Introduction to Supervised Learning

Jakub Powierza

Based on resources shared by Karol Draszawka M.Sc.
“Analysis Methods for Big Data”, Lecture 2

Agenda

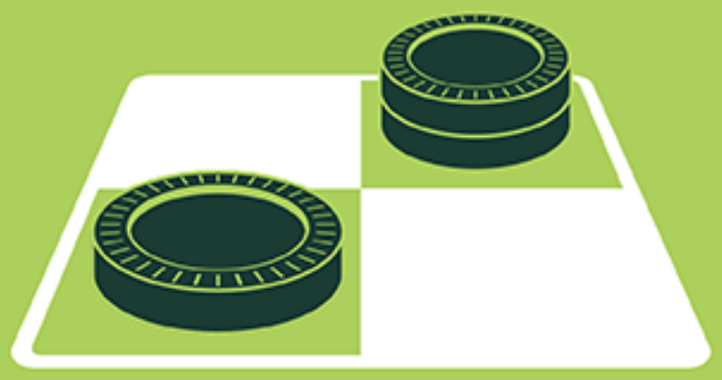
1. Introduction to Machine Learning
2. Supervised Learning
3. Neural Networks
4. Hands-on in TensorFlow
5. Homework

Introduction

Definitions & Problem statement

ARTIFICIAL INTELLIGENCE

Early artificial intelligence stirs excitement.



MACHINE LEARNING

Machine learning begins to flourish.

DEEP LEARNING

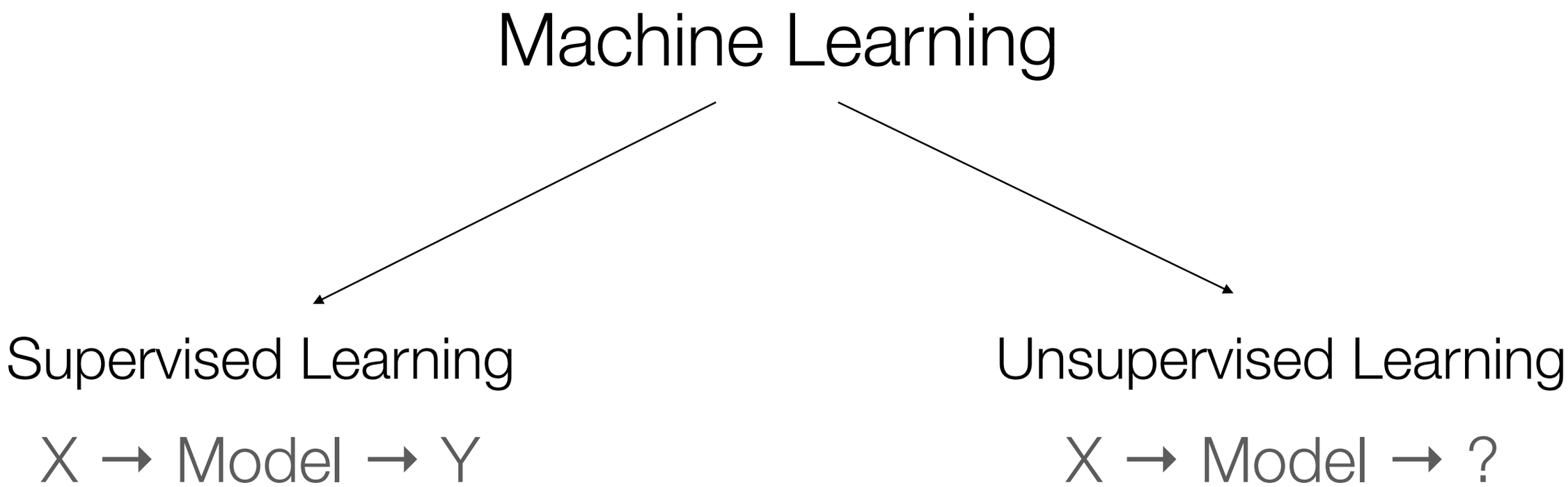
Deep learning breakthroughs drive AI boom.

1950's 1960's 1970's 1980's 1990's 2000's 2010's

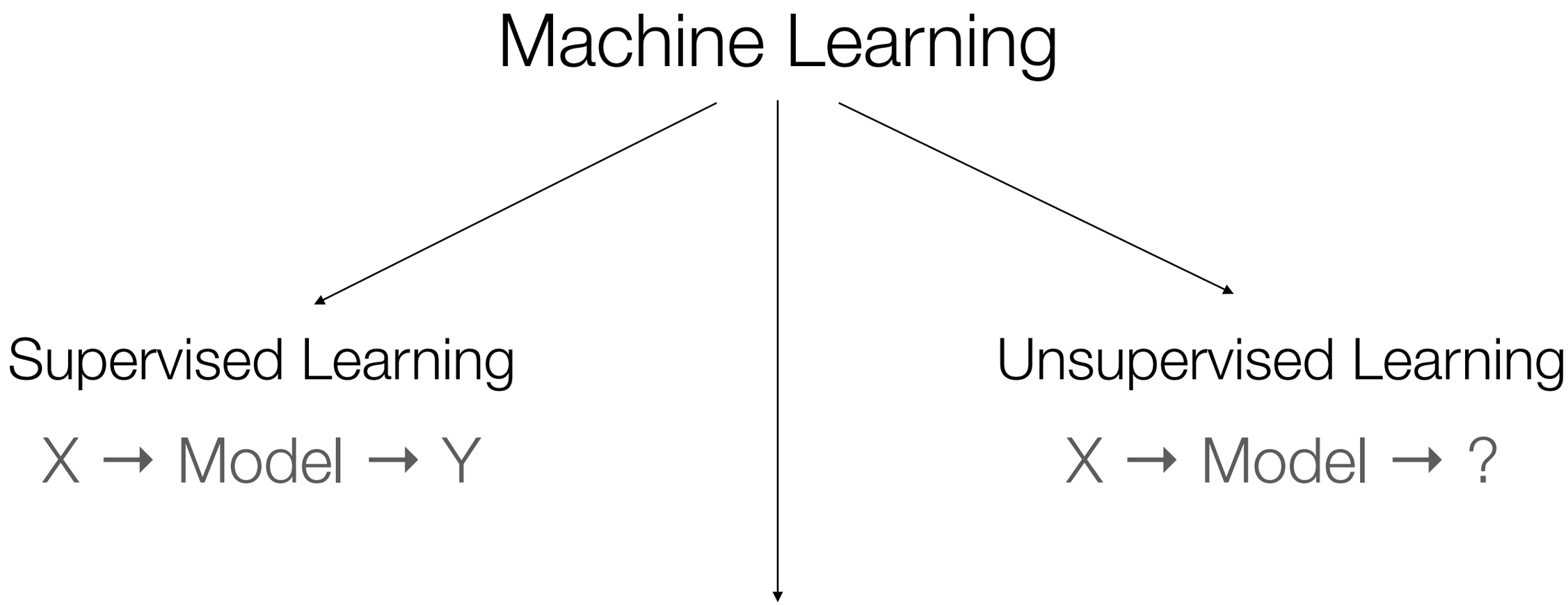
Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Source: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

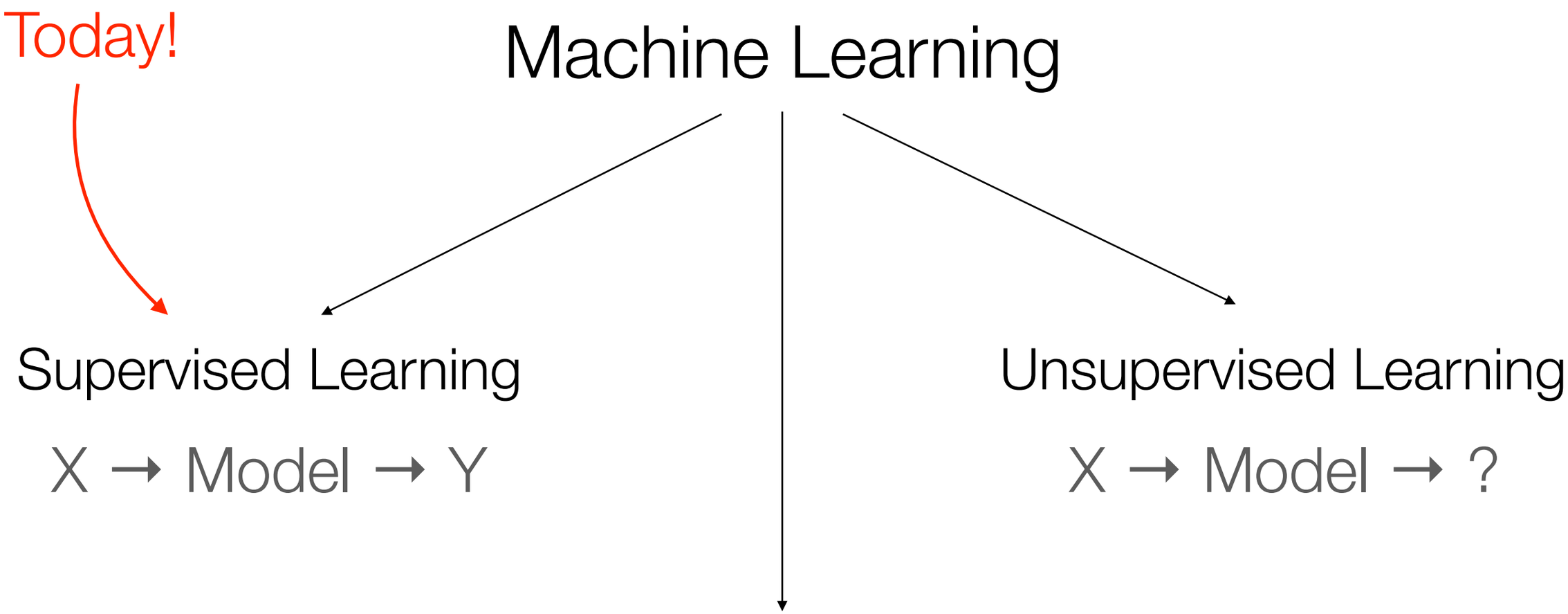
Machine Learning Tasks



Machine Learning Tasks



Machine Learning Tasks



Machine Learning Problems

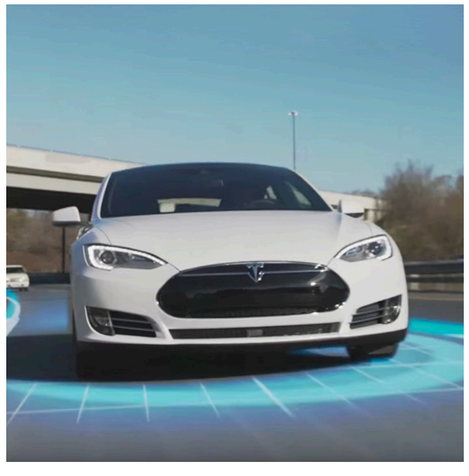
Supervised

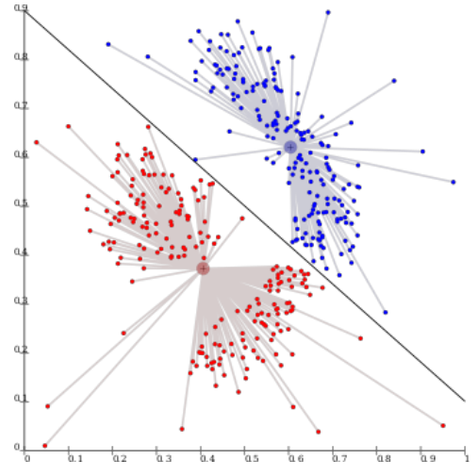
Unsupervised

Classification

Regression

Clustering





“This is a cat!”

“5 degrees to the left”

“Looks abnormal…”

Machine Learning Problems

Today!

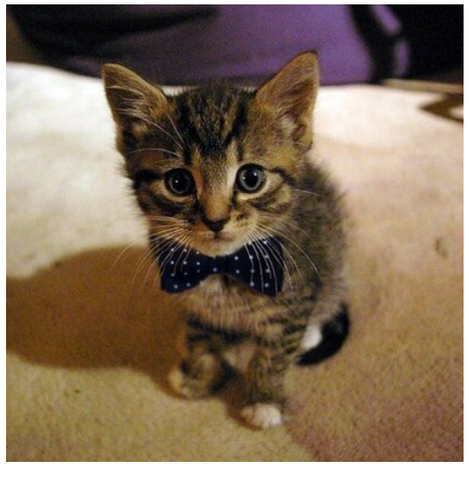
Supervised

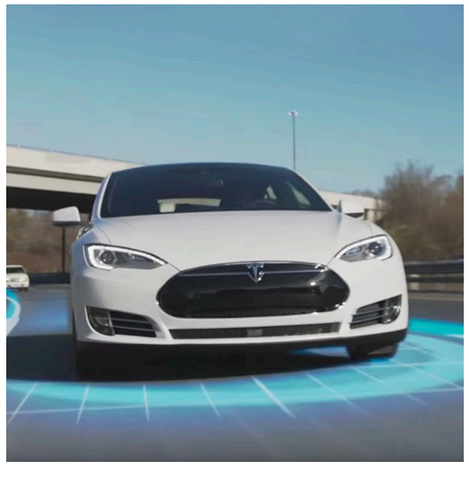
Unsupervised

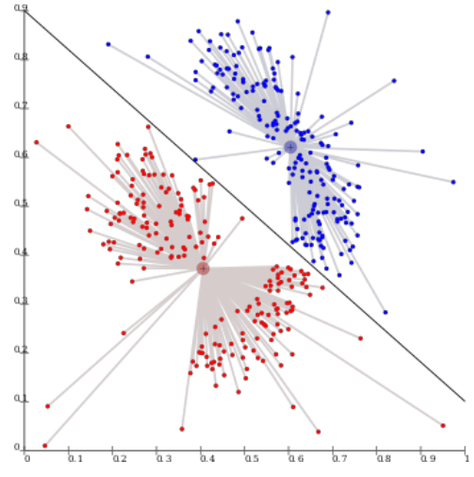
Classification

Regression

Clustering







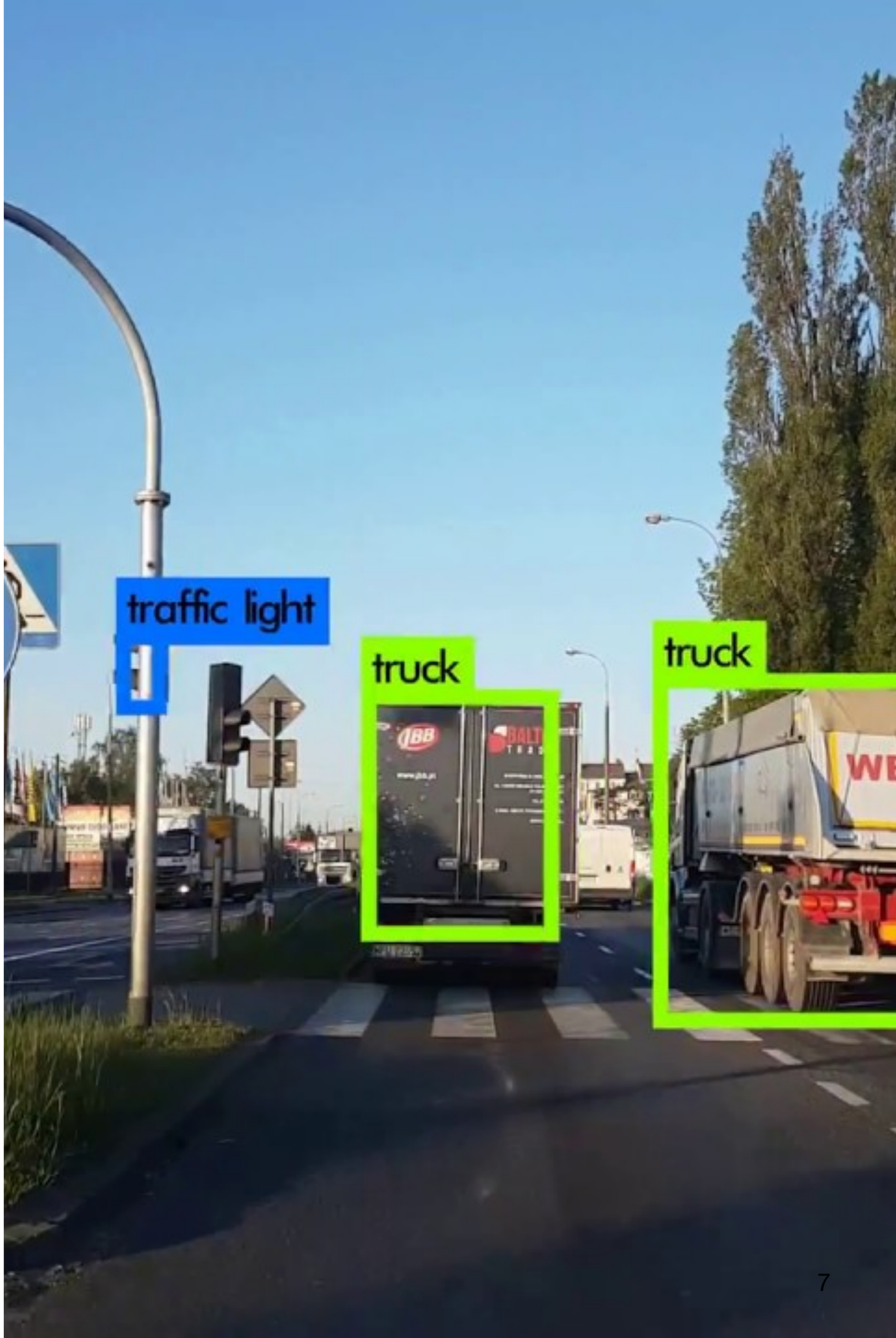
“This is a cat!”

