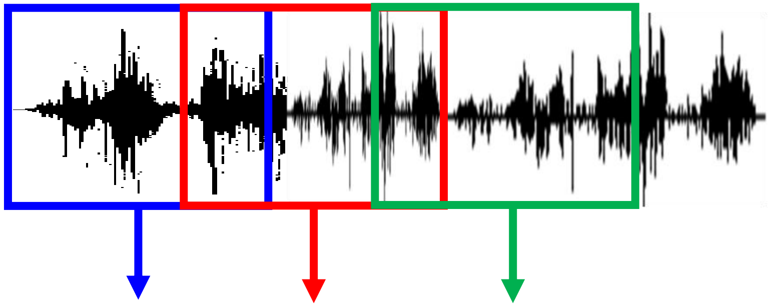
Input:



Classes: “hi”, “how are you”, “I am sorry” …….

Cannot be enumerated

- The HW1 is multi-class classification



frame

/b/ /b/ /ɛ/

The frame belongs to which ***phoneme***.

Classes are the phonemes.

1. What is the model?

What is the function we are looking for?

- ***classification***

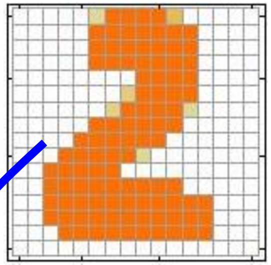
$$y = f(x) \quad \Rightarrow \quad f: R^N \rightarrow R^M$$

  - x: input object to be classified
  - y: class
  - ***Assume both x and y can be represented as fixed-size vector***
    - x is a vector with N dimensions, and y is a vector with M dimensions

What is the function we are looking for?

- ***Handwriting Digit Classification*** $f: R^N \to R^M$

x: image



Each pixel corresponds to an element in the vector

16 x 16

$$\begin{bmatrix}0\\1\\\vdots\end{bmatrix}$$

1: for ink,
0: otherwise

16 x 16 = 256 dimensions

y: class

10 dimensions for digit recognition

$$\begin{array}{ccc} \text{“1”} & & \text{“2”} \\ \downarrow & & \downarrow \\ \begin{bmatrix}1\\0\\0\\\vdots\end{bmatrix} & \begin{matrix}\text{“1”}\\\text{“2”}\\\text{“3”}\\ \end{matrix} & \begin{bmatrix}0\\1\\0\\\vdots\end{bmatrix}\end{array}$$

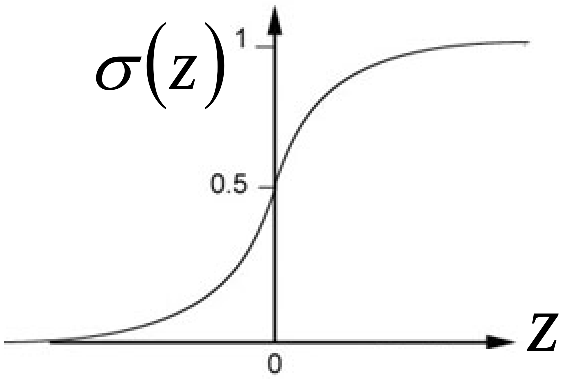
“1” → “1” or not
“2” → “2” or not
“3” → “3” or not

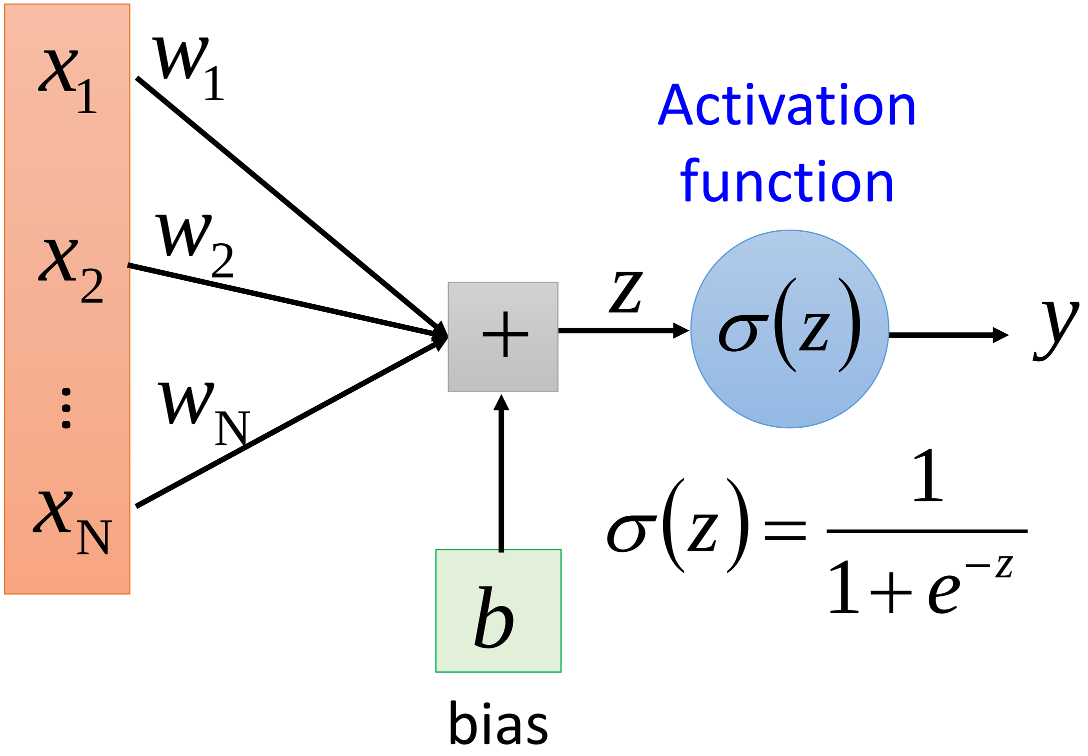
1. What is the model?

A Layer of Neuron

Single Neuron

$$f: R^N \to R$$



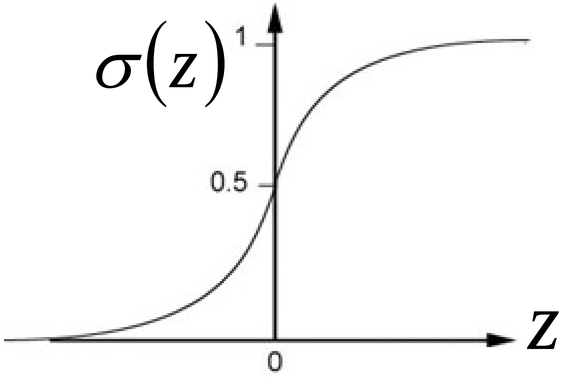


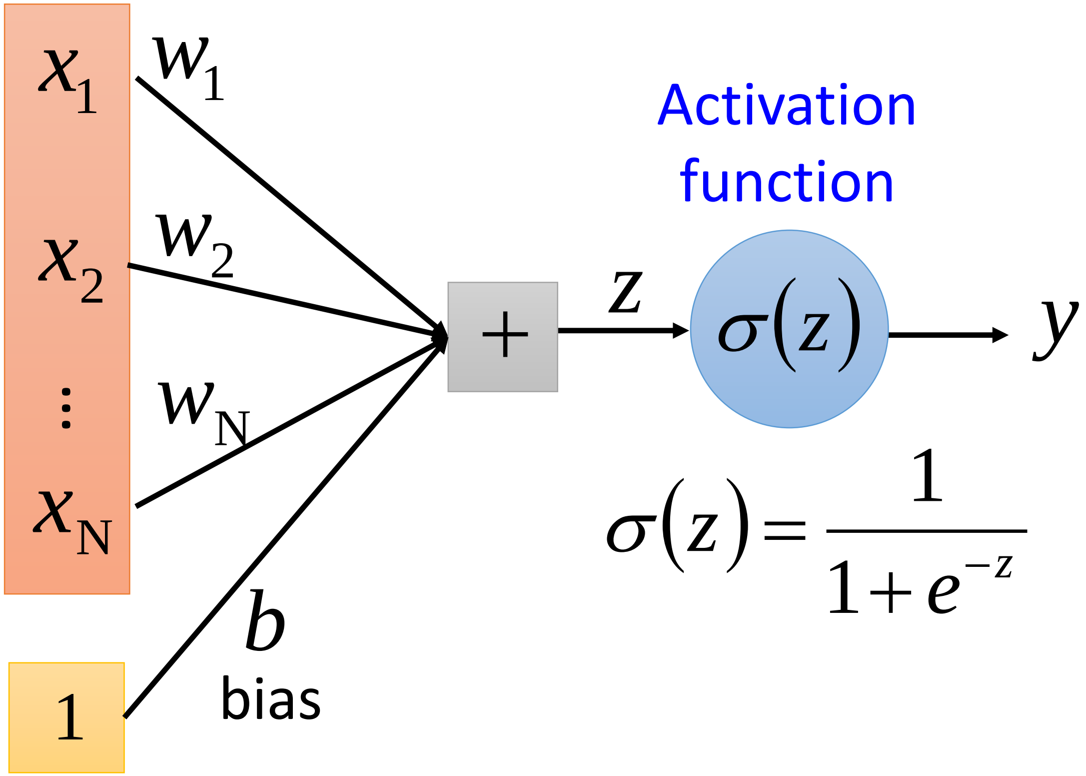
Activation function

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Single Neuron

$$f: R^N \rightarrow R$$

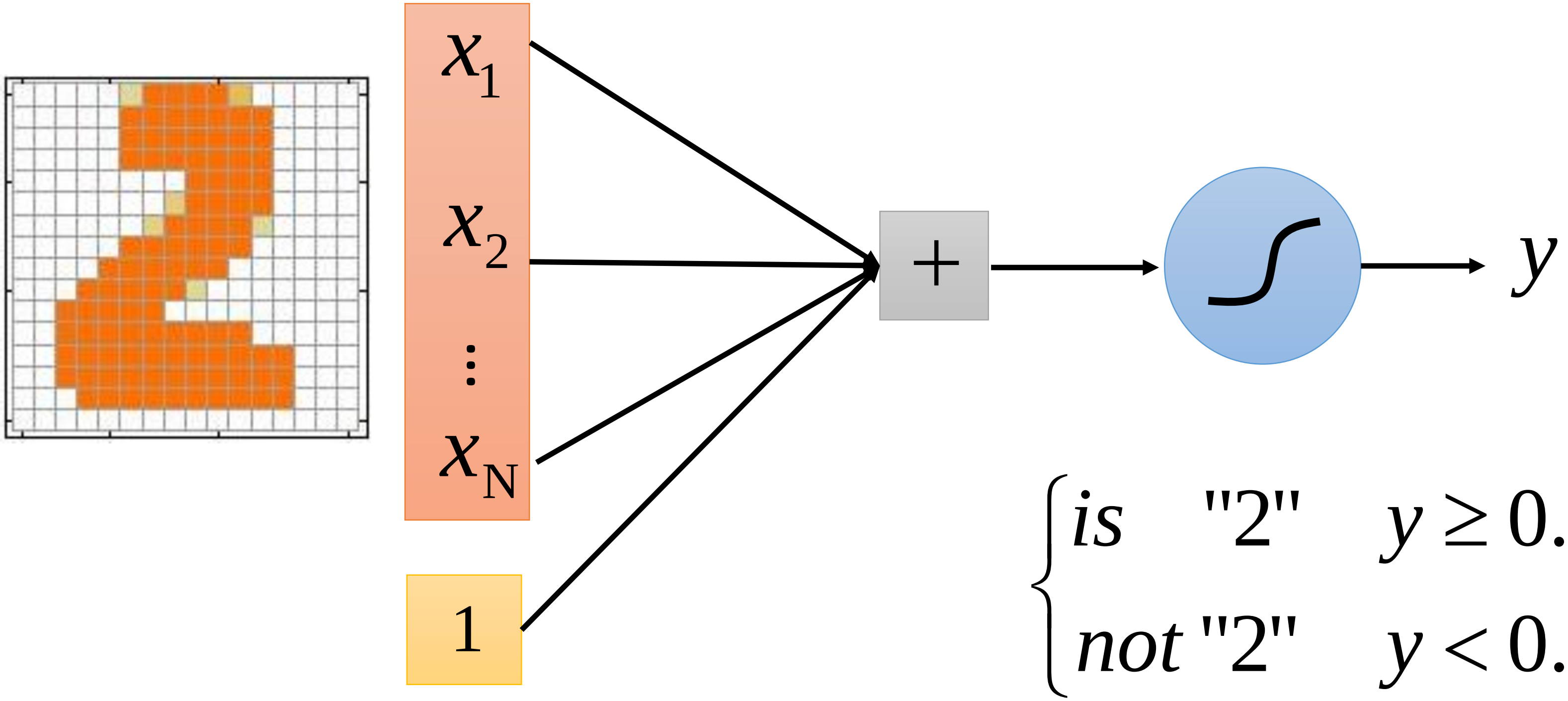




$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Single Neuron $f: R^N \rightarrow R$

- Single neuron can only do binary classification, cannot handle multi-class classification

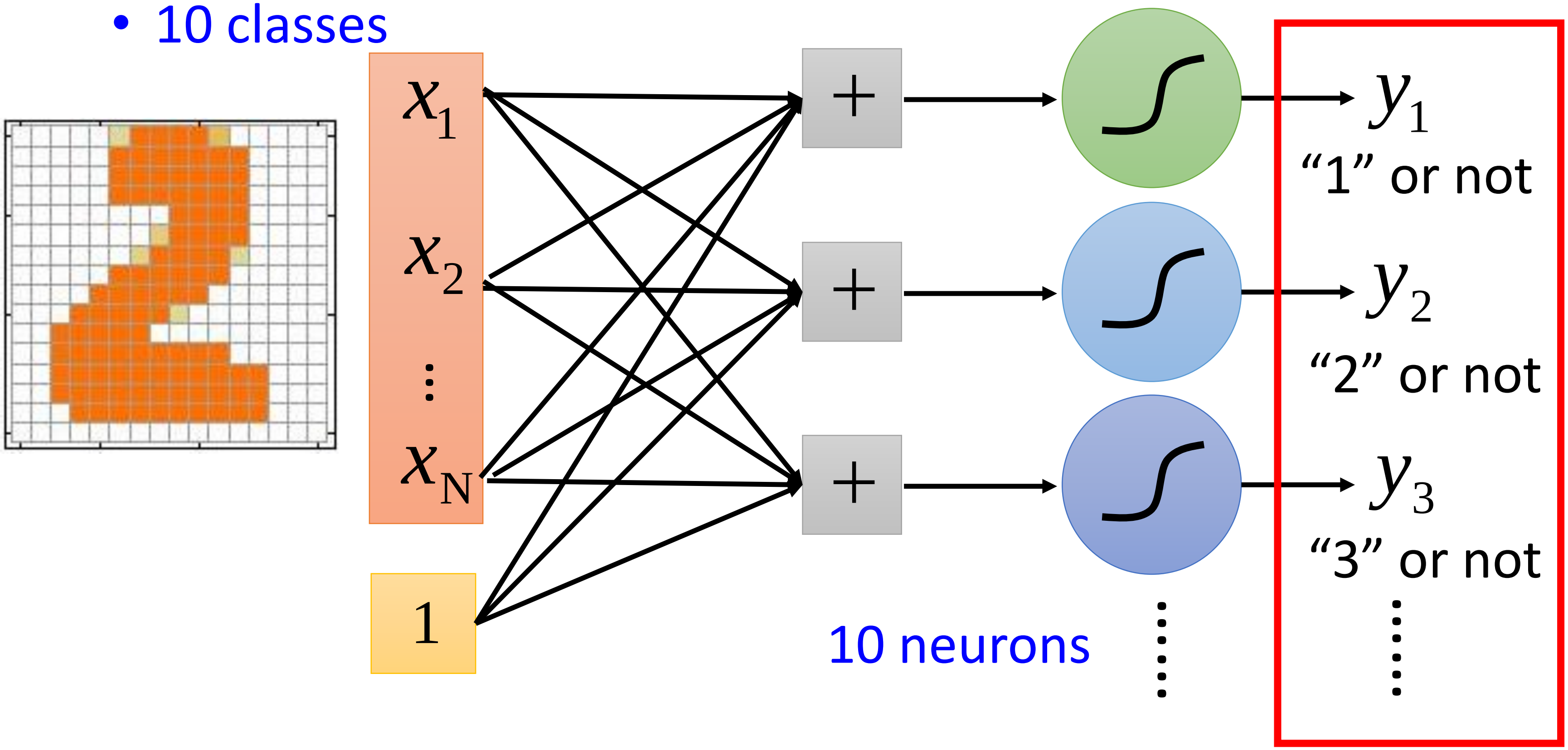


$$\begin{cases} is \quad "2" & y \geq 0.5 \\ not\, "2" & y < 0.5 \end{cases}$$

A Layer of Neuron $f: R^N \to R^M$

- Handwriting digit classification
  - Classes: “1”, “2”, …., “9”, “0”
  - 10 classes

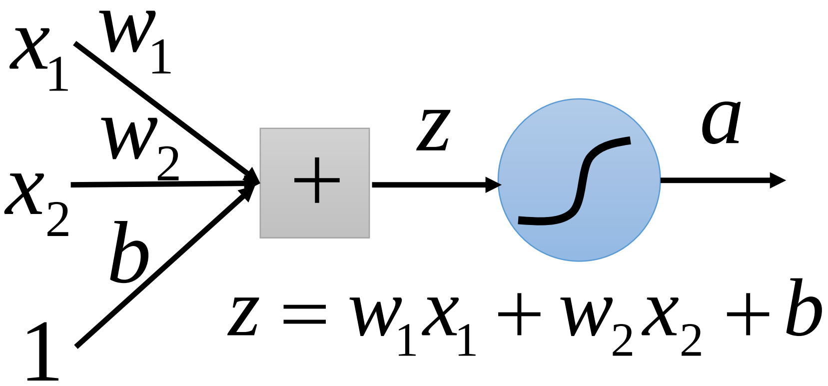
If y_2 is the max, then the image is “2”.



