Welcome to 521153S

# Deep Learning

Assistant Professor **Li Liu**
Center for Machine Vision and Signal analysis
http://www.ee.oulu.fi/~lili/LiLiuHomepage.html
2020 October 26

# Outline

- Course Information
- Introduction to Deep Learning
- Some Deep Learning Basics

# Course Information

UNIVERSITY OF OULU

- Remote Course
  - Zoom
  - Please check course webpage for the zoom links
- Signup link:
  - Please register in weboodi first if you want to obtain credits
- Course webpage:
  - https://moodle.oulu.fi/mod/page/view.php?id=222290
  - Lecture slides, assignments, project, grades

# Course Information

- Lecturer: Li Liu
  - Li Liu
- Teaching assistants
  - Lam Huynh
  - Zhuo Su
  - Yawen Cui
  - Mohammad Tavakolian
- Questions
  - Zhuo Su
  - Mohammad Tavakolian

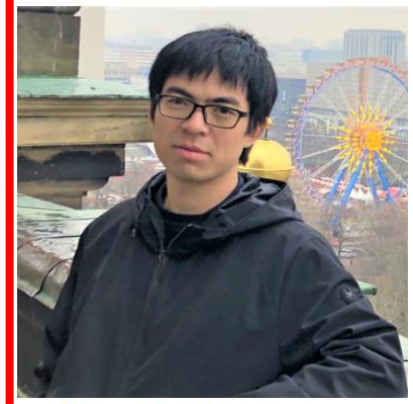
Emails:

firsaname.lastname@oulu.fi

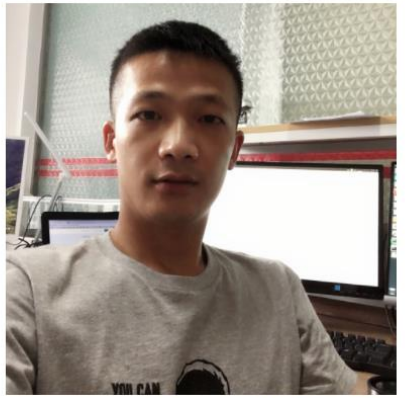
Jie Huang (2020-2024)
PhD student

Wuti Xiong (2020-2024)
PhD student

Yawen Cui (2019-2023)
PhD student

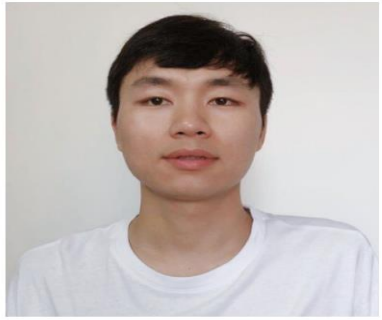


Zhuo Su (2018-2022)
PhD student



Changchong Sheng (2019-2021)
Visiting PhD student

Mohammad Tavakolian
PhD Student



Linpu Fang (2019-2020)
Visiting PhD student


Wanxia Deng (2018-2020)
Visiting PhD student

Xiaoting Wu
PhD Student

# Course Information

UNIVERSITY OF OULU

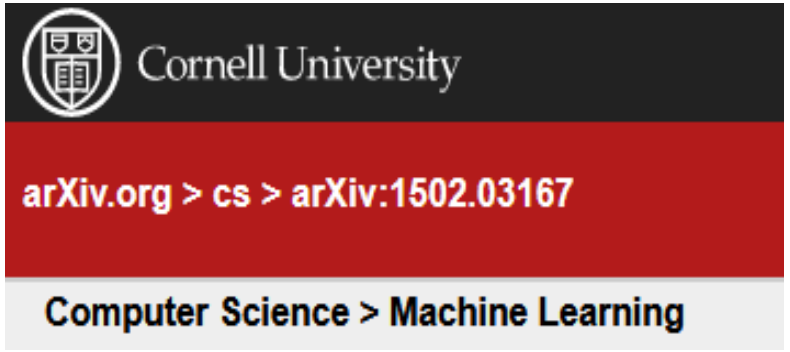
- 11 lectures + 1 tutorial on PyTorch
- 4 assignments + 1 final project
  - Assignment #1: 15%
  - Assignment #2: 15%
  - Assignment #3: 15%
  - Assignment #4: 15%
  - Final Project: 40%
- No exam.
- PyTorch for assignments and final project.

# Course Schedule

| No. | Date | Time | Lecture | Instructor |
|---|---|---|---|---|
| 1 | 2020/10/26 (Mon.) | 14:15~16:00 | Introduction | Li |
| 2 | 2020/10/28 (Wed.) | 14:15~16:00 | Deep Learning Softwares and Pytorch | Zhuo |
| 3 | 2020/11/02 (Mon.) | 14:15~16:00 | Deep Learning Basics: Linear/ Logistic Regression, Loss Functions, Optimization | Li |
| Handle out Assigment 1 (Zhuo) | | | | |
| 4 | 2020/11/09 (Mon.) | 14:15~16:00 | Neural Networks, Deep Neural Networks | Li |
| 5 | 2020/11/11 (Wed.) | 14:15~16:00 | Convolutional Neural Networks | Li |
| Handle out Assigment 2 (Lam) | | | | |
| 6 | 2020/11/16 (Mon.) | 14:15~16:00 | LSTM, RNN, with applications in Lip Reading | Changchong |
| 7 | 2020/11/18 (Wed.) | 14:15~16:00 | Object detection and segmentation | Li |
| Handle out Assigment 3 (Zhuo) | | | | |
| 8 | 2020/11/23 (Mon.) | 14:15~16:00 | Generative Adversarial Networks (GANs) | Lam |
| 9 | 2020/11/25 (Wed.) | 14:15~16:00 | Network Compression | Zhuo |
| Handle out Assigment 4 (Lam) | | | | |
| 10 | 2020/11/30 (Mon.) | 14:15~16:00 | Deep Transfer Learning | Mohammad |
| 11 | 2020/12/03 (Wed.) | 14:15~16:00 | Visualizing and Understanding CNNs | Mohammad |
| Handle out the final project (Mohammad) | | | | |
| 12 | 2020/12/07 (Mon.) | 14:15~16:00 | Deep Adervesarial Attacks | Li |

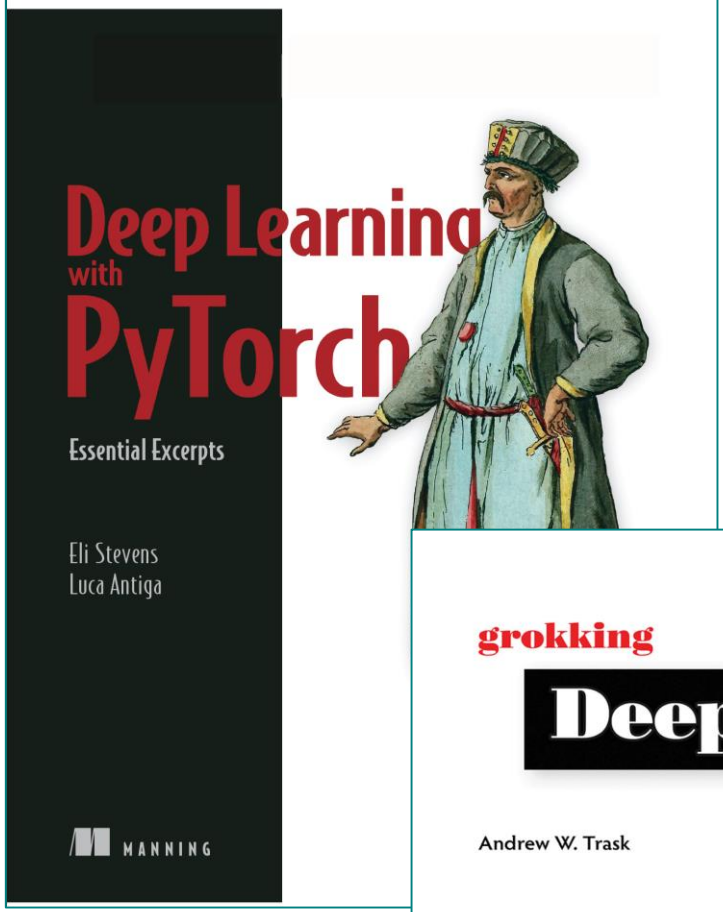
# Selflearning

- Find on social media (Twitter, Facebook …)



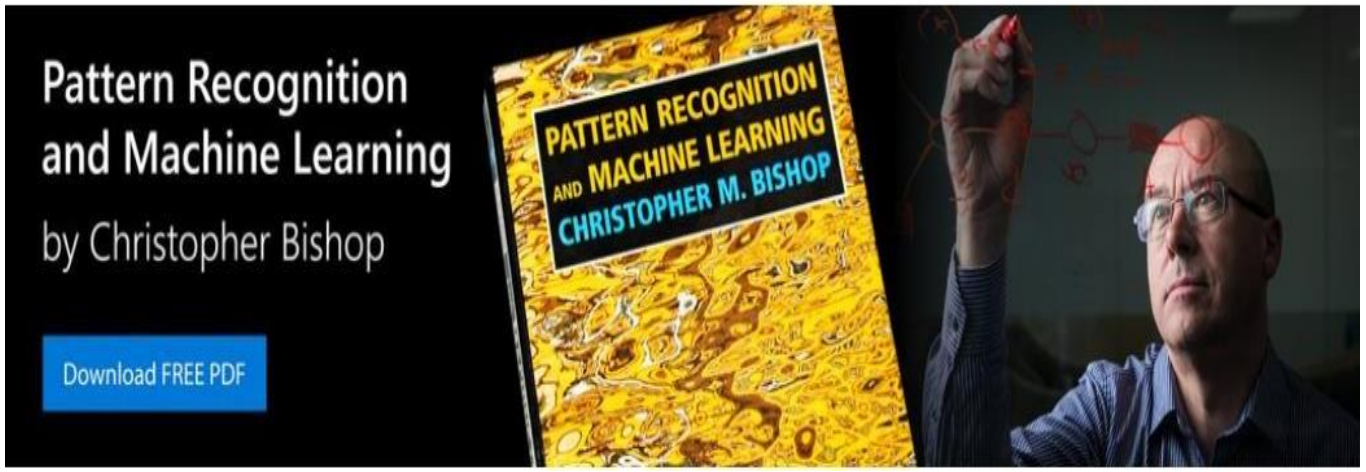

- Read latest papers in a subarea
- Read/Write code to reinforce understanding of concepts

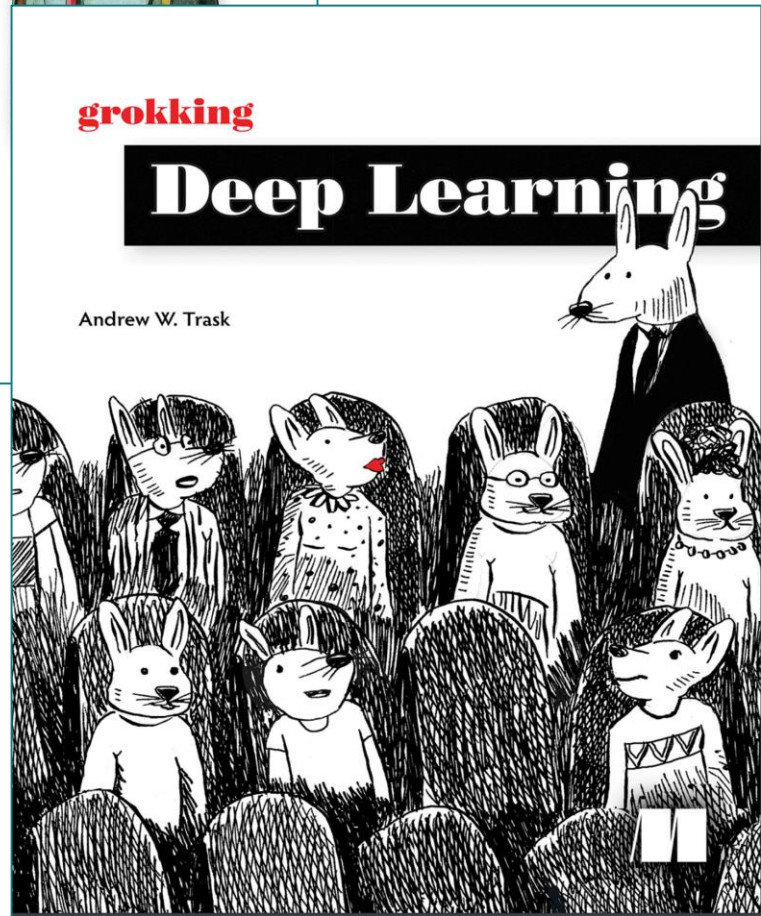
# Books


Form a group to read books and discuss each chapter.

PRML Book https://aka.ms/prml



# Some good blogs

...and many others ...