“5 degrees to the left”

“Looks abnormal…”

Supervised Learning

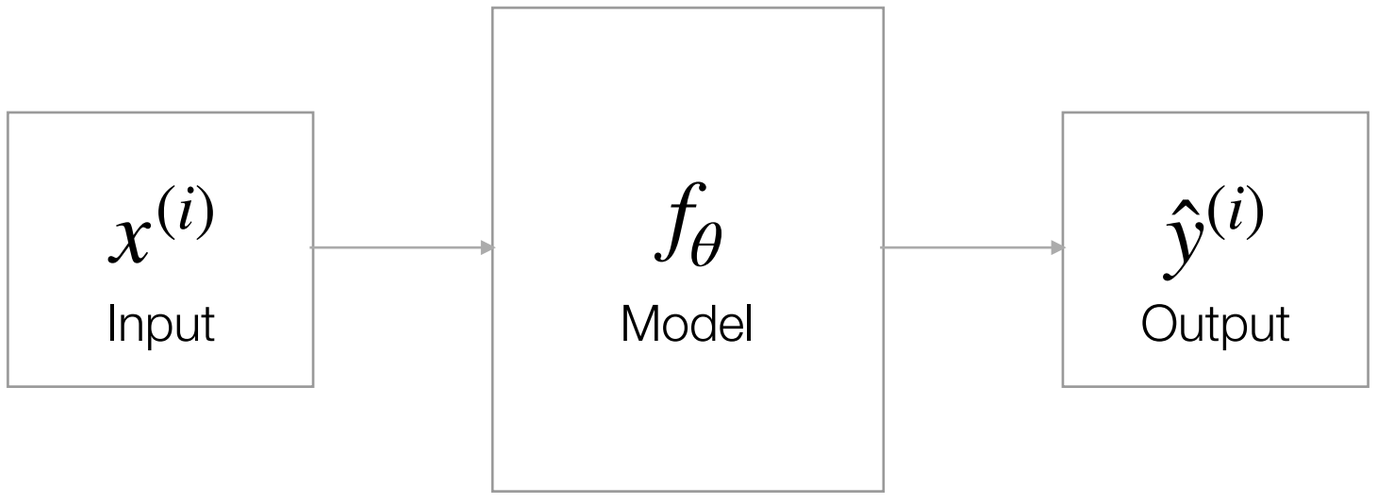
In more details…



Supervised Learning

- **Task:** map input to an output based on example input-output pairs.

$$\hat{y}^{(i)} = f_\theta(x^{(i)})$$

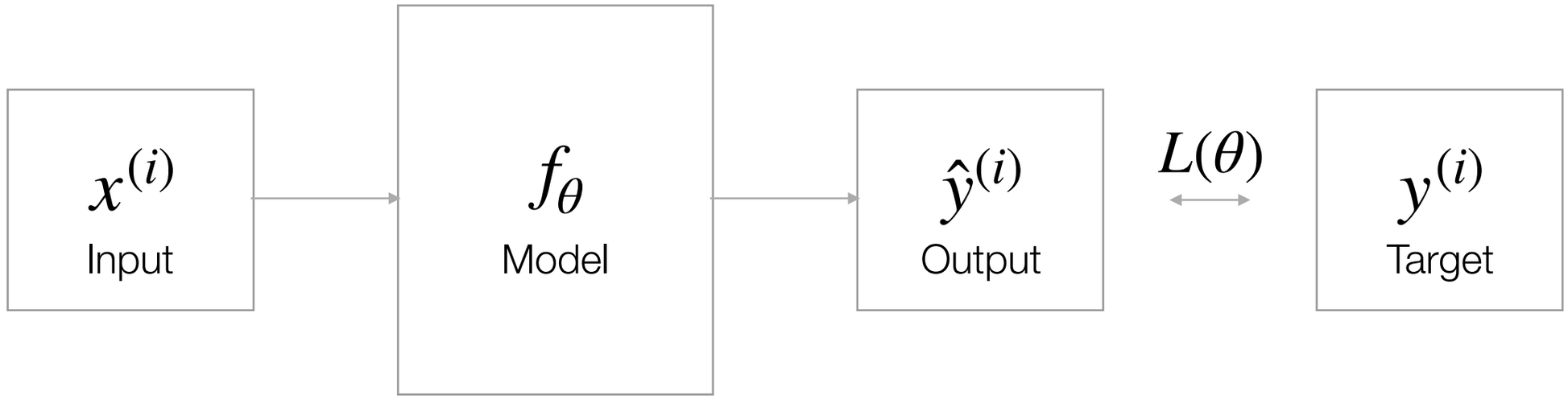


θ - parameters

Loss function

- **Loss functions** tell us how much our model's output $\hat{y}^{(i)}$ is far from target values $y^{(i)}$ defined in given dataset.

$$\hat{y}^{(i)} = f_\theta(x^{(i)})$$



Mean Absolute Error

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} | y^{(i)} - \hat{y}^{(i)} |$$

Mean Squared Error

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} (y^{(i)} - \hat{y}^{(i)})^2$$

Cross Entropy Loss

- Great for classification problem, where $\hat{y}^{(i)}$ is a probability distribution.

$$L(\theta) = -\frac{1}{N}\sum_{i=1}^{N}(y^{(i)}log(\hat{y}^{(i)}))$$

Cross Entropy Loss

- Great for classification problem, where $\hat{y}^{(i)}$ is a probability distribution.

Today!

$$L(\theta) = -\frac{1}{N}\sum_{i=1}^{N}(y^{(i)}log(\hat{y}^{(i)}))$$

Cross Entropy Loss

- Great for classification problem, where $\hat{y}^{(i)}$ is a probability distribution.

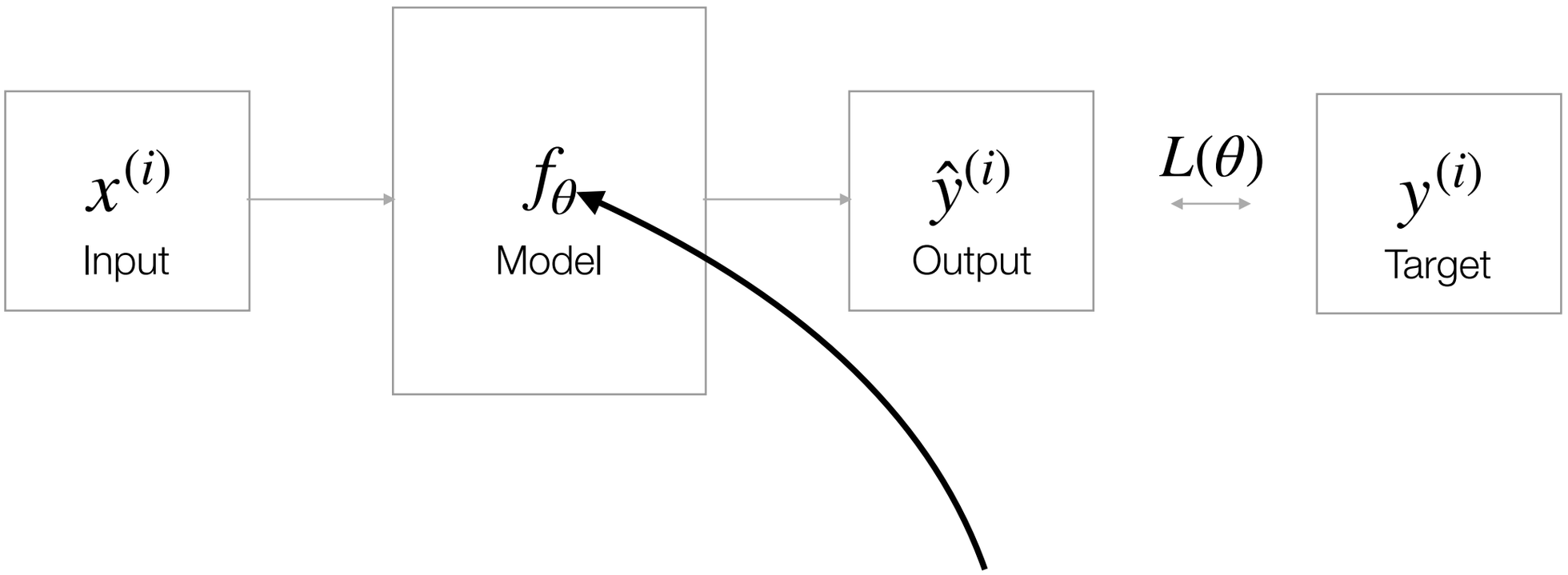
Today!

$$L(\theta) = -\frac{1}{N}\sum_{i=1}^{N}(y^{(i)}log(\hat{y}^{(i)}))$$

$$L(\theta) = -\frac{1}{N}\sum_{i=1}^{N}[y^{(i)}log(\hat{y}^{(i)}) + (1-y^{(i)})log(1-\hat{y}^{(i)})]$$

Optimization problem

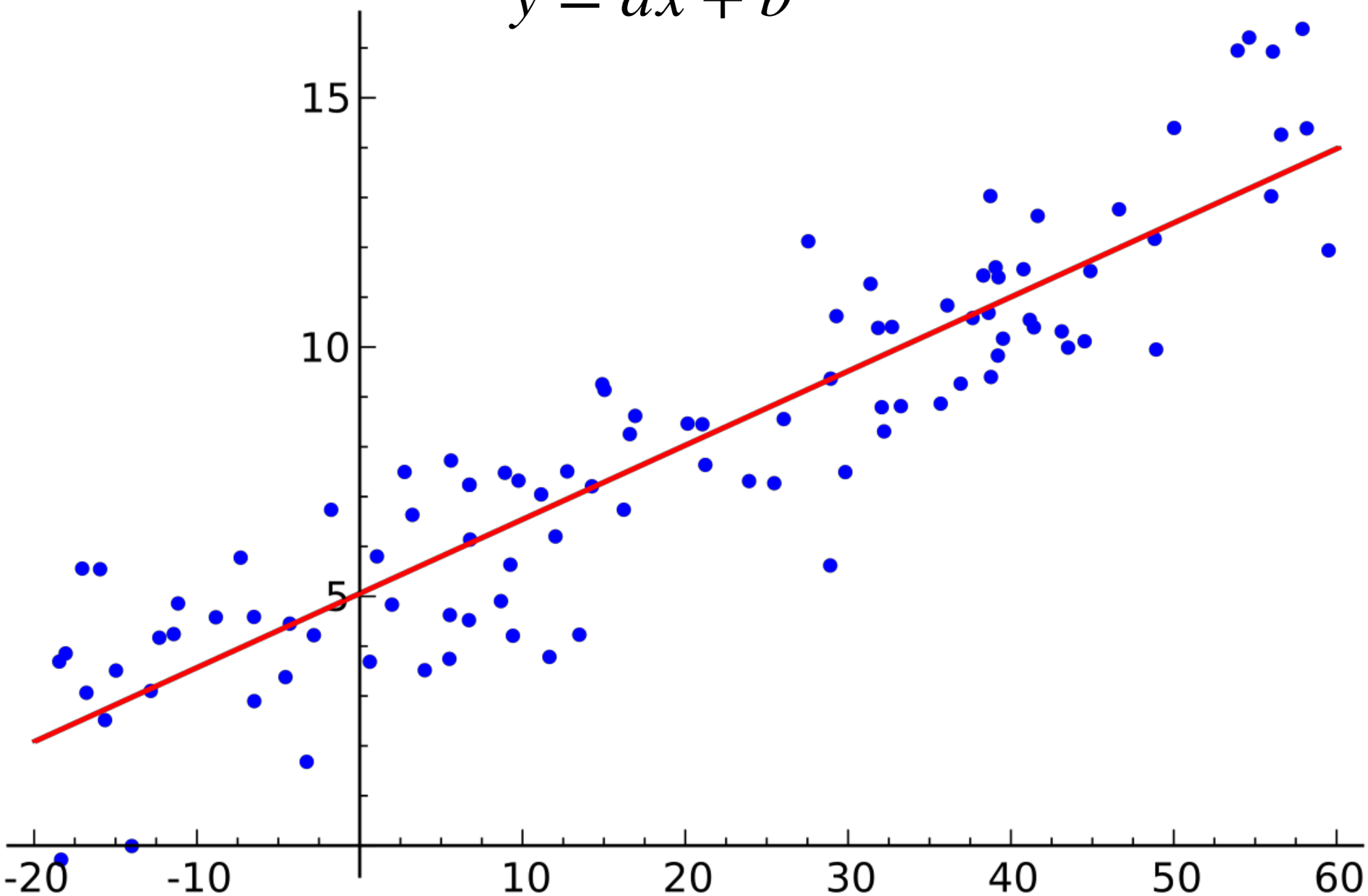
$$\hat{y}^{(i)} = f_\theta(x^{(i)})$$



How to find the best **parameters**?

Optimization problem

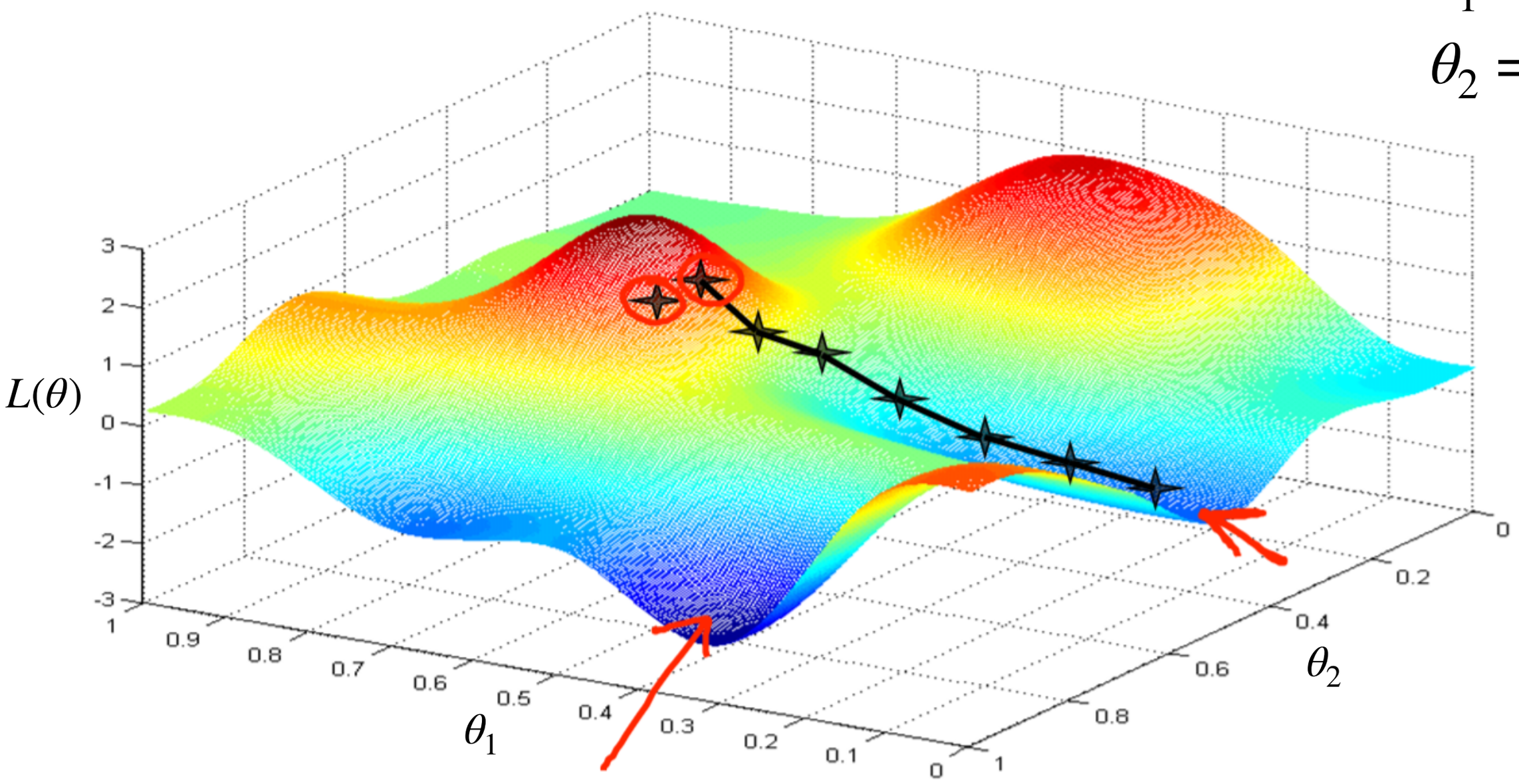
$$y = ax + b$$



Optimization problem

$\theta_1 = a$

$\theta_2 = b$



Gradient to the rescue!

- As our loss can be treated as a **function***, let's calculate a **gradient** (steepness).

- **Gradient** is a vector of partial derivatives of a function with respect to all its arguments.

$$\nabla L(\theta) = \frac{\partial L}{\partial \theta} = \begin{bmatrix} \frac{\partial L}{\partial \theta_1} \\ \frac{\partial L}{\partial \theta_2} \\ \vdots \\ \frac{\partial L}{\partial \theta_P} \end{bmatrix}$$

Gradient to the rescue!

- As our loss can be treated as a **function***, let's calculate a **gradient** (steepness).