1. What is the model?

Limitation of Single Layer

Limitation of Single Layer

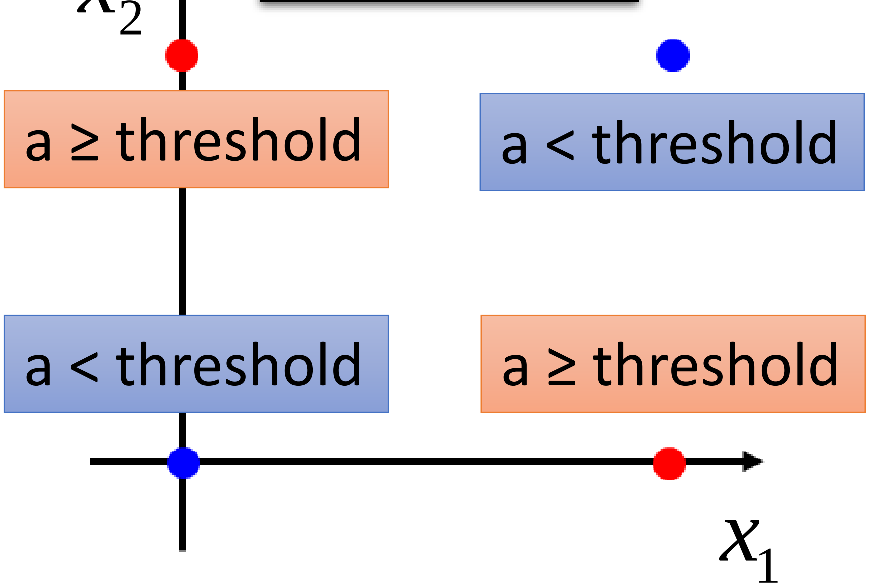


$$z = w_1 x_1 + w_2 x_2 + b$$

$$\begin{cases} yes & a \geq threshold \\ no & a < threshold \end{cases}$$

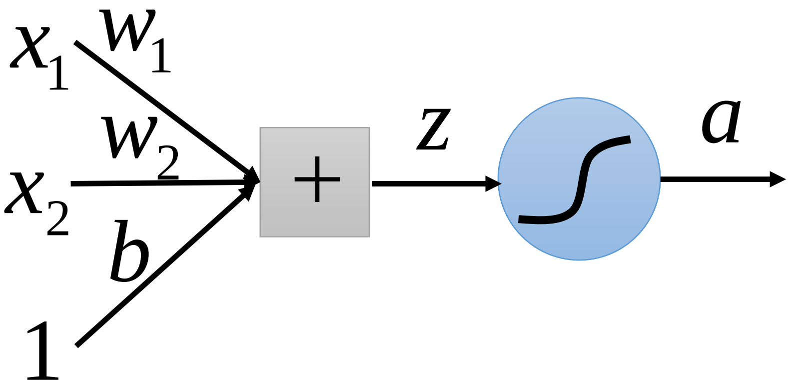
Input		Output
x_1	x_2	
0	0	No
0	1	Yes
1	0	Yes
1	1	No

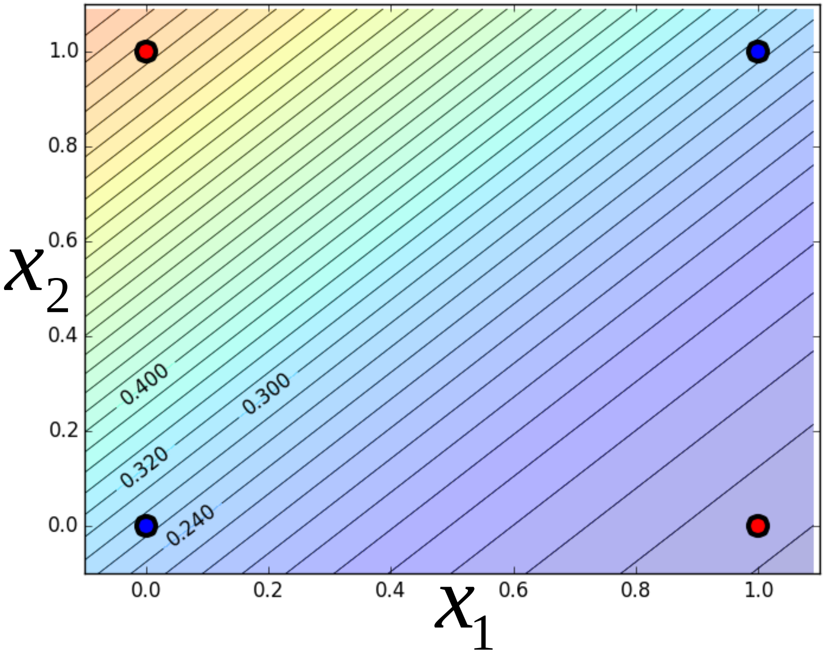
Can we?

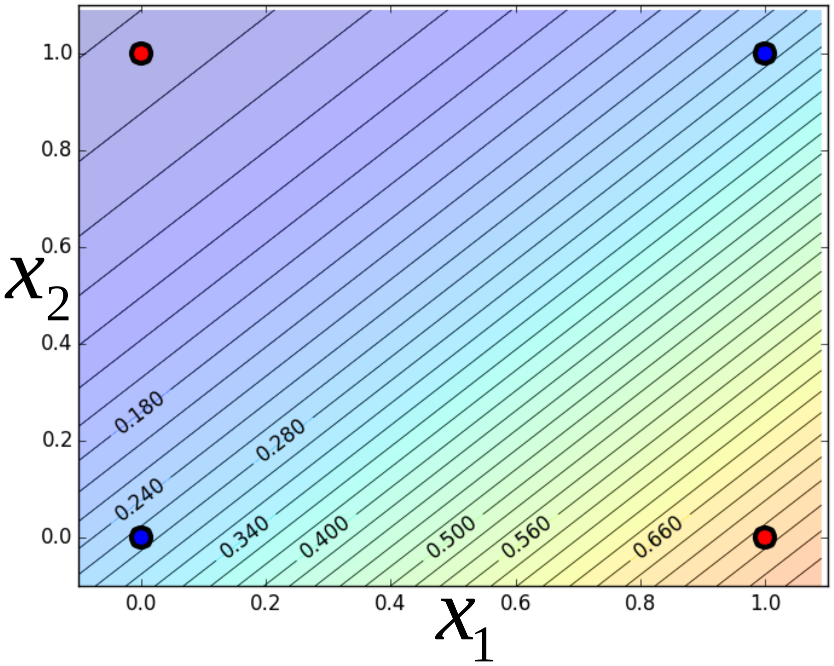


Limitation of Single Layer

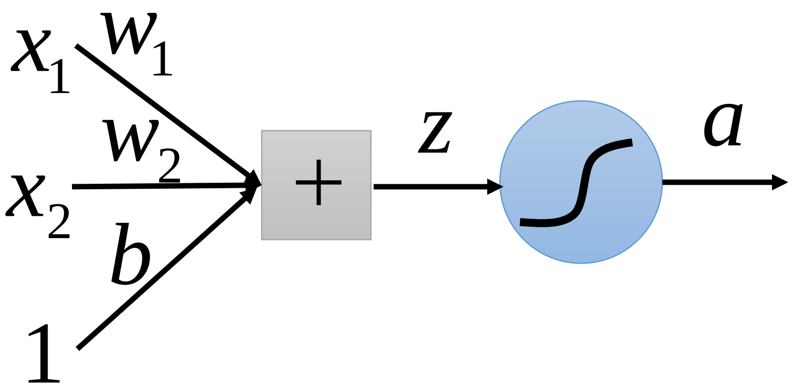
- No, we can't ……



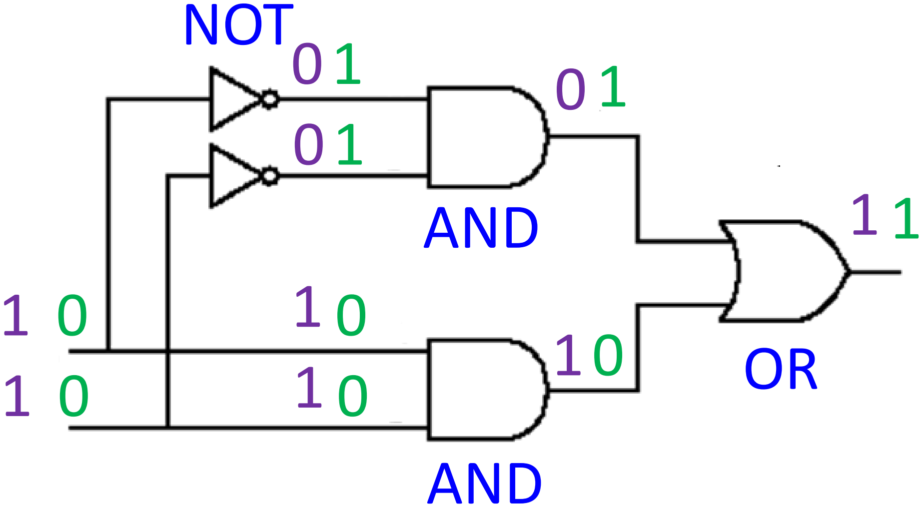




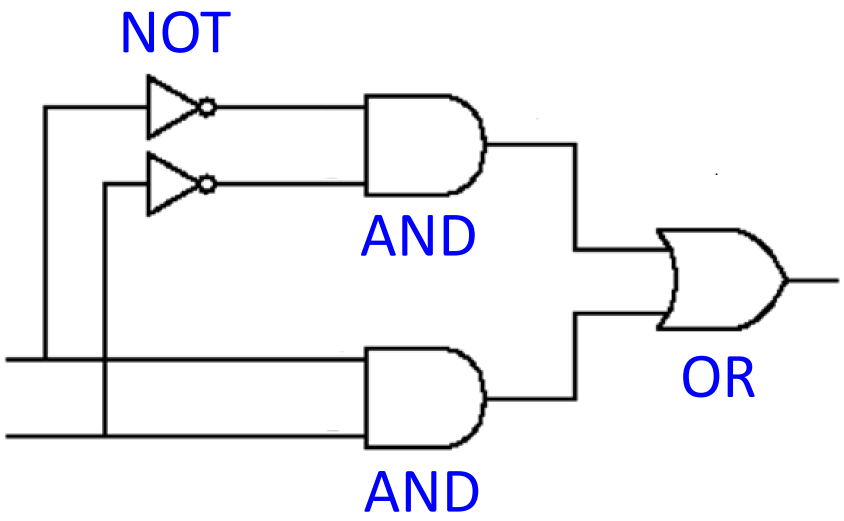
Limitation of Single Layer



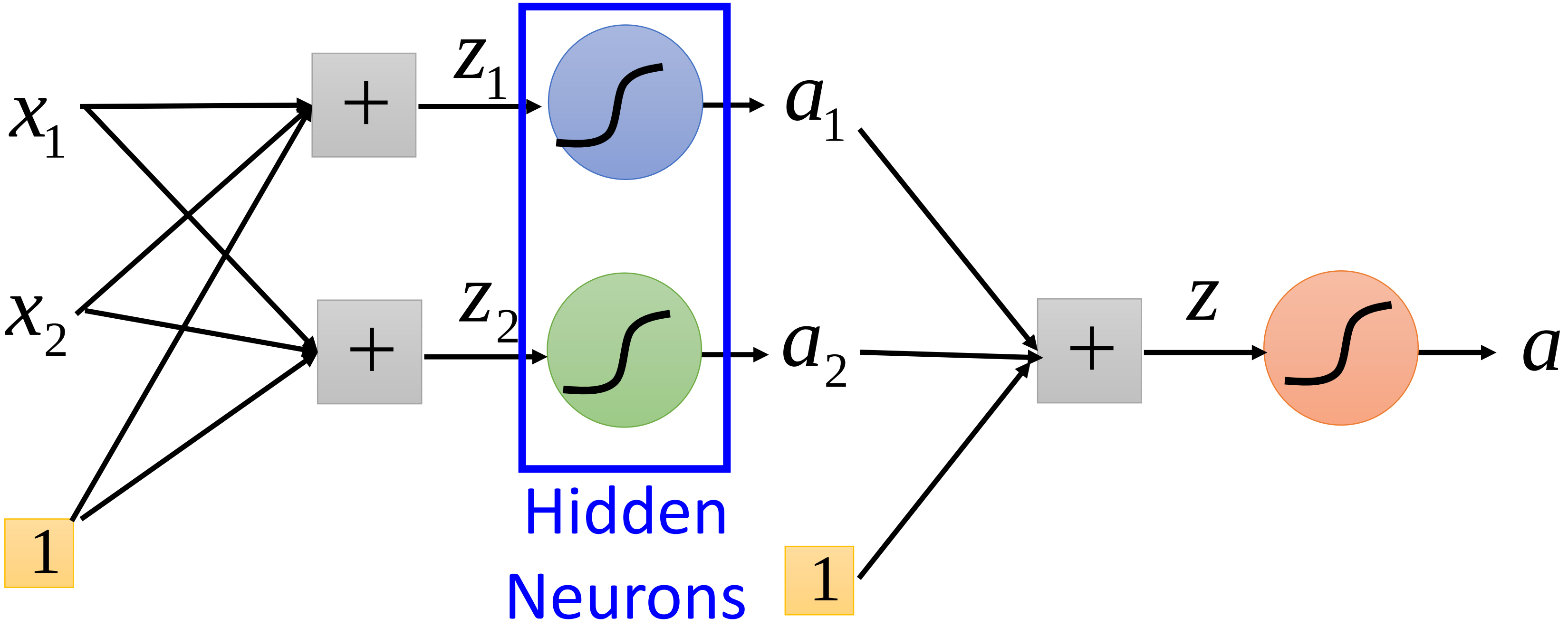
Input		Output
x_1	x_2	
0	0	No
0	1	Yes
1	0	Yes
1	1	No

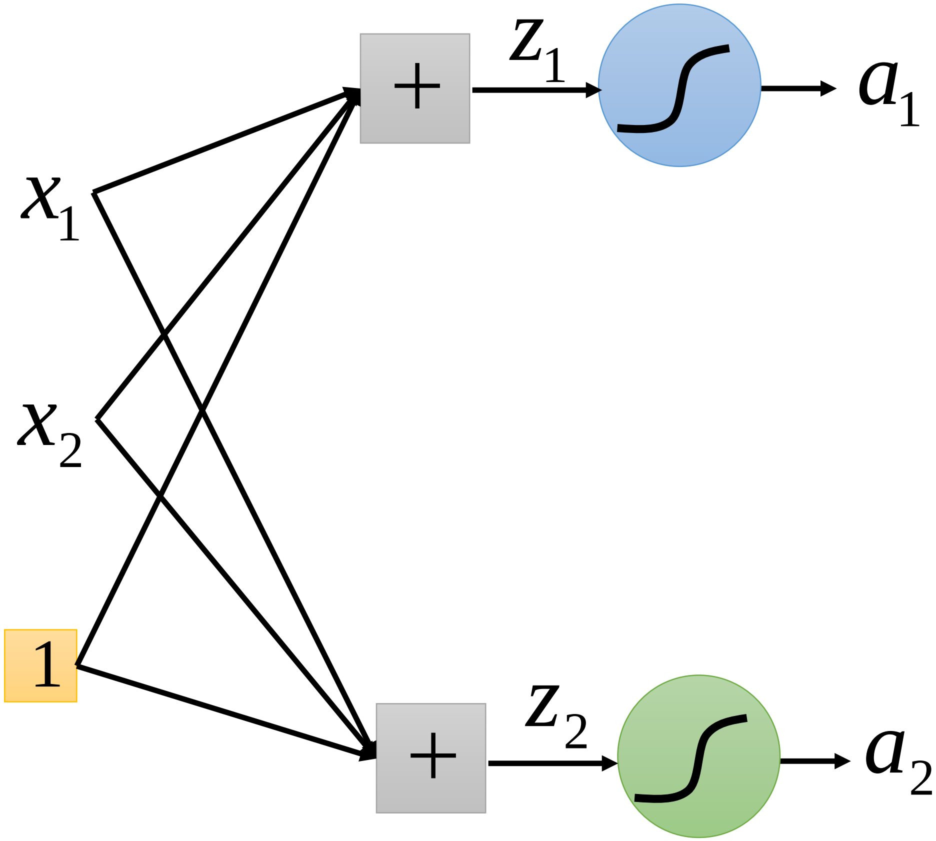


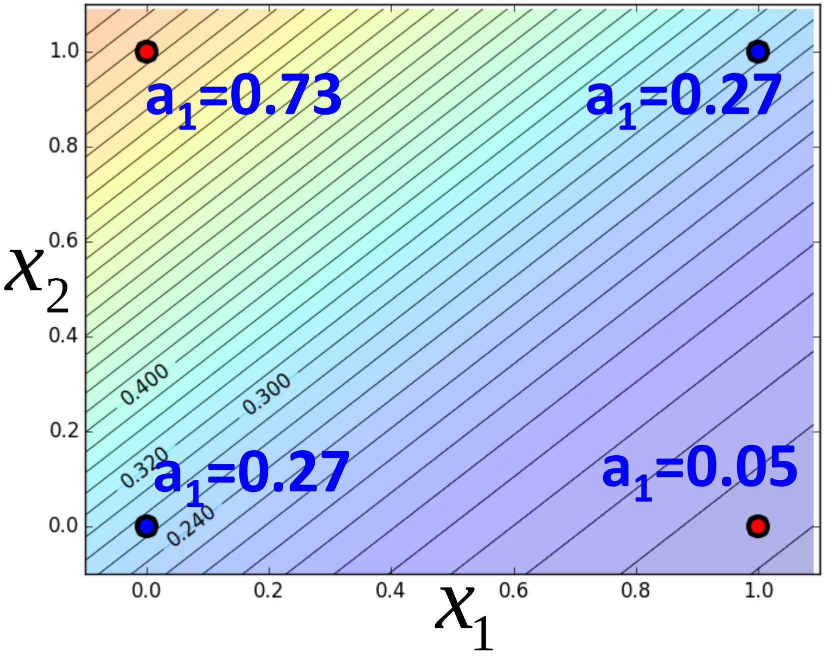
Neural Network

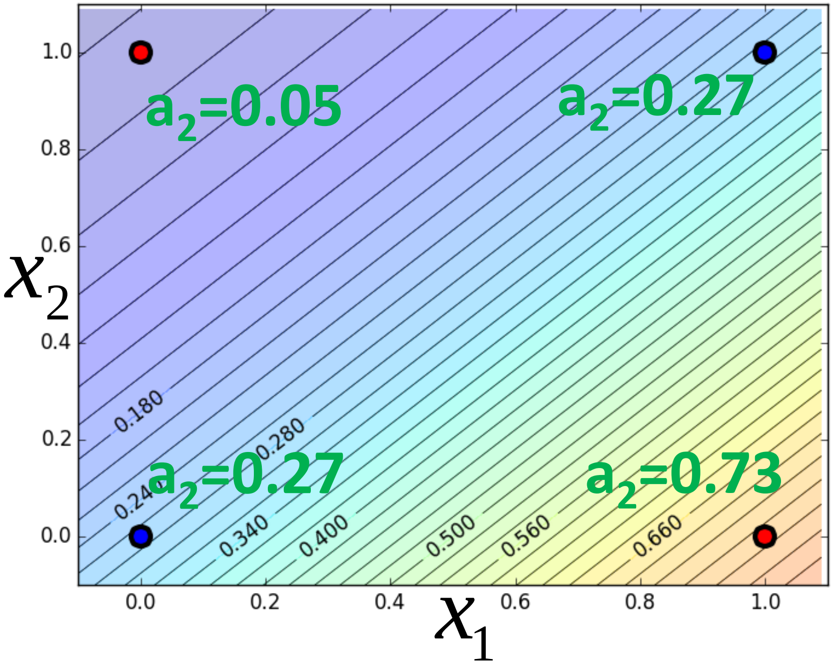


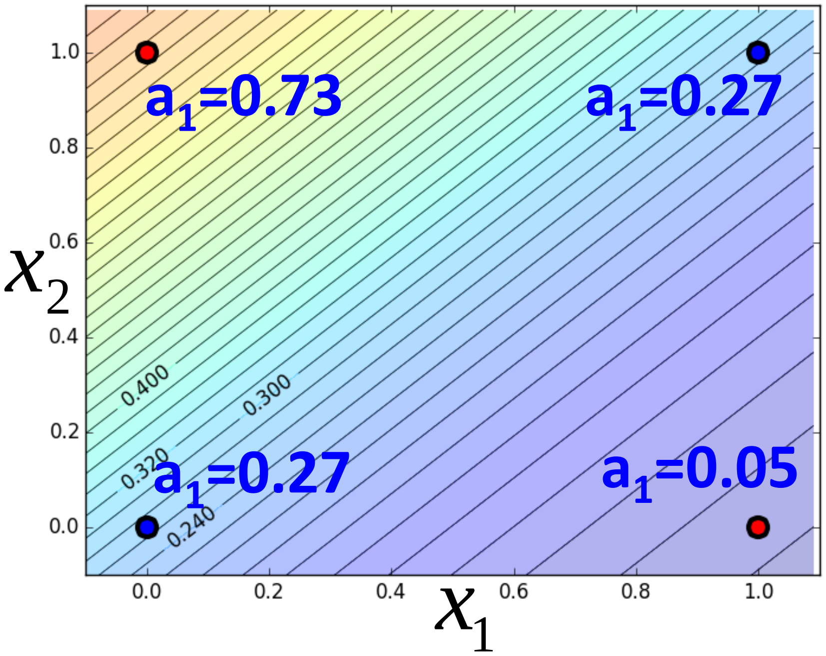
Neural Network



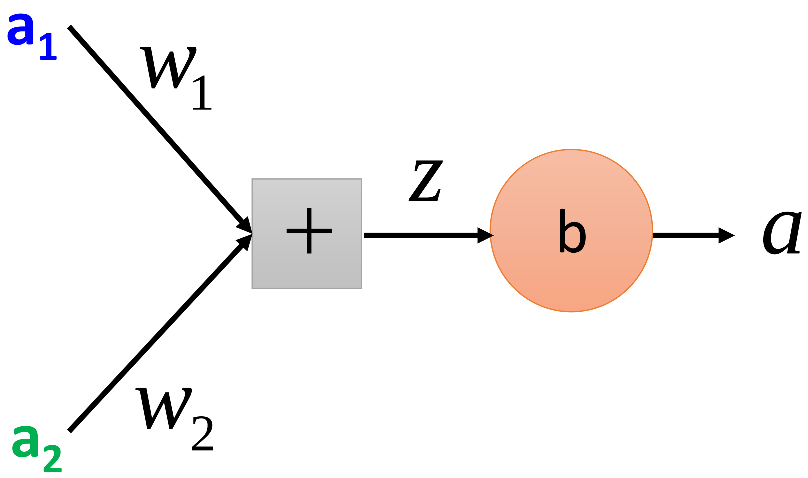

x_1
x_2
1
+
z_1
a_1
+
z_2
a_2


a_1=0.73
a_1=0.27
a_1=0.27
a_1=0.05
0.400
0.300
0.320
0.240
x_2
x_1
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