UNIVERSITY OF OULU

Andrej Karpathy

- Yes you should understand backprop
- A Recipe for Training Neural Networks
- The Unreasonable Effectiveness of Recurrent Neural Networks
- A Survival Guide to a PhD

Chris Olah

- Understanding LSTM Networks
- Calculus on Computational Graphs
- Attention and Augmented Neural Networks

Alexander Rush, Vincent Nguyen, Guillaume Klien

- Annotated Transformer

Robbie Allen

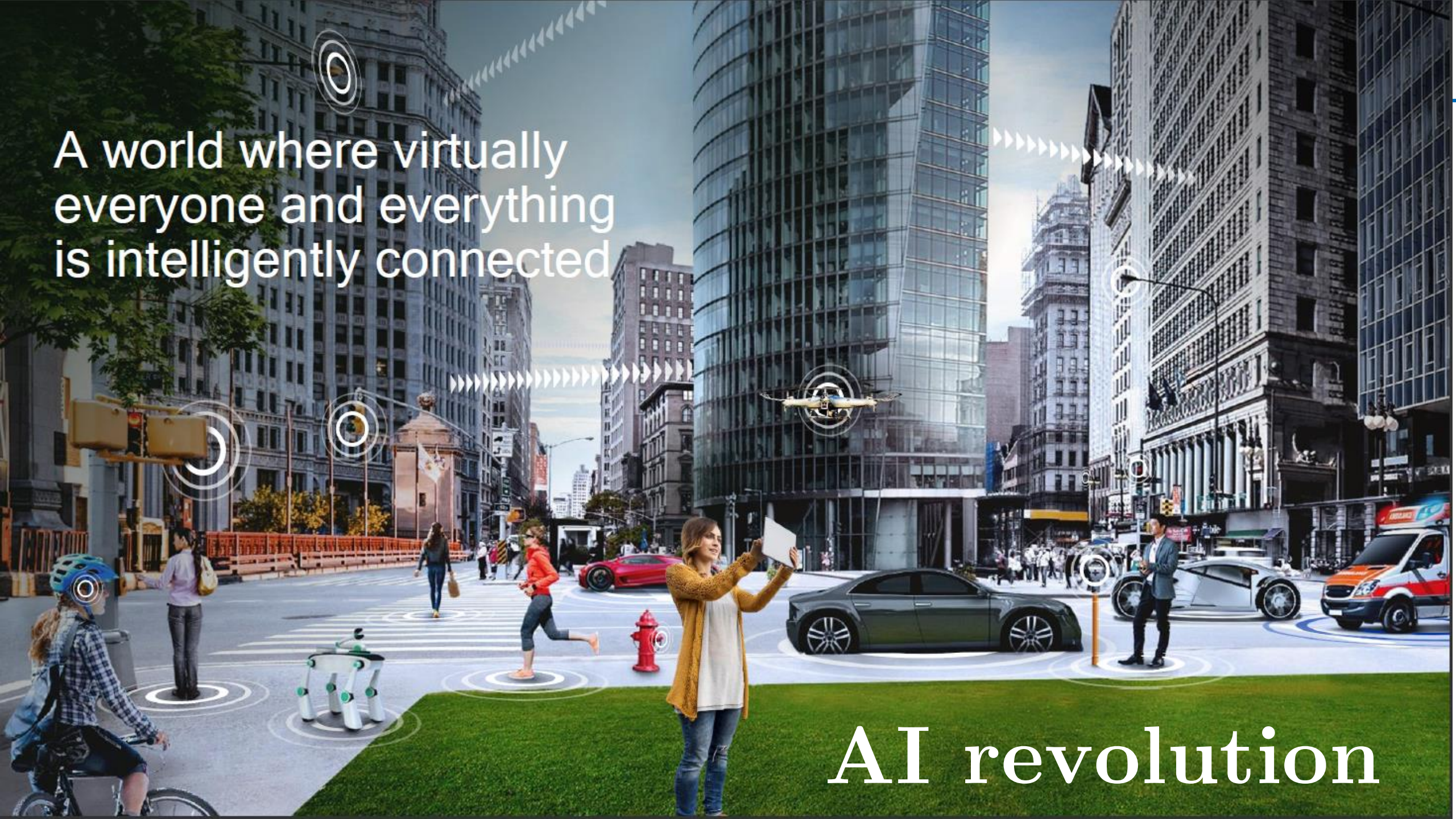
Tutorials: Over 200 of the Best Machine Learning, NLP, and Python Tutorials

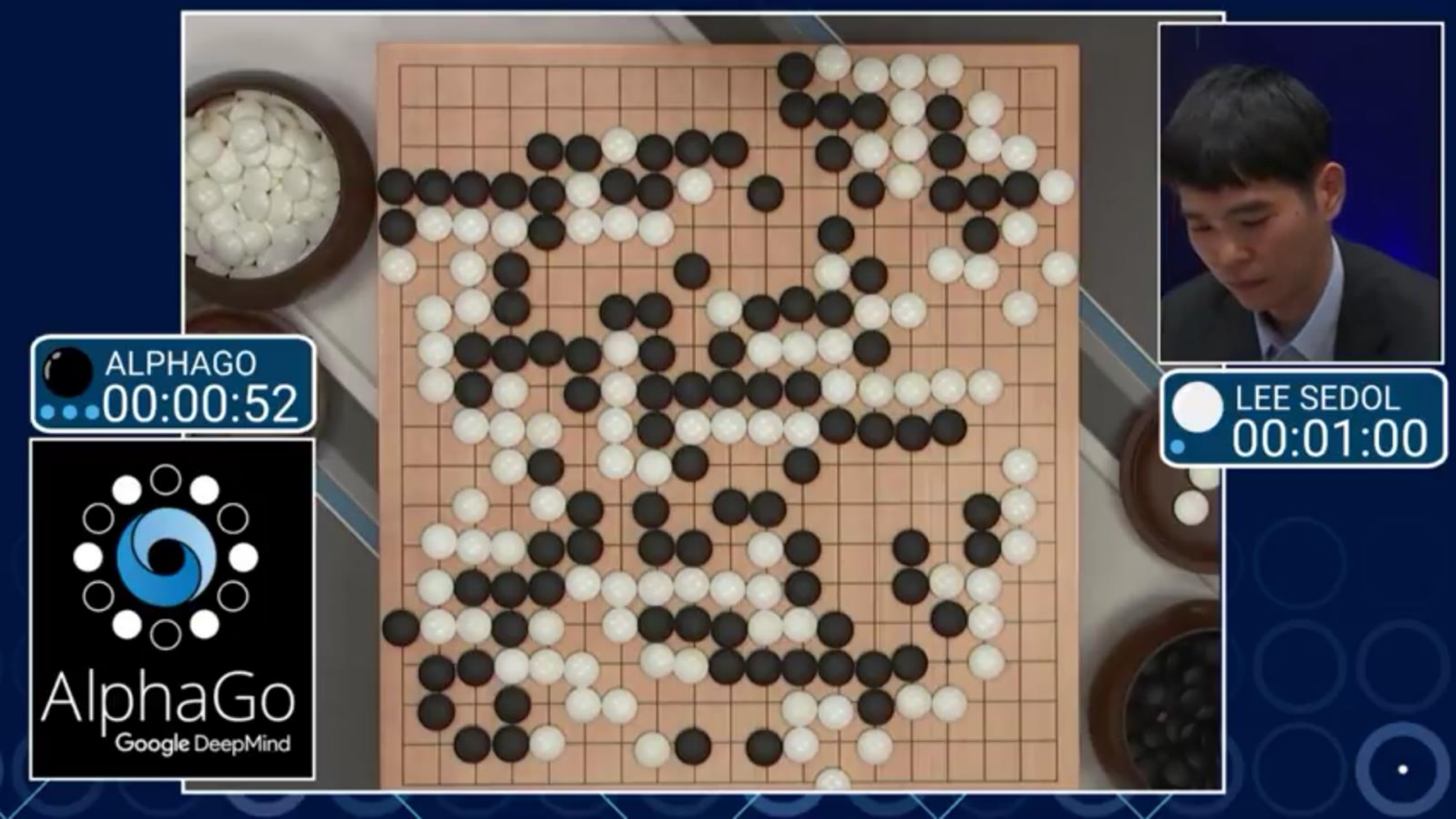
Link: http://bit.ly/36skFE7

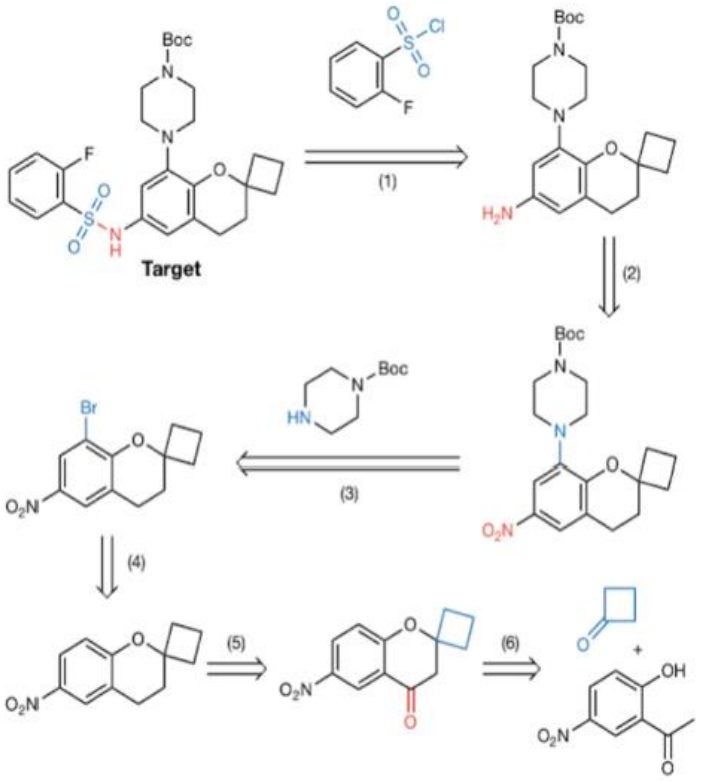
# Outline

UNIVERSITY OF OULU

- Course Information
- Introduction to Deep Learning
- Some Deep Learning Basics


A world where virtually
everyone and everything
is intelligently connected
AI revolution

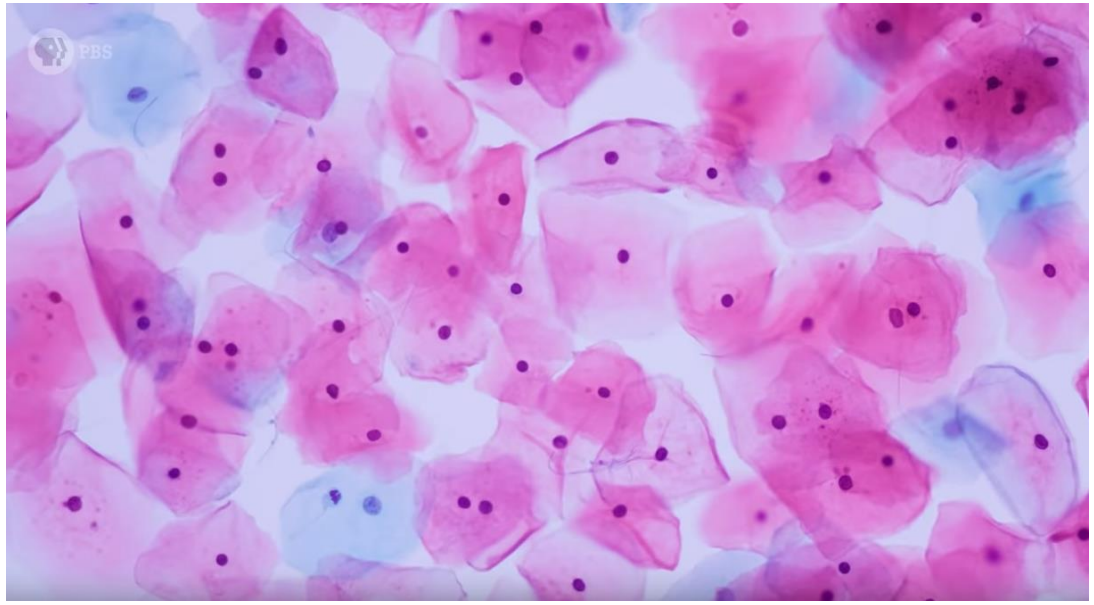

ALPHAGO
00:00:52
AlphaGo
Google DeepMind
LEE SEDOL
00:01:00

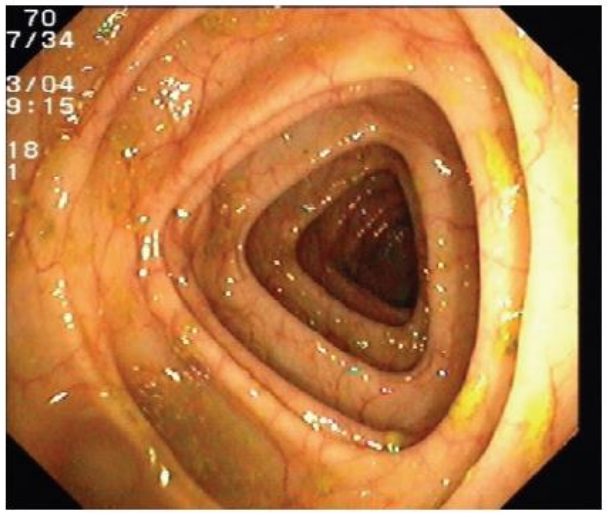



# Algorithm-assisted research

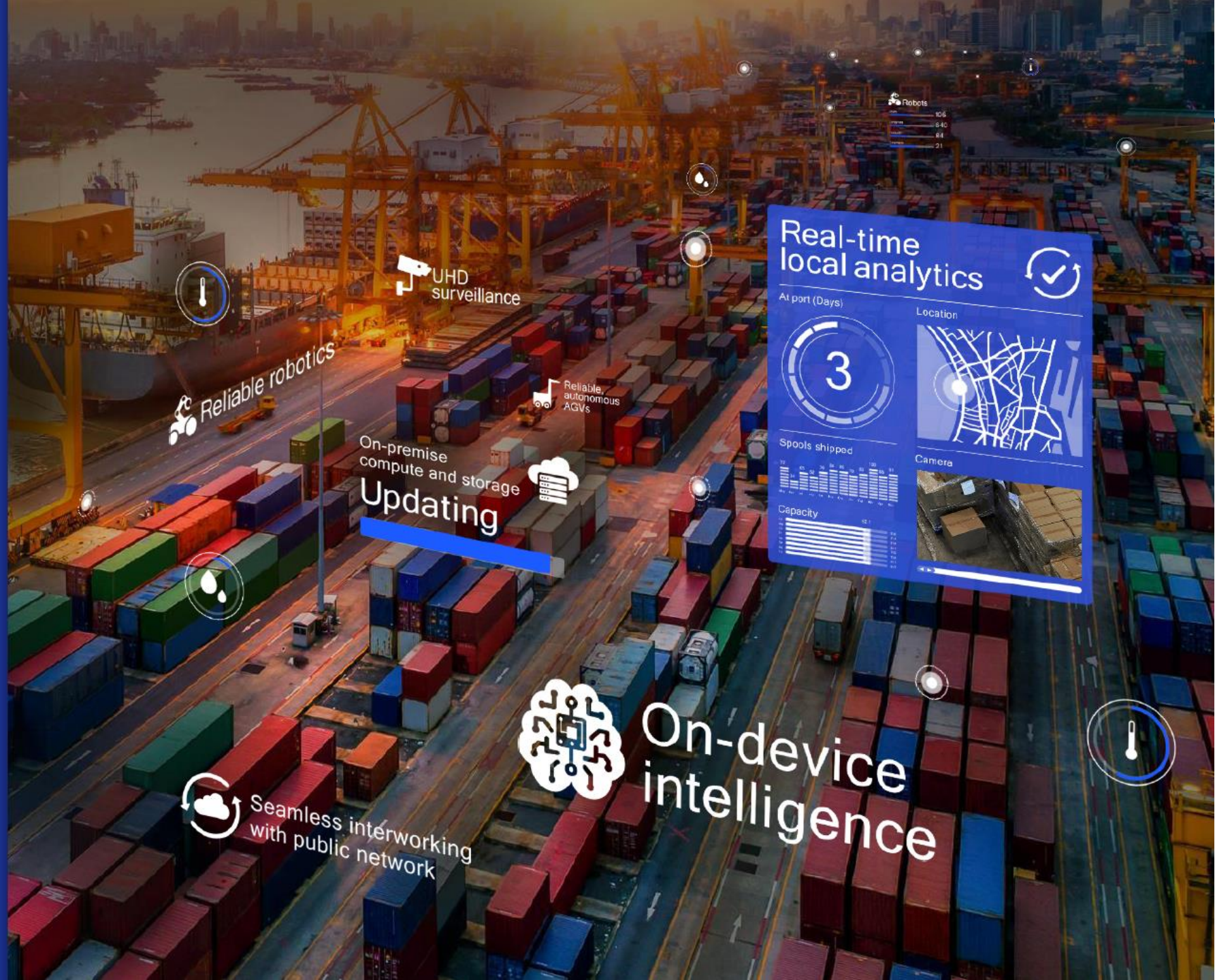
M. H. S. Segler, M. Preuss & M. P. Waller, *Nature*, 2018