- **Gradient** is a vector of partial derivatives of a function with respect to all its arguments.

$$\nabla L(\theta) = \frac{\partial L}{\partial \theta} = \begin{bmatrix} \frac{\partial L}{\partial \theta_1} \\ \frac{\partial L}{\partial \theta_2} \\ \vdots \\ \frac{\partial L}{\partial \theta_P} \end{bmatrix}$$

* Needs to be **differentiable**!

Stochastic Gradient Descent

- Once we know how loss fluctuates for parameters changes, let's **update them** and move into this direction,

- …but just a little bit (multiply by alpha)!

- Alpha is known as a **learning rate**.

$$\theta^{(i+1)} = \theta^{(i)} - \alpha \cdot \nabla L(\theta)$$

Stochastic Gradient Descent

- Once we know how loss fluctuates for parameters changes, let's **update them** and move into this direction,

- …but just a little bit (multiply by alpha)!

- Alpha is known as a **learning rate**.

$$\theta^{(i+1)} = \theta^{(i)} - \alpha \cdot \nabla L(\theta)$$

Gradient show us the way to the maximum,
so let's go in the opposite direction!

Batch training

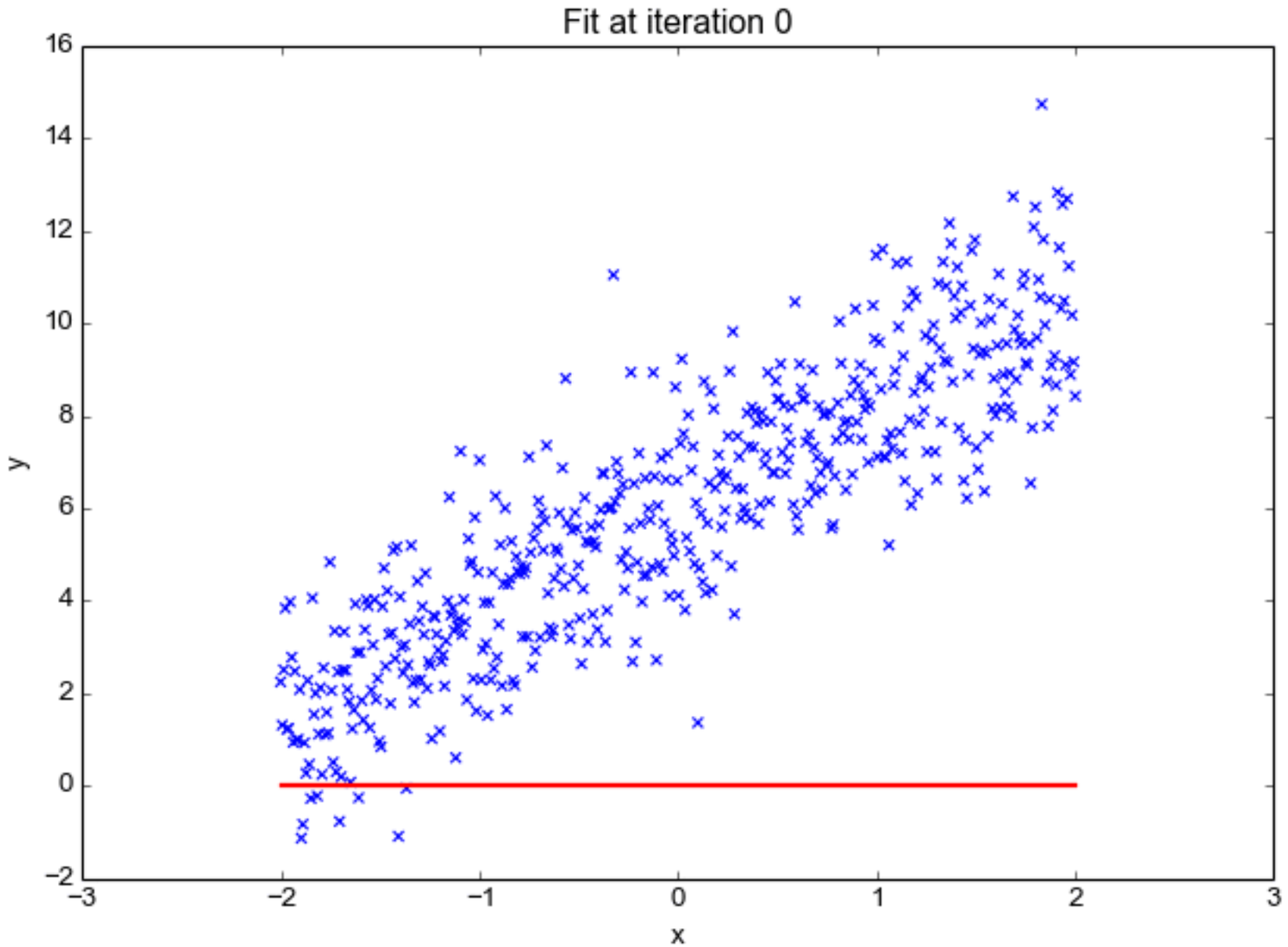
$$\theta^{(i+1)} = \theta^{(i)} - \alpha \cdot \nabla L(\theta)$$

- Instead of passing all dataset at once, it is better to pass smaller packages - **batches**.

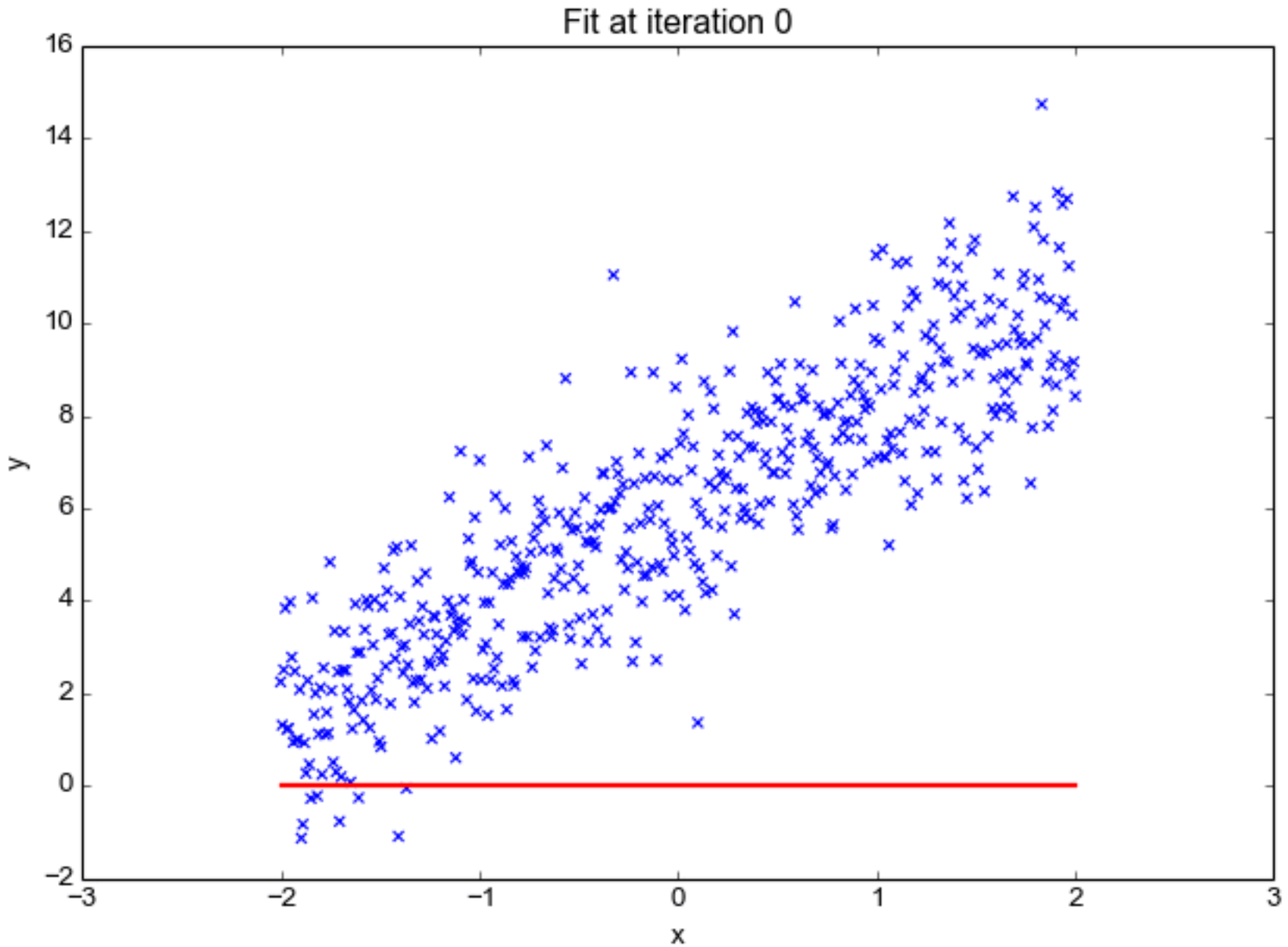
- Split your training dataset into batches and update your model more frequently!

- Keep your batches small (eg. up to 32 examples) and follow **mini-batch** approach.

Linear Regression Training



Linear Regression Training



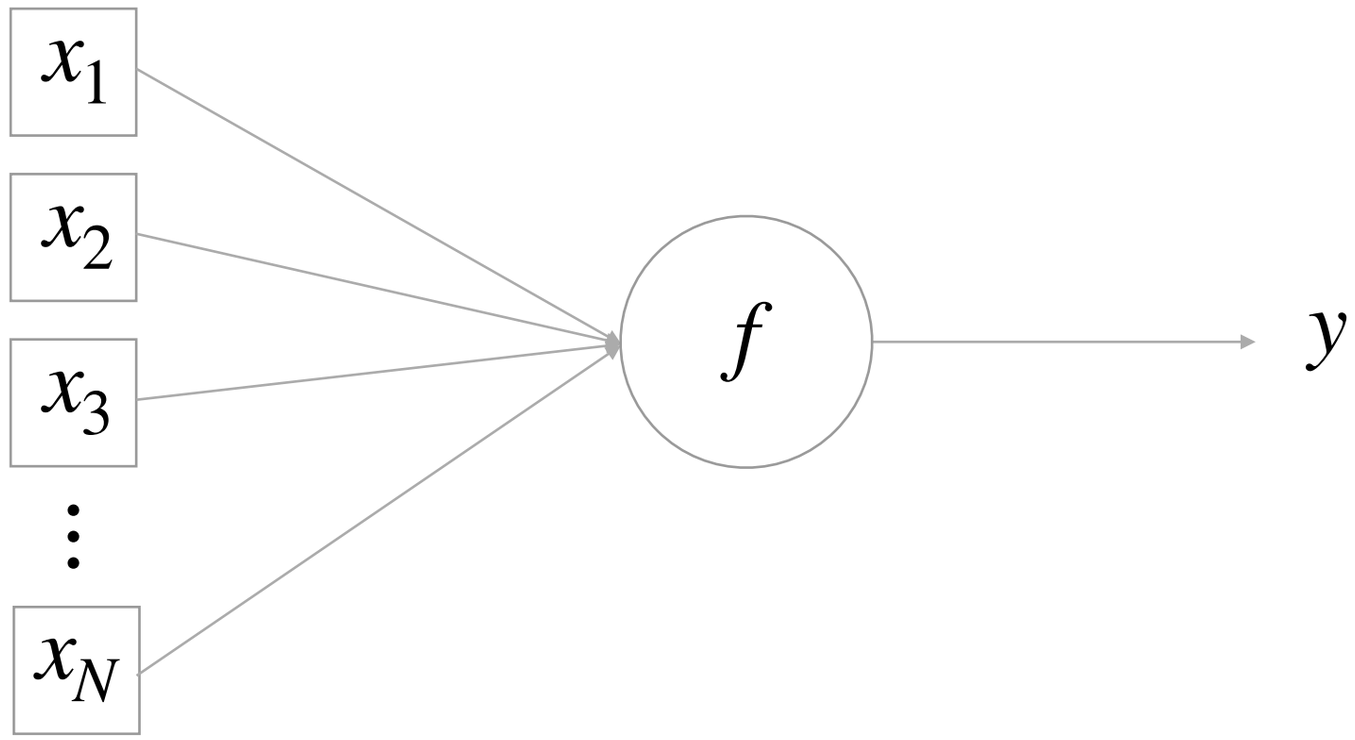
Ufff… Any questions so far?

Neural Networks

More math!

What is a neuron?

- **Neuron** interprets input values to produce a single output.



What is a neuron?

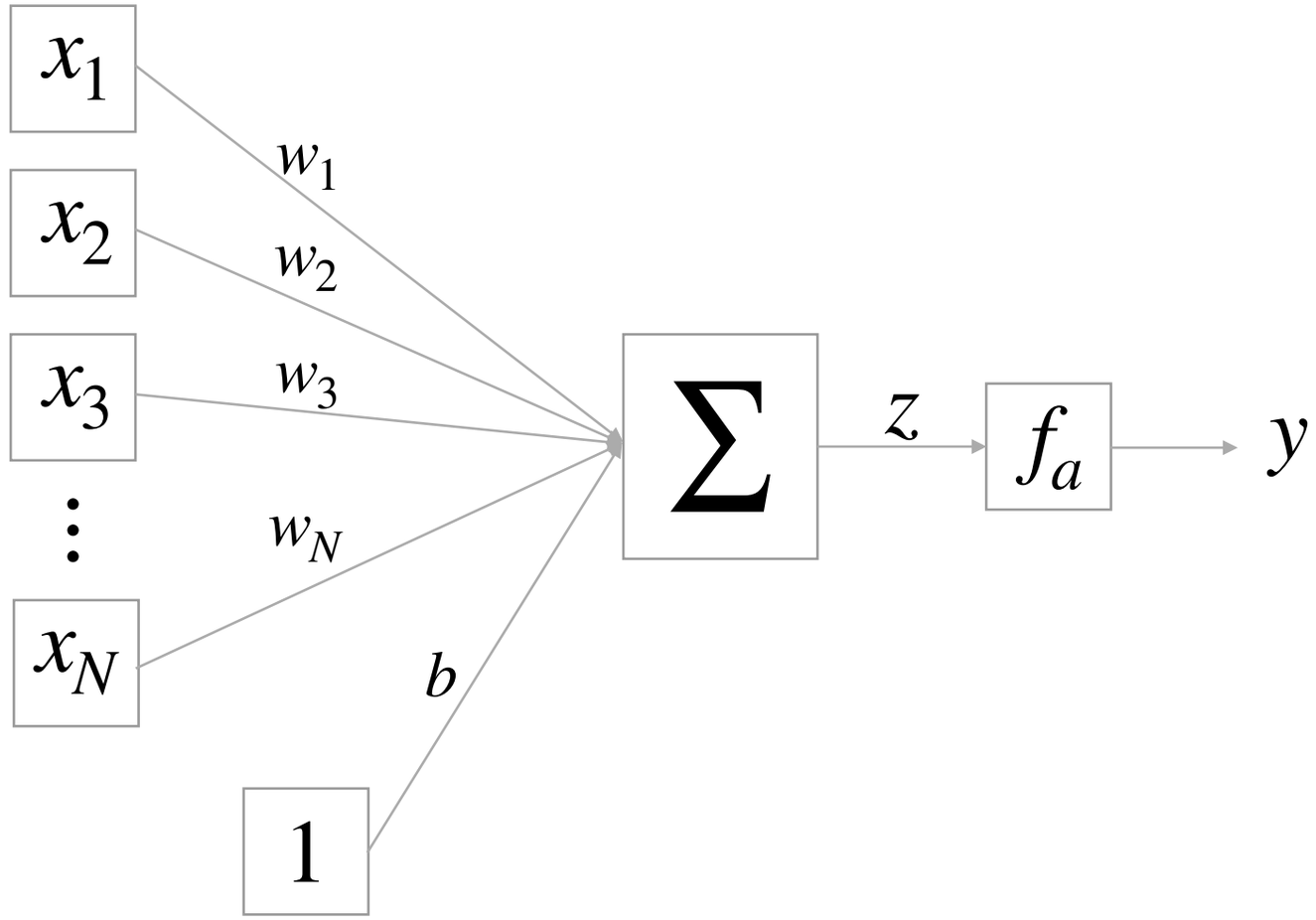
- **Neuron** is just a weighted sum of all inputs (and bias) passed through **activity function**.