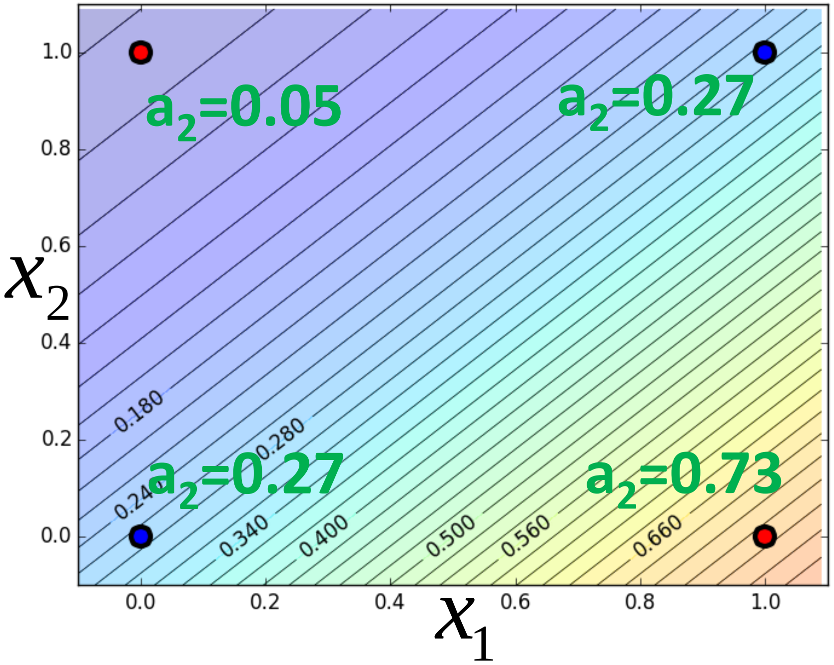

a_2=0.05
a_2=0.27
a_2=0.27
a_2=0.73
0.180
0.280
0.240
0.340
0.400
0.500
0.560
0.660
x_2
x_1
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0



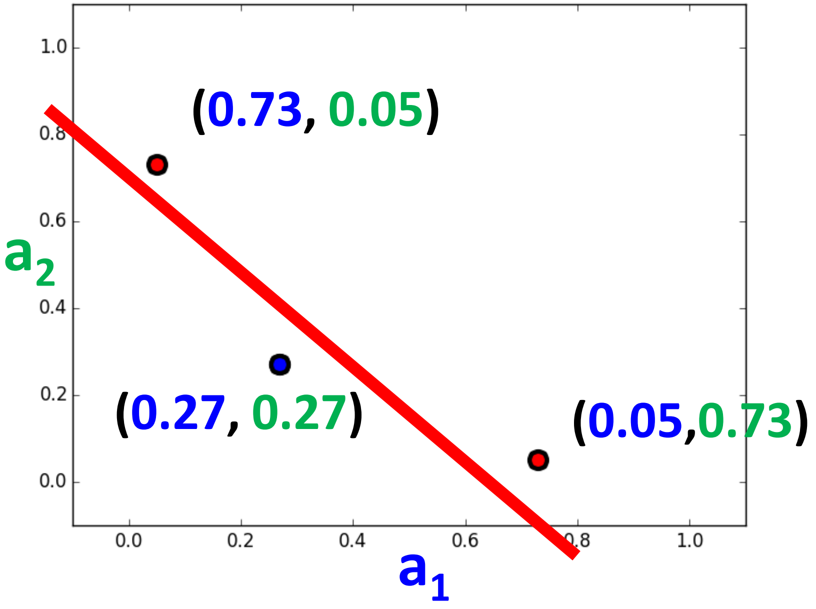
a_1=0.73
a_1=0.27
a_1=0.27
a_1=0.05
0.400
0.300
0.320
0.240
x_2
x_1
0.0
0.2
0.4
0.6
0.8
1.0



a_1
w_1
a_2
w_2
+
z
b
a



a_2=0.05
a_2=0.27
a_2=0.27
a_2=0.73
0.180
0.280
0.240
0.340
0.400
0.500
0.560
0.660
x_2
x_1
0.0
0.2
0.4
0.6
0.8
1.0



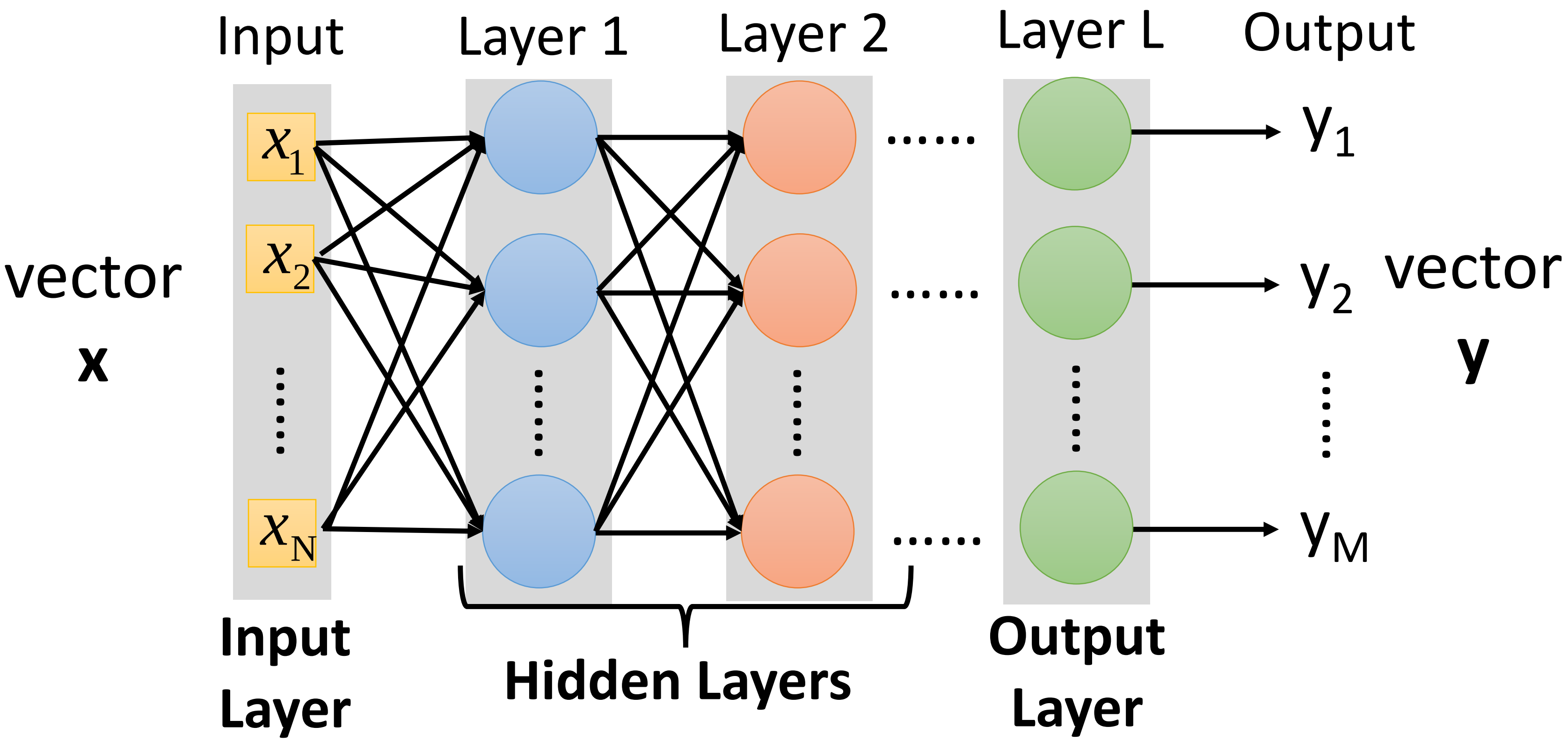
(0.73, 0.05)
(0.27, 0.27)
(0.05,0.73)
a_2
a_1
0.0
0.2
0.4
0.6
0.8
1.0

1. What is the model?

Neural Network

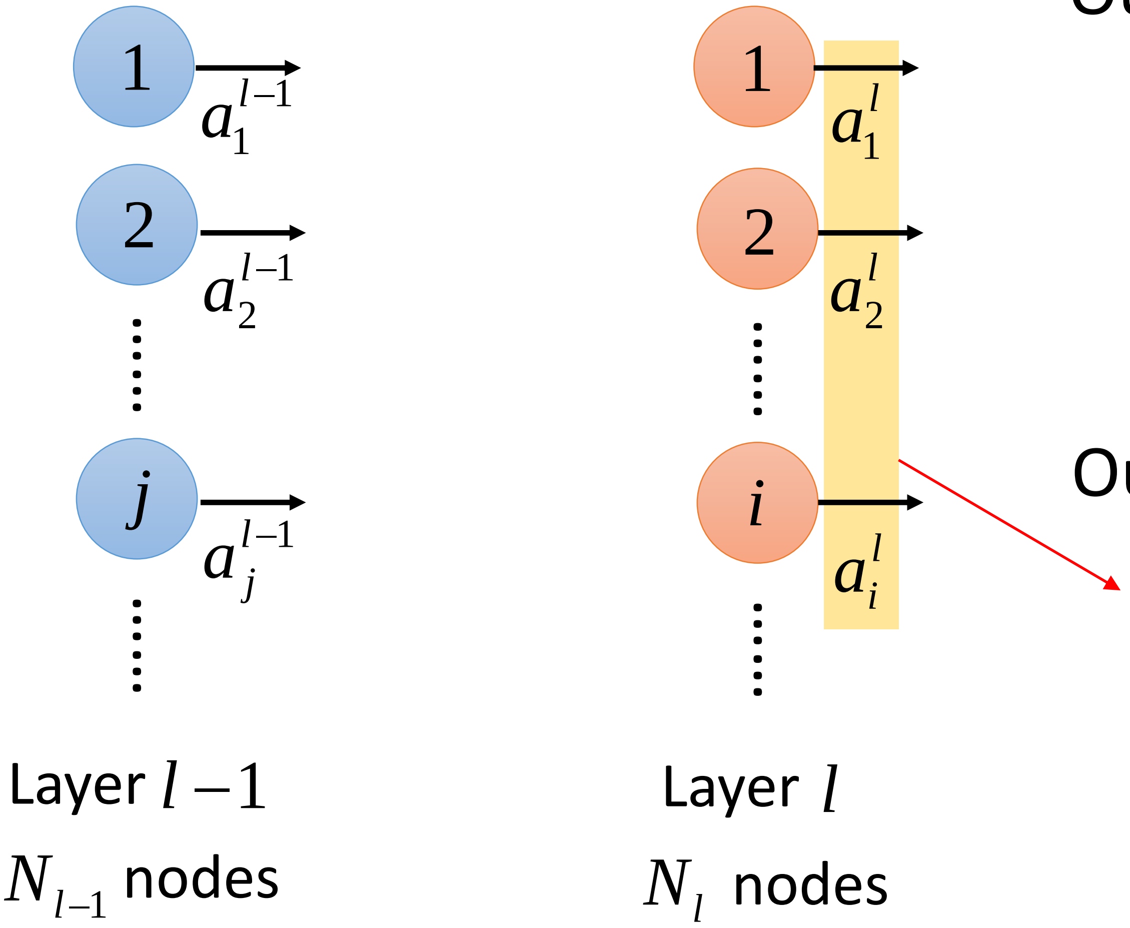
Neural Network as Model

$$f: R^N \to R^M$$



- Fully connected feedforward network
- Deep Neural Network: many hidden layers

Notation



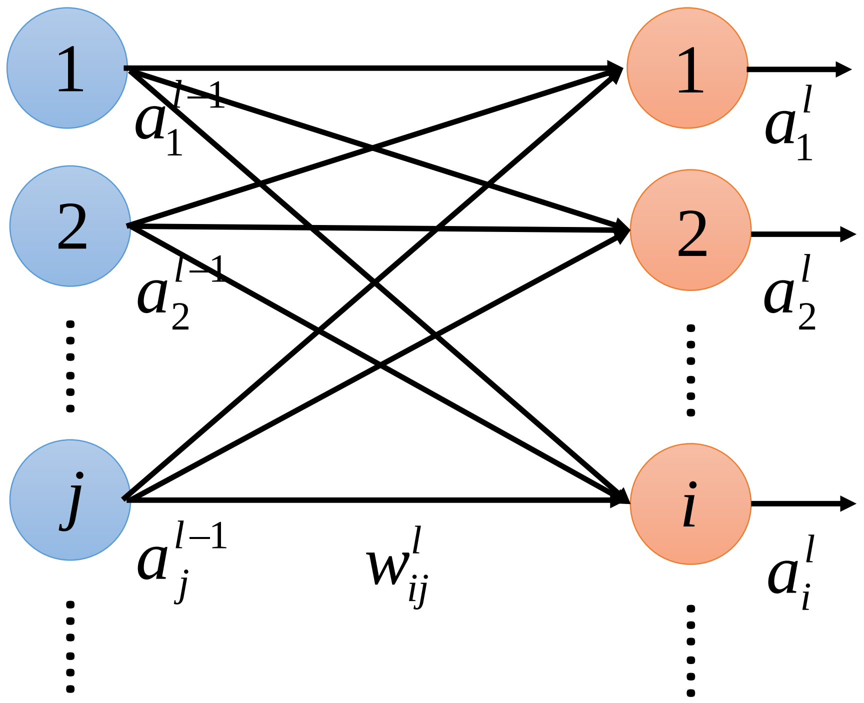
Output of a neuron:

a_i^l → Layer l, → Neuron i

Output of one layer:

a^l : a vector

Notation



Layer $l-1$
N_{l-1} nodes

Layer l
N_l nodes

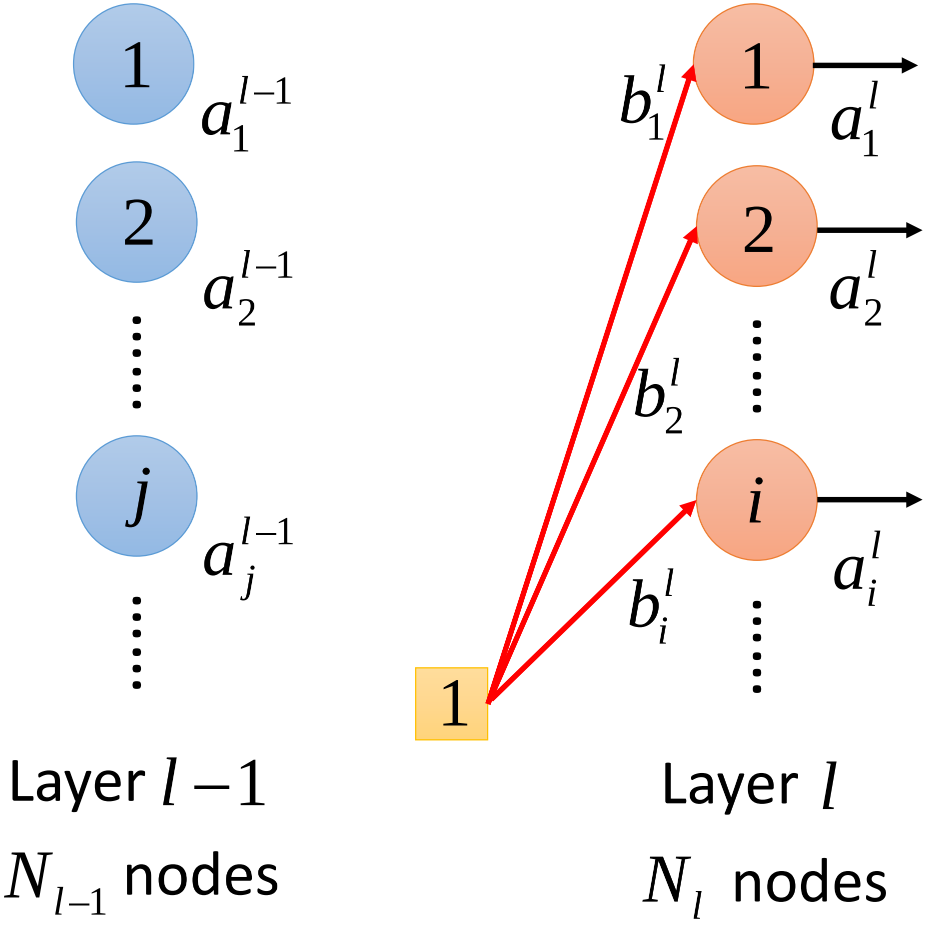
w_{ij}^l → Layer $l-1$
to Layer l

↓

from neuron j (Layer $l-1$)
to neuron i (Layer l)

$$W^l = \left.\overbrace{\begin{bmatrix} w_{11}^l & w_{12}^l & \cdots \\ w_{21}^l & w_{22}^l & \\ \vdots & & \ddots \\ & & \end{bmatrix}}^{\mathrm{N}_{l-1}}\right\} \mathrm{N}_l$$