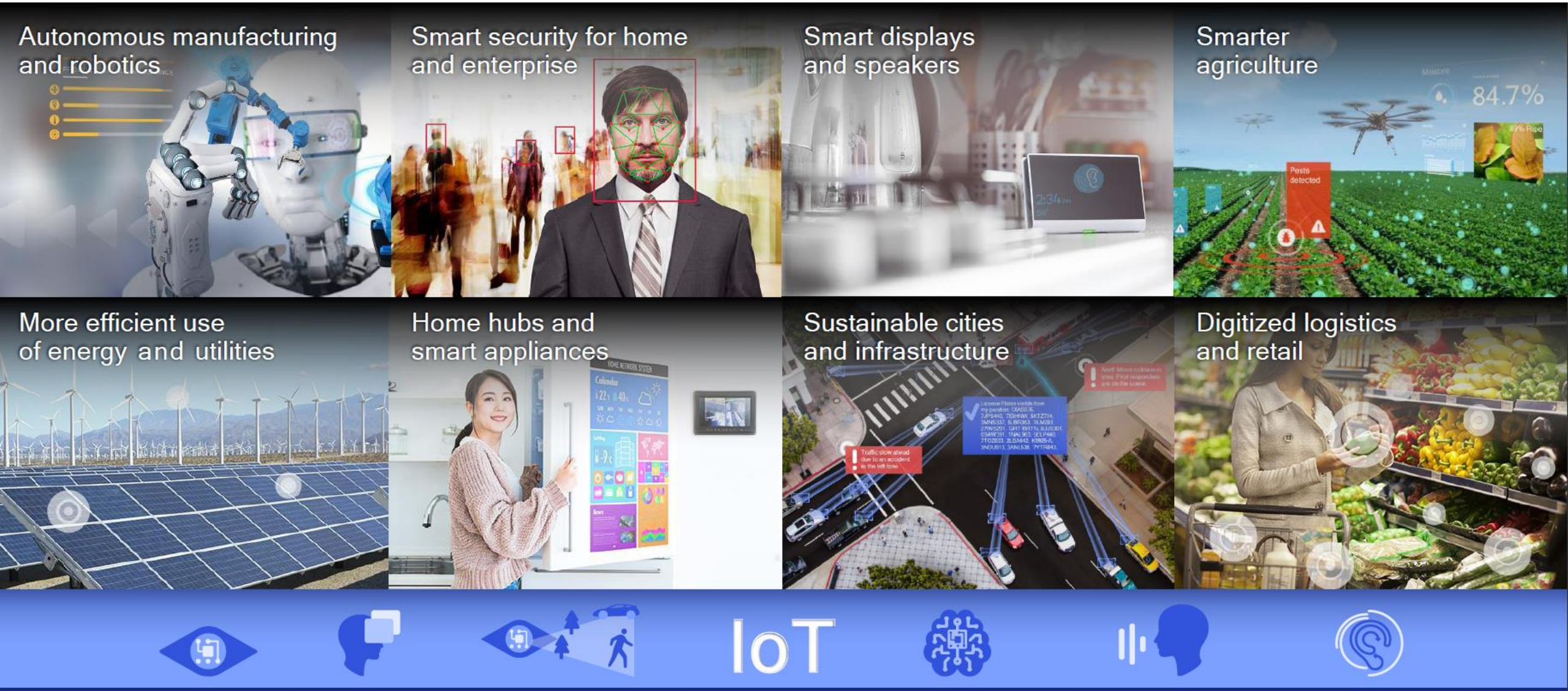
# Assist Medical Diagnosis

# AI will drive transformation across industries



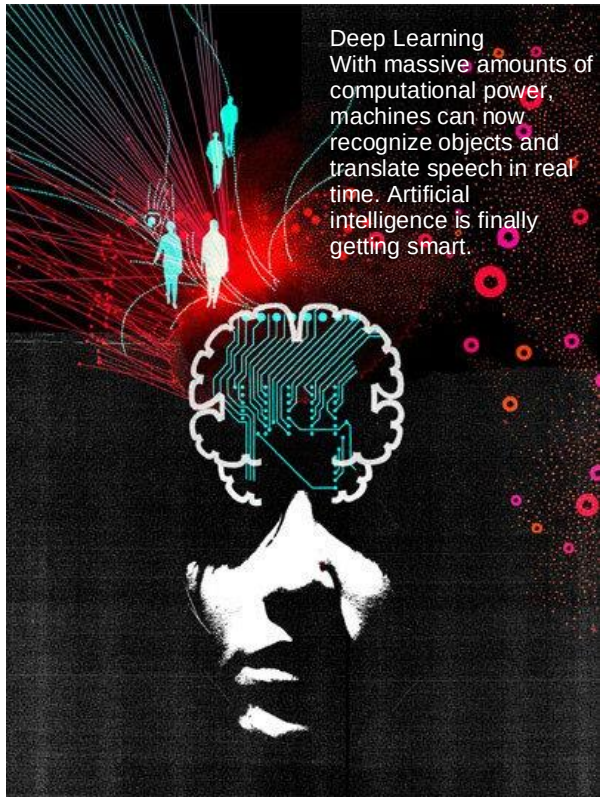



IoT

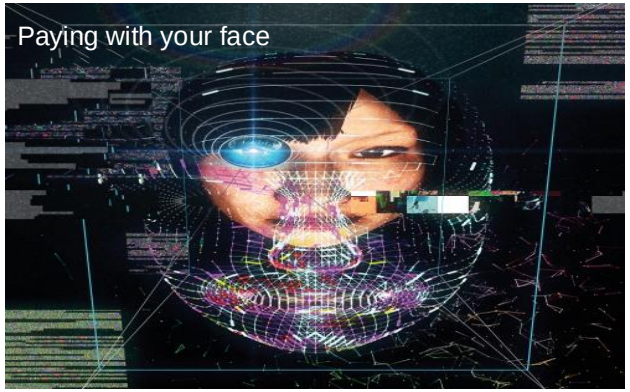
# AI for IoT across the home, industrial/enterprise, and Smart Cities

# Deep learning attracts lots of attention

- I believe you have seen lots of exciting results before.
- MIT Technology Review's 10 Breakthrough Technologies.




2013


2017


2018


2020

This course focuses on the basic techniques.

# Deep Learning in **One** Slide

UNIVERSITY OF OULU

- **What is it**
  Extract useful patterns from data.
- **How**
  Neural network + optimization
- **How (Practical)**
  Python, TensorFlow, PyTorch etc.
- **Hard Part**
  Good Questions + Good Data
- **Why now:**
  Data, hardware, community, tools, investment
- **Where do we stand?**
  Most big questions of intelligence have not been answered nor properly formulated

**Exciting progress:**


Object recognition


Face recognition


Machine translation


Speech recognition


Security authentication


Medical Diagnostics


Play complex games (AlphaGo, DeepStack)


Selfdriving car

Recommendation systems, Robotics…

# Visual Data: The Biggest Big Data


Large Amount of Surveillance Videos

- About 5.9 millions CCTV cameras in the UK, 2016 (1 billion images per day)

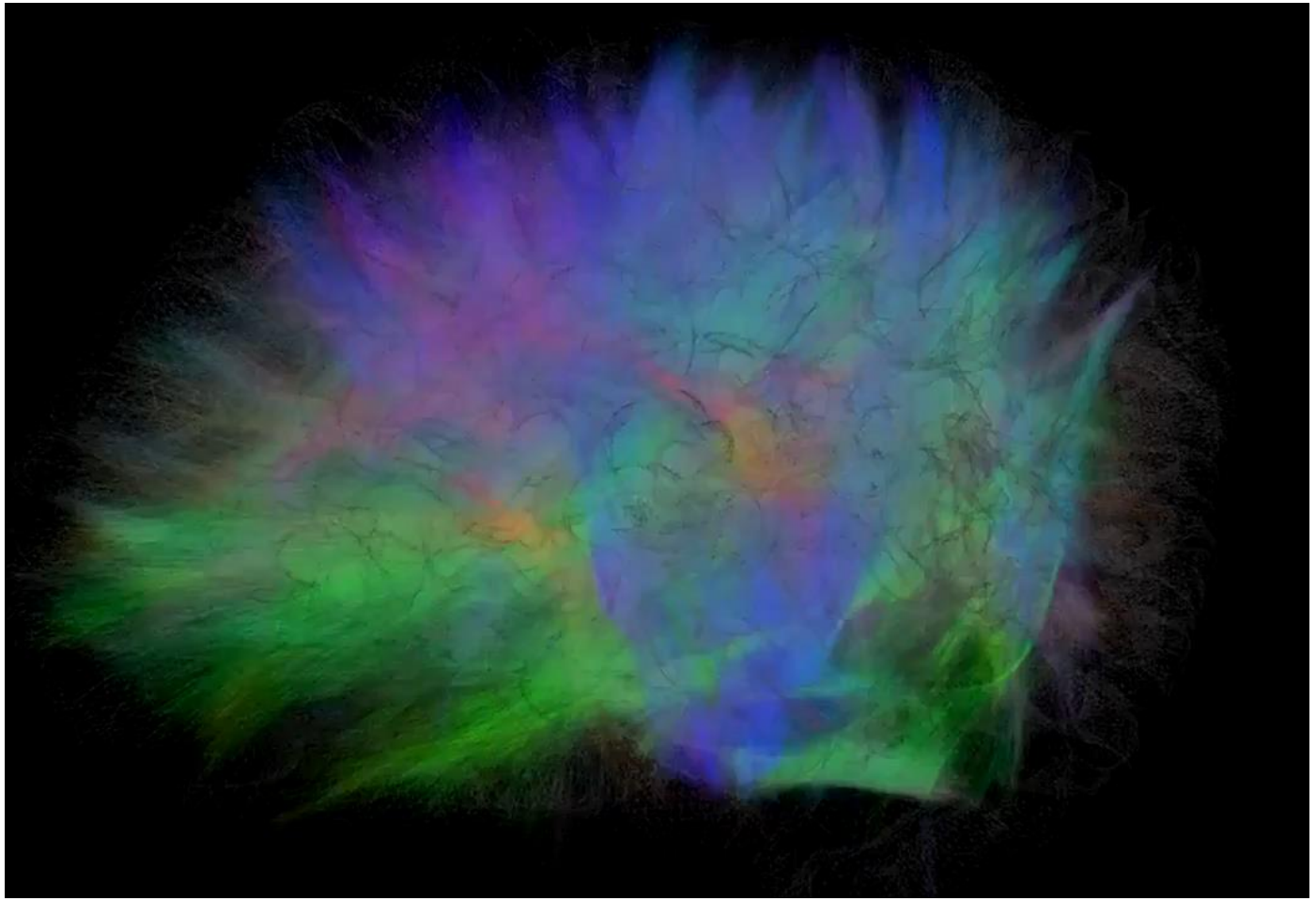
# Visual Data: The Biggest Big Data


- 300 hours' video are uploaded to YouTube every minute!
- More than 90 PB ($1PB=10^6GB$) of videos data every year!

# Visual Data: The Biggest Big Data


“AI began with an ancient wish to forge the gods.”

→Pamela McCorduck , Machines Who Think, 1979



Visualization of 3% of the neurons and 0.0001% of the synapses in the brain.