- A single neuron solves linear-separable problems.

$$y = f_a\left(\sum_i^N x_i w_i + b\right) = f_a(wx + b) = f_a(z)$$

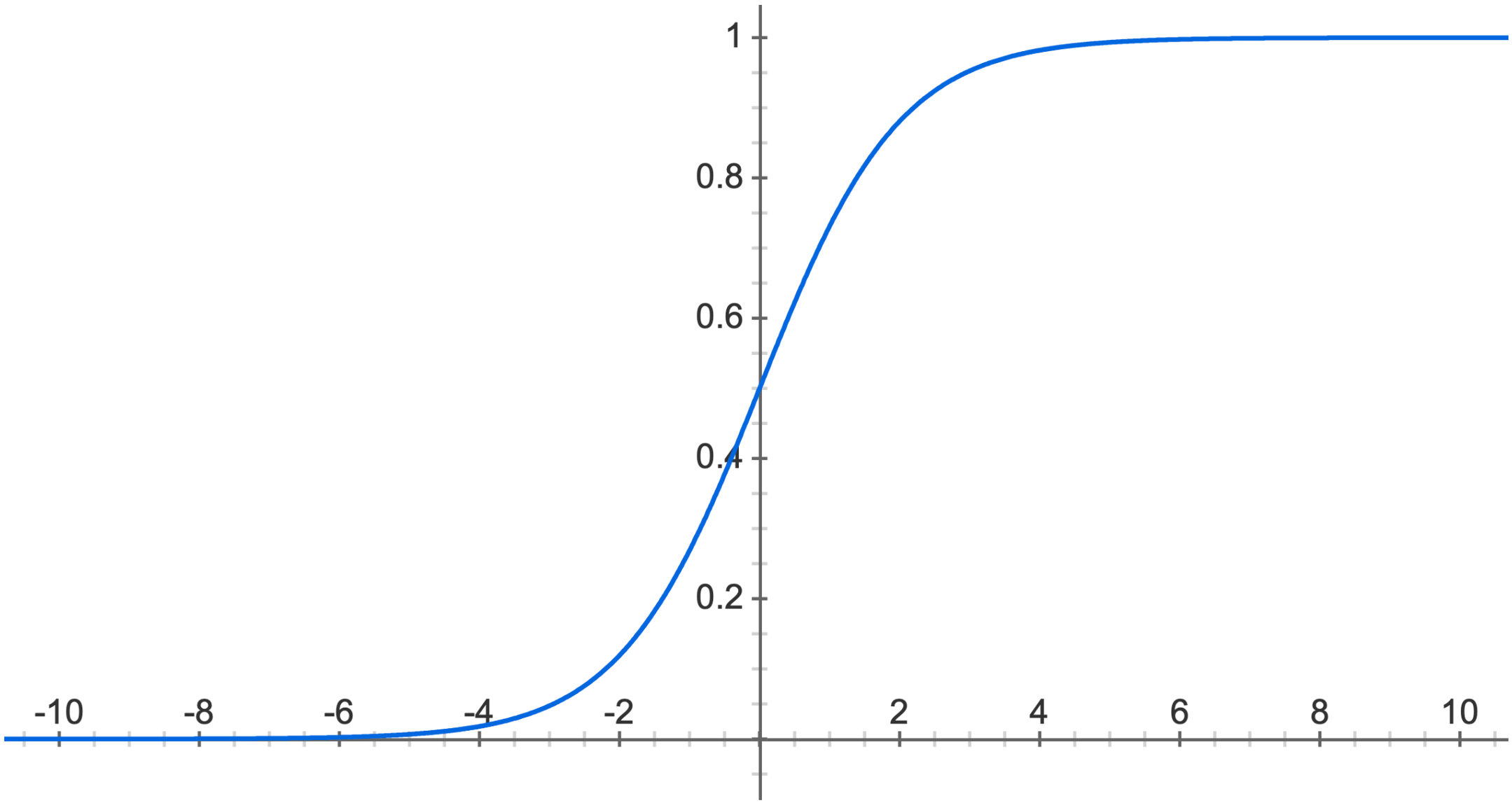
What is a neuron?

$$y = f_a\left(\sum_i^N x_i w_i + b\right) = f_a(wx + b) = f_a(z)$$



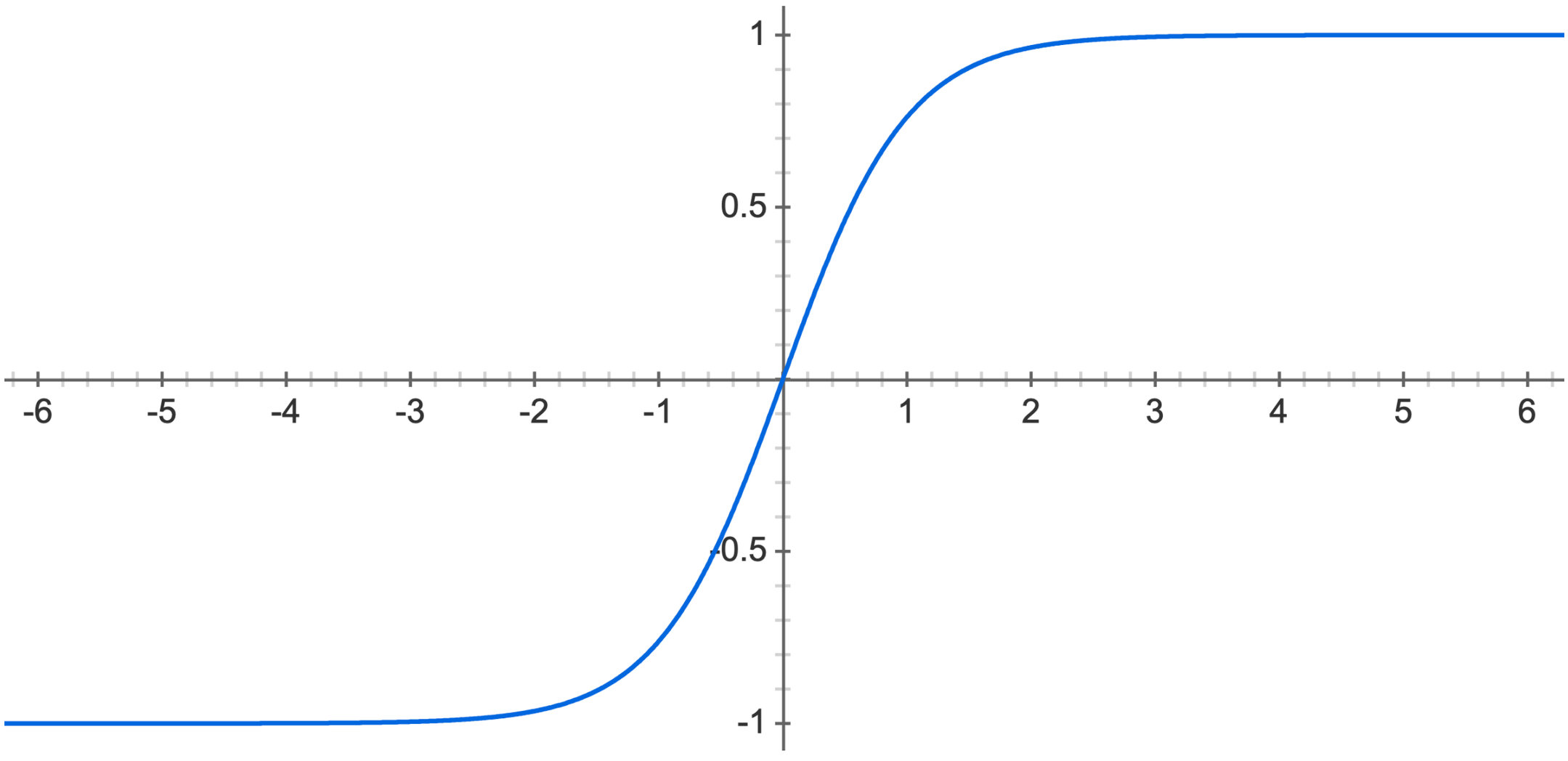
Sigmoid

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



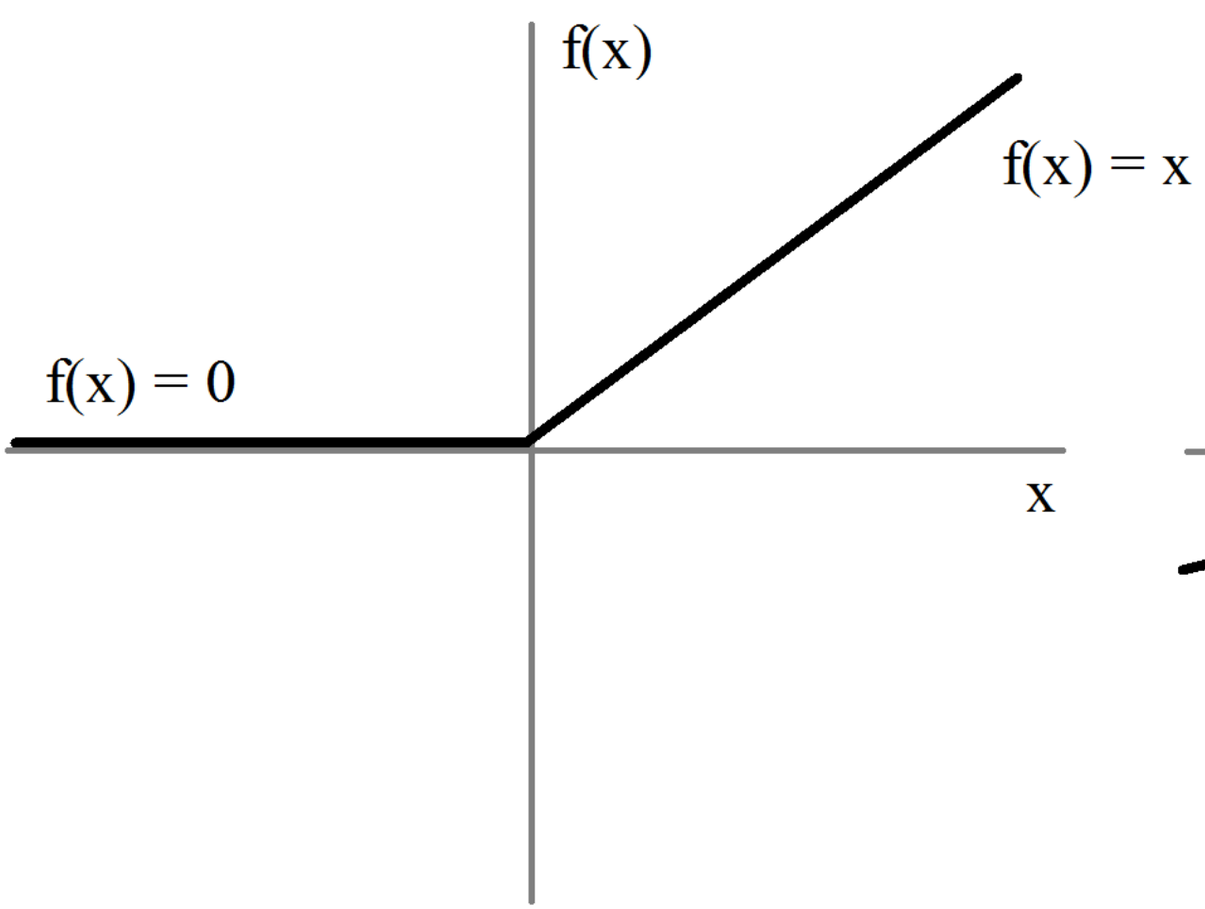
Hyperbolic Tangent

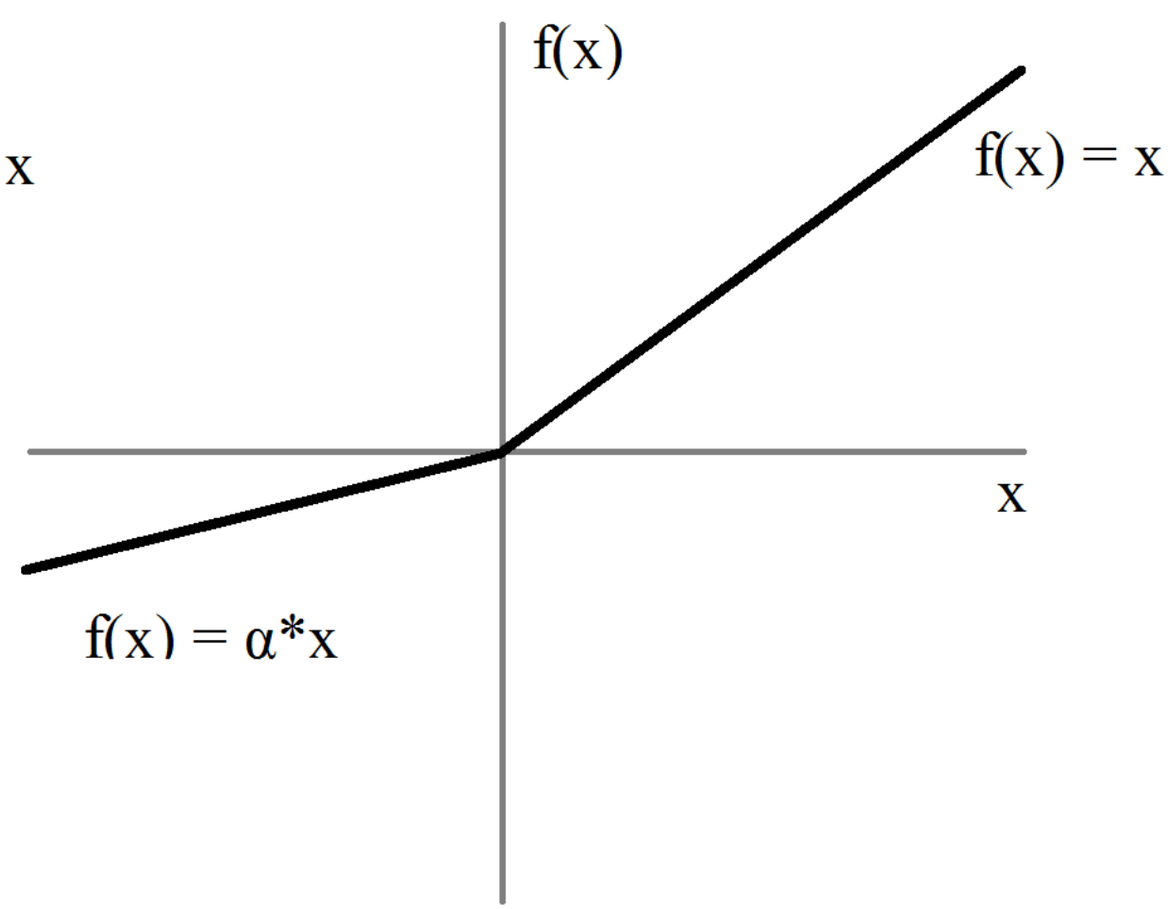
$$f_a(z) = tanh(z)$$



ReLU & Leaky ReLU

$$f_a(z) = \begin{cases} x & , x \geq 0 \\ 0 & , x < 0 \end{cases} \qquad f_a(z) = \begin{cases} x & , x \geq 0 \\ \alpha x & , x < 0 \end{cases}$$