Notation

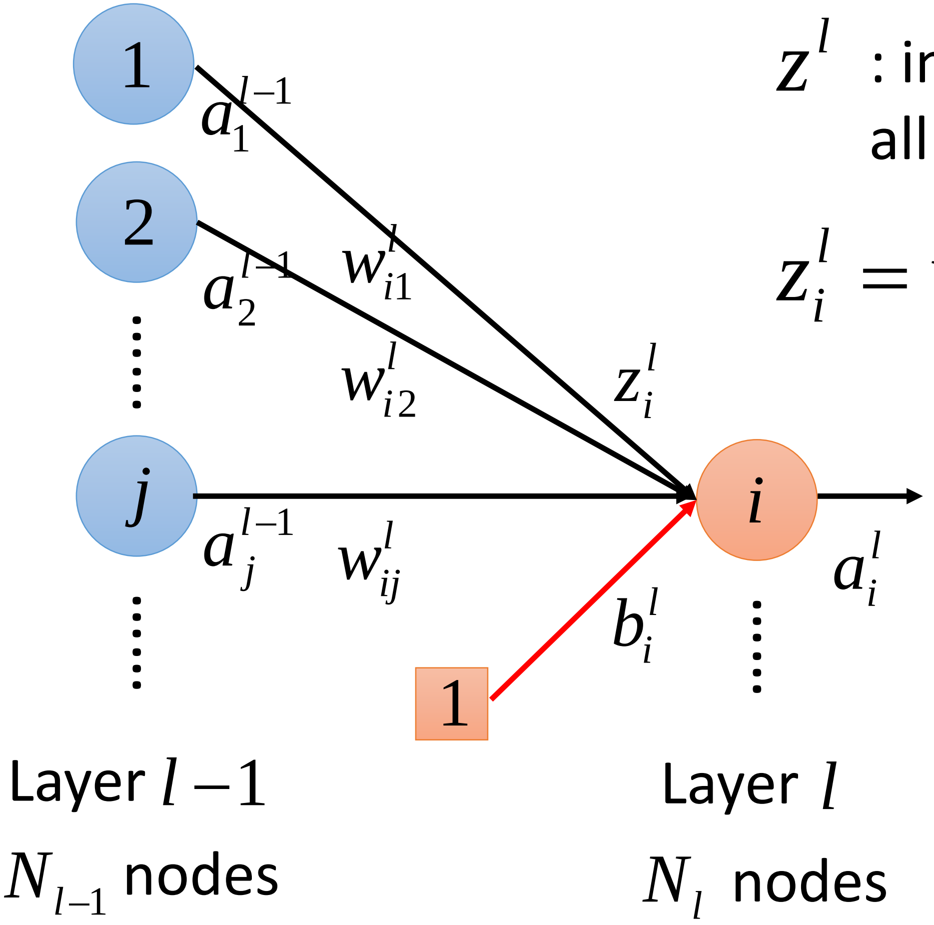


b_i^l : bias for neuron i at layer l

$$b^l = \begin{bmatrix} b_1^l \\ b_2^l \\ \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$

bias for all neurons in layer l

Notation



z_i^l : input of the activation function for neuron i at layer l

z^l : input of the activation function all the neurons in layer l

$$z_i^l = w_{i1}^l a_1^{l-1} + w_{i2}^l a_2^{l-1} \ldots + b_i^l$$

$$z_i^l = \sum_{j=1}^{N_{l-1}} w_{ij}^l a_j^{l-1} + b_i^l$$

Notation - Summary

a_i^l :output of a neuron

a^l :output of a layer

z_i^l : input of activation function

z^l : input of activation function for a layer

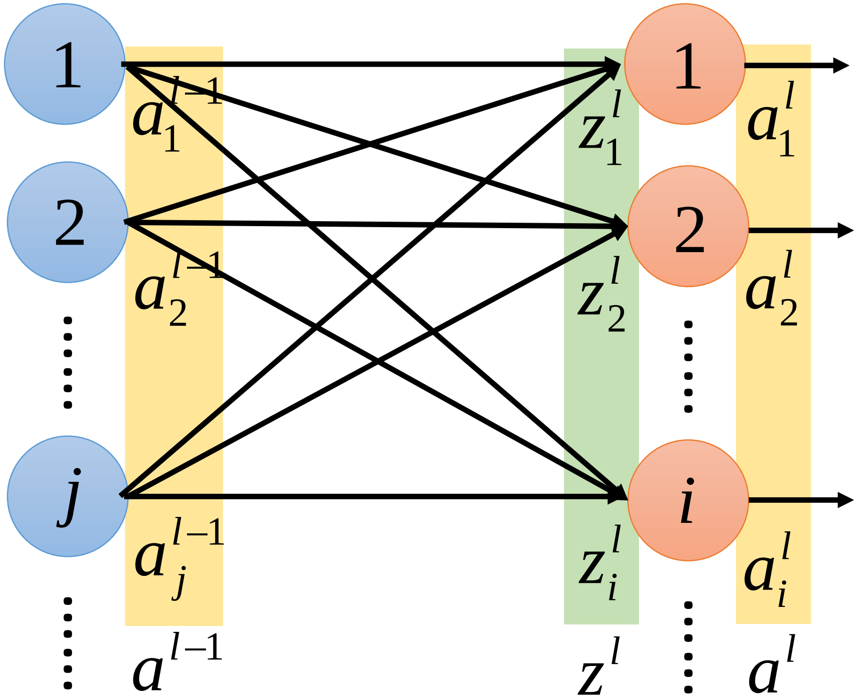
w_{ij}^l : a weight

W^l : a weight matrix

b_i^l : a bias

b^l : a bias vector

Relations between Layer Outputs



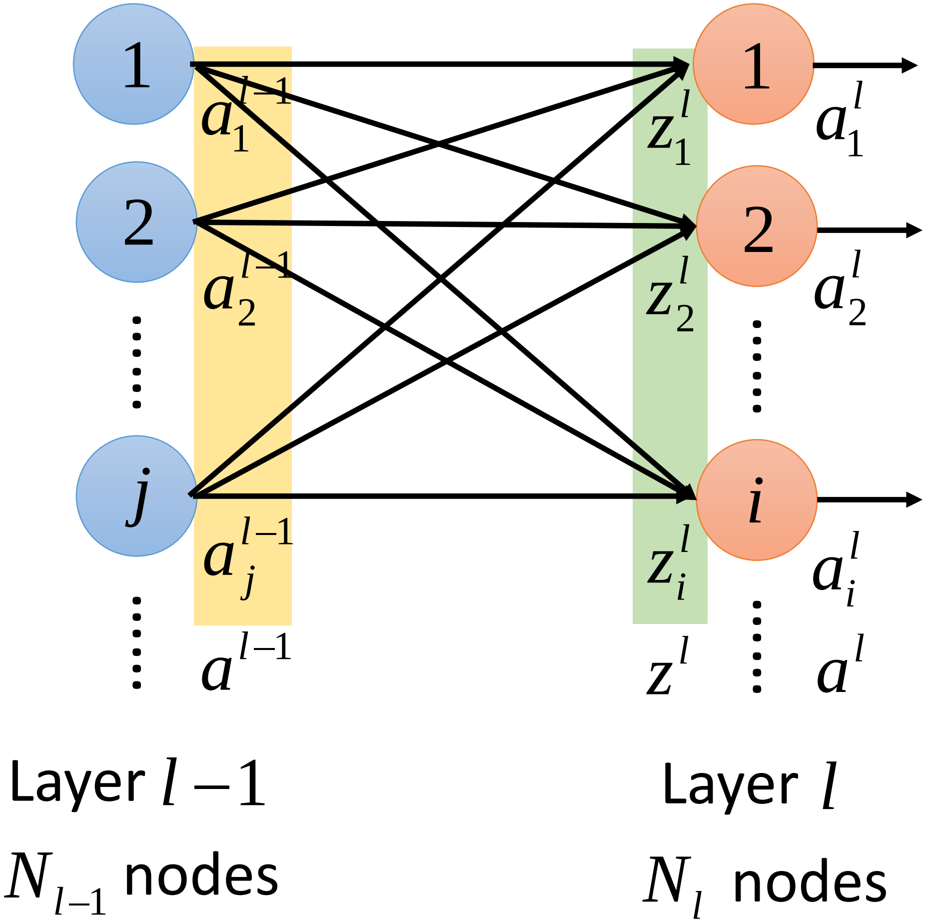
Layer $l-1$

N_{l-1} nodes

Layer l

N_l nodes

Relations between Layer Outputs



$$z_1^l = w_{11}^l a_1^{l-1} + w_{12}^l a_2^{l-1} + \cdots + b_1^l$$

$$z_2^l = w_{21}^l a_1^{l-1} + w_{22}^l a_2^{l-1} + \cdots + b_2^l$$

$$\vdots$$

$$z_i^l = w_{i1}^l a_1^{l-1} + w_{i2}^l a_2^{l-1} + \cdots + b_i^l$$

$$\vdots$$

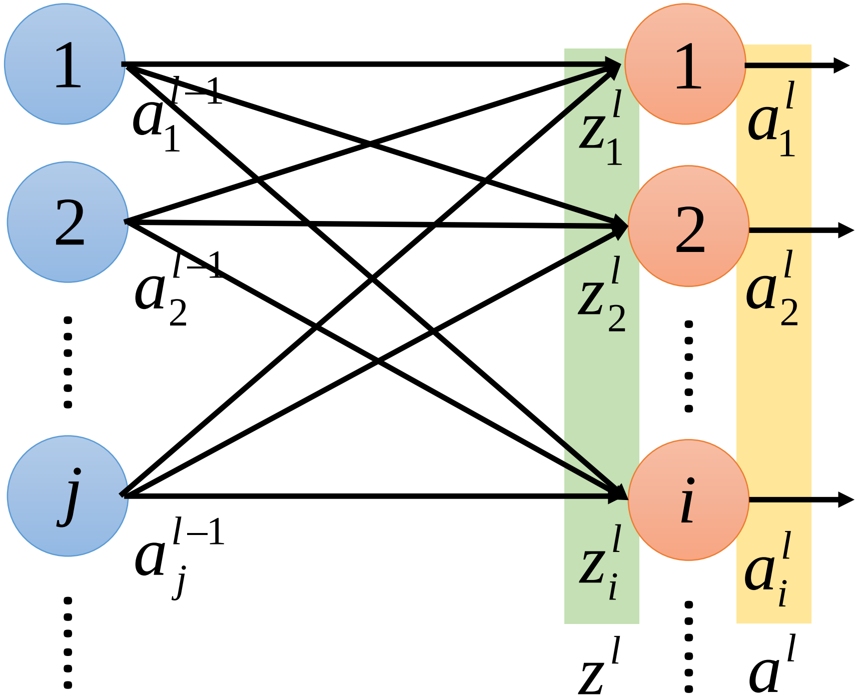
$$\Downarrow$$

$$\begin{bmatrix} z_1^l \\ z_2^l \\ \vdots \\ z_i^l \\ \vdots \end{bmatrix} = \begin{bmatrix} w_{11}^l & w_{12}^l & \cdots \\ w_{21}^l & w_{22}^l & \\ \vdots & & \ddots \end{bmatrix} \begin{bmatrix} a_1^{l-1} \\ a_2^{l-1} \\ \vdots \\ a_i^{l-1} \\ \vdots \end{bmatrix} + \begin{bmatrix} b_1^l \\ b_2^l \\ \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$

$$\Downarrow$$

$$z^l = W^l a^{l-1} + b^l$$

Relations between Layer Outputs



Layer $l-1$

N_{l-1} nodes

Layer l

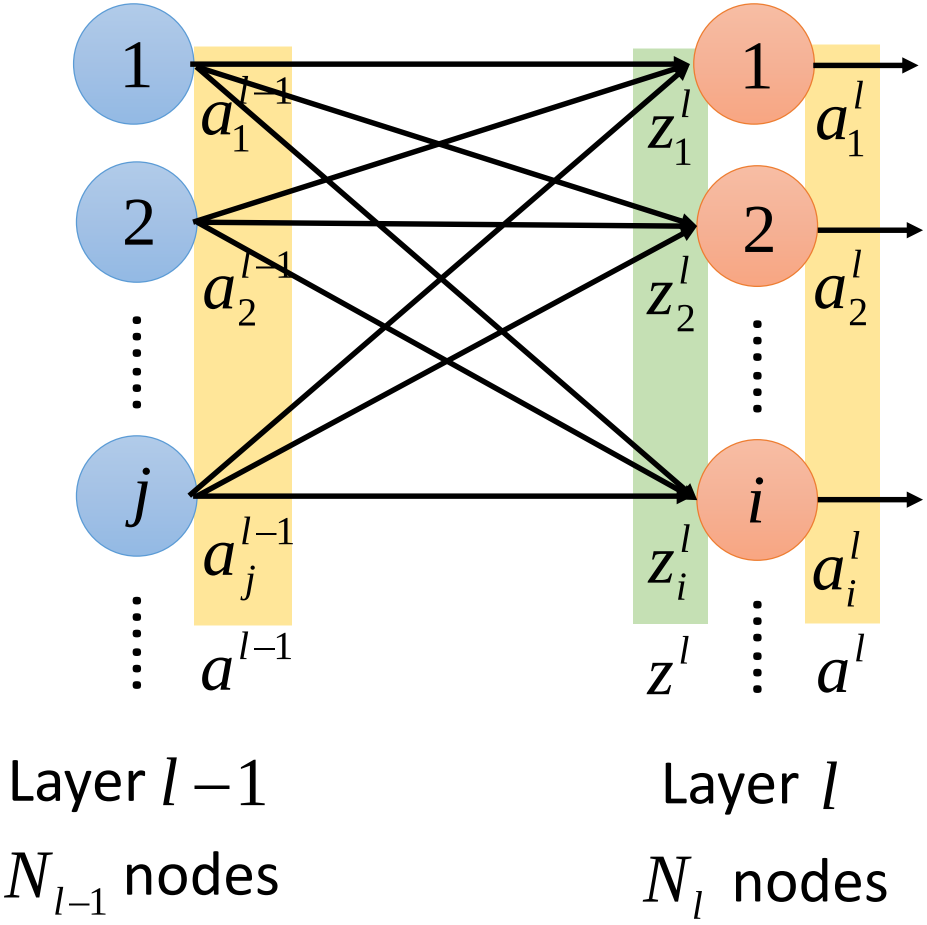
N_l nodes

$$a_i^l = \sigma\left(z_i^l\right)$$

$$\begin{bmatrix} a_1^l \\ a_2^l \\ \vdots \\ a_i^l \\ \vdots \end{bmatrix} = \begin{bmatrix} \sigma\left(z_1^l\right) \\ \sigma\left(z_2^l\right) \\ \vdots \\ \sigma\left(z_i^l\right) \\ \vdots \end{bmatrix}$$