Thalamocortical system visualization via DigiCortex Engine

Slides from Lex Fridman (MIT)

# A Incomplete History of Deep Learning

UNIVERSITY OF OULU

- 1943: Neural networks (McCulloch and Pitts )
- 1958: Perceptron (Rosenblatt)
- 1974-1986: Backpropagation, RBM, RNN
- 1989-1998: CNN (LeNet), LSTM
- 2012: AlexNet
- 2014: Generative Adversarial Networks (GANs)
- 2014: DeepFace
- 2016: AlphaGo
- 2017: AlphaZero, Capsule Networks
- 2018: BERT

For much, much, *much* more detail, see Schmidhuber's historical overview (Neural Networks, 2015)

# Turing Award for Deep Learning

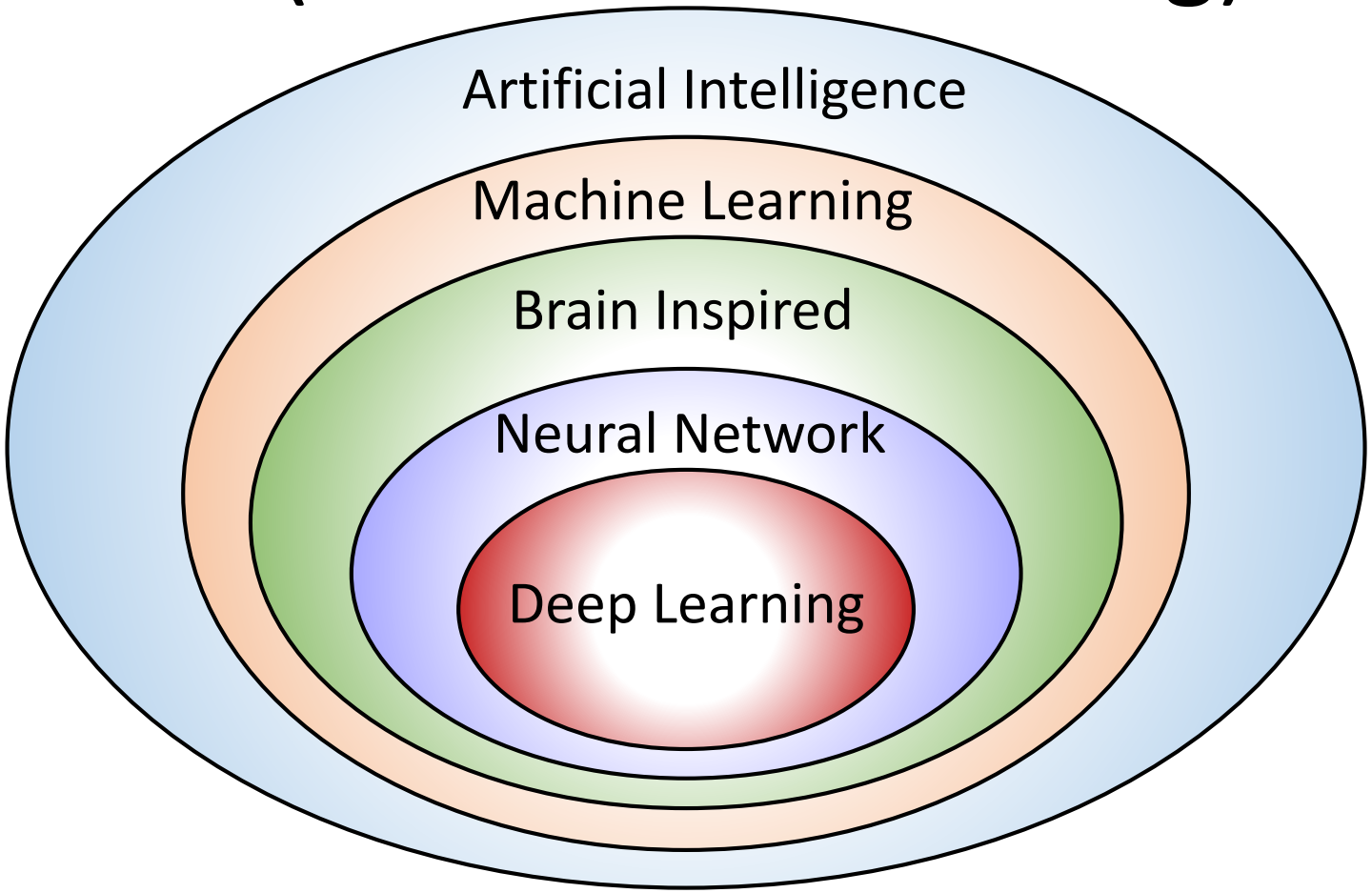
# History of DL Tools

- Mark 1 Perceptron – 1960
- Torch – 2002
- CUDA – 2007
- Theano – 2008
- Caffe – 2014
- TensorFlow 0.1 – 2015
- PyTorch 0.1 – 2017
- TensorFlow 1.0 – 2017
- PyTorch 1.0 – 2017
- TensorFlow 2.0 – 2019
- PyTorch 1.3 – 2019



This course: PyTorch the most popular today

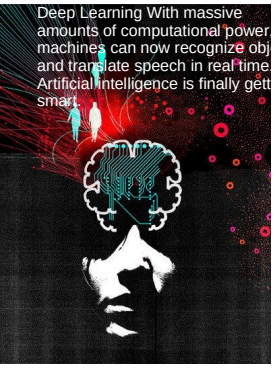
# Deep learning is representation learning (aka feature learning)



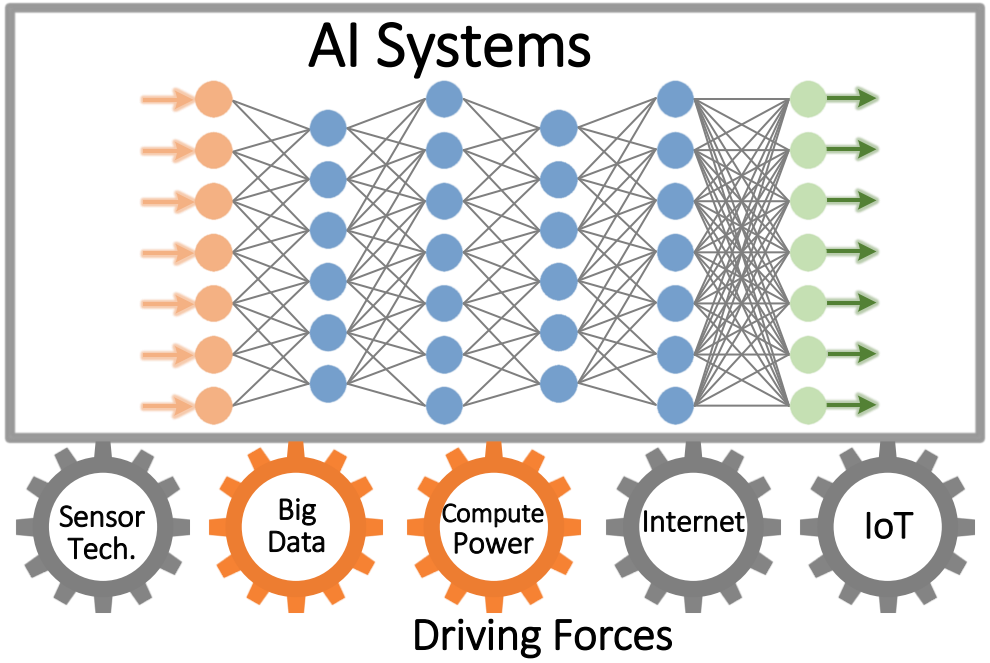

We will focus on **supervised learning** in this course.

Here the data consists of input and output pairs.

# Limitations of Deep Learning






## AI Applications

- Object Detection
- Face Detection
  Facial Expression
- Natural Language
  Understanding
- Speech Recognition
- Security
  Authentication
- Medical Diagnostics
- Playing Complex
  Games
- Transportation,
  Finance, Legal ...

## Key Challenges

**1 High Computational Complexity**
- Requires power hungry computing resources (*e.g.* GPUs)
- Energy hungry
- Very time consuming

**2 Data and Label Hungry**
- Labeling data is labor intensive
- Collecting many labeled data may be hard or impossible

**3 Vulnerable to attacks**
- Fundamentally brittle
- Vulnerable to adversarial attacks

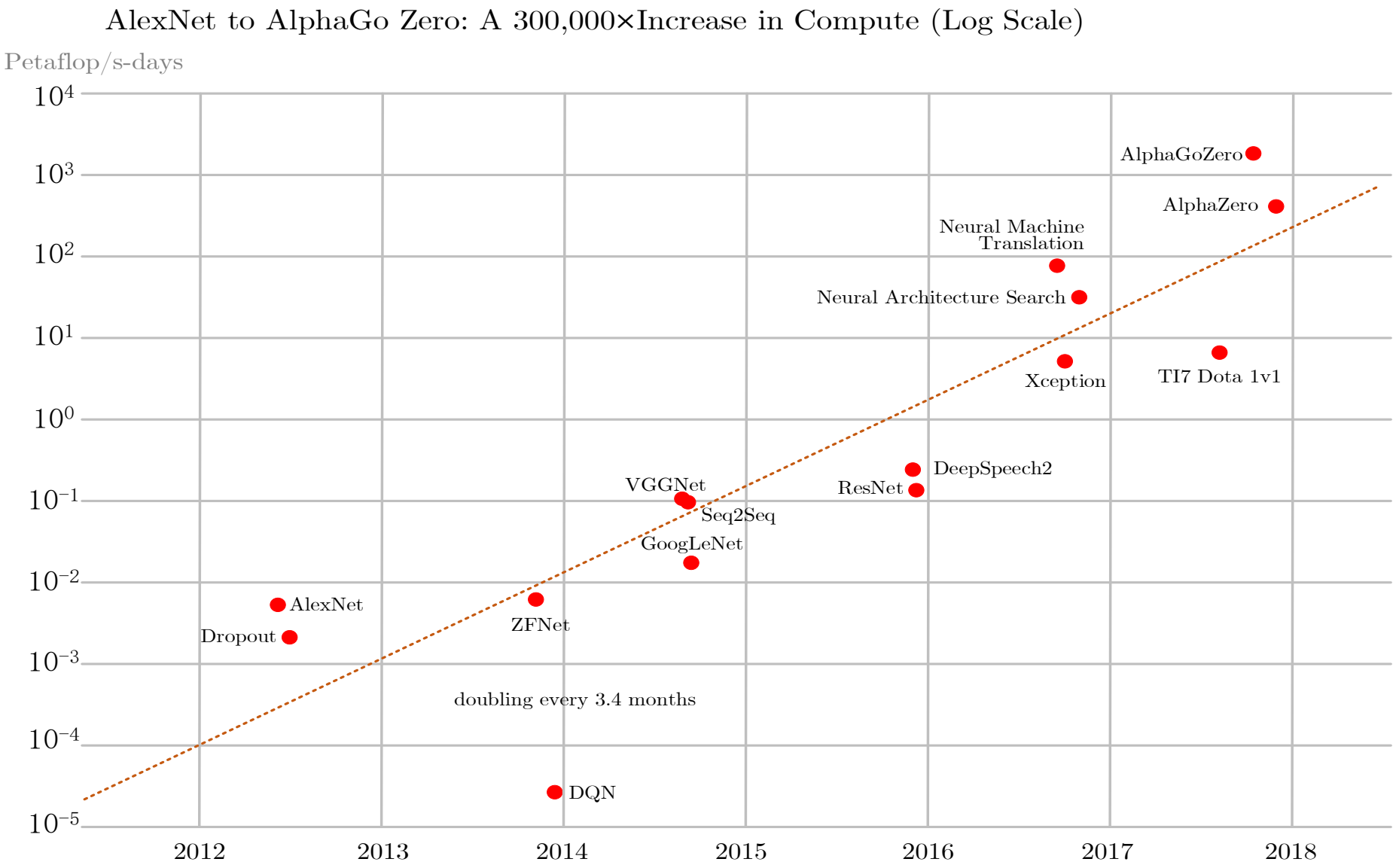
**4 Weak generalizability**

**5 Hard to Be Explainable**
- Why did you do that?
- When do you succeed or fail?
- When can I trust you?
- How do I correct an error?

**Prediction from Rodney Brooks:**

"By 2020, the popular press starts having stories that the era of Deep Learning is over."

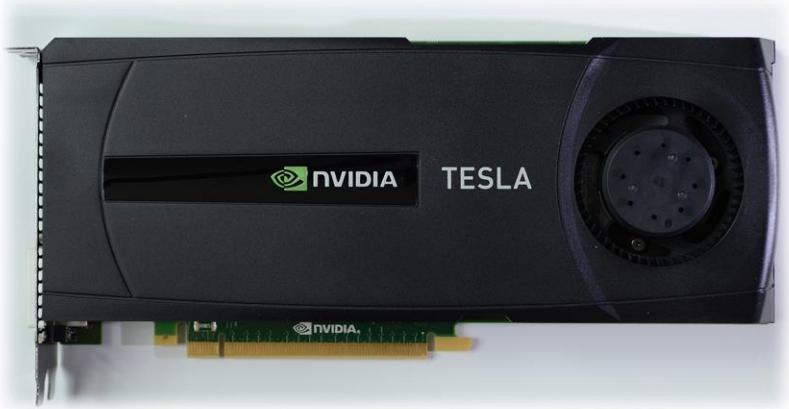
# High Computational Complexity of DL



# Energy Hungry

## Energy Efficiency is Vital at all levels, including cloud, edge and chip

### Cloud AI




- Memory: 32 GB
- Computation: TFLOPS/s

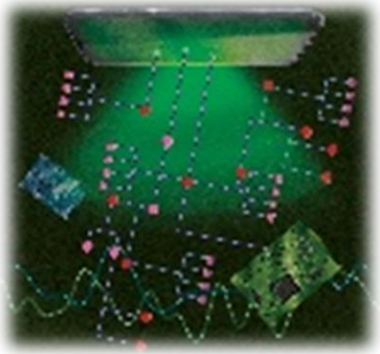
Less Resources

### Mobile AI


- Memory: 4 GB
- Computation: GFLOPS/s

Less Resources

### Tiny AI



- Memory: <100 KB
- Computation: <MFLOPS/s

# Edge AI/Tiny AI

Cloud

AI is Moving from the Cloud to the Edge.


Edge Devices
On Devices

<1 Watt >10 Watts

## On Device AI Benefits

- Privacy protection
- Low latency
- Low power and low cost
- Efficient use of network bandwidth

## On Device AI Challenges

- Thermally efficient for ultralight designs
- Constrained computing power
- Lower power, storage, memory, bandwidth

# Not Environment Friendly

MIT Technology Review

## Training a single AI model can emit as much carbon as five cars in their lifetimes

Deep learning has a terrible carbon footprint.

by **Karen Hao** Jun 6, 2019

### Common carbon footprint benchmarks

in lbs of CO2 equivalent

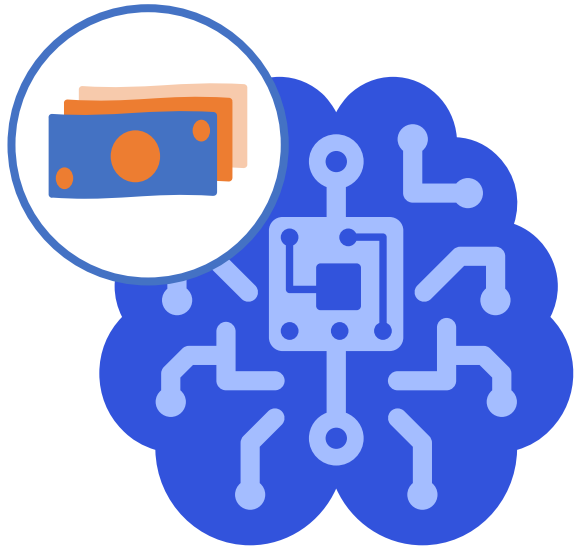




Chart: MIT Technology Review • Source: Strubell et al. • Created with Datawrapper

# Energy Efficient AI

## Value created by AI must exceed the cost to run the service



Economic feasibility per transaction may require cost as low as a microdollar (1/10,000$^{th}$ of a cent)