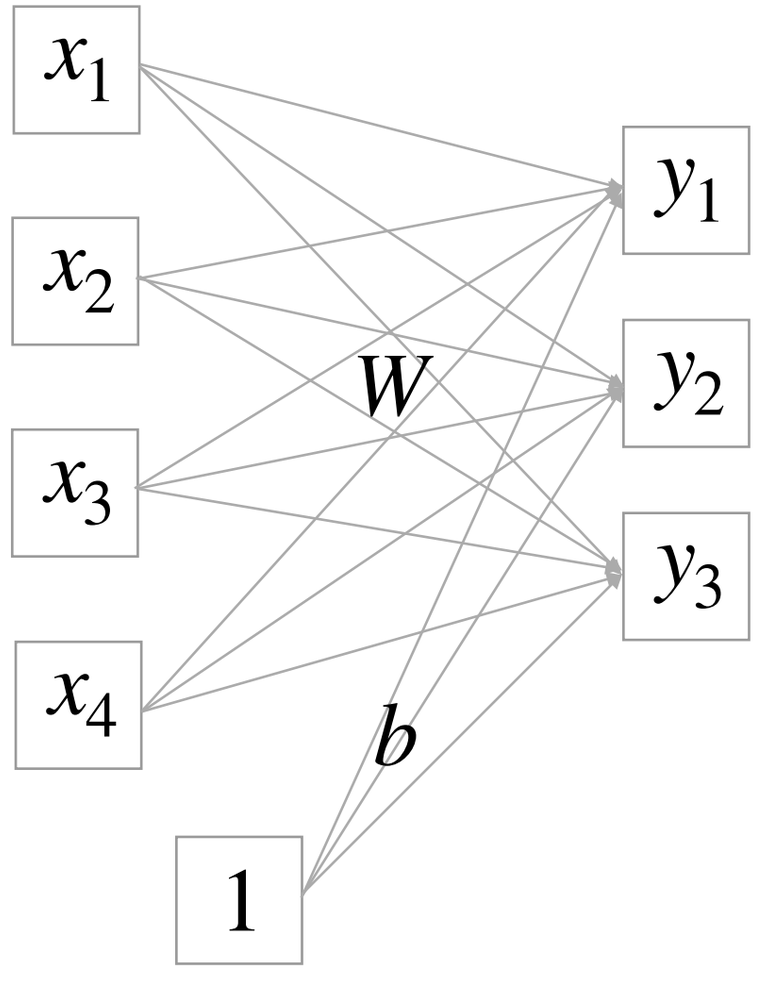




Dense Layers

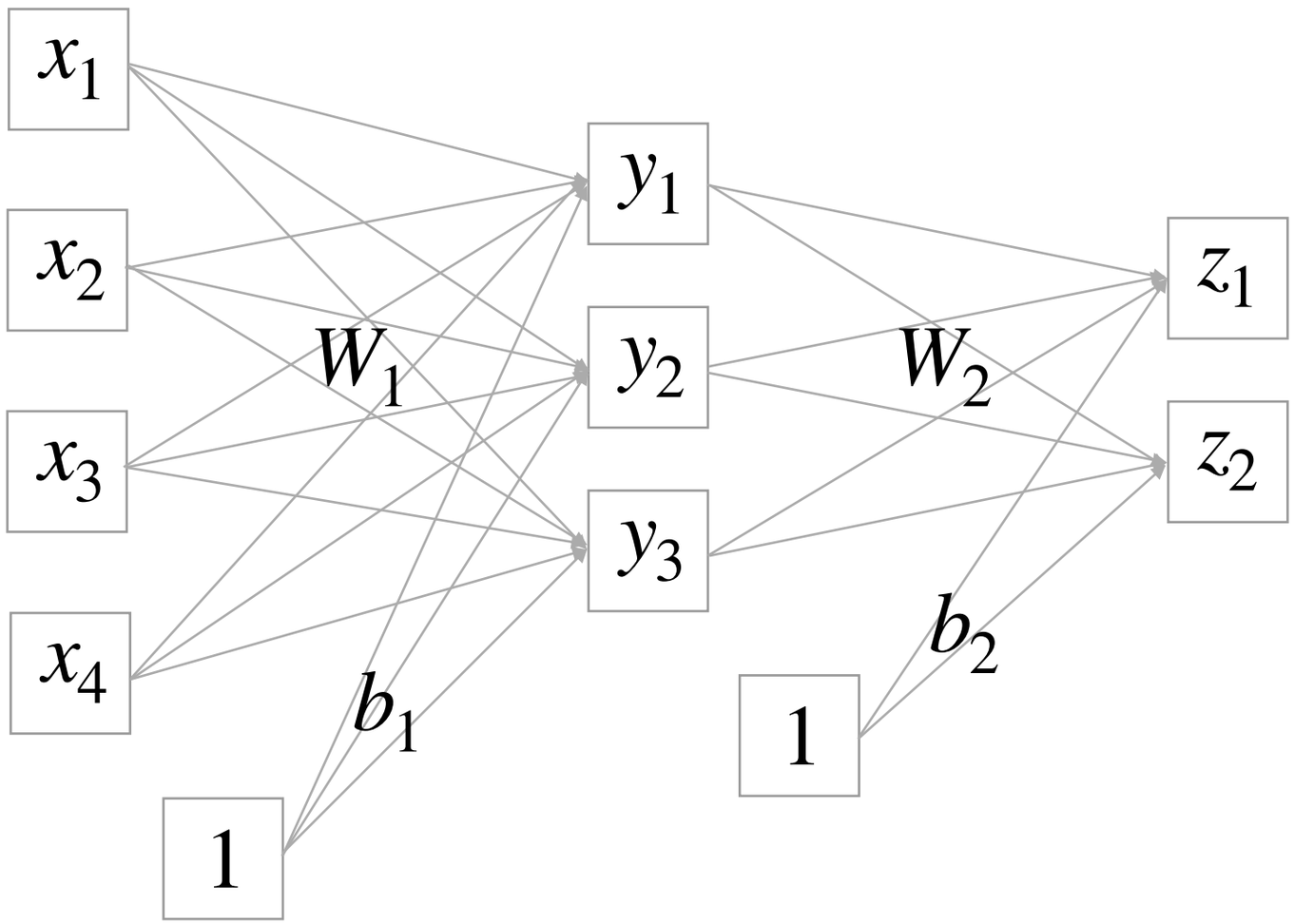
- We can **stack multiple neurons** into layers.
- Then, we can connect these neurons **densly**.

$$y = f_a(xW + b)$$



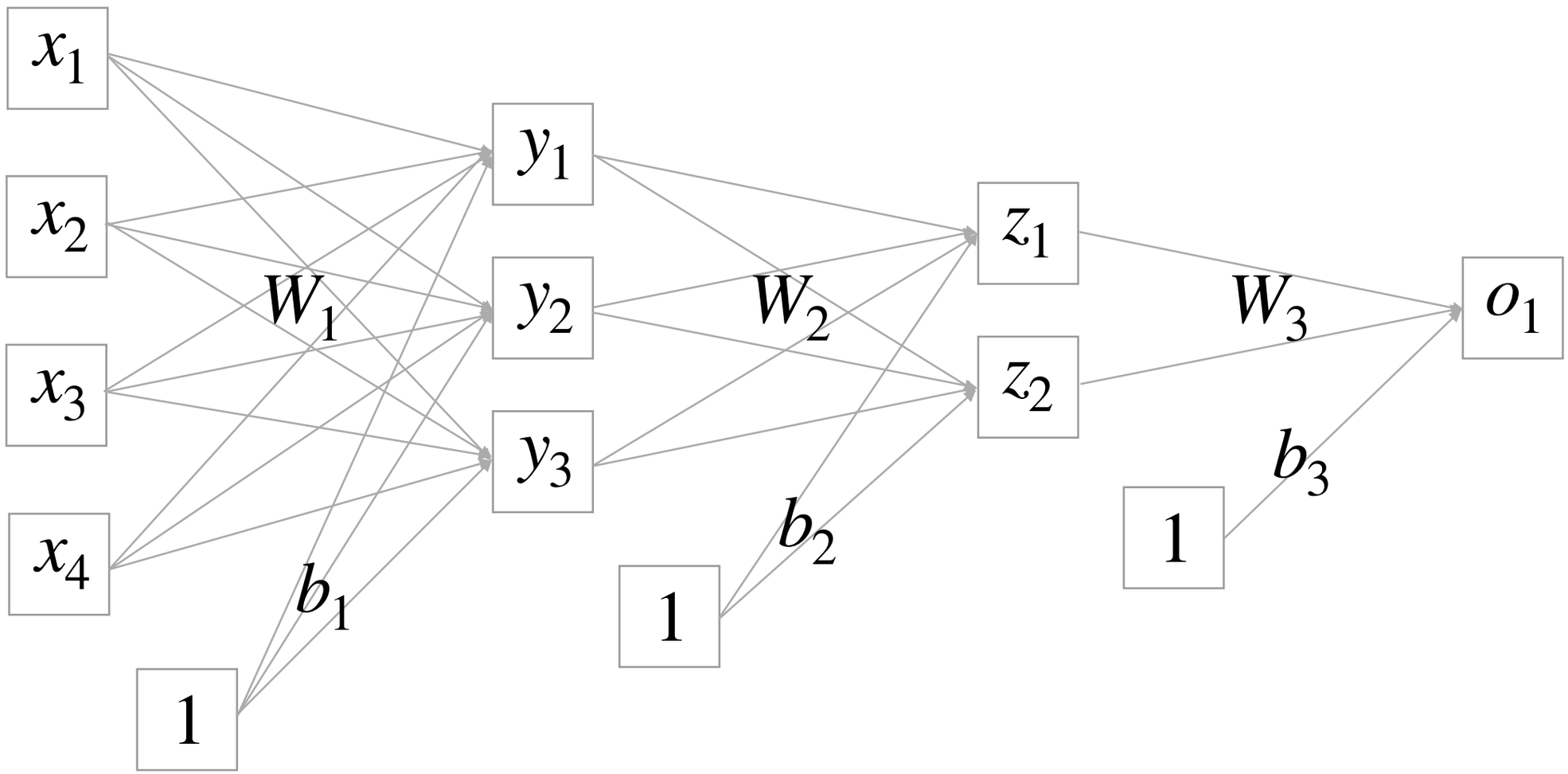
Neural networks

- Then we can **stack multiple layers** into networks.