$$a^l = \sigma\left(z^l\right)$$

Relations between Layer Outputs

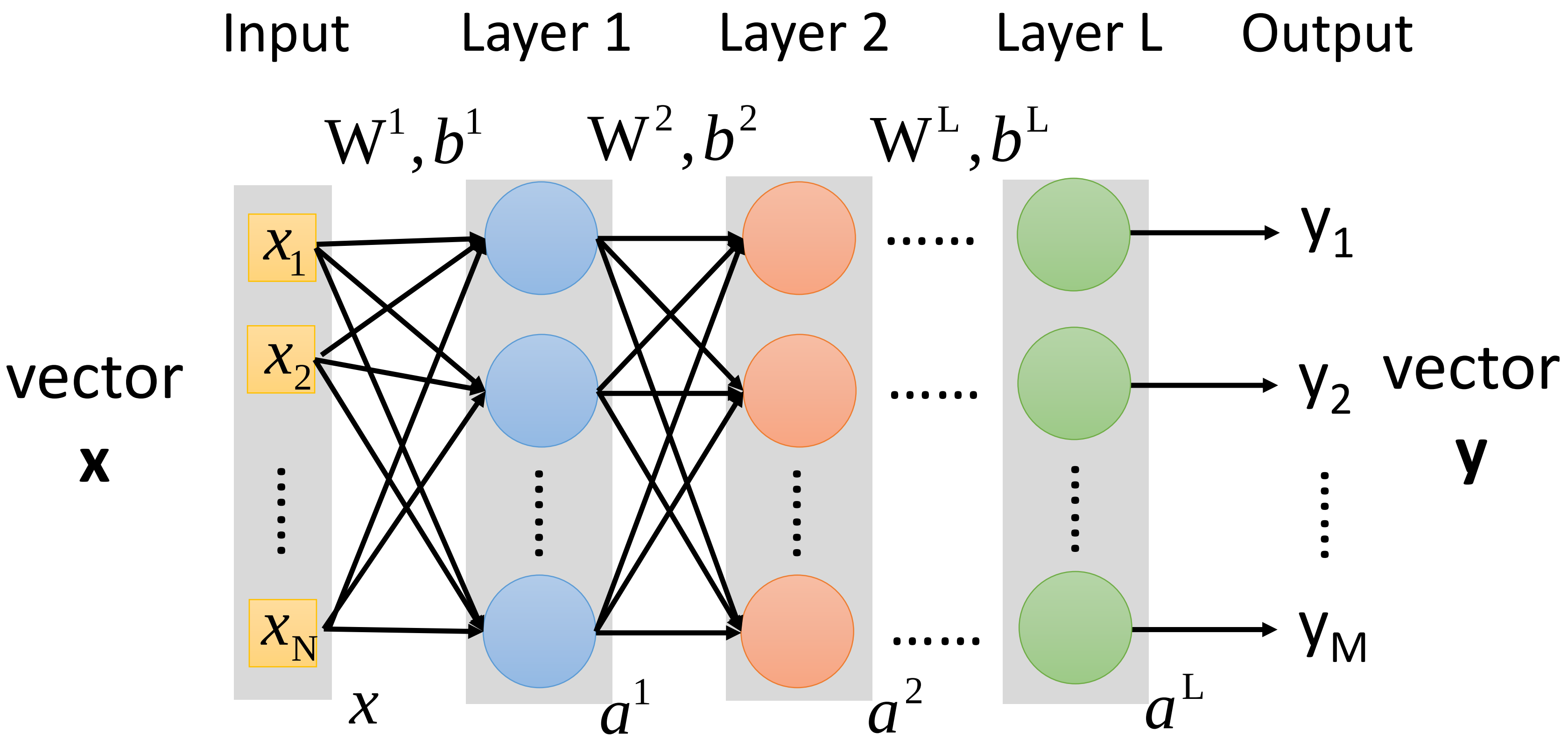


$$z^l = W^l a^{l-1} + b^l$$

$$a^l = \sigma\left(z^l\right)$$

$$a^l = \sigma\left(W^l a^{l-1} + b^l\right)$$

Function of Neural Network

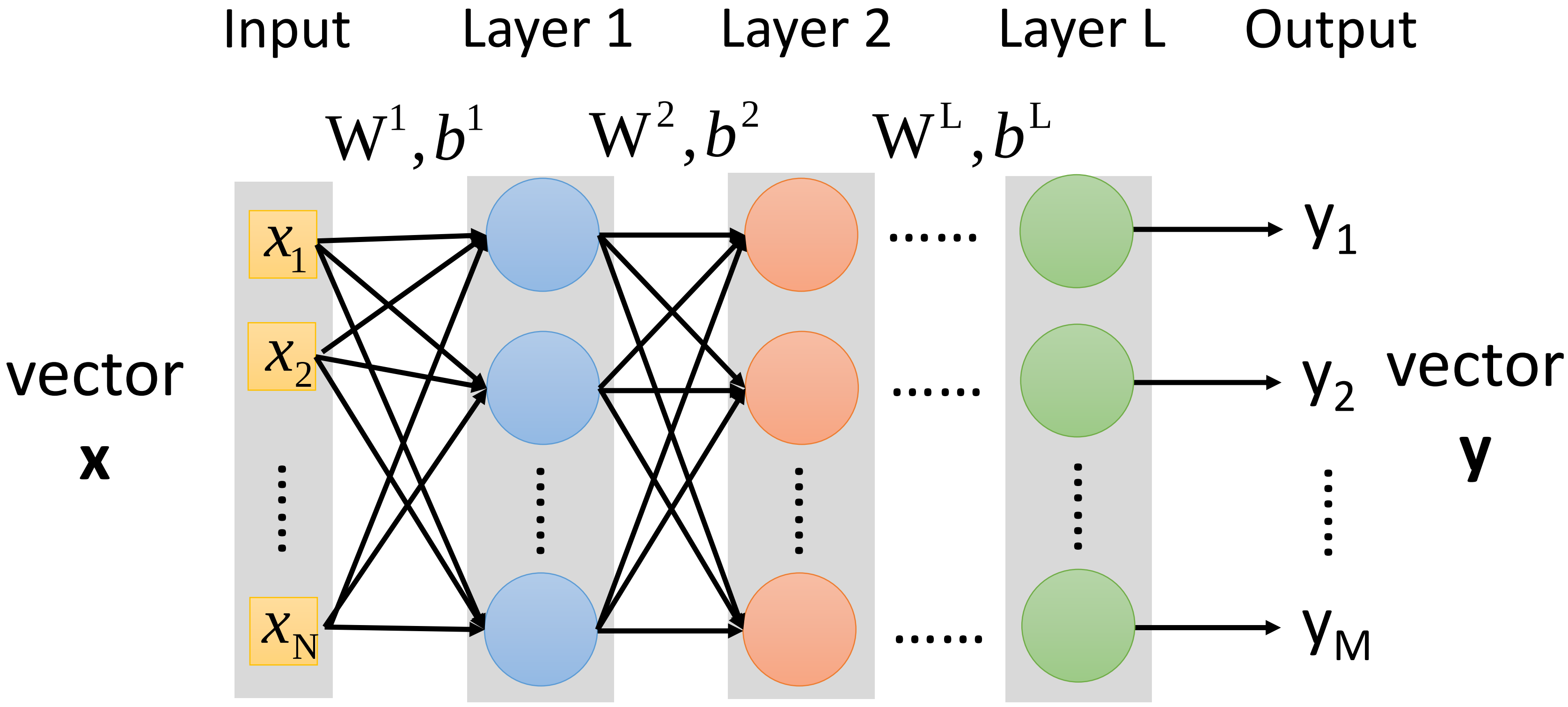


$$\sigma\left(W^1 x + b^1\right) = a^1$$

$$\sigma\left(W^2 a^1 + b^2\right) = a^2$$

$$\sigma\left(W^L a^{L\text{-}1} + b^L\right) = a^L = y$$

Function of Neural Network



$$y = f(x)$$

$$= \sigma\left(W^L \ldots \sigma\left(W^2 \sigma\left(W^1 x + b^1\right) + b^2\right) \ldots + b^L\right)$$

2. What is the “best” function?

Best Function = Best Parameters

$$y = f(x) = \sigma\left(W^L \ldots \sigma\left(W^2 \sigma\left(W^1 x + b^1\right) + b^2\right) \ldots + b^L\right)$$

function set

because different parameters W and b lead to different function

Formal way to define a function set:

$f(x;\underline{\theta})$ → parameter set

$$\theta = \left\{W^1, b^1, W^2, b^2 \cdots W^L, b^L\right\}$$

Pick the “best” function f*

→

Pick the “best” parameter set θ*

Cost Function

- Define a function for parameter set $C(\theta)$
  - $C(\theta)$ evaluate how bad a parameter set is
  - The best parameter set θ^* is the one that minimizes $C(\theta)$

$$\theta^* = arg\min_{\theta} C(\theta)$$

- $C(\theta)$ is called ***cost/loss/error function***
  - If you define the goodness of the parameter set by another function $O(\theta)$
  - $O(\theta)$ is called objective function

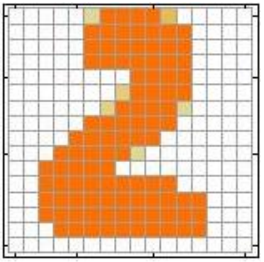
Cost Function

Given training data:

$$\left\{\left(x^1,\hat{y}^1\right)\dots\left(x^r,\hat{y}^r\right)\dots\left(x^R,\hat{y}^R\right)\right\}$$

- ***<u>Handwriting Digit Classification</u>***

sum over all training examples



$$C(\theta)=\frac{1}{R}\sum_r\left\|\underline{f\left(\boxed{x^r};\theta\right)}-\underline{\hat{y}^r}\right\|$$

Minimize distance

$$\left[\begin{array}{c}0.1\\0.4\\0.2\\\vdots\end{array}\right]\begin{array}{l}\text{“1”}\\\text{“2”}\\\text{“3”}\\\end{array}\qquad\left[\begin{array}{c}0\\1\\0\\\vdots\end{array}\right]\begin{array}{l}\text{“1”}\\\text{“2”}\\\text{“3”}\\\end{array}$$

3. How to pick the “best” function?

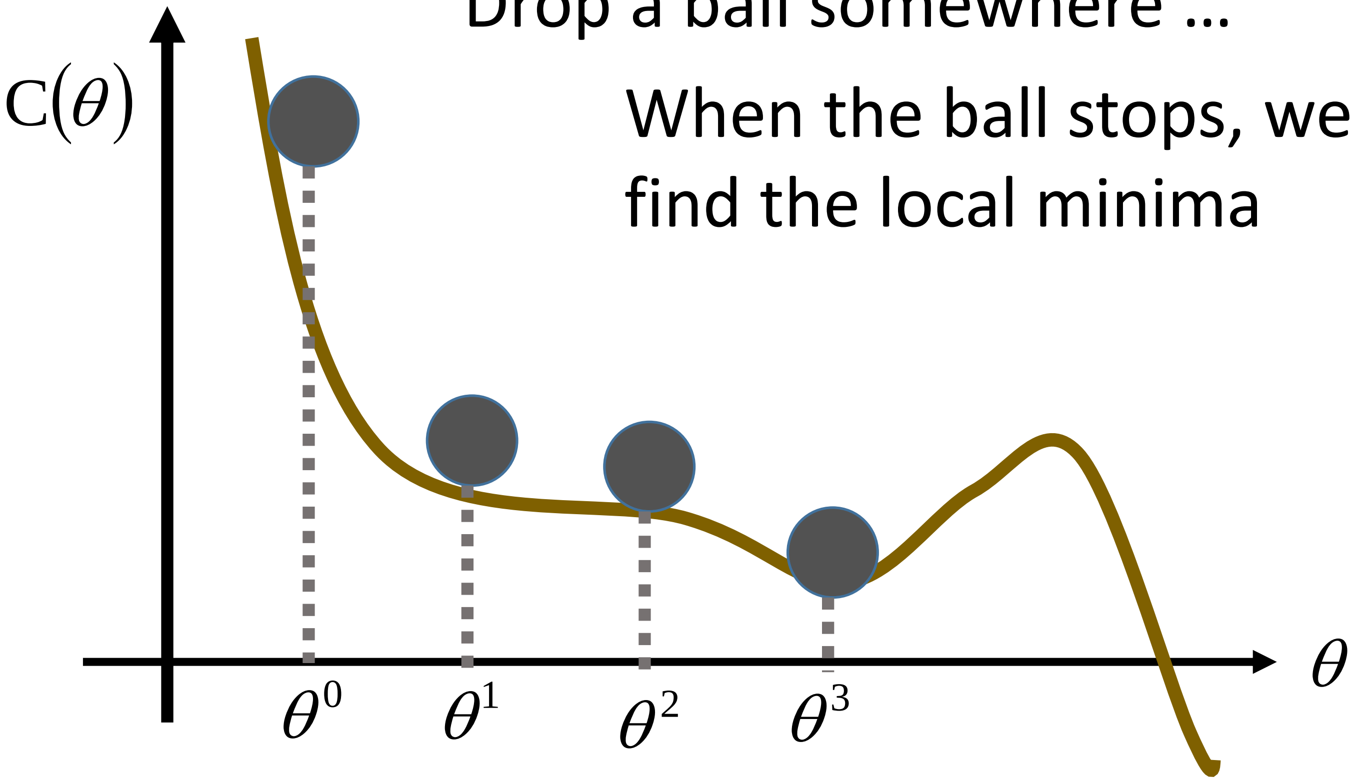
Gradient Descent

Statement of Problems

- Statement of problems:
  - There is a function C(θ)
    - θ represents parameter set
    - $\theta = \{\theta_1, \theta_2, \theta_3, \ldots\ldots\}$
  - Find θ^* that minimizes C(θ)
- Brute force?
  - Enumerate all possible θ
- Calculus?
  - Find θ^* such that $\left.\frac{\partial C(\theta)}{\partial \theta_1}\right|_{\theta=\theta^*} = 0, \left.\frac{\partial C(\theta)}{\partial \theta_2}\right|_{\theta=\theta^*} = 0, \ldots$

Gradient Descent – Idea

- For simplification, first consider that θ has only one variable



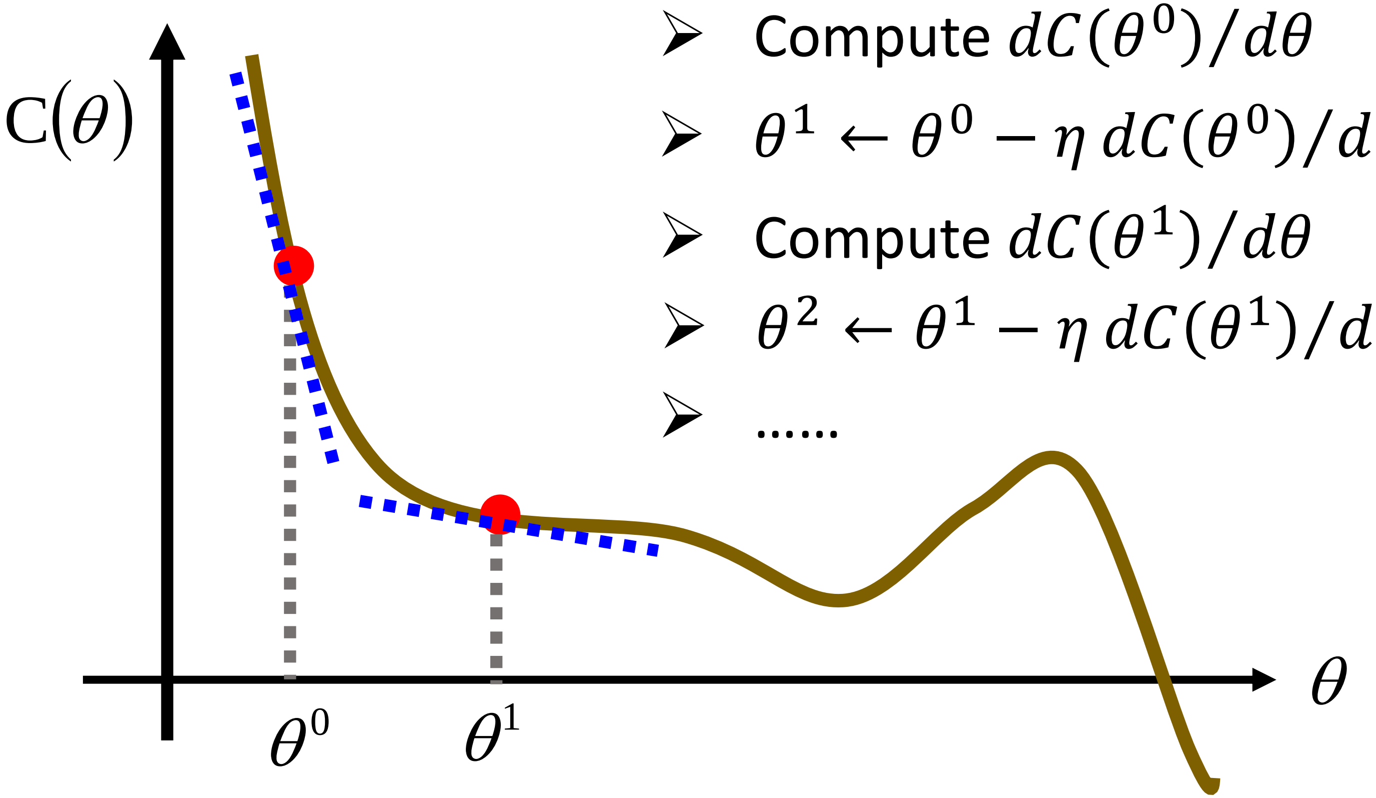
Drop a ball somewhere …

When the ball stops, we find the local minima

Gradient Descent – Idea

η is called *"learning rate"*

- For simplification, first consider that θ has only one variable



- ➢ Randomly start at θ^0
- ➢ Compute $dC(\theta^0)/d\theta$
- ➢ $\theta^1 \leftarrow \theta^0 - \eta\, dC(\theta^0)/d\theta$
- ➢ Compute $dC(\theta^1)/d\theta$
- ➢ $\theta^2 \leftarrow \theta^1 - \eta\, dC(\theta^1)/d\theta$
- ➢ ……

Gradient Descent

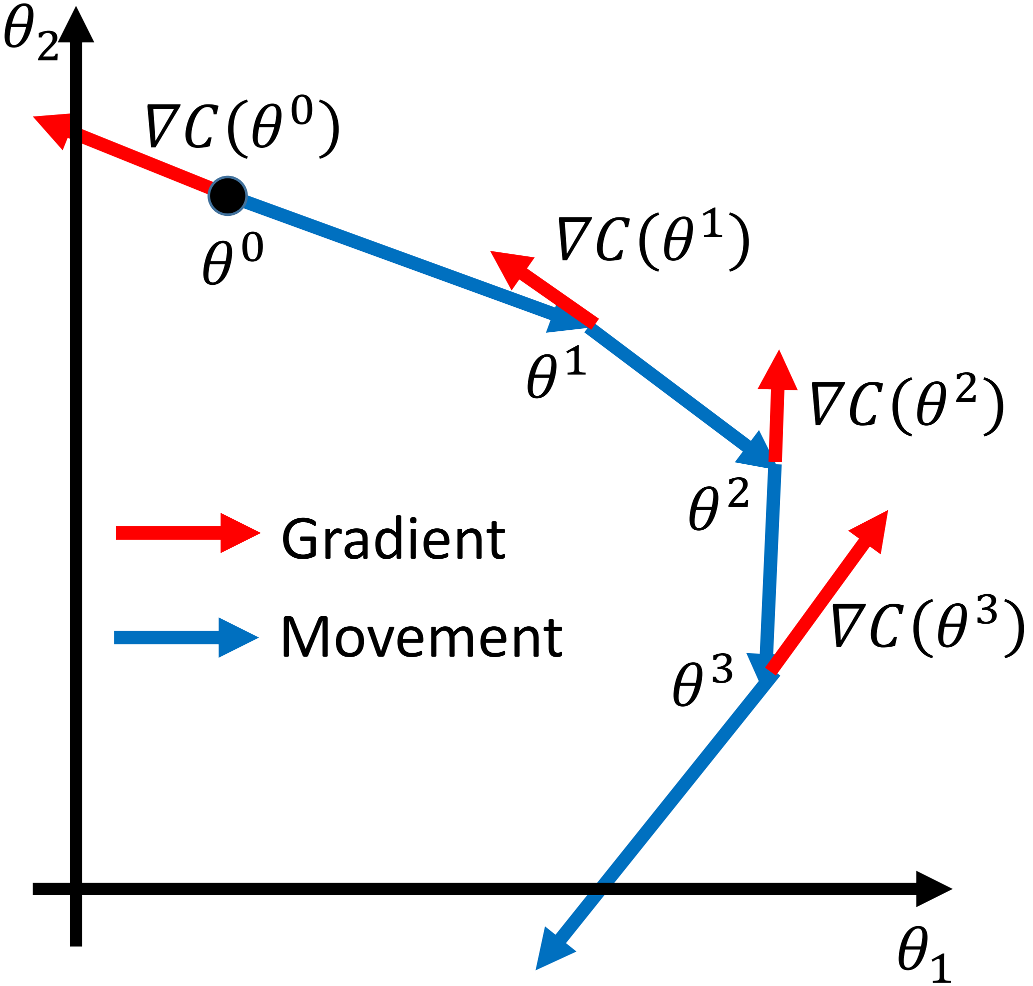
- Suppose that θ has two variables $\{\theta_1, \theta_2\}$

- Randomly start at $\theta^0 = \begin{bmatrix} \theta_1^0 \\ \theta_2^0 \end{bmatrix}$
- Compute the gradients of $C(\theta)$ at θ^0: $\nabla C(\theta^0) = \begin{bmatrix} \partial C(\theta_1^0)/\partial \theta_1 \\ \partial C(\theta_2^0)/\partial \theta_2 \end{bmatrix}$
- Update parameters

$$\begin{bmatrix} \theta_1^1 \\ \theta_2^1 \end{bmatrix} = \begin{bmatrix} \theta_1^0 \\ \theta_2^0 \end{bmatrix} - \eta \begin{bmatrix} \partial C(\theta_1^0)/\partial \theta_1 \\ \partial C(\theta_2^0)/\partial \theta_2 \end{bmatrix} \quad ➡ \quad \theta^1 = \theta^0 - \eta \nabla C(\theta^0)$$

- Compute the gradients of $C(\theta)$ at θ^1: $\nabla C(\theta^1) = \begin{bmatrix} \partial C(\theta_1^1)/\partial \theta_1 \\ \partial C(\theta_2^1)/\partial \theta_2 \end{bmatrix}$
- ……

Gradient Descent



Start at position θ^0

Compute gradient at θ^0

Move to $\theta^1 = \theta^0 - \eta \nabla C(\theta^0)$

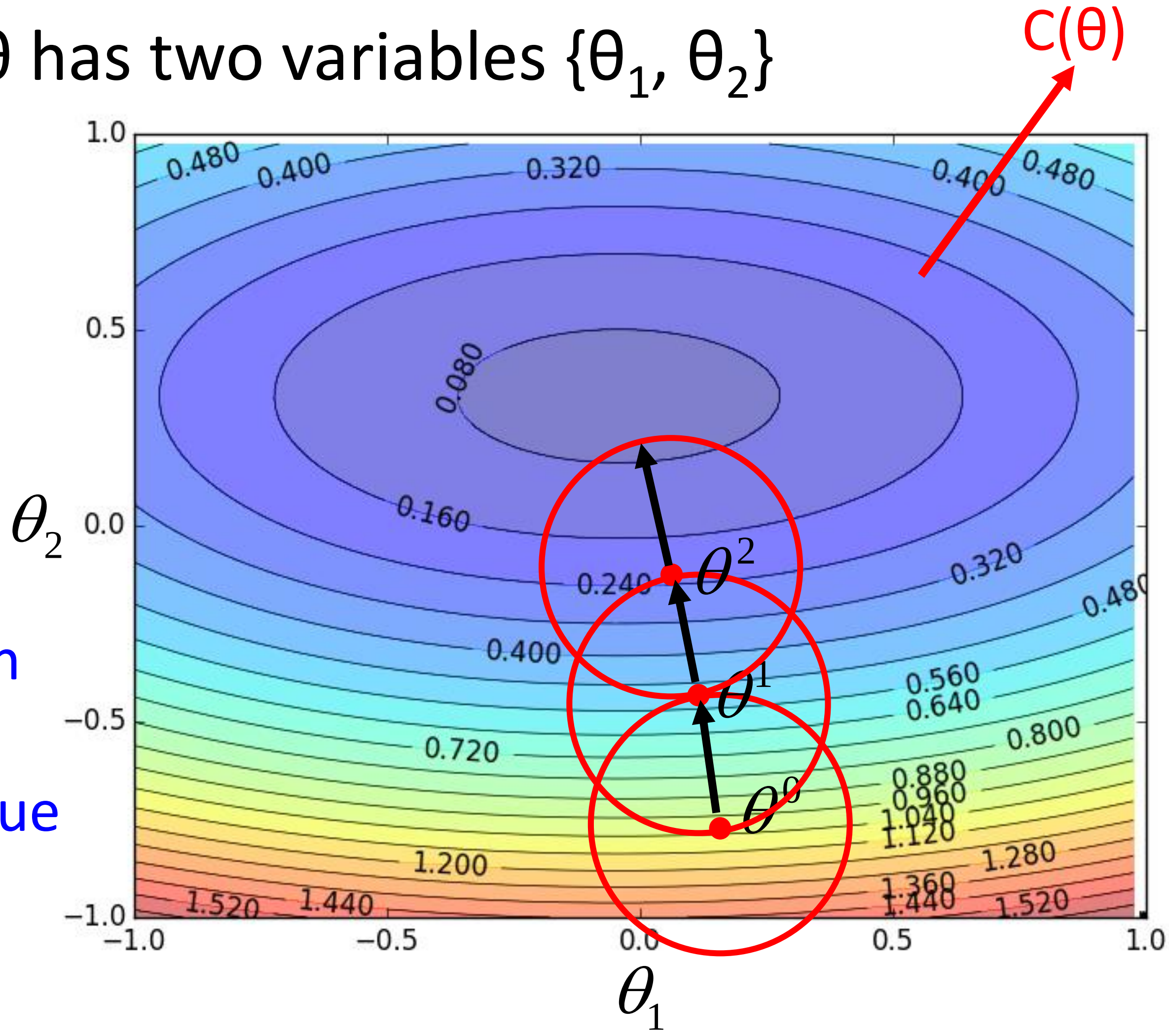
Compute gradient at θ^1

Move to $\theta^2 = \theta^1 - \eta \nabla C(\theta^1)$

⋮

Formal Derivation of Gradient Descent

- Suppose that θ has two variables $\{\theta_1, \theta_2\}$



Given a point, we can easily find the point with the smallest value nearby. How?