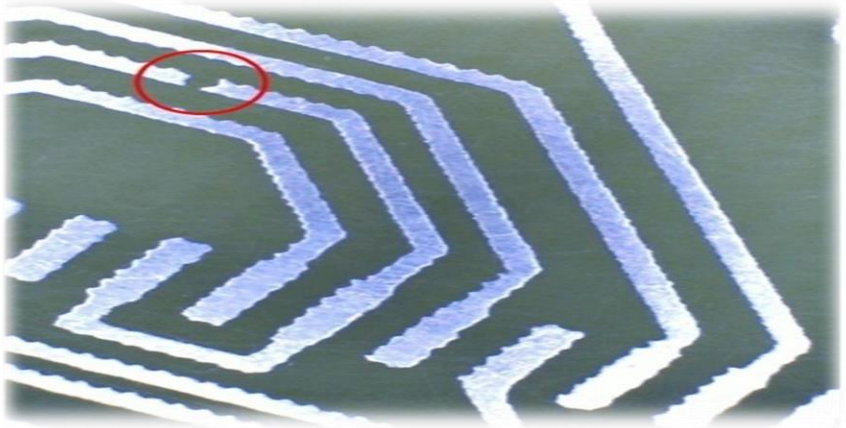
Broad economic viability requires energy efficient AI

# Data and Label Hungry

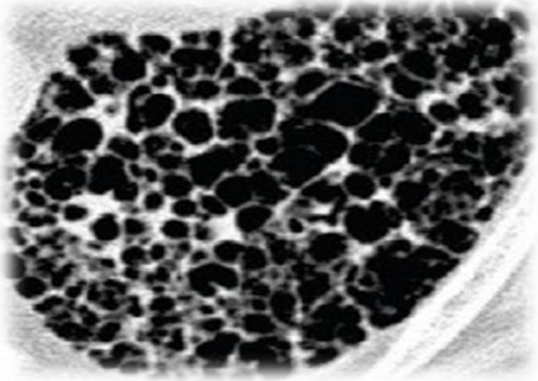
# Data and Label Hungry

UNIVERSITY OF OULU

Industrial inspection



Medical domain

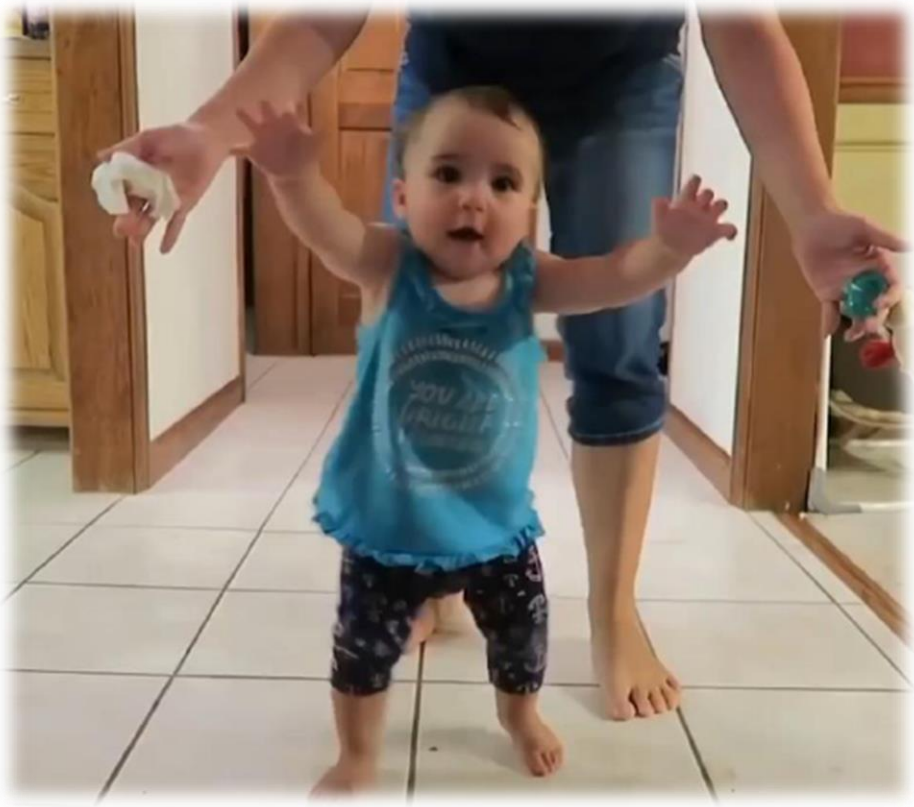


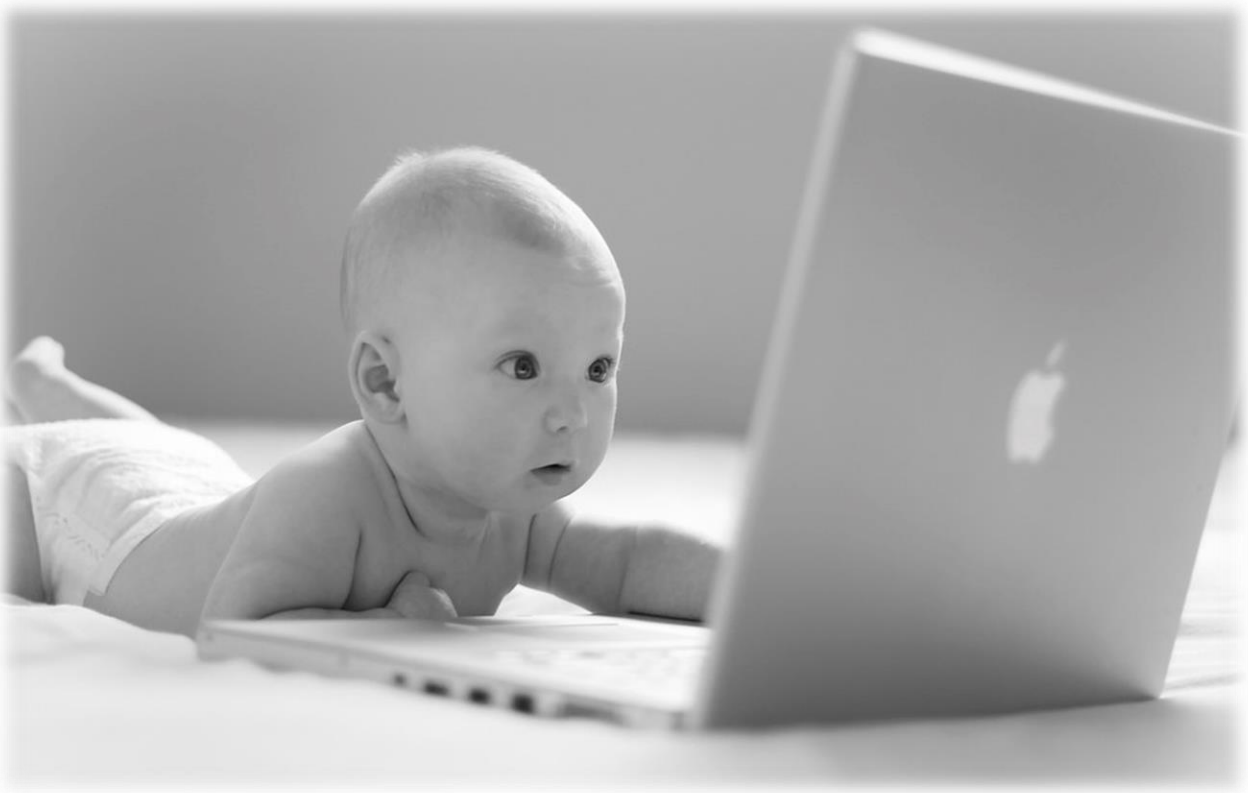
Endangered species

# Data and Label Hungry

"Labels are the opium of the machine learning researcher."

——Jitendra Malik





A baby doesn't learn by downloading data from the Internet.

# Deep Learning Deep Trouble (Nature, 2019)

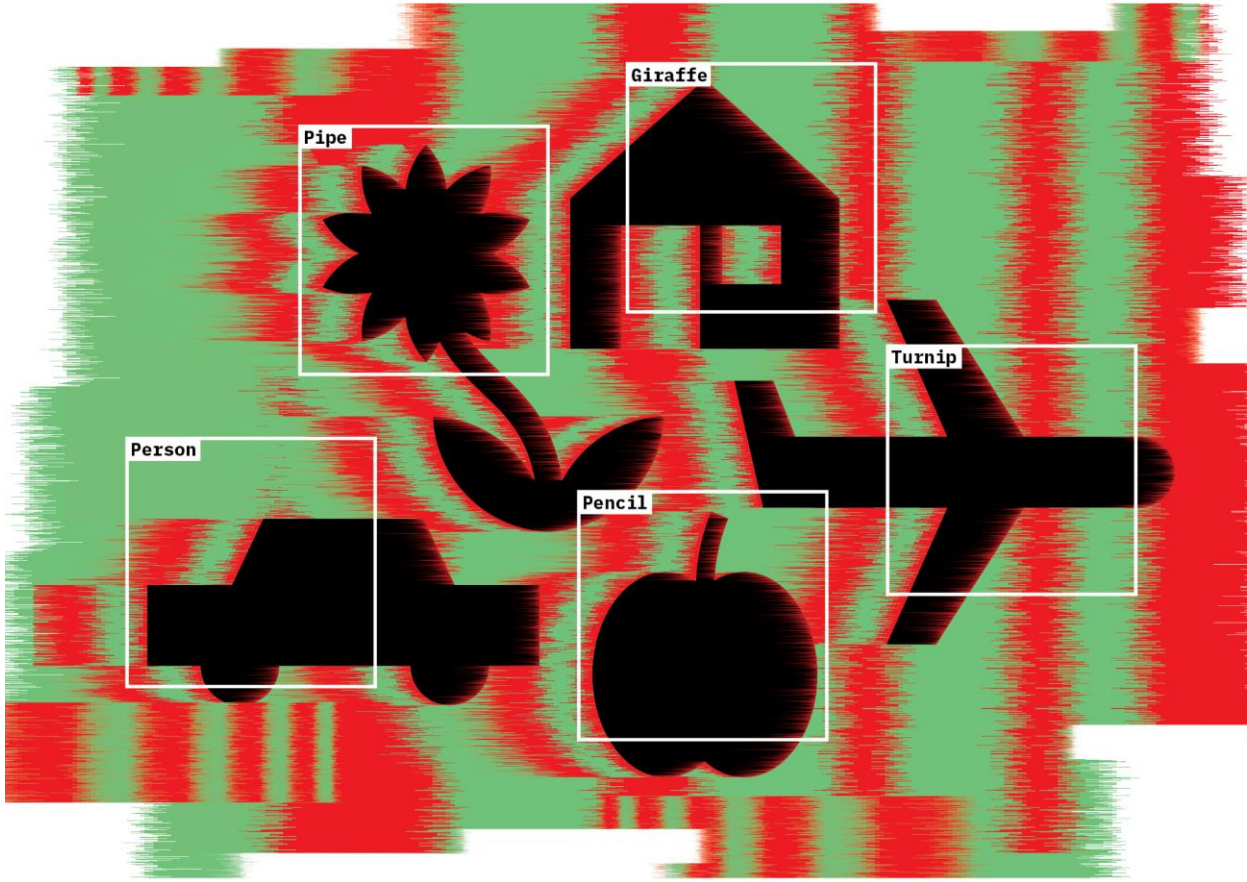



ILLUSTRATION BY EDGAR BAK

## DEEP TROUBLE FOR DEEP LEARNING

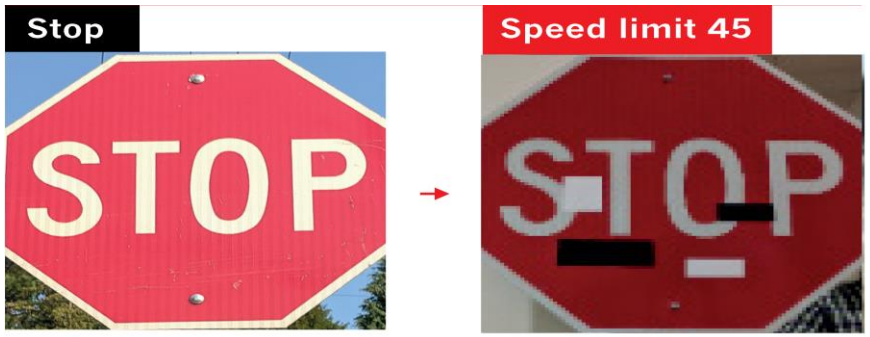
BY DOUGLAS HEAVEN

### ARTIFICIAL-INTELLIGENCE RESEARCHERS ARE TRYING TO FIX THE FLAWS OF NEURAL NETWORKS.

A self-driving car approaches a stop sign, but instead of slowing down, it accelerates into the busy intersection. An accident report later reveals that four small rectangles had been stuck to the face of the sign. These fooled the car's onboard artificial intelligence (AI) into misreading the word 'stop' as 'speed limit 45'.