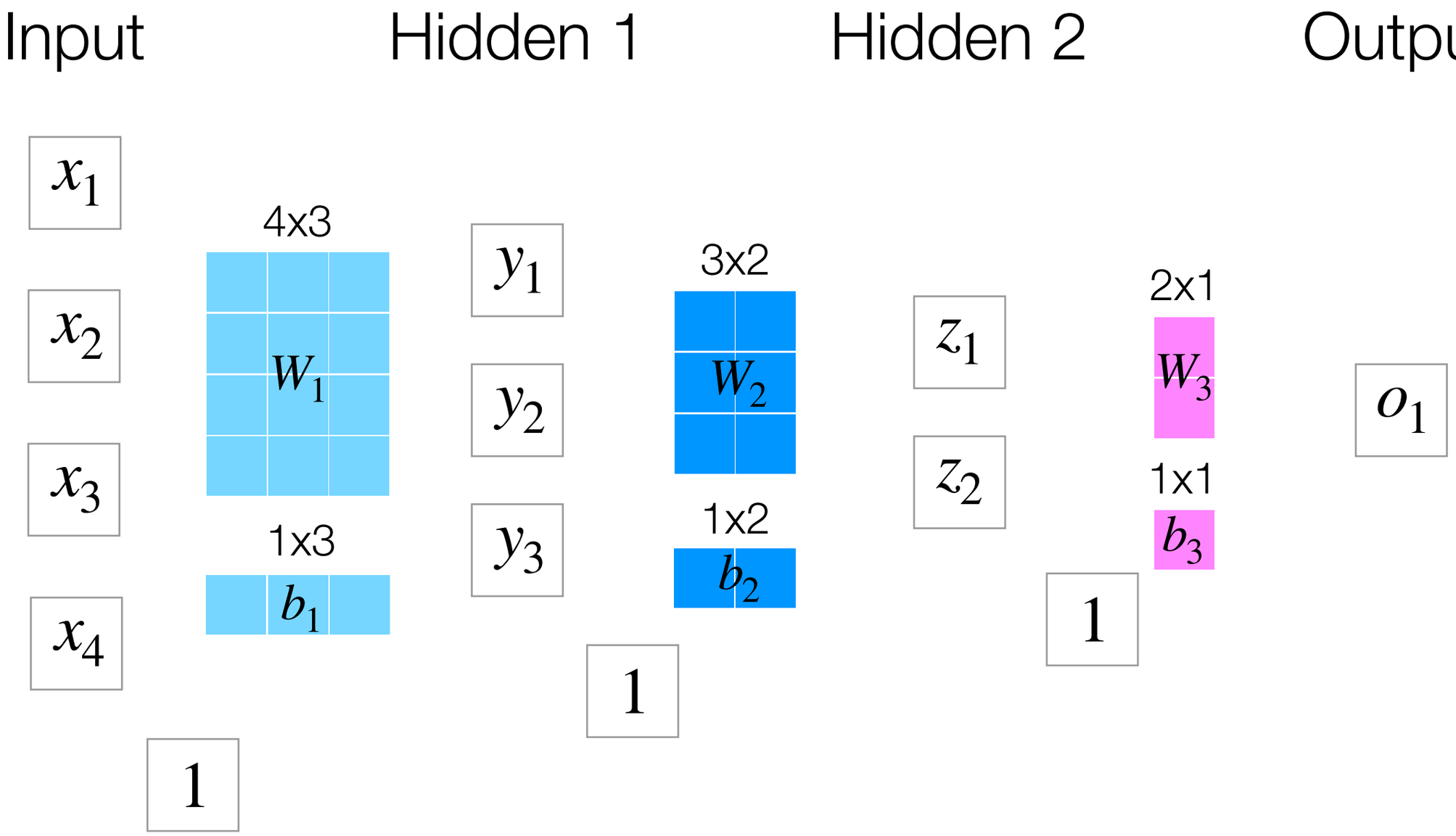


Neural networks

Input Hidden 1 Hidden 2 Output

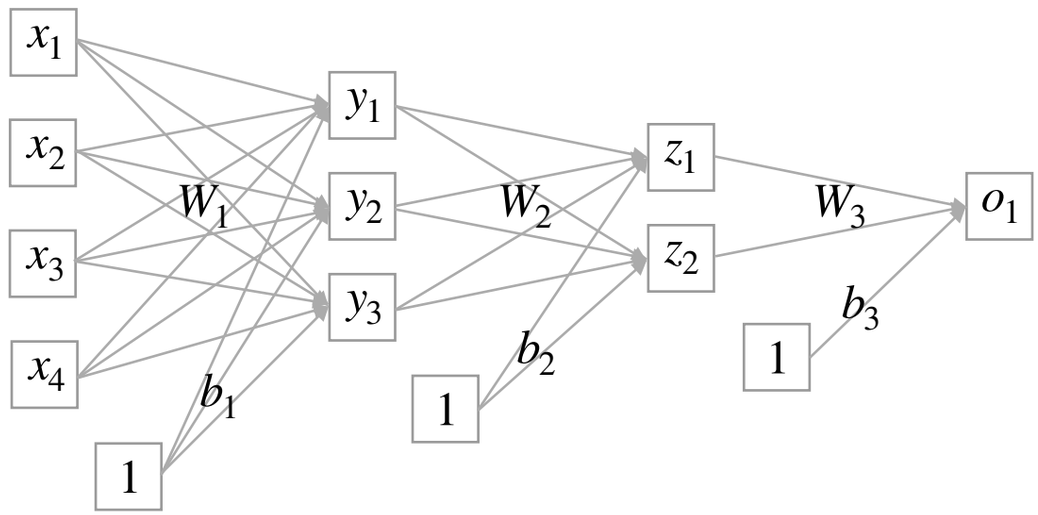


Neural networks



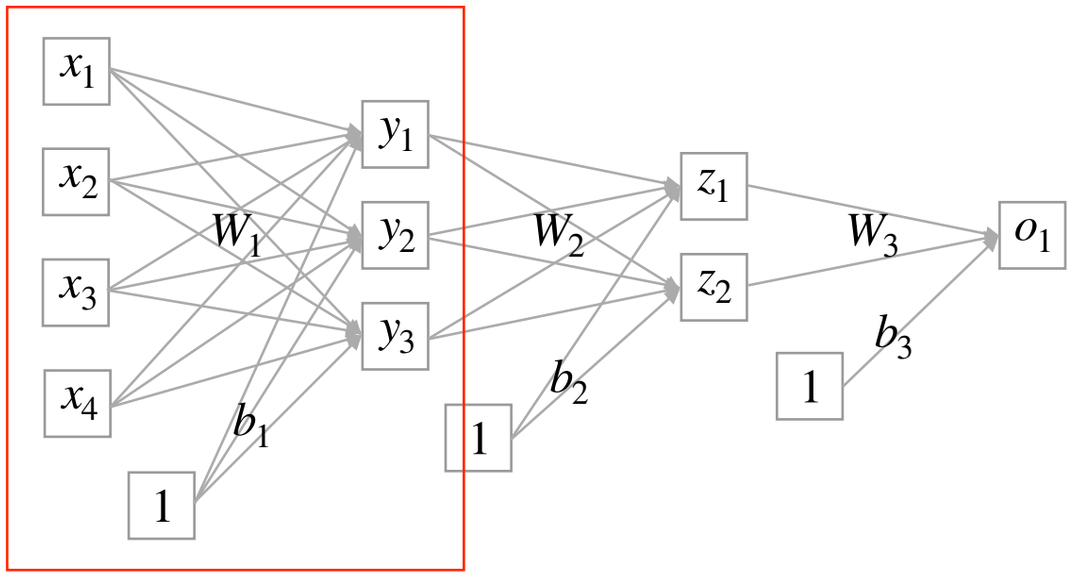
Neural networks

$$y = f_a(xW + b)$$



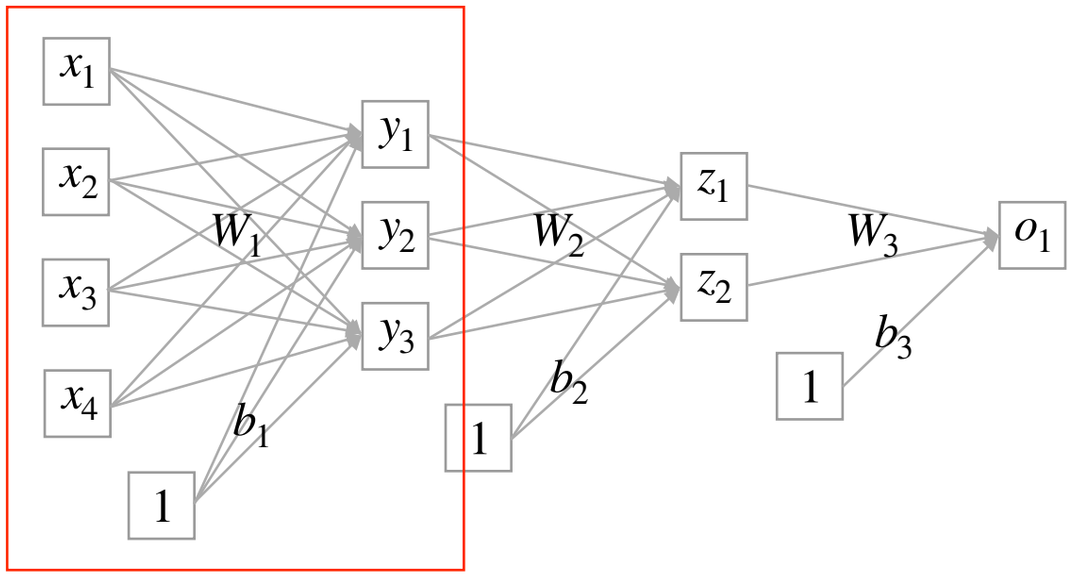
Neural networks

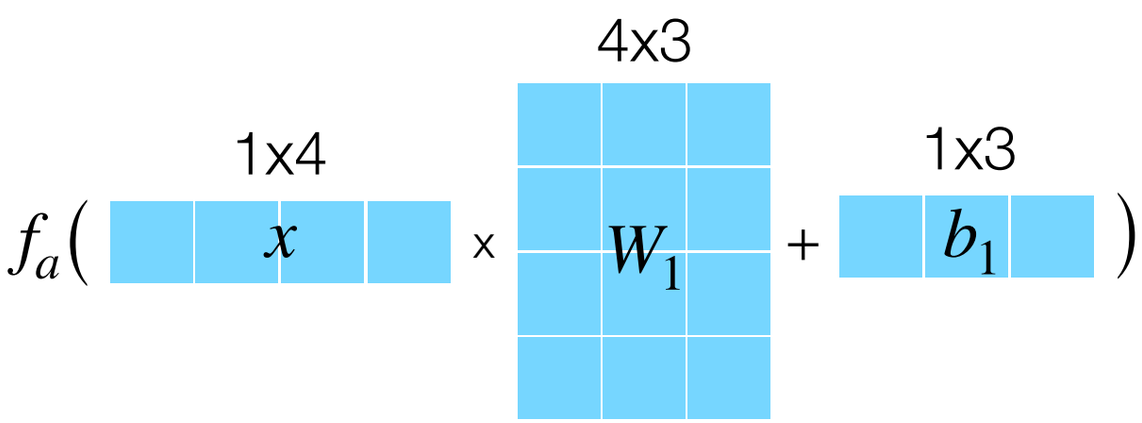
$$y = f_a(xW + b)$$



Neural networks

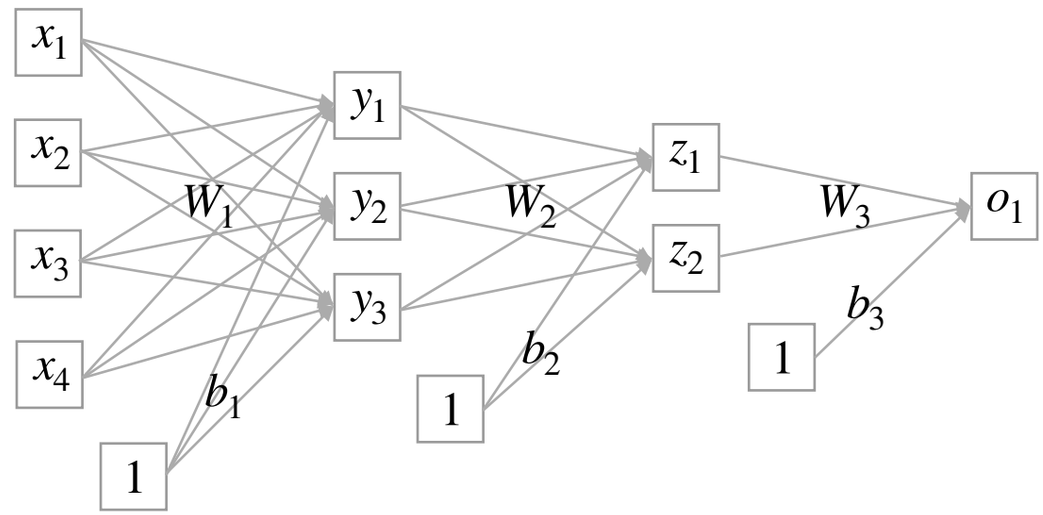
$$y = f_a(xW + b)$$

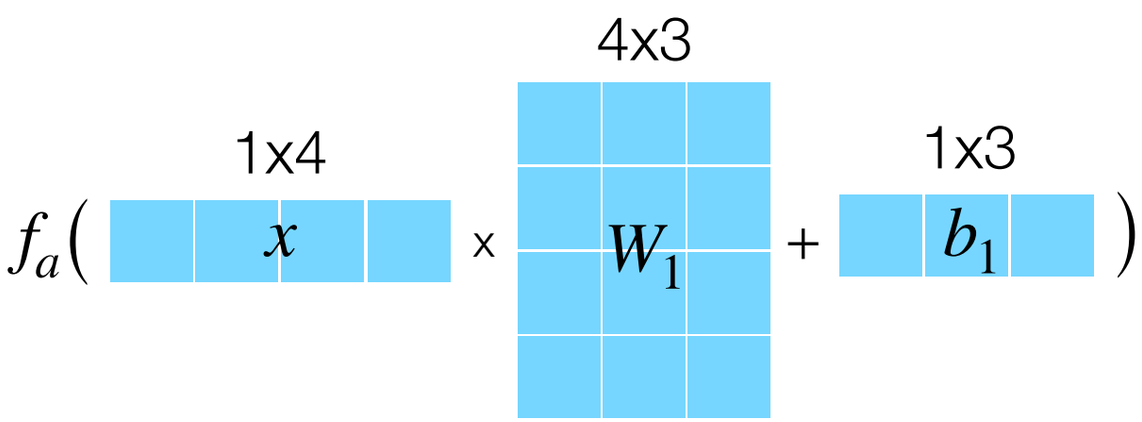




Neural networks

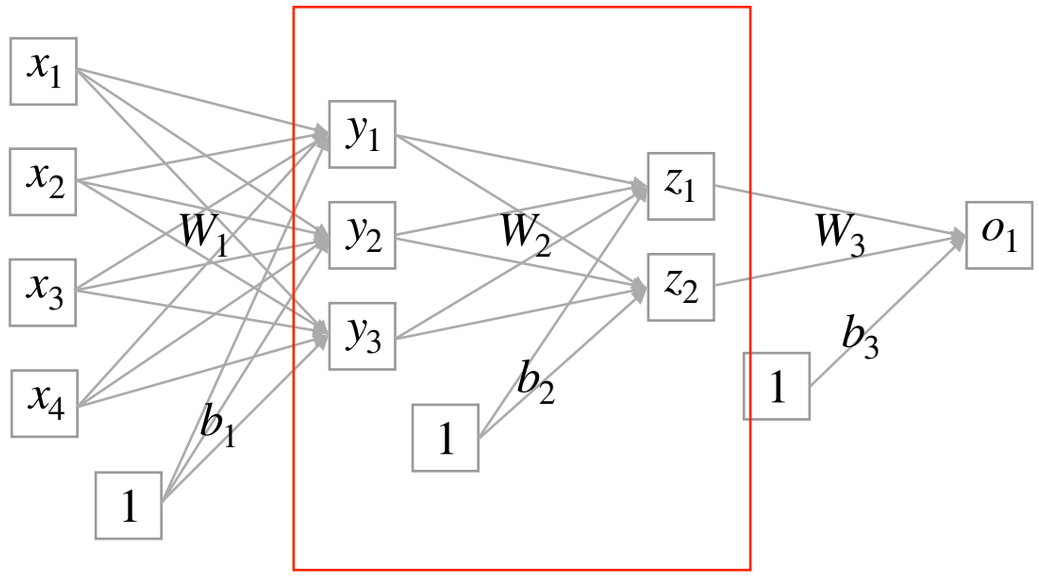
$$y = f_a(xW + b)$$

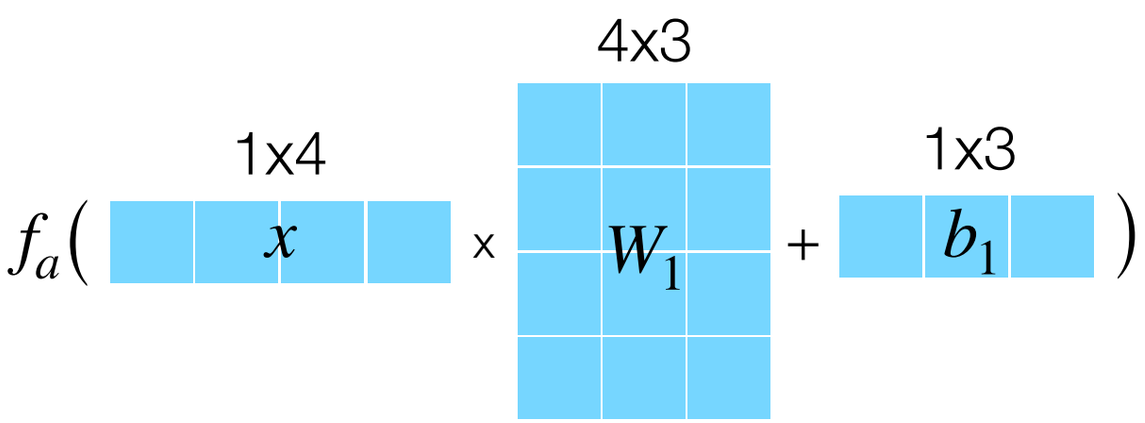




Neural networks

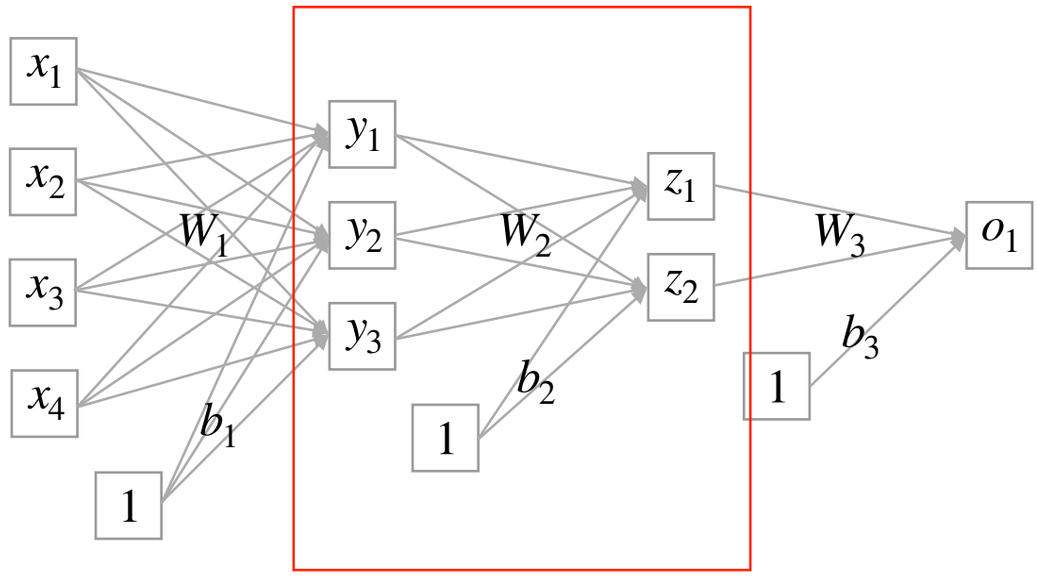
$$y = f_a(xW + b)$$





Neural networks

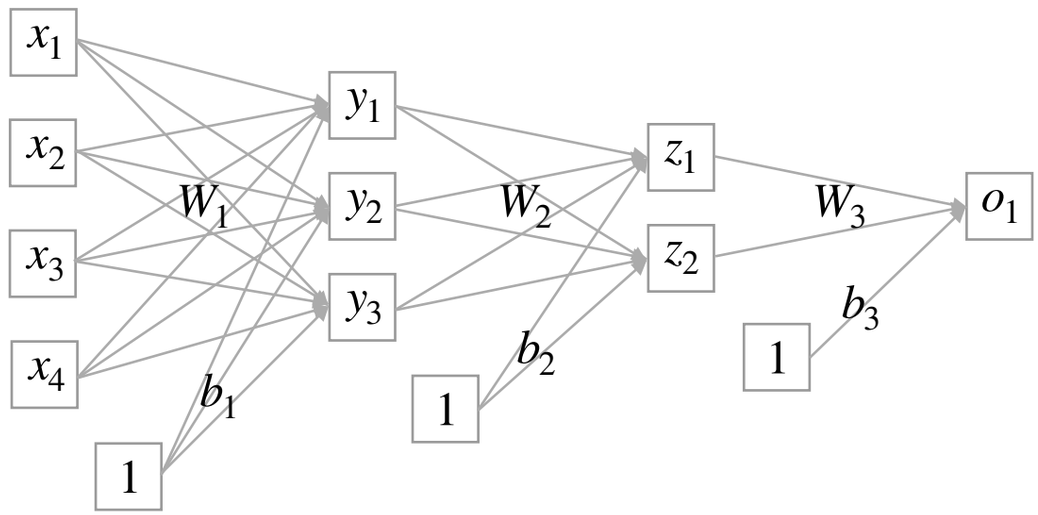
$$y = f_a(xW + b)$$



$$f_a\Big(\ f_a\big(\ \underset{1\text{x}4}{x}\ \text{x}\ \underset{4\text{x}3}{W_1} + \underset{1\text{x}3}{b_1}\ \big)\ \text{x}\ \underset{3\text{x}2}{W_2} + \underset{1\text{x}2}{b_2}\ \Big)$$

Neural networks

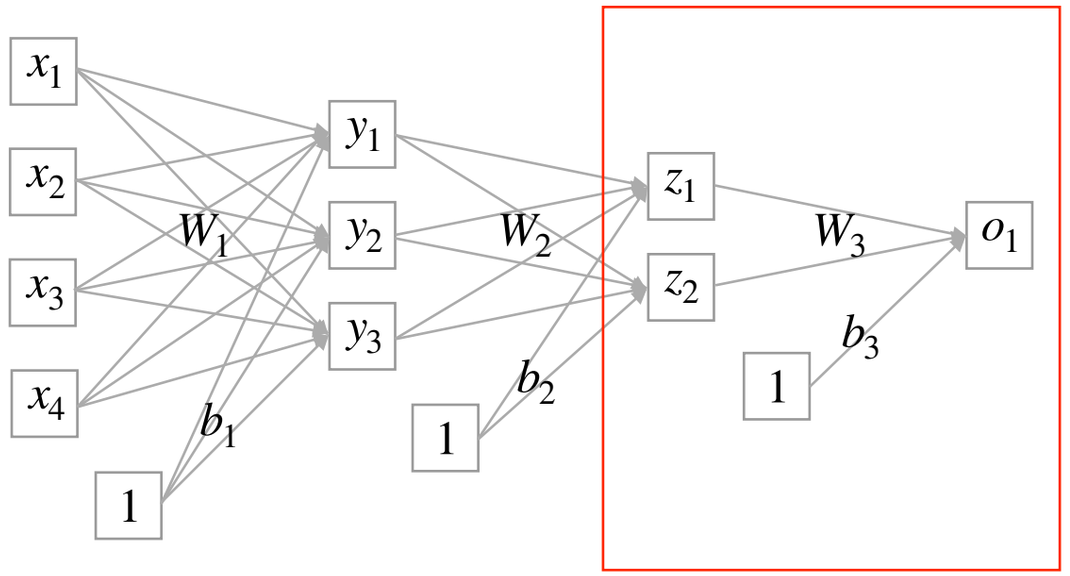
$$y = f_a(xW + b)$$

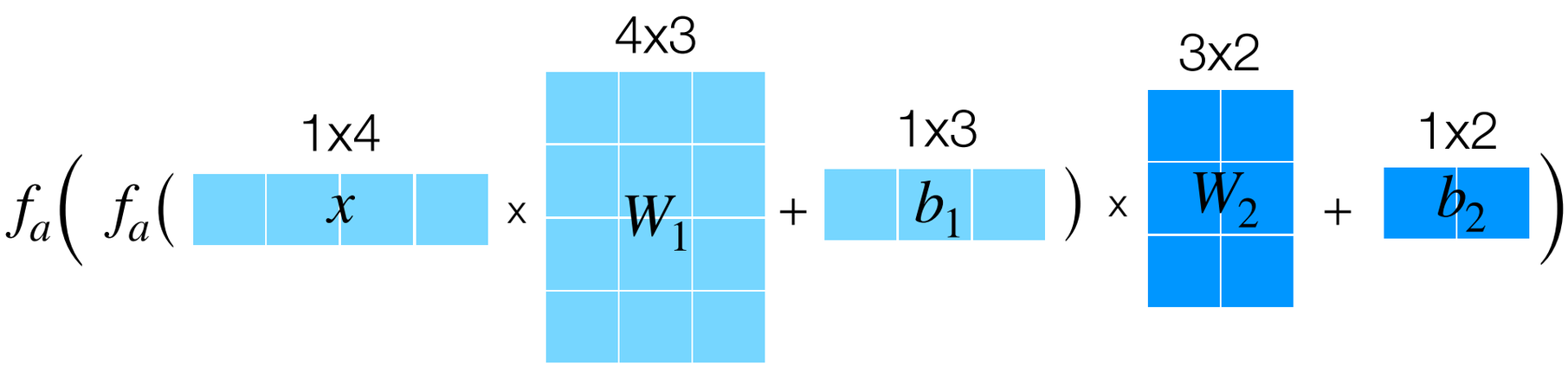


$$f_a\Big(\ f_a\big(\ \underset{1\times4}{x}\ \times\ \underset{4\times3}{W_1}\ +\ \underset{1\times3}{b_1}\ \big)\ \times\ \underset{3\times2}{W_2}\ +\ \underset{1\times2}{b_2}\ \Big)$$