Formal Derivation of Gradient Descent

- **Taylor series**: Let h(x) be infinitely differentiable around x = x_0.

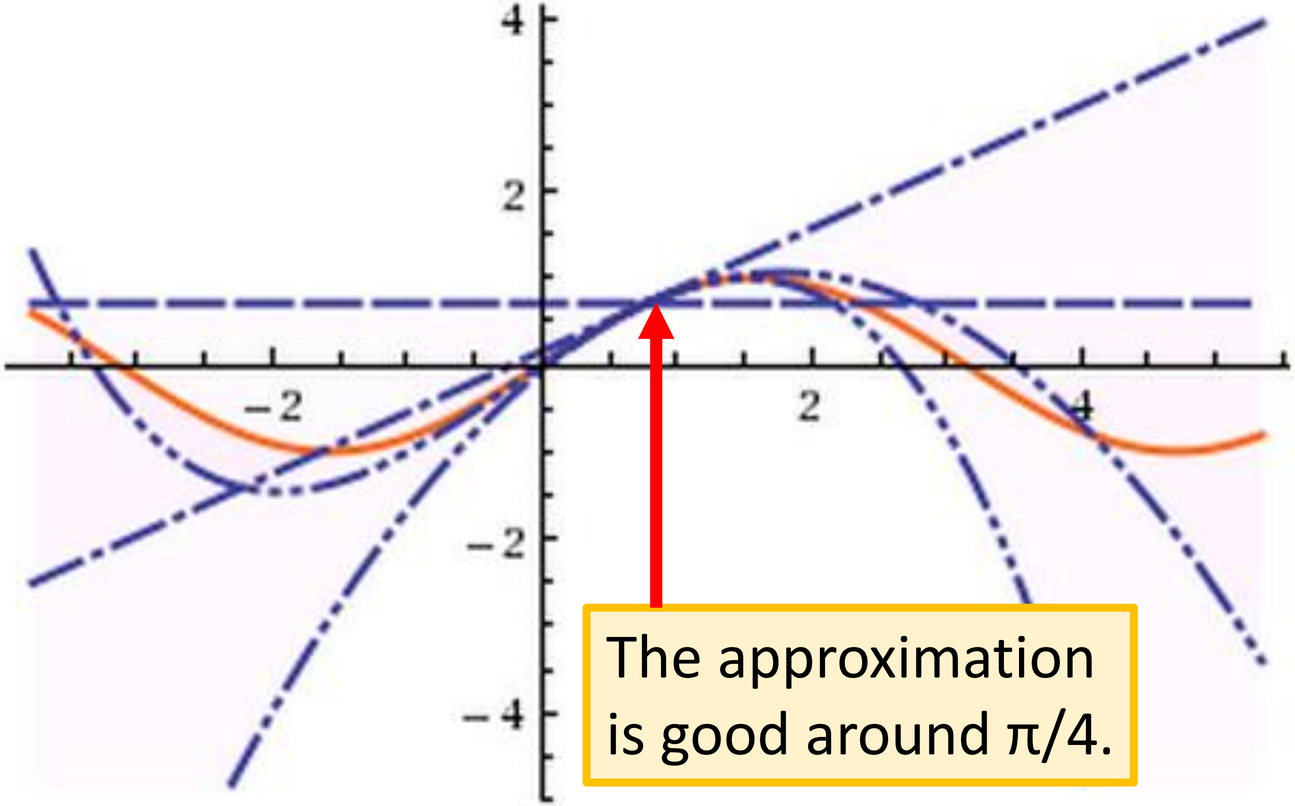
$$\mathrm{h}(\mathrm{x}) = \sum_{k=0}^{\infty} \frac{\mathrm{h}^{(k)}(x_0)}{k!}(x - x_0)^k$$

$$= h(x_0) + h'(x_0)(x - x_0) + \frac{h''(x_0)}{2!}(x - x_0)^2 + \ldots$$

When x is close to x_0 ⇨ $h(x) \approx h(x_0) + h'(x_0)(x - x_0)$

E.g. Taylor series for h(x)=sin(x) around $x_0=\pi/4$

$$\sin(x)= \frac{1}{\sqrt{2}} + \frac{x-\frac{\pi}{4}}{\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^2}{2\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^3}{6\sqrt{2}} + \frac{\left(x-\frac{\pi}{4}\right)^4}{24\sqrt{2}} + \frac{\left(x-\frac{\pi}{4}\right)^5}{120\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^6}{720\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^7}{5040\sqrt{2}} + \frac{\left(x-\frac{\pi}{4}\right)^8}{40\,320\sqrt{2}} + \frac{\left(x-\frac{\pi}{4}\right)^9}{362\,880\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^{10}}{3\,628\,800\sqrt{2}} + \ldots\ldots$$



The approximation is good around π/4.

Multivariable Taylor series

$$h(x,y)=h(x_0,y_0)+\frac{\partial h(x_0,y_0)}{\partial x}(x-x_0)+\frac{\partial h(x_0,y_0)}{\partial y}(y-y_0)$$

+ something related to $(x-x_0)^2$ and $(y-y_0)^2$ + ……

When x and y is close to x_0 and y_0

⬇

$$h(x,y)\approx h(x_0,y_0)+\frac{\partial h(x_0,y_0)}{\partial x}(x-x_0)+\frac{\partial h(x_0,y_0)}{\partial y}(y-y_0)$$

Formal Derivation of Gradient Descent

Based on Taylor Series:

If the red circle is ***small enough***, in the red circle

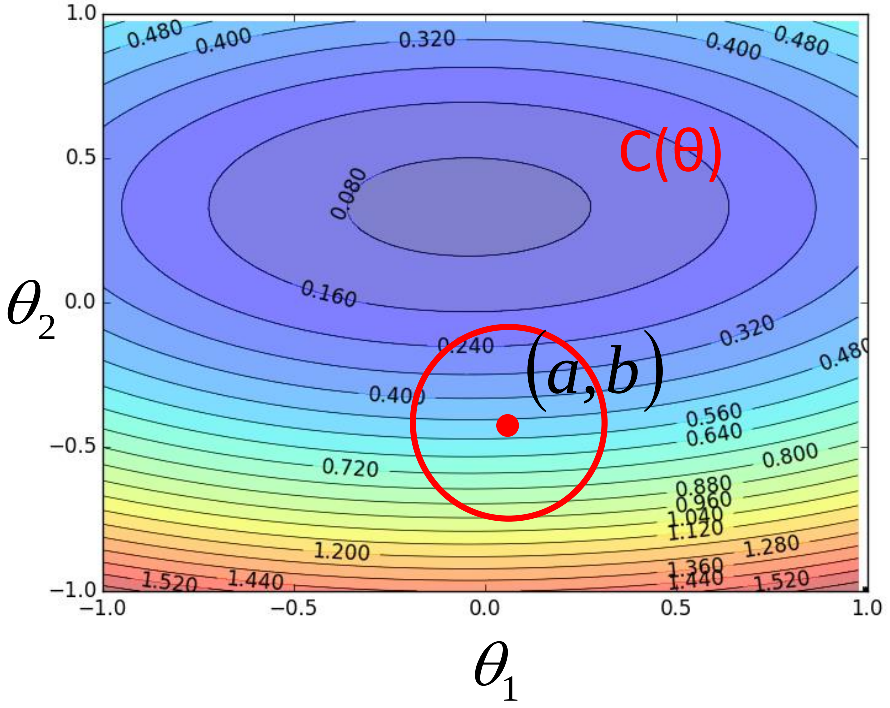
$$C(\theta) \approx C(a,b) + \frac{\partial C(a,b)}{\partial \theta_1}(\theta_1 - a) + \frac{\partial C(a,b)}{\partial \theta_2}(\theta_2 - b)$$

$$s = C(,b)$$

$$u = \frac{\partial C(a,b)}{\partial \theta_1}, v = \frac{\partial C(a,b)}{\partial \theta_2}$$

$$C(\theta)$$

$$\approx s + u(\theta_1 - a) + v(\theta_2 - b)$$



Formal Derivation of Gradient Descent

$$u = \frac{\partial \mathrm{C}(a,b)}{\partial \theta_1}, v = \frac{\partial \mathrm{C}(a,b)}{\partial \theta_2}$$

Based on Taylor Series:

If the red circle is ***small enough***, in the red circle

$$\mathrm{C}(\theta) \approx s + u(\theta_1 - a) + v(\theta_2 - b)$$

Find θ_1 and θ_2 yielding the smallest value of $C(\theta)$ in the circle

$$\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix} - \eta \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial \mathrm{C}(a,b)}{\partial \theta_1} \\ \frac{\partial \mathrm{C}(a,b)}{\partial \theta_2} \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix} - \eta \nabla \mathrm{C}(a,b)$$

Its value depending on the radius of the circle, u and v.

This is how gradient descent updates parameters.

Gradient Descent for Neural Network

$$Compute\ \nabla C\left(\theta^0\right)$$

$$\theta^1 = \theta^0 - \eta \nabla C\left(\theta^0\right)$$

$$Compute\ \nabla C\left(\theta^1\right)$$

$$\theta^2 = \theta^1 - \eta \nabla C\left(\theta^1\right)$$

Starting Parameters

$$\theta^0 \longrightarrow \theta^1 \longrightarrow \theta^2 \longrightarrow \ldots\ldots$$

$$\nabla C(\theta) = \begin{bmatrix} \vdots \\ \dfrac{\partial C(\theta)}{\partial w_{ij}^l} \\ \vdots \\ \dfrac{\partial C(\theta)}{\partial b_i^l} \\ \vdots \end{bmatrix}$$

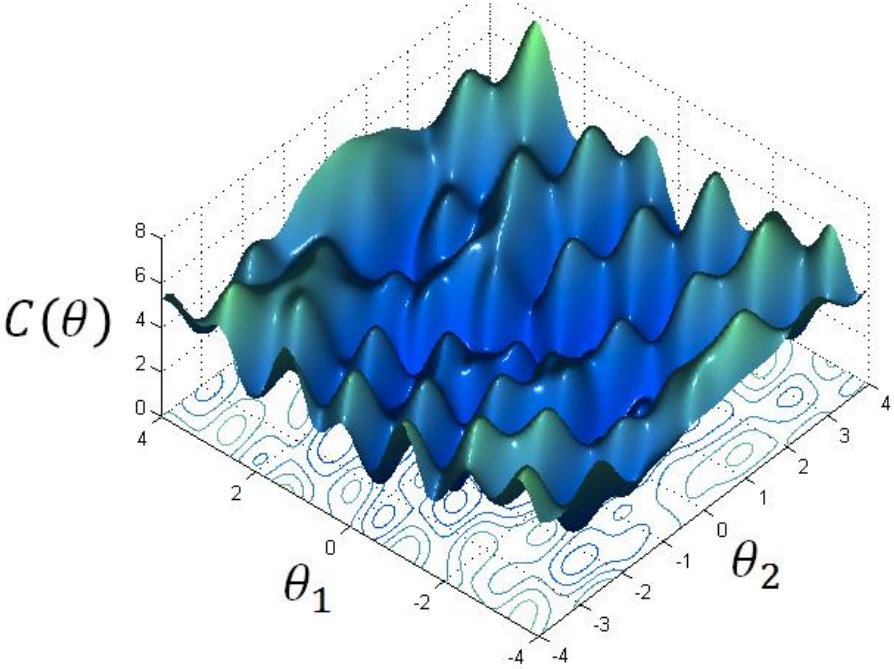
$$\theta = \left\{ W^1, b^1, W^2, b^2, \cdots, W^l, b^l, \cdots, W^L, b^L \right\}$$

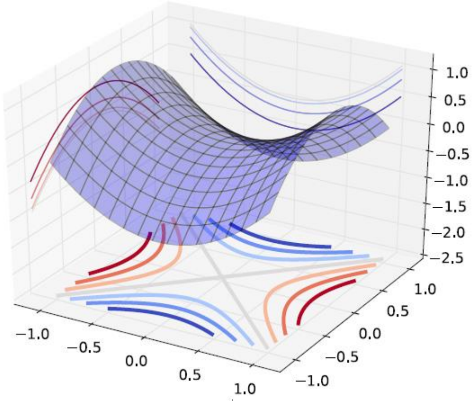
$$\begin{bmatrix} w_{11}^l & w_{12}^l & \cdots \\ w_{21}^l & w_{22}^l & \\ \vdots & & \ddots \\ & & \end{bmatrix} \qquad \begin{bmatrix} b_1^l \\ b_2^l \\ \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$

Millions of parameters ……

To compute the gradients efficiently, we use ***backpropagation***.

Stuck at local minima?





Saddle point

- Who is Afraid of Non-Convex Loss Functions?
- http://videolectures.net/eml07_lecun_wia/
- Deep Learning: Theoretical Motivations
- http://videolectures.net/deeplearning2015_bengio_theoretical_motivations/

3. How to pick
the “best” function?

Practical Issues
for neural network

Practical Issues for neural network

- Parameter Initialization
- Learning Rate
- Stochastic gradient descent and Mini-batch
- Recipe for Learning

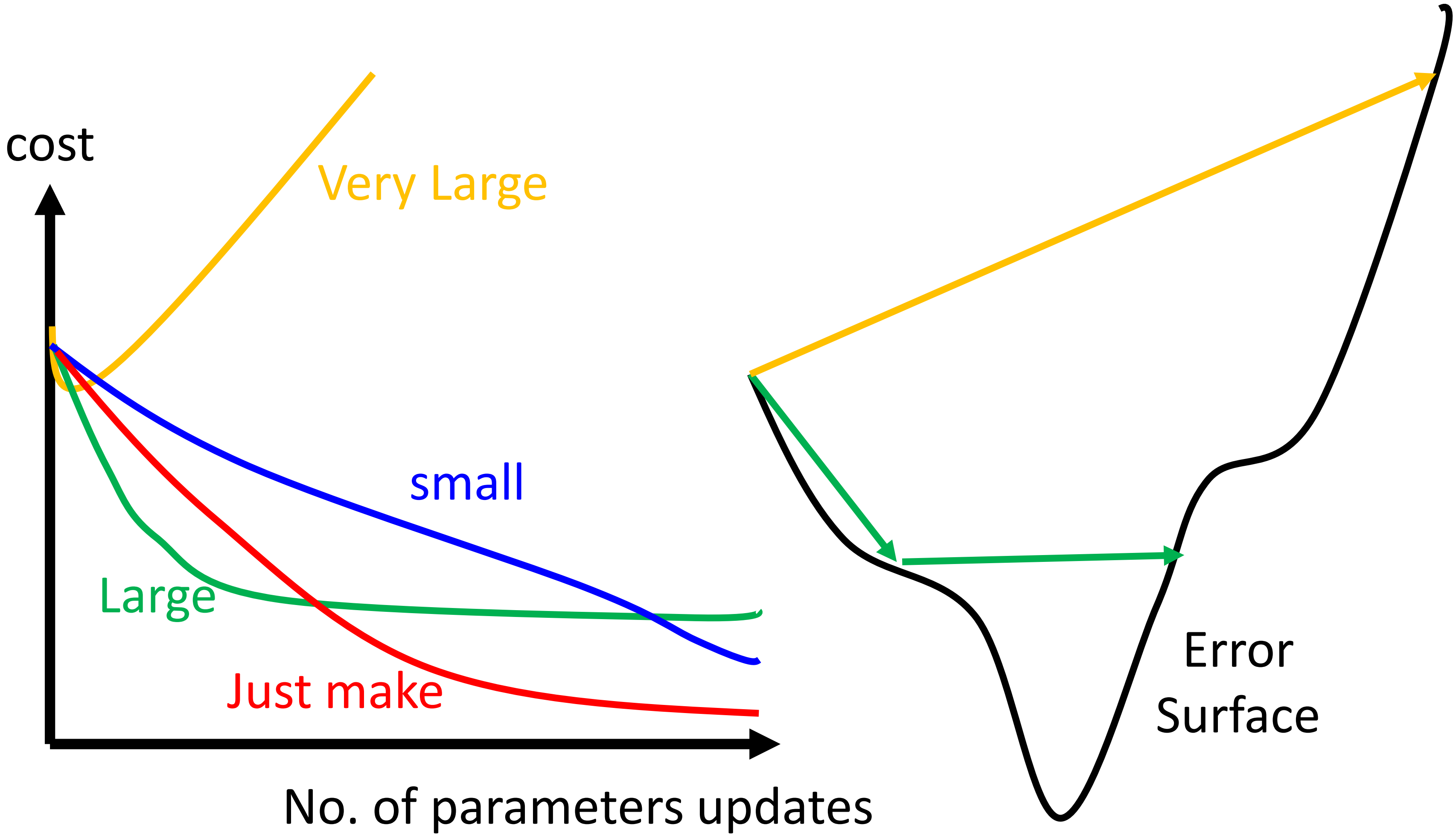
Parameter Initialization

- For gradient Descent, we need to pick an initialization parameter θ^0.
- The initialization parameters have some influence to the training.
  - We will go back to this issue in the future.
- Suggestion today:
  - Do not set all the parameters θ^0 equal
  - Set the parameters in θ^0 randomly