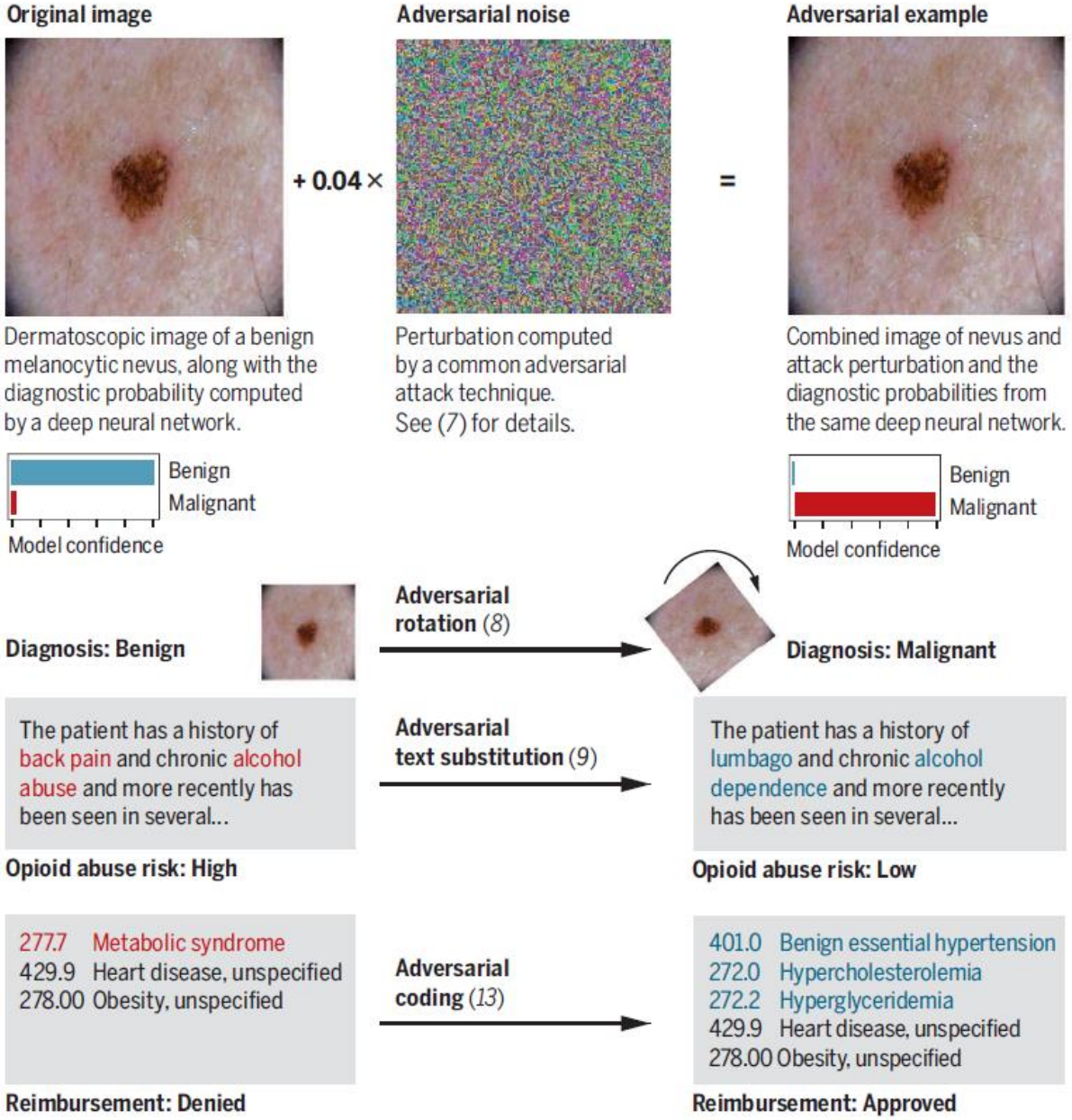
Such an event hasn't actually happened, but the potential for sabotaging AI is very real. Researchers have already demonstrated how to fool an AI system into misreading a stop sign, by carefully positioning stickers on it[1]. They have deceived facial-recognition systems by sticking a printed pattern on glasses or hats. And they have tricked speech-recognition systems into hearing phantom phrases by inserting patterns of white noise in the audio.

# Paper published in Science 2019


## The anatomy of an adversarial attack

Demonstration of how adversarial attacks against various medical AI systems might be executed without requiring any overtly fraudulent misrepresentation of the data.



# Outline

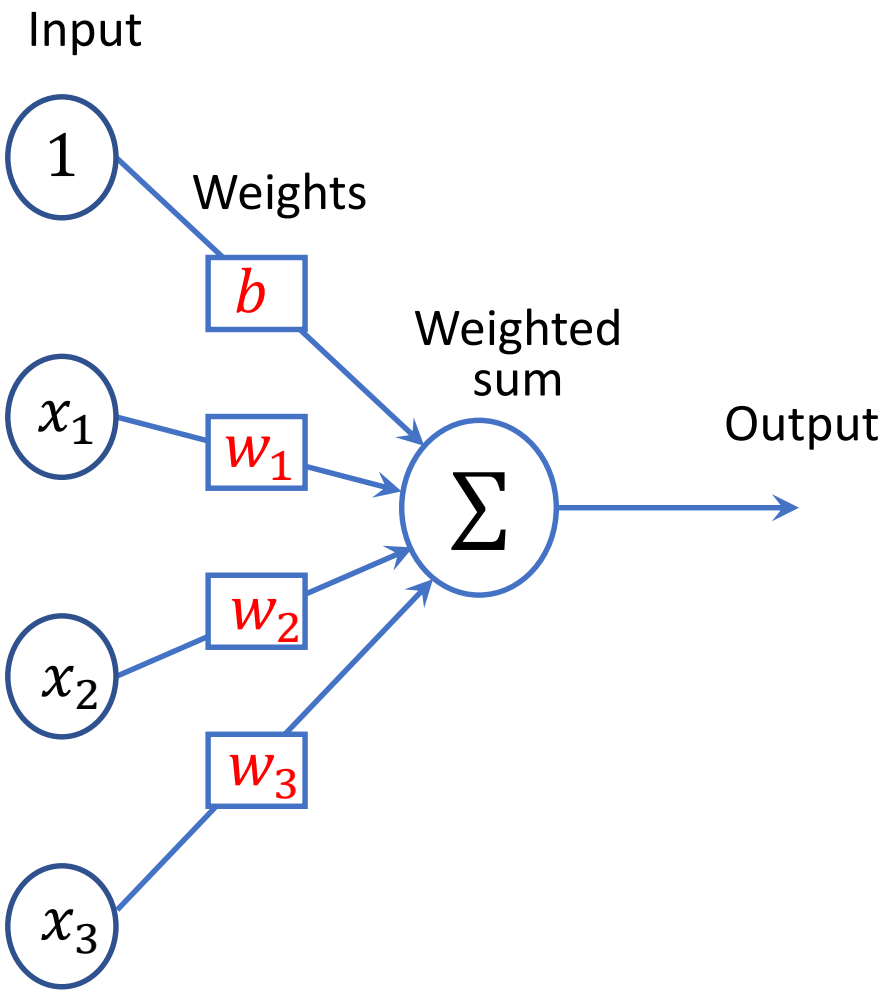
- Course Information
- Introduction to Deep Learning
- Some Deep Learning Basics

# Deep Learning Basics

UNIVERSITY OF OULU

- Linear regression
- Model complexity
- Regularization
- Gradient descent
- Loss Functions
- Stochastic gradient descent
- Logistic regression

# Linear Regression



# Linear Regression

- Linear hypothesis
- Many real processes can be approximated with linear models.
- Linear regression often appears as a module of larger systems.
- Linear problems can be solved analytically.
- Linear prediction provides an introduction to many of the core concepts of machine learning and deep learning.
- Let's focus on training of a *parametric model* in a supervised scenario.

# Linear Regression: Formulation

- Given: a training dataset of *N* instances of (input, output) pairs

  $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ , where $\boldsymbol{x}^{(i)} \in \mathbb{R}^{n\times 1}$ and $y^{(i)} \in \mathbb{R}$

- Notation:

*N* = number of features

$\boldsymbol{x}^{(i)}$ = input (features) of $i^{th}$ training example

$x_j^{(i)}$ = value of feature $j$ in $i^{th}$ training example

$y^{(i)}$ = output of $i^{th}$ training example

$$\boldsymbol{x}^{(i)} = \begin{bmatrix} x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

For simplicity, the superscript is omitted when unnecessary.

# Linear Regression: Formulation

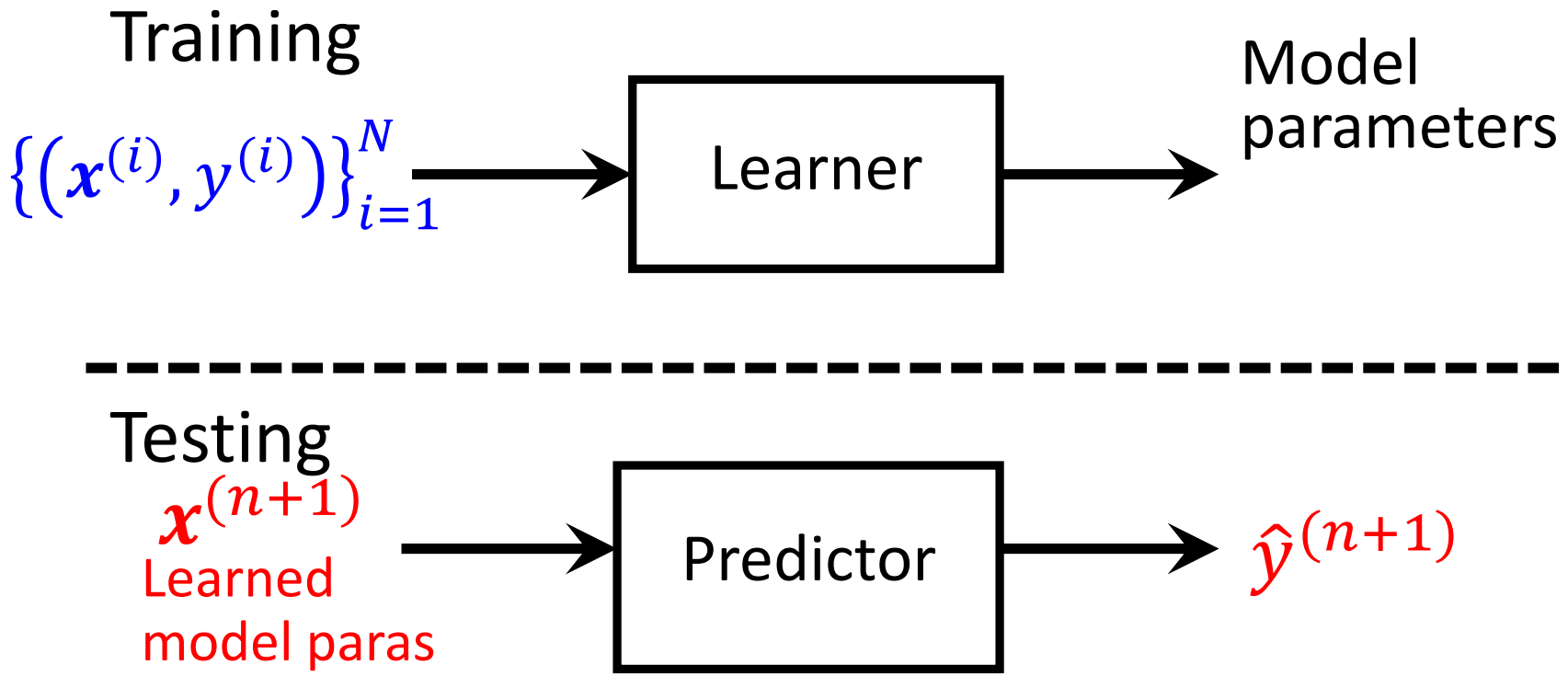
- Given: a training dataset of *N* instances of (input, output) pairs $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ , where $\boldsymbol{x}^{(i)} \in \mathbb{R}^{n\times 1}$ and $y^{(i)} \in \mathbb{R}$
  - A typical dataset with N=4 instances and n=4 variables (features)

| | Size ($m^2$) | Number of bedrooms | Number of floors | Age of home (years) | Price (*1000$) $y^{(i)}$ |
|---|---|---|---|---|---|
| $\boldsymbol{x}^{(1)}$ | 180 | 4 | 2 | 45 | 460 |
| | 140 | 3 | 1 | 40 | 232 |
| | 120 | 3 | 1 | 30 | 315 |
| | 90 | 2 | 1 | 36 | 178 |
| | Feature 1 $x_1$ | Feature 2 $x_2$ | Feature 3 $x_3$ | Feature 4 $x_4$ | |

Example from Andrew Ng

# Linear Regression: Formulation

- Goal: Learn a model of how the inputs affect the outputs, and use the learned model to predict the output of a new value of the input.



# Linear Hypothesis

Hypothesis:

$$y = b + x_1 w_1 + x_2 w_2 + \cdots x_n w_n$$

Parameters: $b, w_1, \cdots, w_n$ (can be any value)

$b$ is known as the bias term. For convenience of notation, it can be represented in vector form:

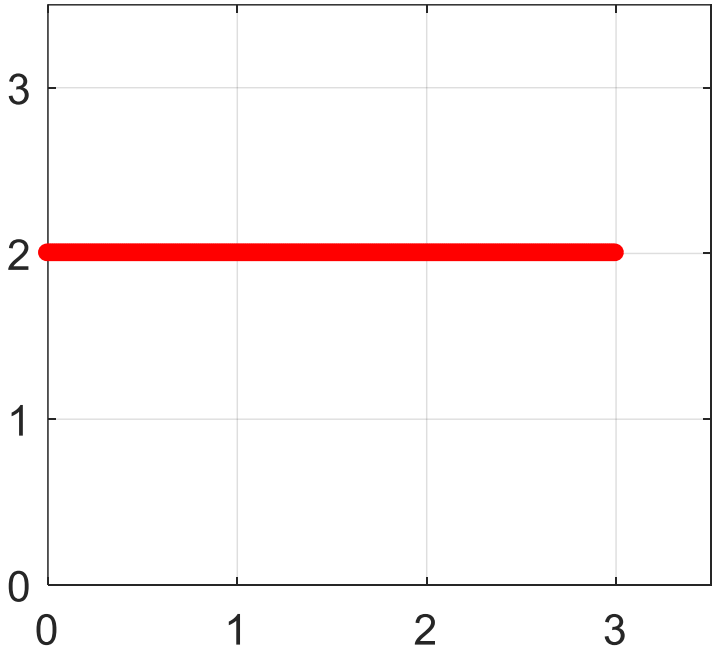
$$\hat{y}(\boldsymbol{x}) = b + \boldsymbol{w}^T \boldsymbol{x}$$ inner product

where $\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n$, $\boldsymbol{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} \in \mathbb{R}^n$