Neural networks

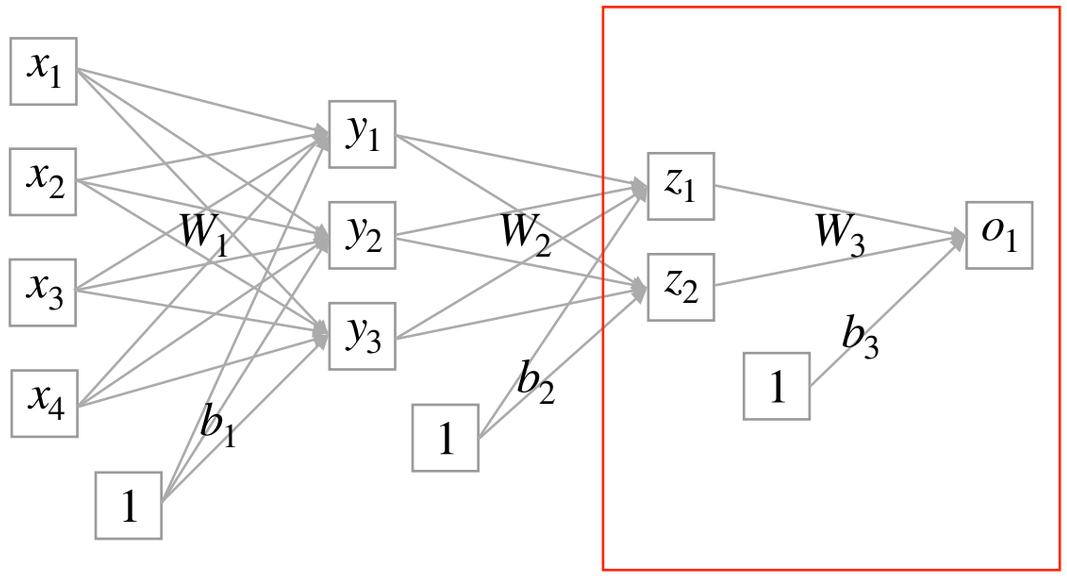
$$y = f_a(xW + b)$$





Neural networks

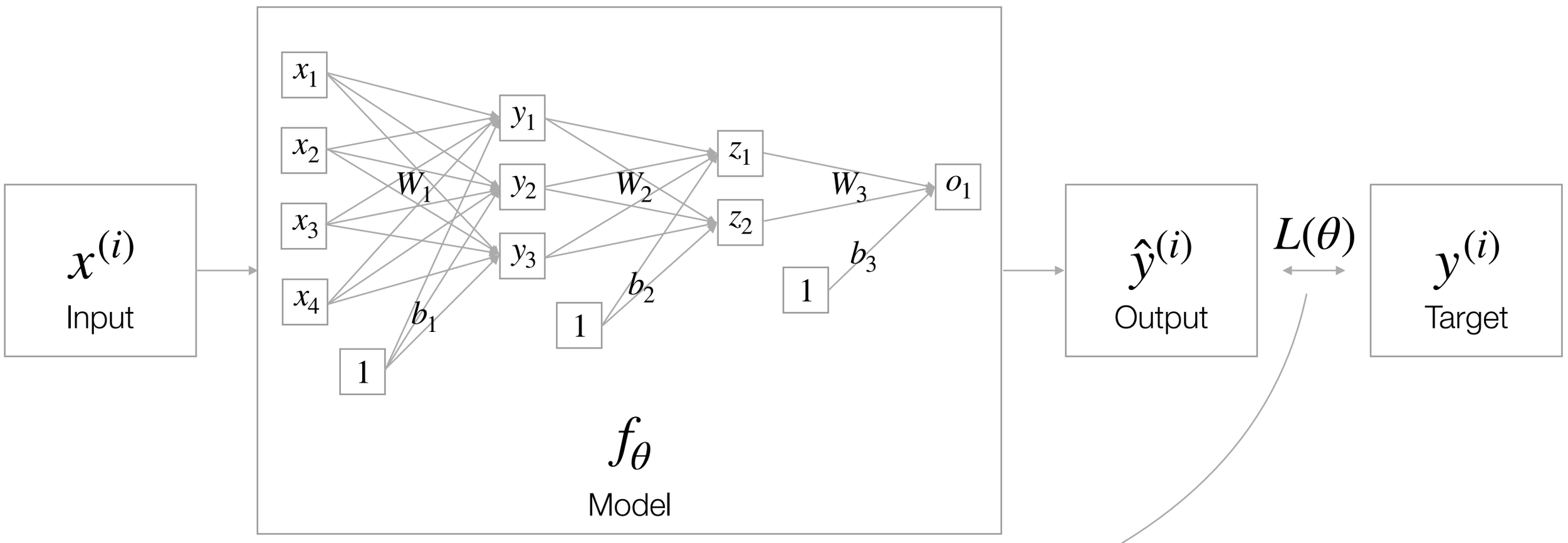
$$y = f_a(xW + b)$$



$$f_a\Big(f_a\Big(f_a\Big(\underset{1\text{x}4}{x} \times \underset{4\text{x}3}{W_1} + \underset{1\text{x}3}{b_1} \Big) \times \underset{3\text{x}2}{W_2} + \underset{1\text{x}2}{b_2} \Big) \times \underset{2\text{x}1}{W_3} + \underset{1\text{x}1}{b_3} \Big)$$

Recap to the “Supervised Learning”

$$\hat{y}^{(i)} = f_\theta(x^{(i)})$$



$$\theta = \{W_1, b_1, W_2, b_2, W_3, b_3\}$$

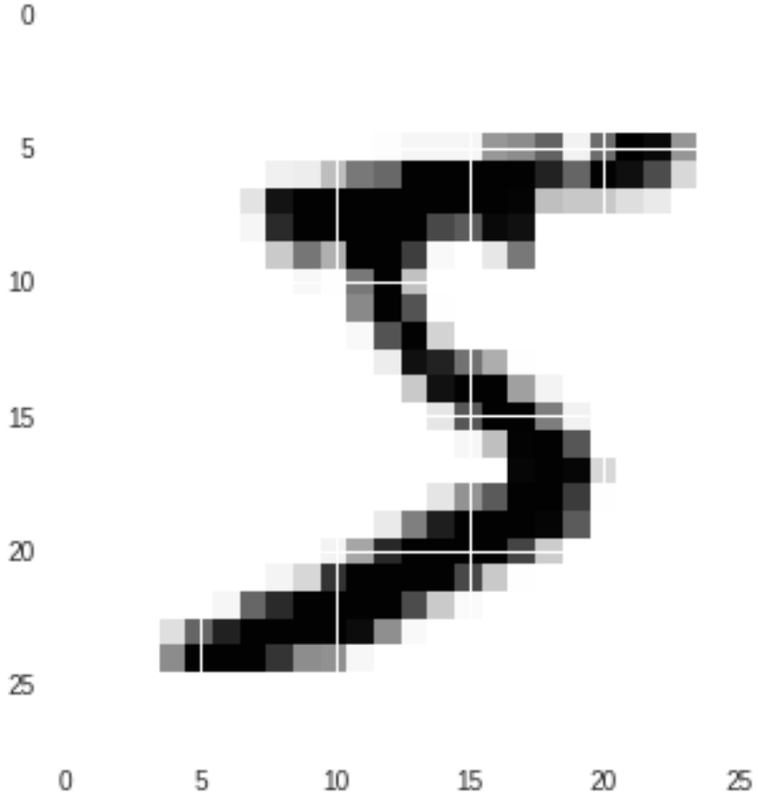
$$\theta^{(i+1)} = \theta^{(i)} - \alpha \cdot \nabla L(\theta)$$

That’s all! Questions?

Hands-on time!

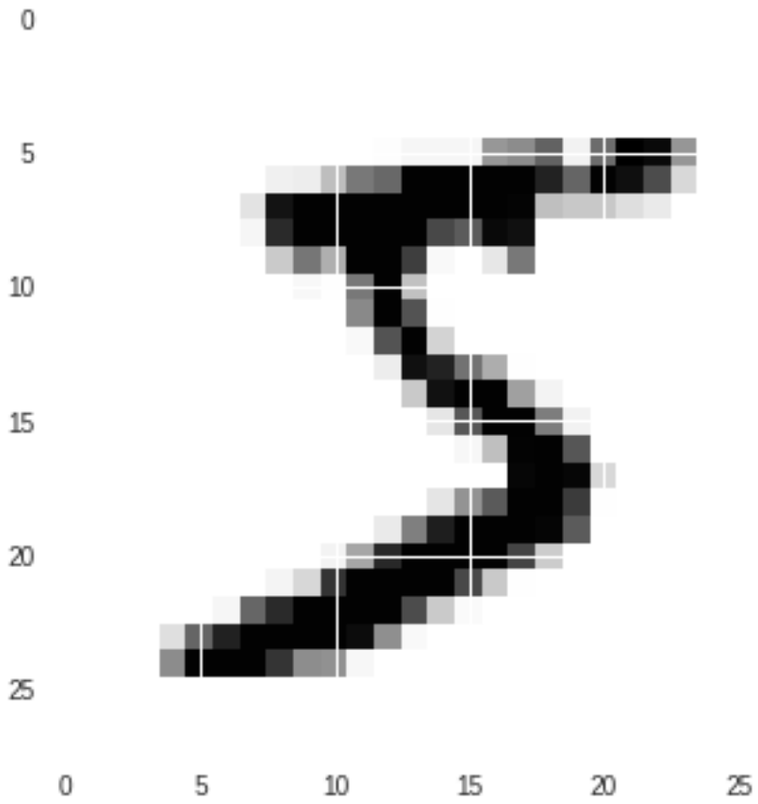
Hands-on problem

1x28x28



Hands-on problem

1x28x28

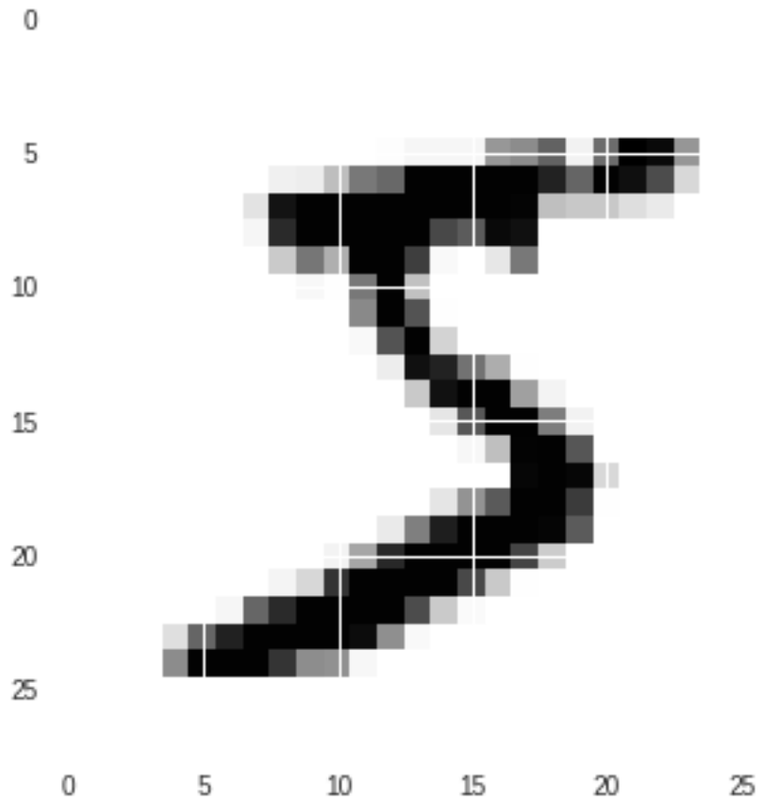


1x10

0	0
1	0
2	0
3	0
4	0
5	1
6	0
7	0
8	0
9	0

Hands-on problem

1x28x28



1x784

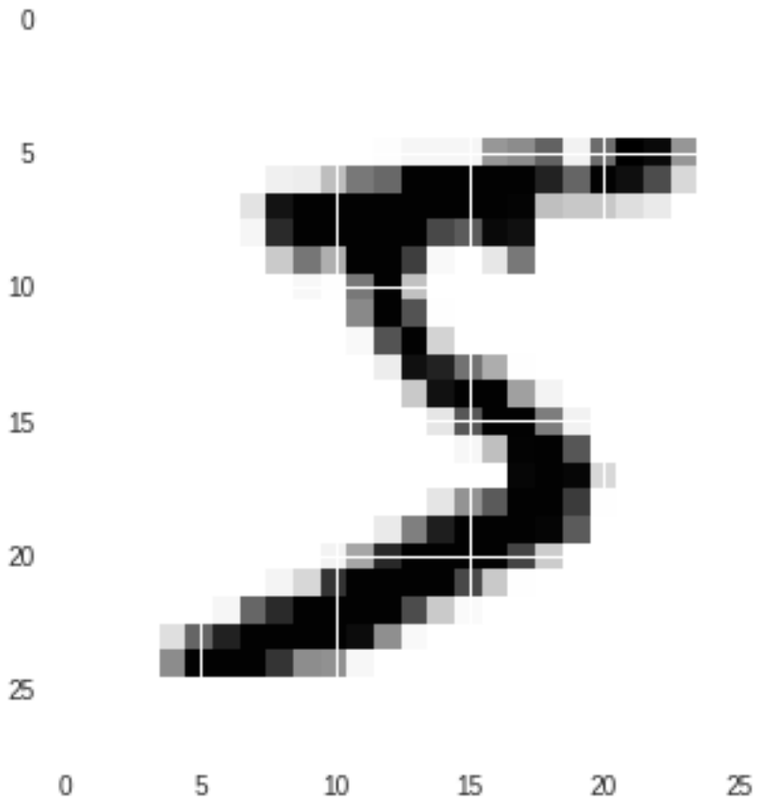
0	0
1	0
2	0
3	0
4	0.5
5	0.8
...	
781	1.0
782	0
783	0

1x10

0	0
1	0
2	0
3	0
4	0
5	1
6	0
7	0
8	0
9	0

Hands-on problem

1x28x28



1x784

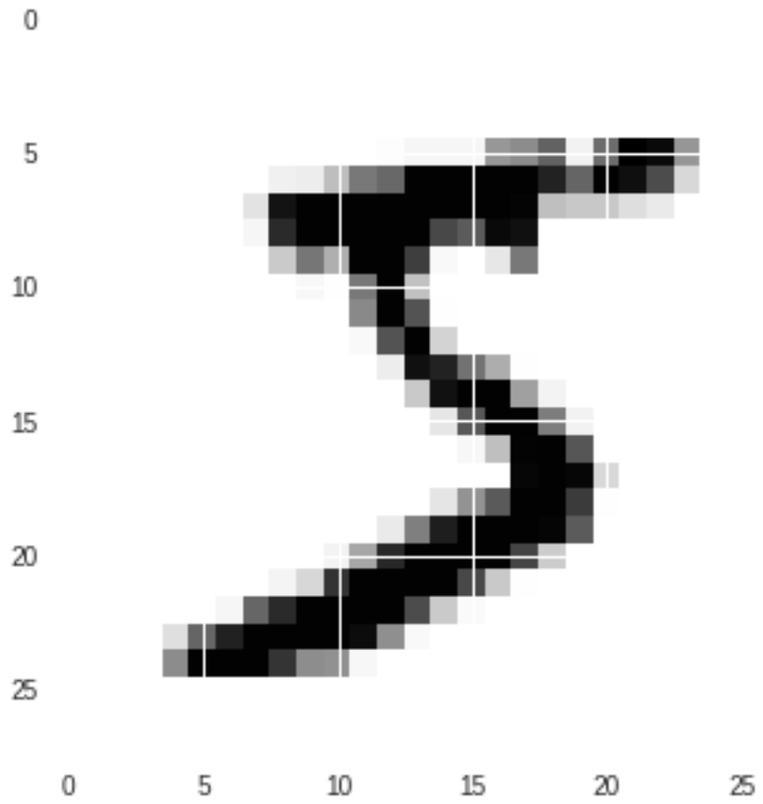
0	0
1	0
2	0
3	0
4	0.5
5	0.8
...	
781	1.0
782	0
783	0