Learning Rate

$$\theta^{i} = \theta^{i-1} - \eta \nabla C\left(\theta^{i-1}\right)$$

- Set the learning rate η carefully

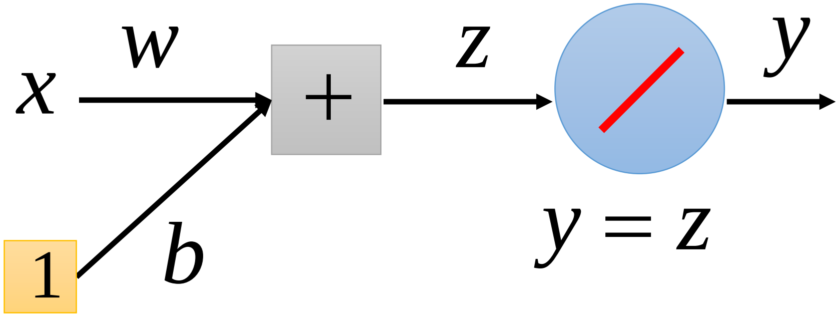


Learning Rate

$$\theta^{i} = \theta^{i-1} - \eta \nabla C\left(\theta^{i-1}\right)$$

- Set the learning rate η carefully

- ***Toy Example***



$$\theta^{*} = \begin{bmatrix} w = 1 \\ b = 0 \end{bmatrix}$$

Training Data (20 examples)

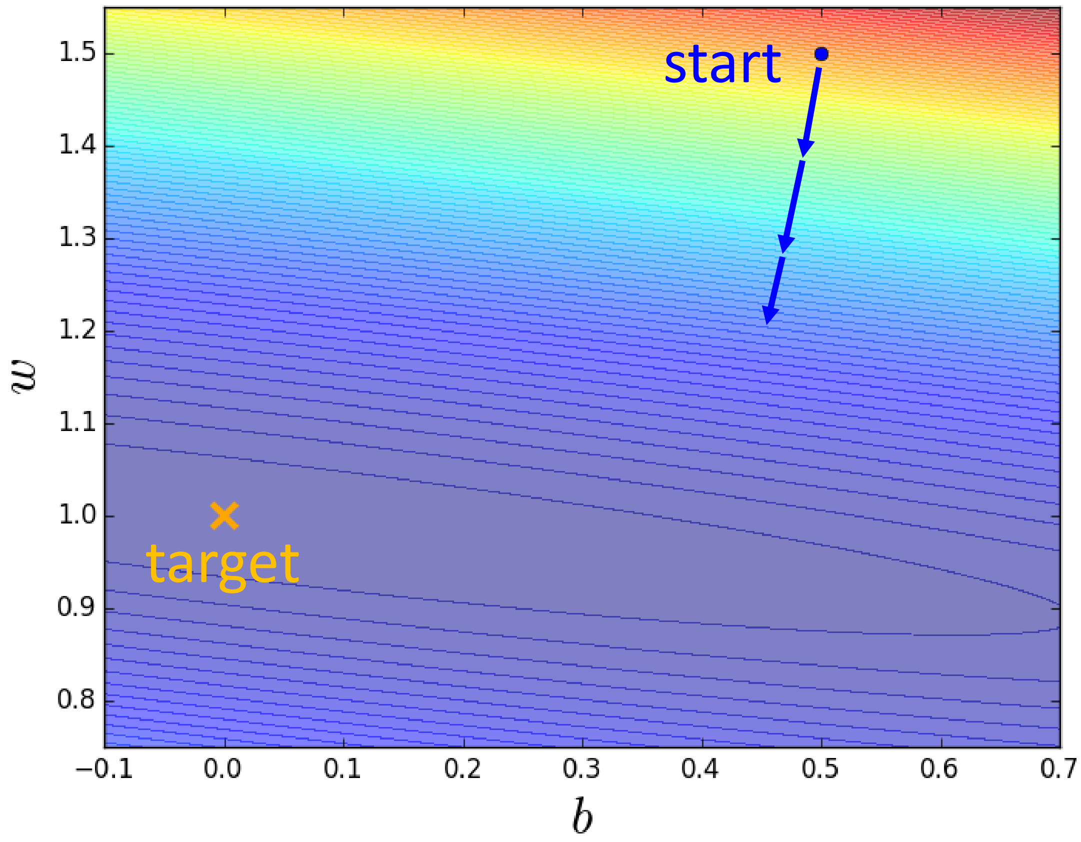
x = [0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0, 6.5, 7.0, 7.5, 8.0, 8.5, 9.0, 9.5]
y = [0.1, 0.4, 0.9, 1.6, 2.2, 2.5, 2.8, 3.5, 3.9, 4.7, 5.1, 5.3, 6.3, 6.5, 6.7, 7.5, 8.1, 8.5, 8.9, 9.5]

Learning Rate

$$\theta^{i} = \theta^{i-1} - \eta \nabla C\left(\theta^{i-1}\right)$$

- ***Toy Example***

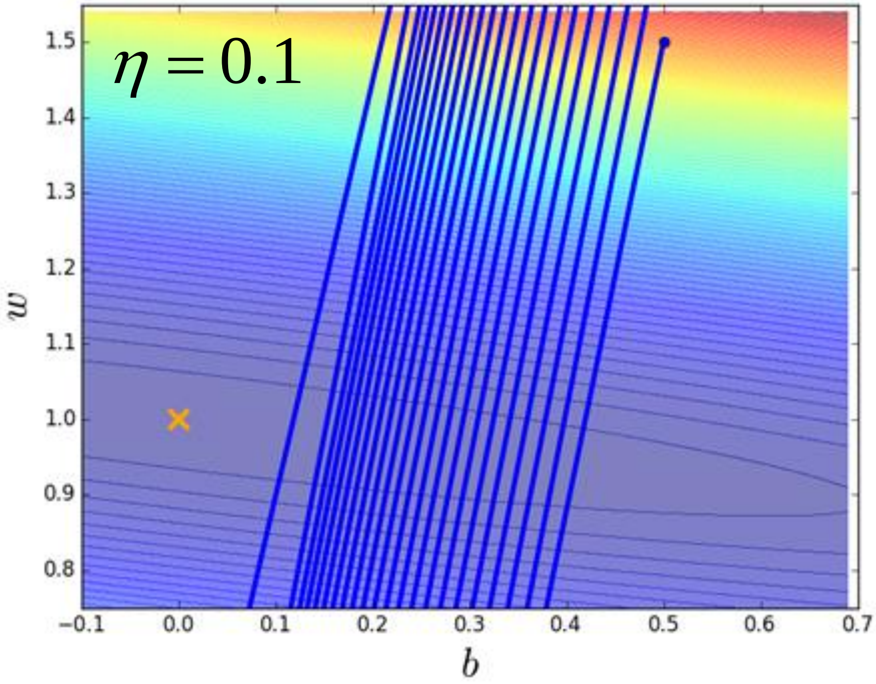
Error Surface: C(w,b)

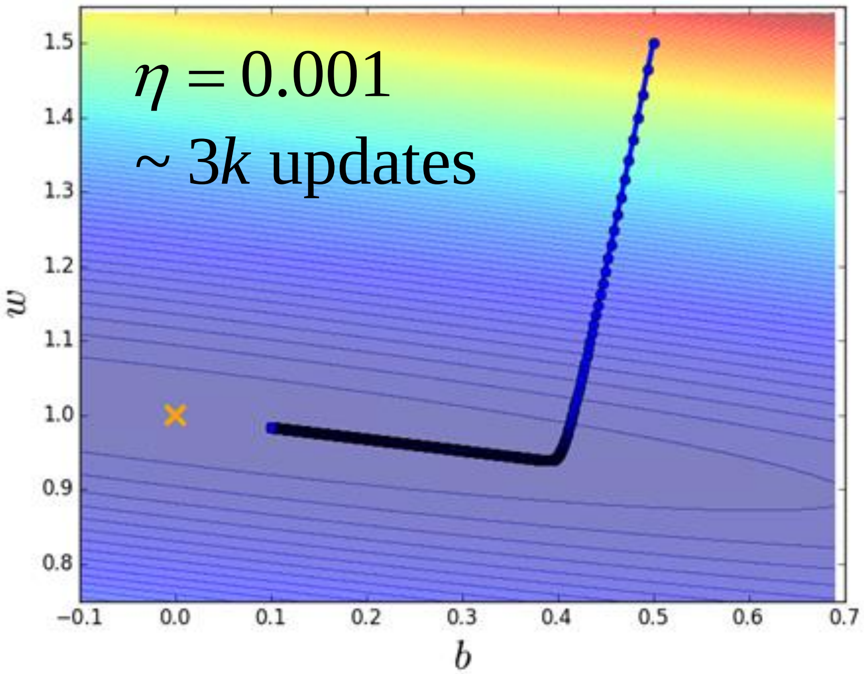


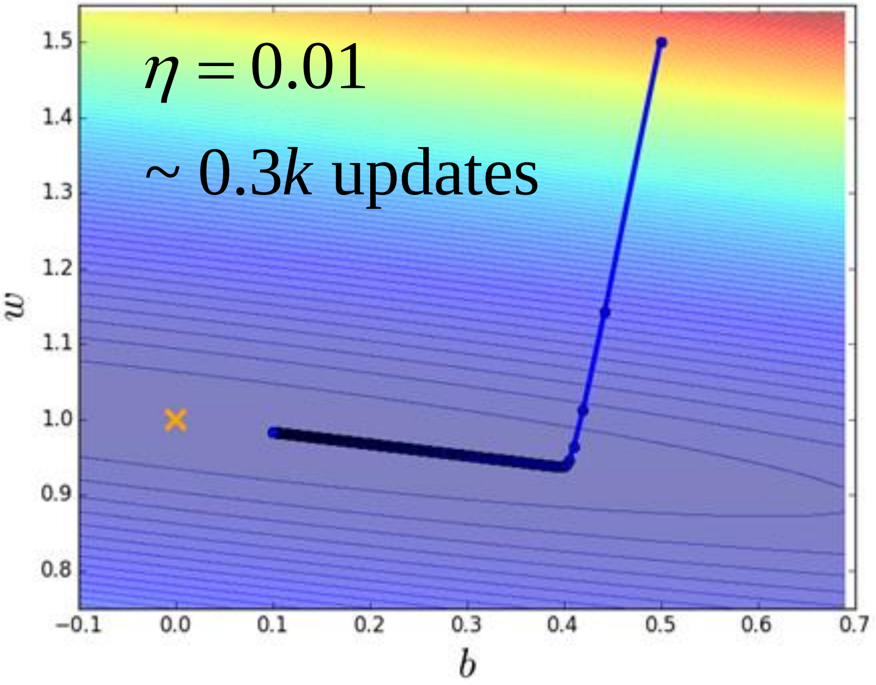
Learning Rate

- ***Toy Example***

  Different learning rate η







Stochastic Gradient Descent and Mini-batch

$$C(\theta)=\frac{1}{R}\sum_r\left\|f\left(x^r;\theta\right)-\hat{y}^r\right\|$$

$$=\frac{1}{R}\sum_r \mathrm{C}^r(\theta)$$

- ***Gradient Descent***

$$\theta^i=\theta^{i-1}-\eta\nabla C\left(\theta^{i-1}\right)$$

$$\nabla C\left(\theta^{i-1}\right)=\frac{1}{R}\sum_r\nabla C^r\left(\theta^{i-1}\right)$$

- ***Stochastic Gradient Descent***

Faster! Better!

Pick an example x^r

$$\theta^i=\theta^{i-1}-\eta\nabla C^r\left(\theta^{i-1}\right)$$

If all example x^r have equal probabilities to be picked

$$E\left[\nabla C^r\left(\theta^{i-1}\right)\right]=\frac{1}{R}\sum_r\nabla C^r\left(\theta^{i-1}\right)$$

Stochastic Gradient Descent and Mini-batch

What is epoch?

Training Data: $\left\{\left(x^1, \hat{y}^1\right), \left(x^2, \hat{y}^2\right), \cdots \left(x^r, \hat{y}^r\right), \cdots \left(x^R, \hat{y}^R\right)\right\}$

When using stochastic gradient descent

Starting at θ_0

pick x^1 $\theta^1 = \theta^0 - \eta \nabla C^1\left(\theta^0\right)$

pick x^2 $\theta^2 = \theta^1 - \eta \nabla C^2\left(\theta^1\right)$

⋮ ⋮

pick x^r $\theta^r = \theta^{r-1} - \eta \nabla C^r\left(\theta^{r-1}\right)$

⋮ ⋮

pick x^R $\theta^{\mathrm{R}} = \theta^{\mathrm{R}-1} - \eta \nabla C^{\mathrm{R}}\left(\theta^{\mathrm{R}-1}\right)$

Seen all the examples once

One epoch

---

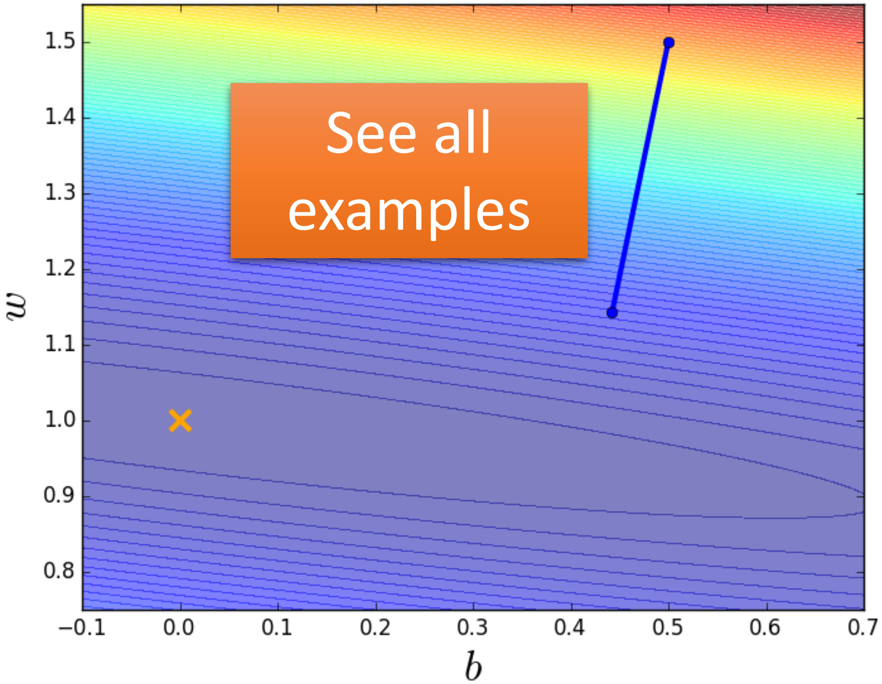
pick x^1 $\theta^{\mathrm{R}+1} = \theta^{\mathrm{R}} - \eta \nabla C^1\left(\theta^{\mathrm{R}}\right)$

Stochastic Gradient Descent and Mini-batch

- ***Toy Example***

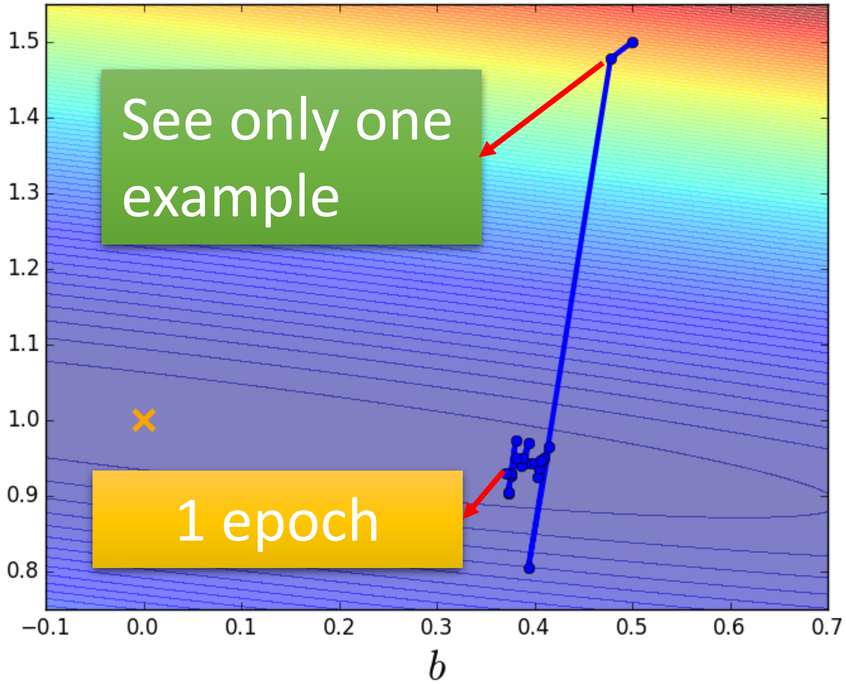
Gradient Descent

Update after seeing all examples



Stochastic Gradient Descent

If there are 20 examples, update 20 times in one epoch.