# Linear Hypothesis

$$f_{w,b}(x) = b + wx$$

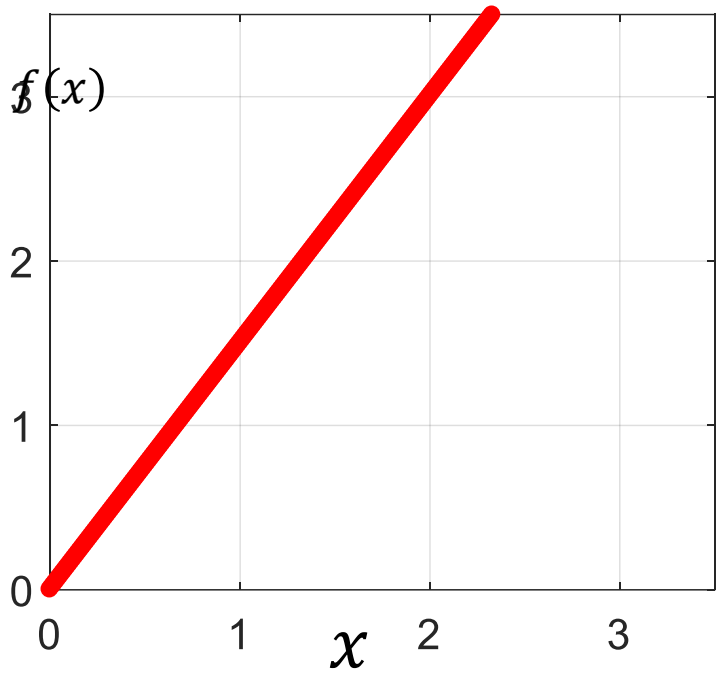
$f(x) = 2 + 0 \cdot x$




$b = 2$

$w = 0$

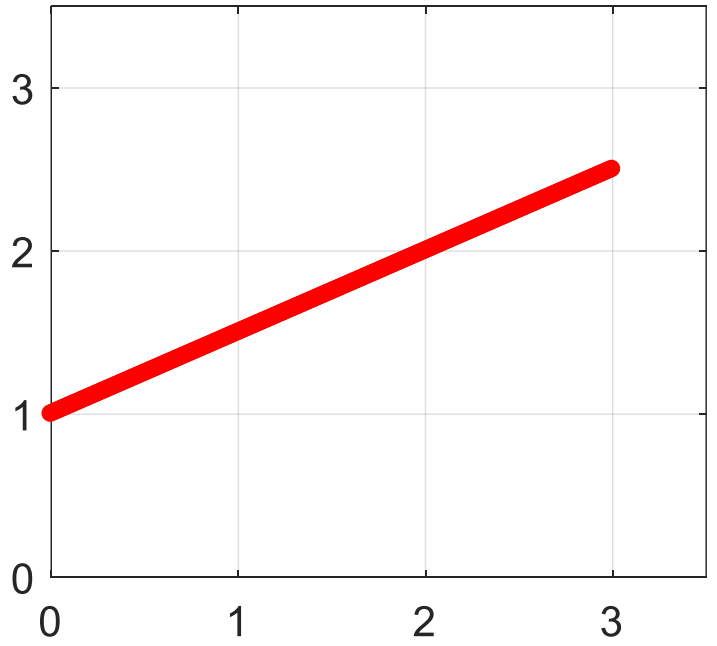
$f(x) = 0 + 1.5 \cdot x$




$b = 0$

$w = 1.5$

$f(x) = 1 + 0.5 \cdot x$




$b = 1$

$w = 0.5$

# Linear Regression: Formulation

- Given: a training dataset of *N* instances of (input, output) pairs

  $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ , where $\boldsymbol{x}^{(i)} \in \mathbb{R}^{n\times 1}$ and $y^{(i)} \in \mathbb{R}$

- Hypothesis: $\hat{y} = f(\boldsymbol{x}) = b + \boldsymbol{w}^T \boldsymbol{x}$

- Question:

  How to choose parameters $\boldsymbol{w}$ and $b$?

  In other words, how to select a function $f(\boldsymbol{x})$ parametered by $\boldsymbol{w}$ and $b$?

# Step 1: Model

$y = b + \boldsymbol{w}^T \boldsymbol{x}$

A set of functions

Model

$f_1, f_2 \cdots$

## One variable example

$\boldsymbol{w}$ and $b$ are parameters

$f_1$: $y = 10.0 + 9.0\ x$

$f_2$: $y = 9.8 + 9.2\ x$

$f_3$: $y = -0.8 - 1.2\ x$

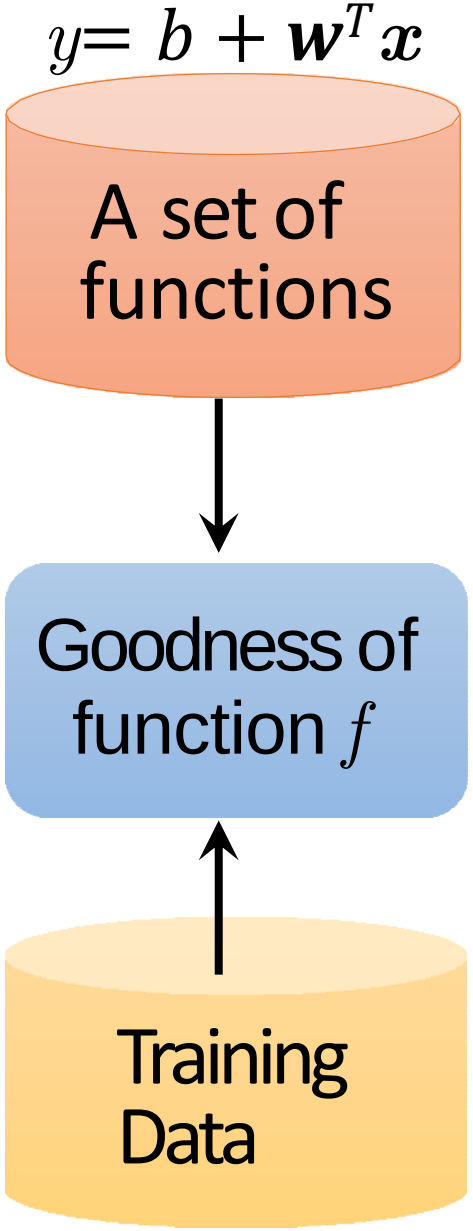
..... infinite

**Linear Model:**

$$\hat{y} = f(\boldsymbol{x}) = b + \boldsymbol{w}^T \boldsymbol{x}$$

Note: for fixed $\boldsymbol{w}$ and $b$, this is a function of $\boldsymbol{x}$

# Step 2: Cost/Loss Function




Model

$f_1, f_2 \cdots$

- **Cost function $L$**
  - Input: a function
  - Output: how good it is

$$L(f) = L(\boldsymbol{w}, b) = \frac{1}{2N} \sum_{i=1}^{N} \left( y^{(i)} - \hat{y}^{(i)} \right)^2$$

$$= \frac{1}{2N} \sum_{i=1}^{N} \left( y^{(i)} - \left( b + \boldsymbol{w}^T \boldsymbol{x}^{(i)} \right) \right)^2$$

Estimated error

Sum over training data

# Now we have…

**Hypothesis:** $\hat{y} = f(\boldsymbol{x}) = b + \boldsymbol{w}^T \boldsymbol{x}$

**Parameters:** $w,\ b$

**Cost function:**

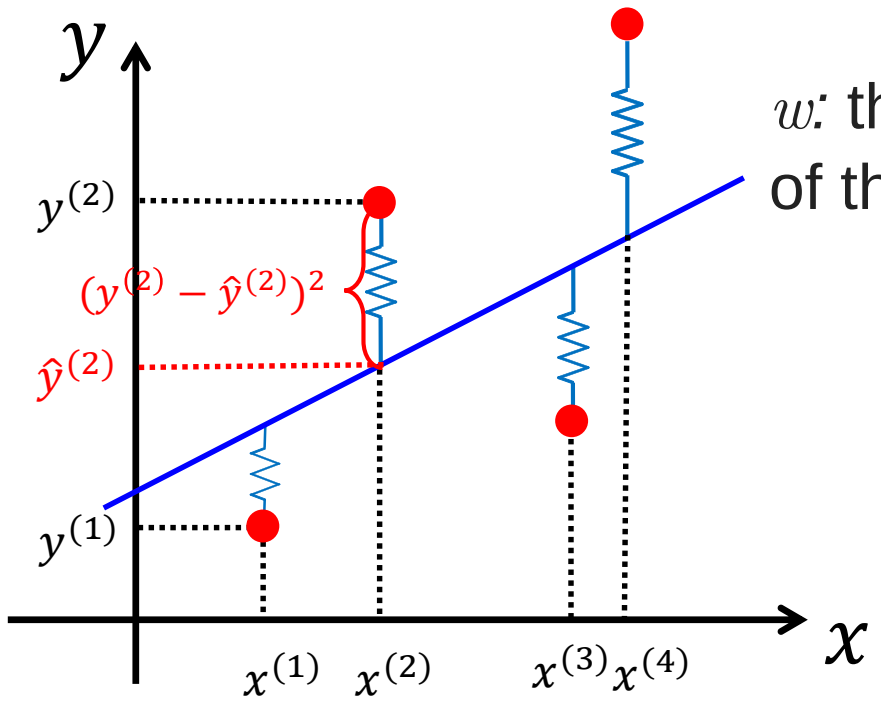
$$L(\boldsymbol{w}, b) = \frac{1}{2N} \sum_{i=1}^{N} \left( y^{(i)} - \left( b + \boldsymbol{w}^T \boldsymbol{x}^{(i)} \right) \right)^2$$

**Goal:** $\underset{\boldsymbol{w}, b}{\text{minimize}}\ L(\boldsymbol{w}, b)$

# Cost Function: Intuitive

$$\hat{y}(x) = b + wx$$

$$L(w,b) = \sum_{i=1}^{N} \left(y^{(i)} - \hat{y}^{(i)}\right)^2 = \sum_{i=1}^{N} \left(y^{(i)} - (b + wx^{(i)})\right)^2$$

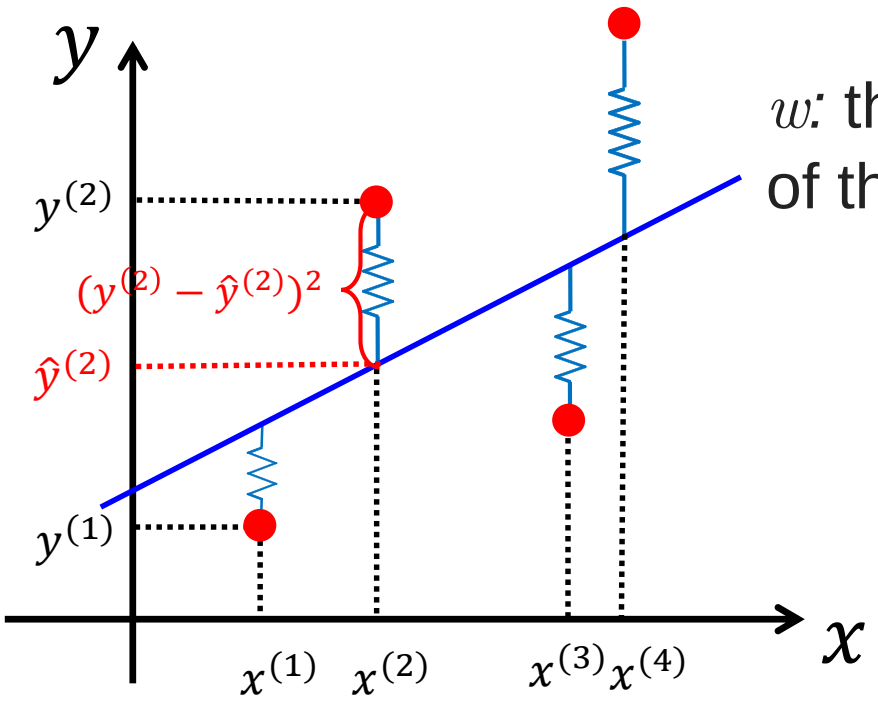



$w$: the slope of the fit line

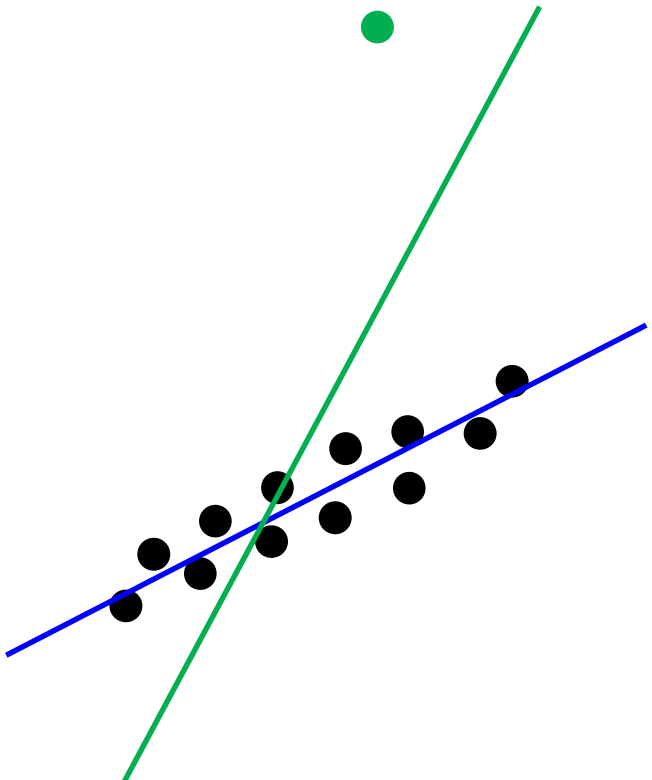
# Cost Function: Intuitive

$$\hat{y}(x) = b + wx$$

$$L(w,b) = \sum_{i=1}^{N} \left(y^{(i)} - \hat{y}^{(i)}\right)^2 = \sum_{i=1}^{N} \left(y^{(i)} - (b + wx^{(i)})\right)^2$$




$w$: the slope of the fit line



# Loss Function: Intuitive

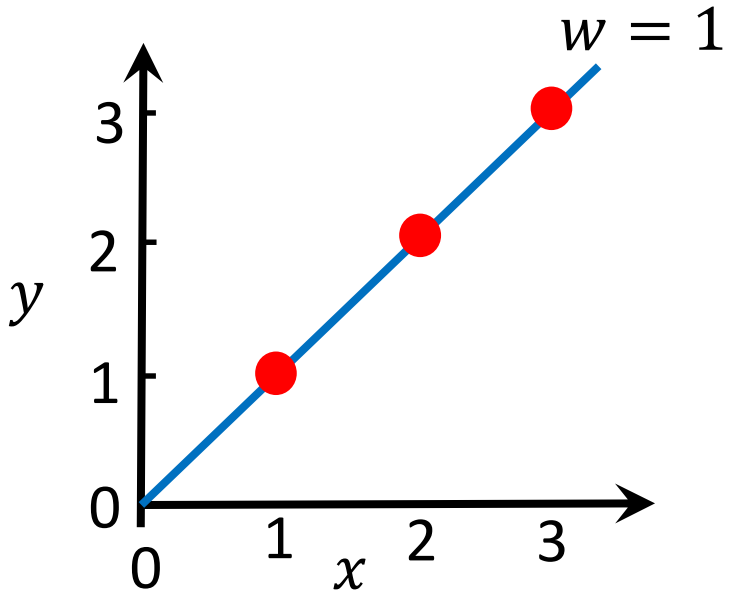
$f_w(x) = wx$

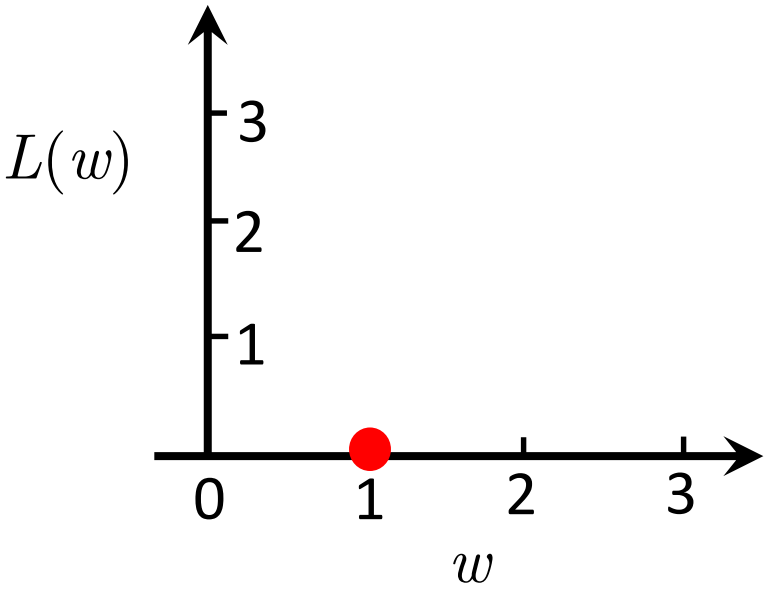
(For fixed $w$, this is a function of $x$)