1x10

0	0
1	0
2	0
3	0
4	0
5	1
6	0
7	0
8	0
9	0

Hands-on problem

1x28x28



1x784

0	0
1	0
2	0
3	0
4	0.5
5	0.8
...	
781	1.0
782	0
783	0

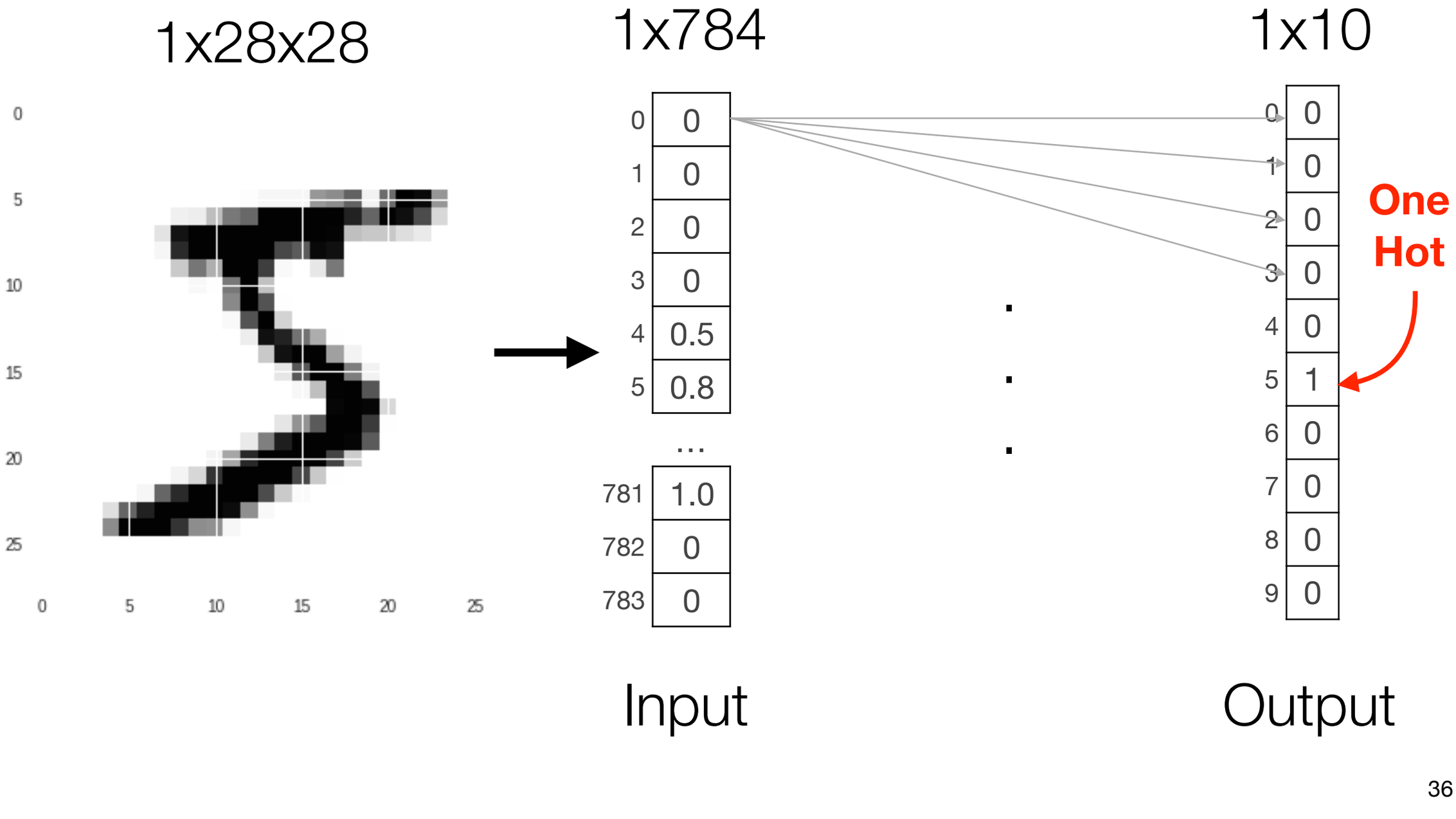
Input

1x10

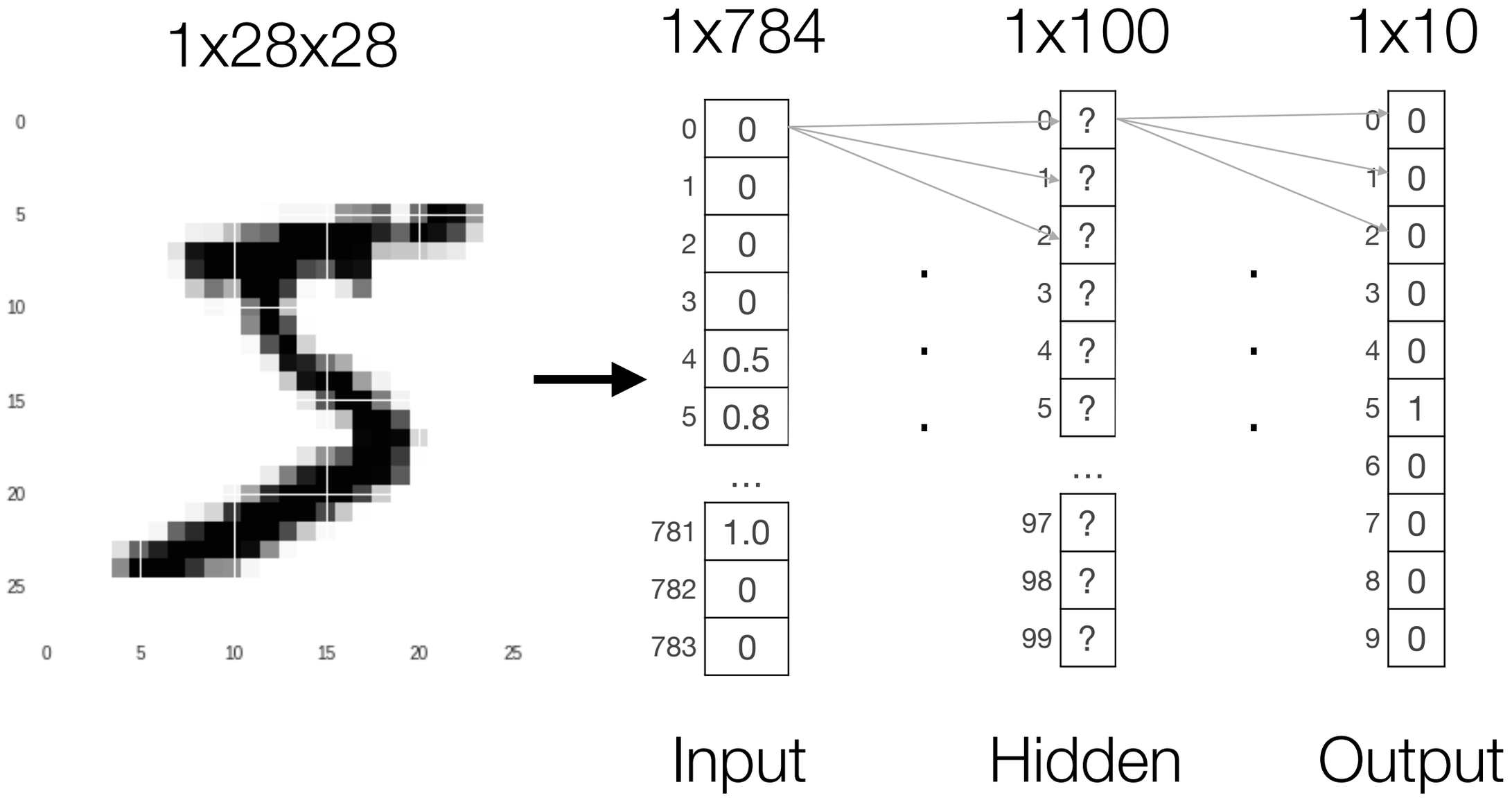
0	0
1	0
2	0
3	0
4	0
5	1
6	0
7	0
8	0
9	0

Output

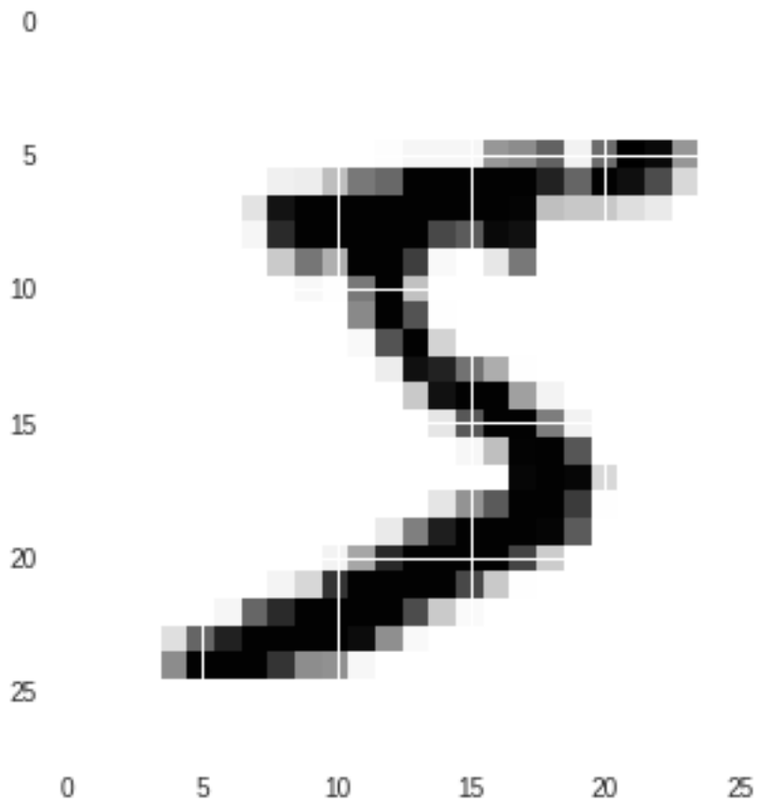
Hands-on problem



Hands-on problem



Hands-on problem

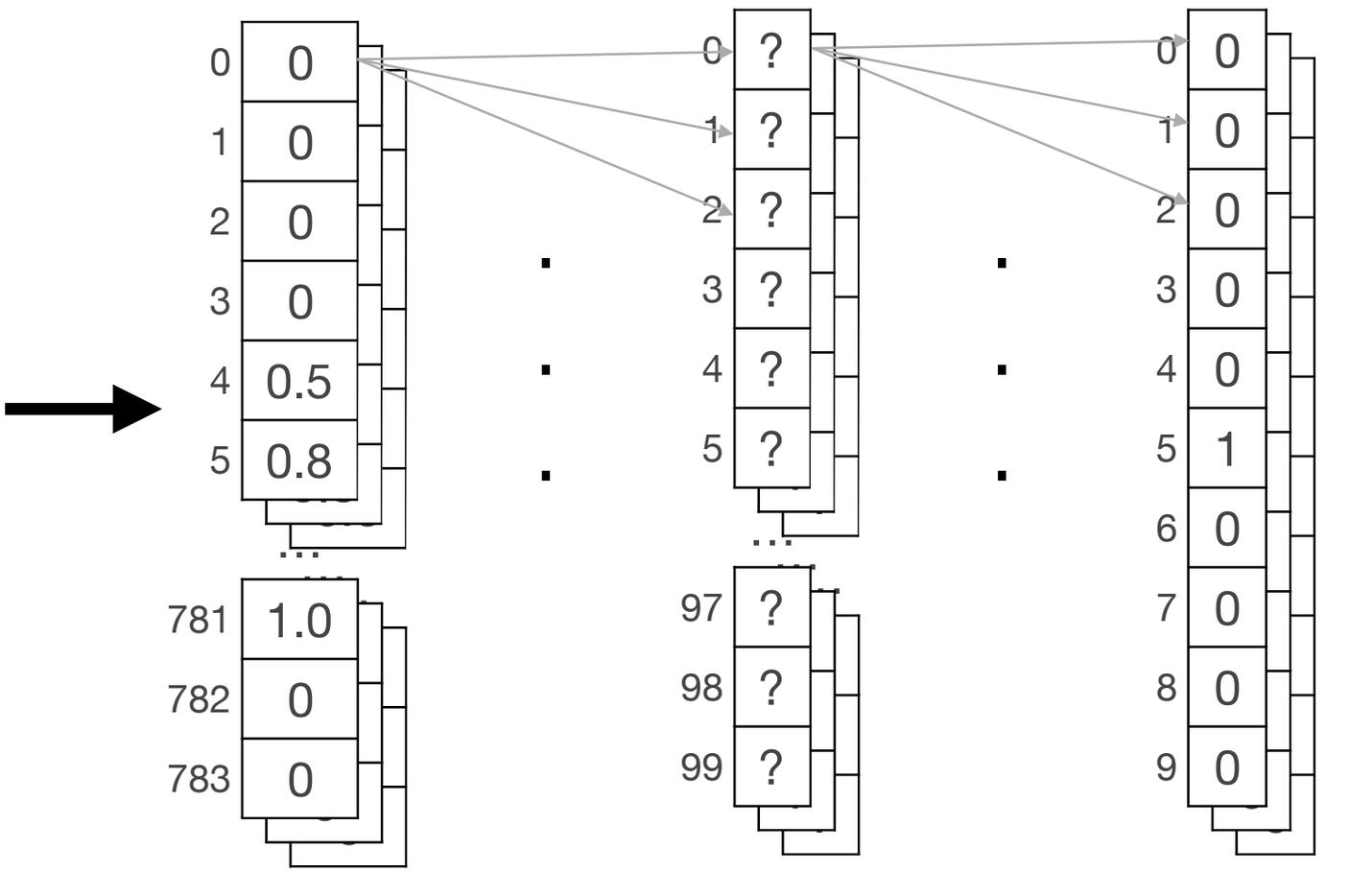


60000x28x28

60000x784

60000x100

60000x10



Input

Hidden

Output

Softmax for classification (on output layer)

$$f_a(z) = \frac{e^z}{\sum_i^N e^{z^{(i)}}}$$

0	0,30
1	0,04
2	2,10
3	0,20
4	0,01
5	7,00
6	0,40
7	0,10
8	0,40
9	0,00
Sum	10,55

Softmax

Probability distribution

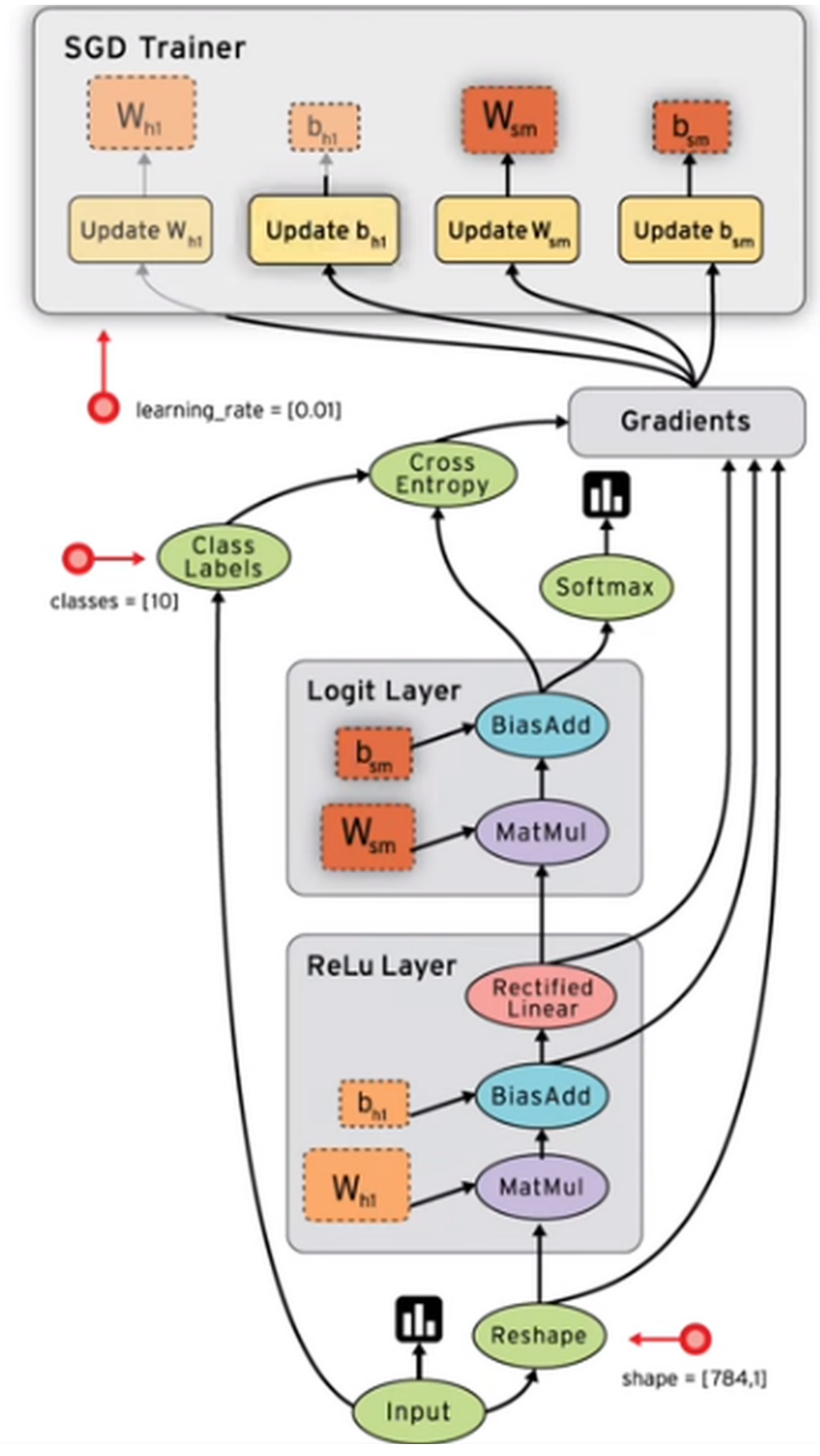
0	0,0013
1	0,0009
2	0,0076
3	0,0010
4	0,0009
5	0,9839
6	0,0013
7	0,0009
8	0,0013
9	0,0009
Sum	1

TensorFlow

Really Quick introduction

Graph & Ops

- **TensorFlow** bases on computational graphs.
- Their compilation speeds up computations!
- Easier backpropagation!
- Graph consists of operations (**ops**).
- Operation can define a single **Tensor** or execute some computations against Tensors.



Example code

```
>>> import tensorflow as tf

>>> a = tf.constant([[1.0, 2.0], [3.0, 4.0]])
>>> b = tf.constant([[1.0, 1.0], [0.0, 1.0]])
>>> c = tf.add(a, b)
>>> print(a)
Tensor("Const:0", shape=(2, 2), dtype=float32)
>>> print(b)
Tensor("Const_1:0", shape=(2, 2), dtype=float32)
>>> print(c)
Tensor("Add:0", shape=(2, 2), dtype=float32)

>>> sess = tf.Session()
>>> result = sess.run(c)
>>> print(result)
[[2. 3.]
 [3. 5.]]
>>> sess.close()
```

Let’s move to the Google Colab!

https://bit.ly/2RFWSsf

Homework

- Experiment with different **learning rate** & **batch size**,
- Check what will happen if you change the way you **initialize layers**,
- Extend your model and add **another fully-connected layer**,
- **Remove data preprocessing**, what will happen?
- Learn about **backpropagation**!

Where to get more information?

- Great introduction & visualization of Neural Networks by 3Blue1Brown on YouTube,
- TensorFlow in 5 minutes by Siraj Raval on YouTube,
- Short videos about Neural Networks by Welch Labs on YouTube,
- Part I: Applied Math and Machine Learning Basics of Deep Learning Book,
- A Visual and Interactive Guide to the Basics of Neural Networks by Jay Alammar's (Blog post),
- CS229 Course from Stanford (lectures can be found on YouTube),
- CS231n Course from Stanford (lectures can be found on YouTube),

Thank you!

Jakub Powierza