Stochastic Gradient Descent and Mini-batch

◆ ***Gradient Descent***

$$\theta^i = \theta^{i-1} - \eta \nabla C\left(\theta^{i-1}\right) \qquad \nabla C\left(\theta^{i-1}\right) = \frac{1}{R}\sum_r \nabla C^r\left(\theta^{i-1}\right)$$

◆ ***Stochastic Gradient Descent***

Pick an example x_r

$$\theta^i = \theta^{i-1} - \eta \nabla C^r\left(\theta^{i-1}\right)$$

◆ ***Mini Batch Gradient Descent***

Pick B examples as a batch b

B is batch size

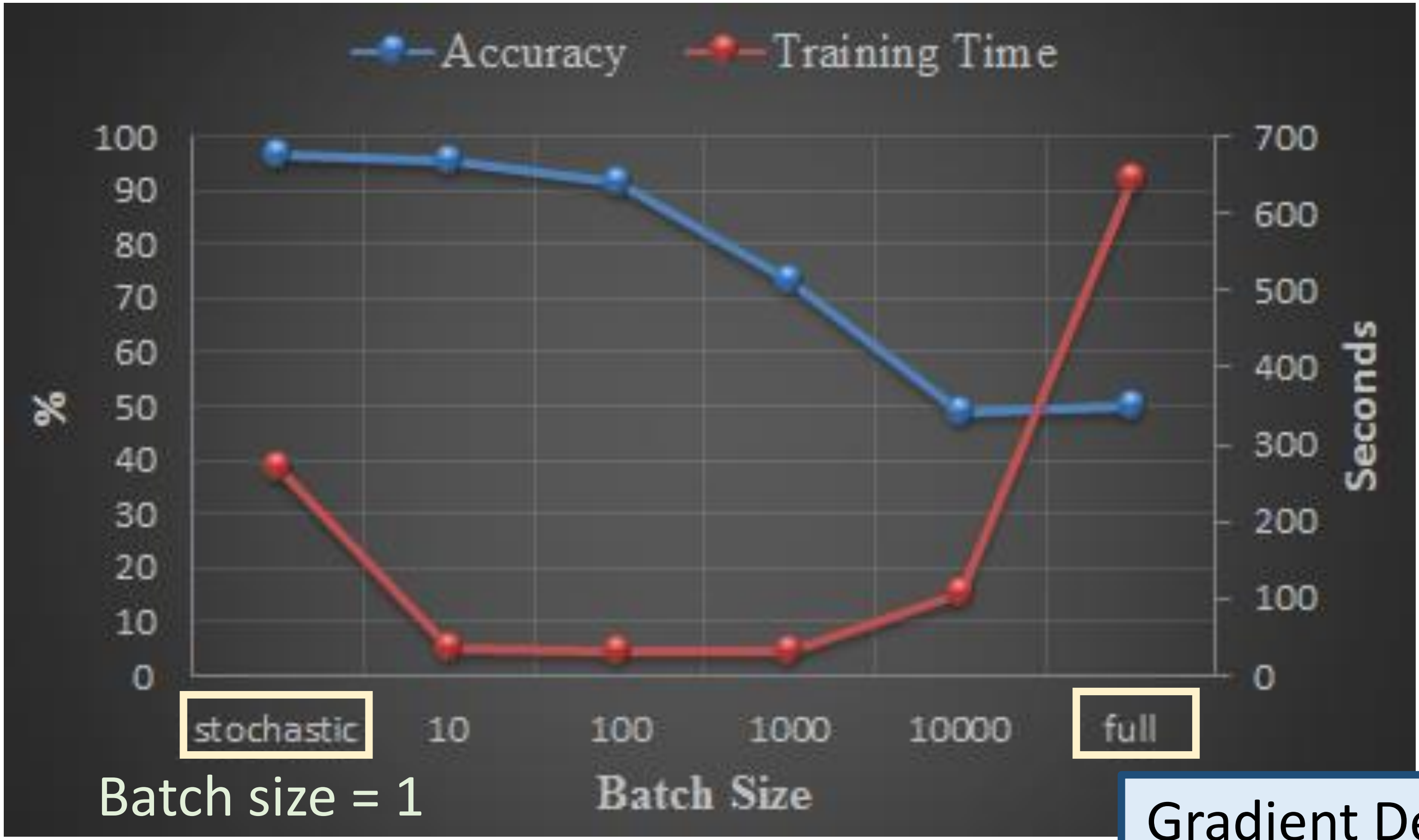
Shuffle your data

$$\theta^i = \theta^{i-1} - \eta \underline{\frac{1}{B}\sum_{x_r \in b} \nabla C^r\left(\theta^{i-1}\right)}$$

Average the gradient of the examples in the batch b

Stochastic Gradient Descent and Mini-batch

- ***Handwriting Digit Classification***



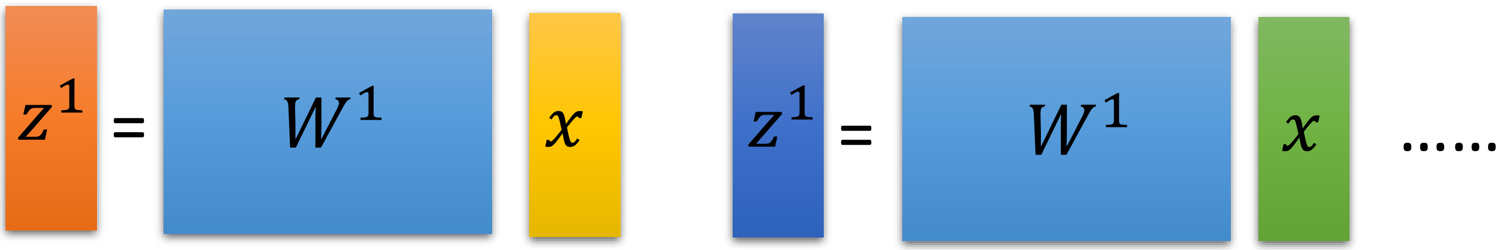
Batch size = 1

Gradient Descent

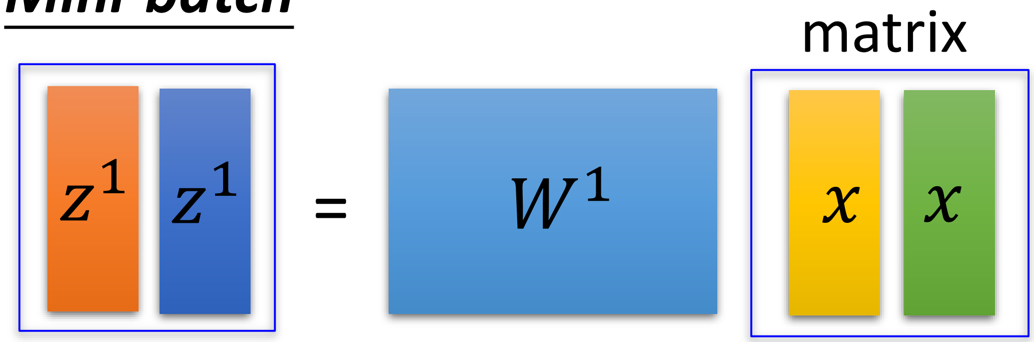
Stochastic Gradient Descent and Mini-batch

- Why mini-batch is faster than stochastic gradient descent?

<u>Stochastic Gradient Descent</u>



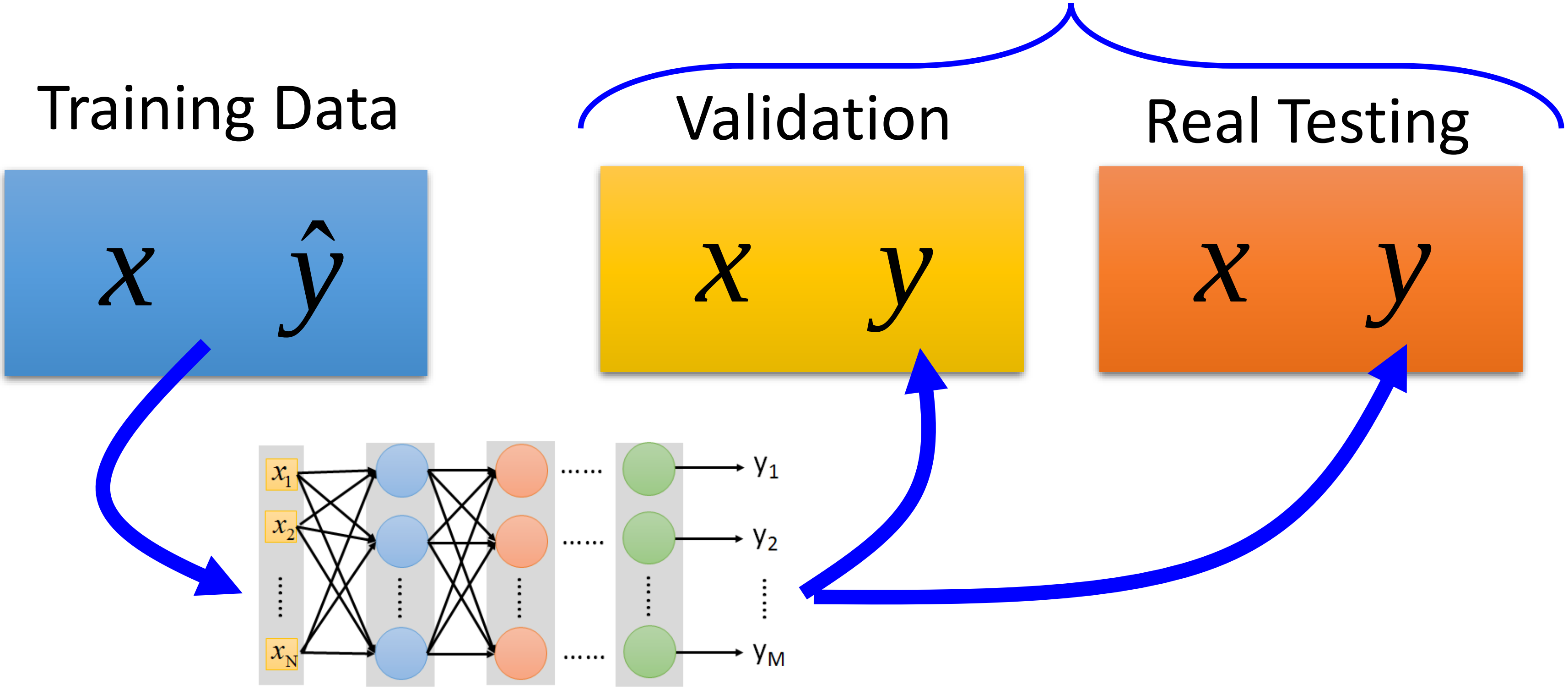
Mini-batch



Practically, which one is faster?

Recipe for Learning

- Data provided in homework



Testing Data

Training Data

Validation

Real Testing

x $\hat{y}$

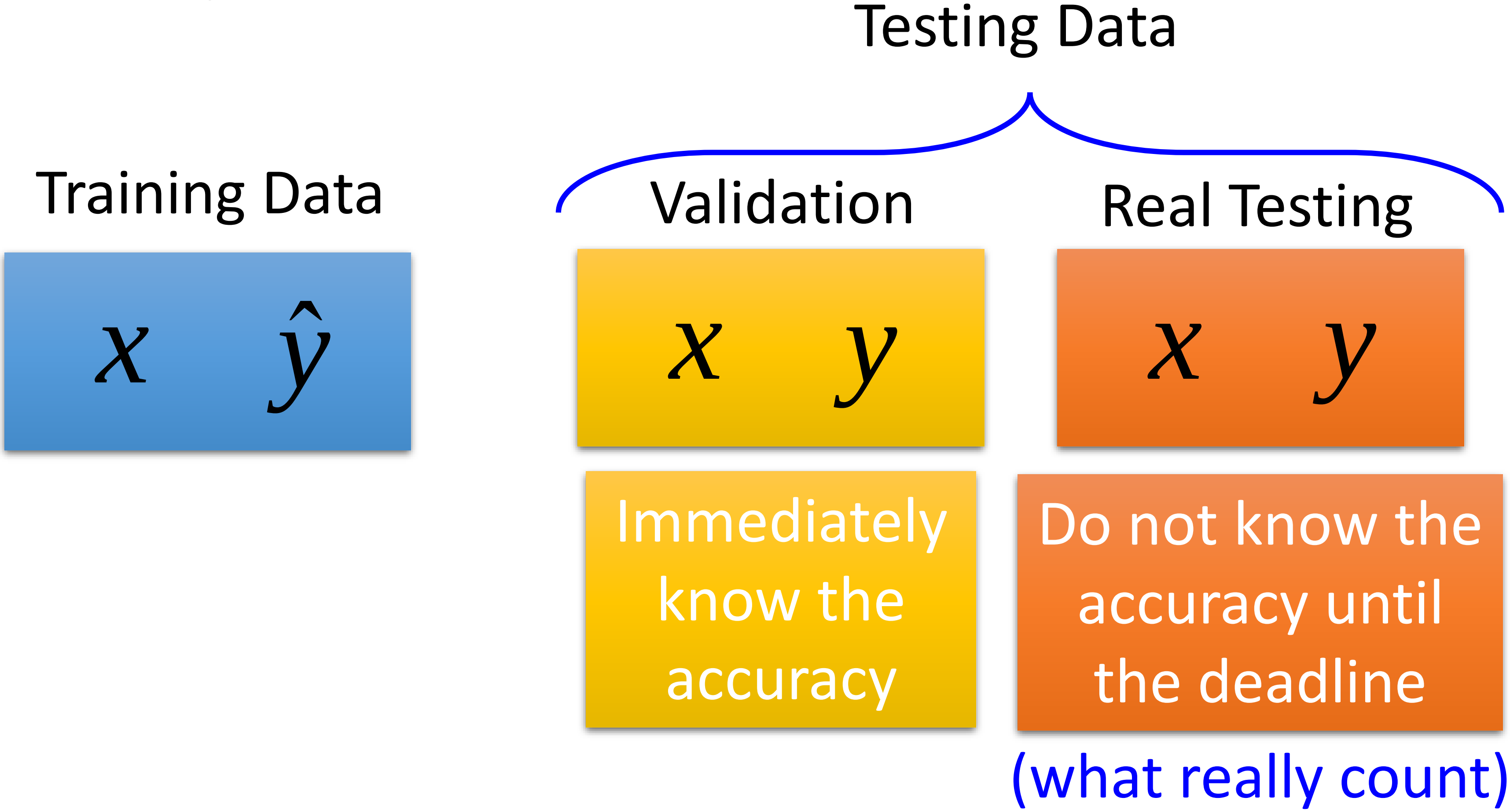
x y

x y

"Best" Function f^*

Recipe for Learning

- Data provided in homework



Testing Data

Training Data

x $\hat{y}$

Validation

x y

Immediately know the accuracy

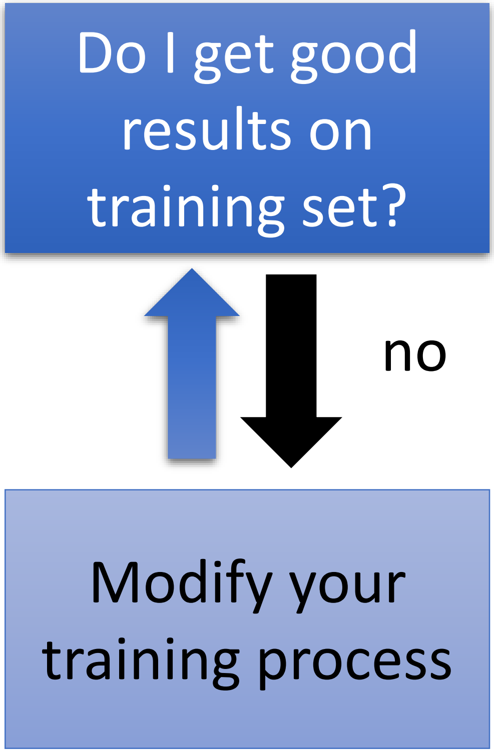
Real Testing

x y

Do not know the accuracy until the deadline

(what really count)

Recipe for Learning



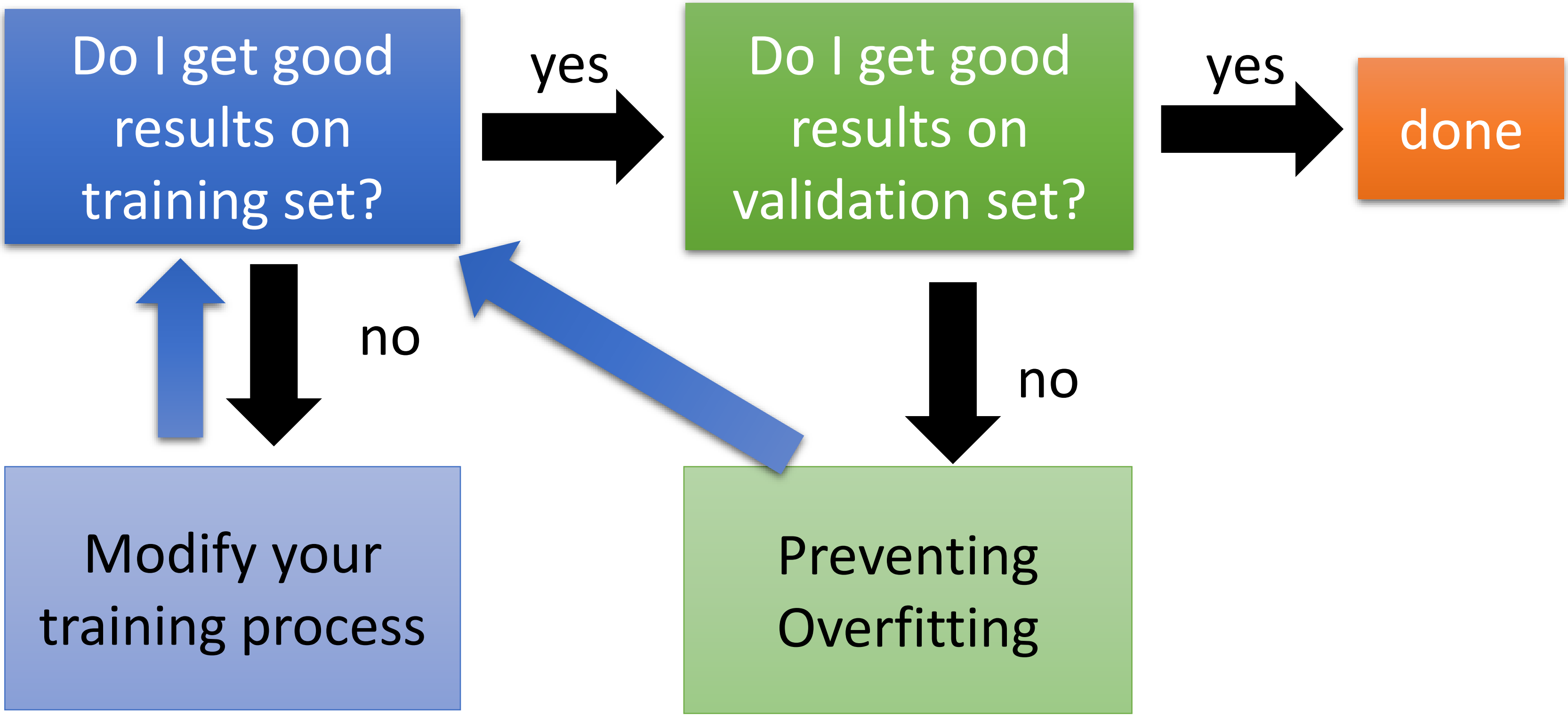
Do I get good results on training set?

no

Modify your training process

- Your code has bug.
- Can not find a good function
  - Stuck at local minima, saddle points ….
  - Change the training strategy
- Bad model
  - There is no good function in the hypothesis function set.
  - Probably you need bigger network

Recipe for Learning



Do I get good results on training set?

yes

Do I get good results on validation set?

yes

done

no

Modify your training process

no

Preventing Overfitting

- Your code usually do not have bug at this situation.

Recipe for Learning - Overfitting

- You pick a "best" parameter set θ^*

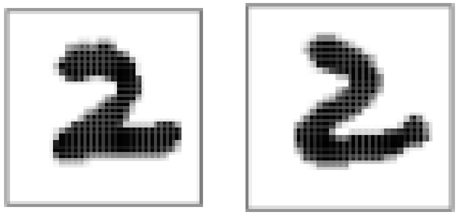
Training Data: $\{\ldots(x^r, \hat{y}^r)\ldots\} \Rightarrow \forall r: f(x^r;\theta^*) = \hat{y}^r$

However,

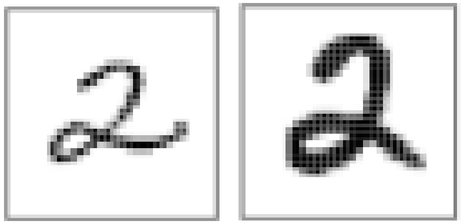
Testing Data: $\{\ldots x^u \ldots\} \Rightarrow f(x^u;\theta^*) \neq \hat{y}^u$

Training data and testing data have different distribution.

Training Data:



Testing Data:



Recipe for Learning - Overfitting

- Panacea: Have more training data
  - You can do that in real application, but you can’t do that in homework.
- We will go back to this issue in the future.

Concluding Remarks

1. What is the model (function hypothesis set)?

Neural Network

2. What is the "best" function?

Cost Function

3. How to pick the "best" function?

Gradient Descent

- Parameter Initialization
- Learning Rate
- Stochastic gradient descent, Mini-batch
- Recipe for Learning

Acknowledgement

- 感謝 余朗祺 同學於上課時糾正投影片上的拼字錯誤
- 感謝 吳柏瑜 同學糾正投影片上的 notation 錯誤
- 感謝 Yes Huang 糾正投影片上的打字錯誤