$L(w)$

(Function of the parameter $w$)

| $(x^{(i)}, y^{(i)})$ |
|---|
| $(1,1)$ |
| $(2,2)$ |
| $(3,3)$ |




$$L(w) = \frac{1}{2N}\sum_{i=1}^{N}\left(f_w\left(x^{(i)}\right) - y^{(i)}\right)^2$$

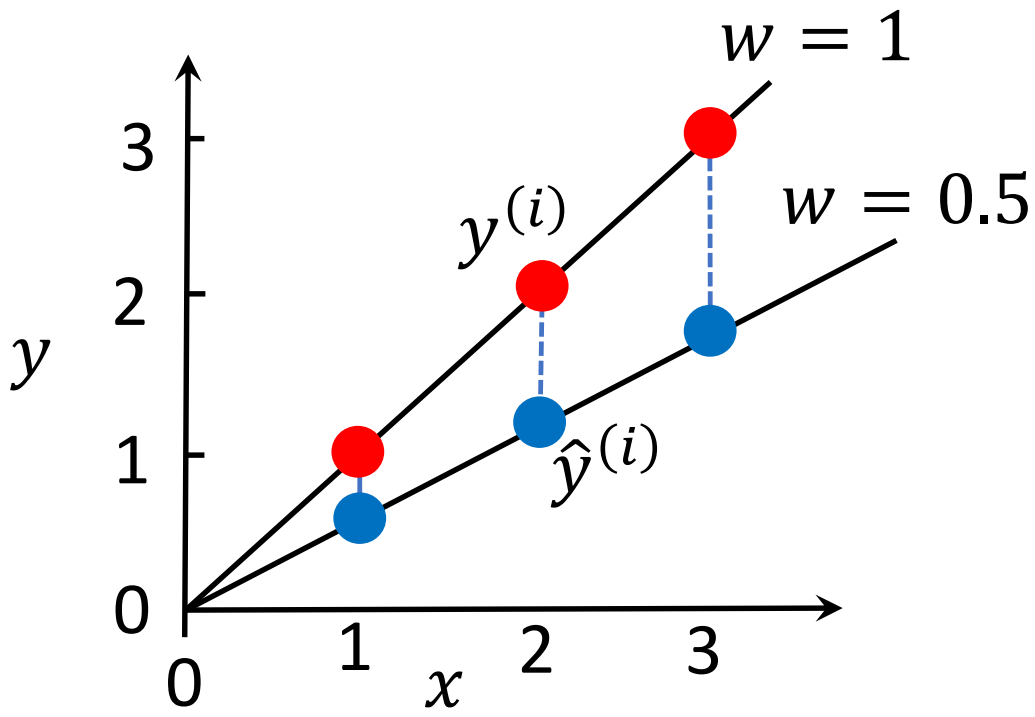
$L(0.5) = ?$

$$= \frac{1}{2N}\sum_{i=1}^{N}\left(wx^{(i)} - y^{(i)}\right)^2 = \frac{1}{2m}(0^2 + 0^2 + 0^2) = 0$$

# Cost Function: Intuitive

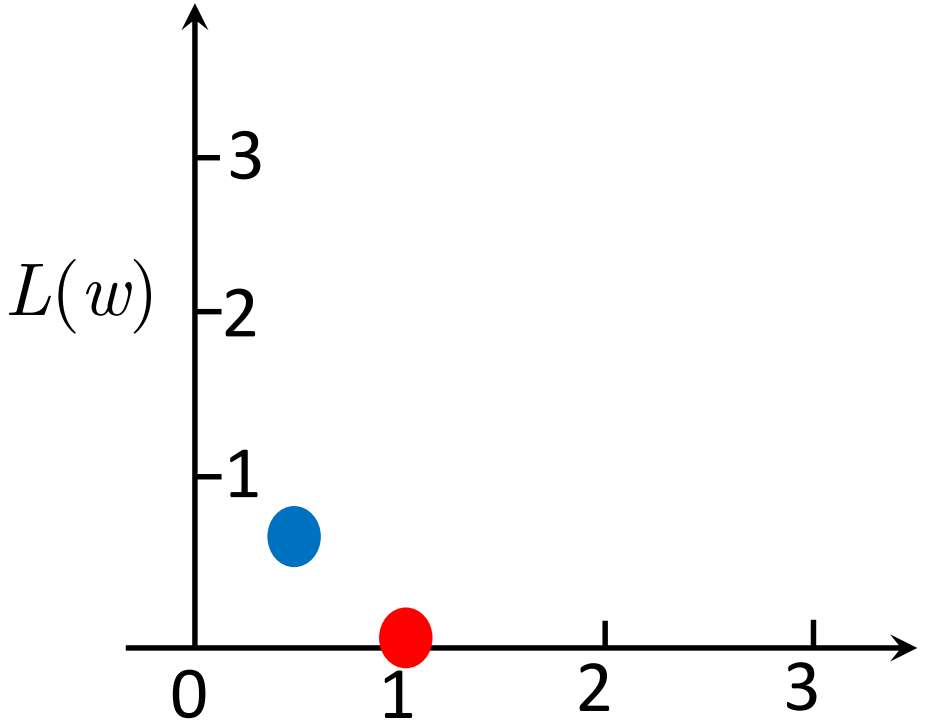
$f_w(x)$

(For fixed $w$, this is a function of $x$)




$L(w)$

(Function of the parameter $w$)



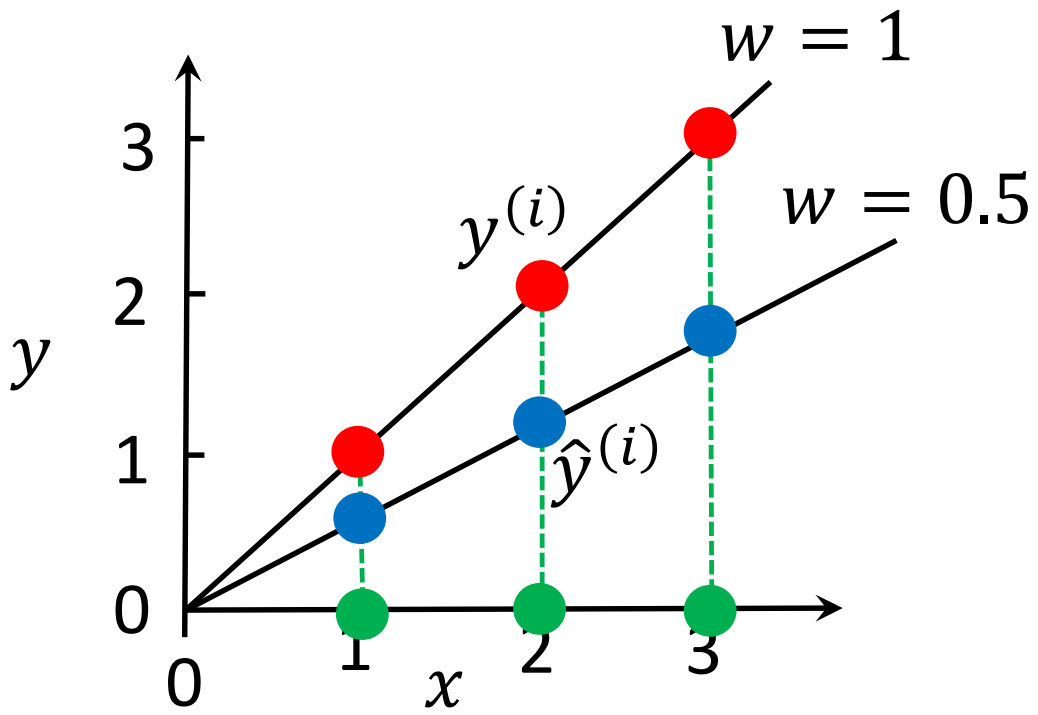

$$L(0.5) = \frac{1}{2N}\left((0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right)$$

$$= \frac{1}{2 \times 3}(3.5) \approx 0.58$$

# Cost Function: Intuitive

$f_w(x)$

(For fixed $w$, this is a function of $x$)

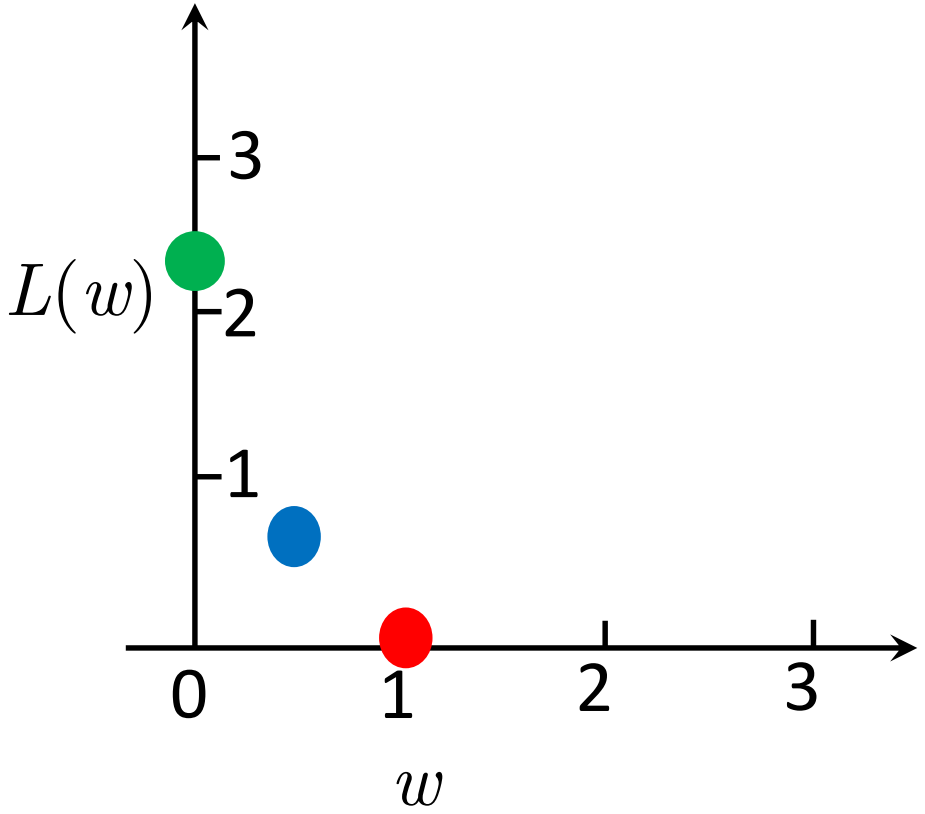



$$L(0) = \frac{1}{2N}\left((1)^2 + (2)^2 + (3)^2\right)$$

$$= \frac{1}{2 \times 3}(14) \approx 2.3$$

$L(w)$

(Function of the parameter $w$)

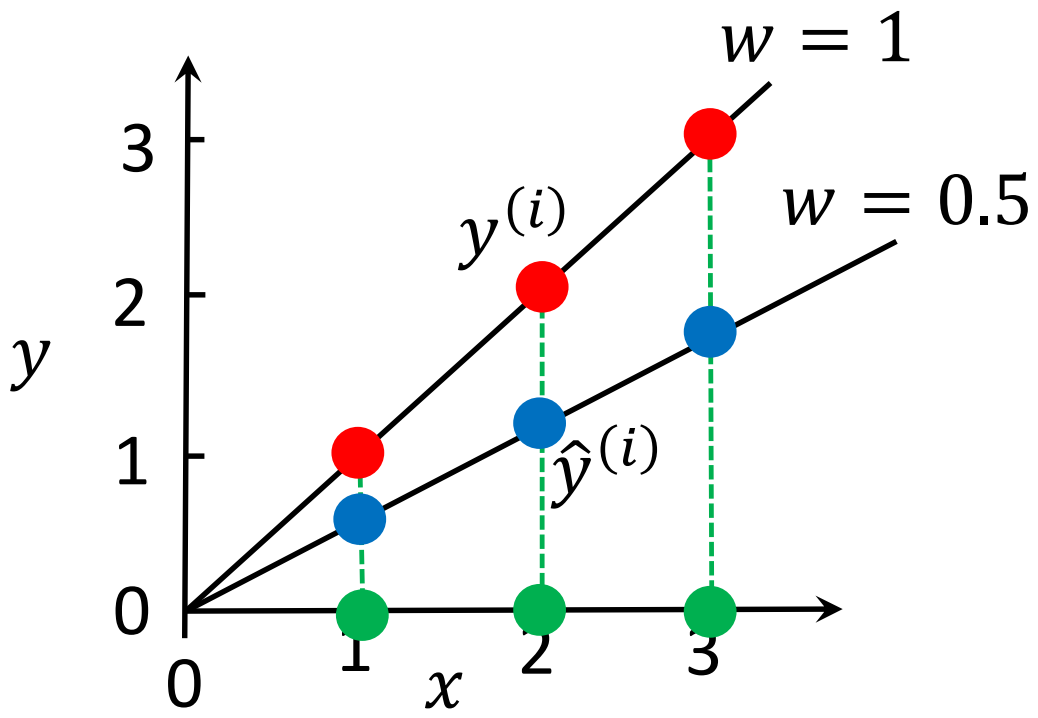



$$\min_w L(w)$$

# Cost Function: Intuitive

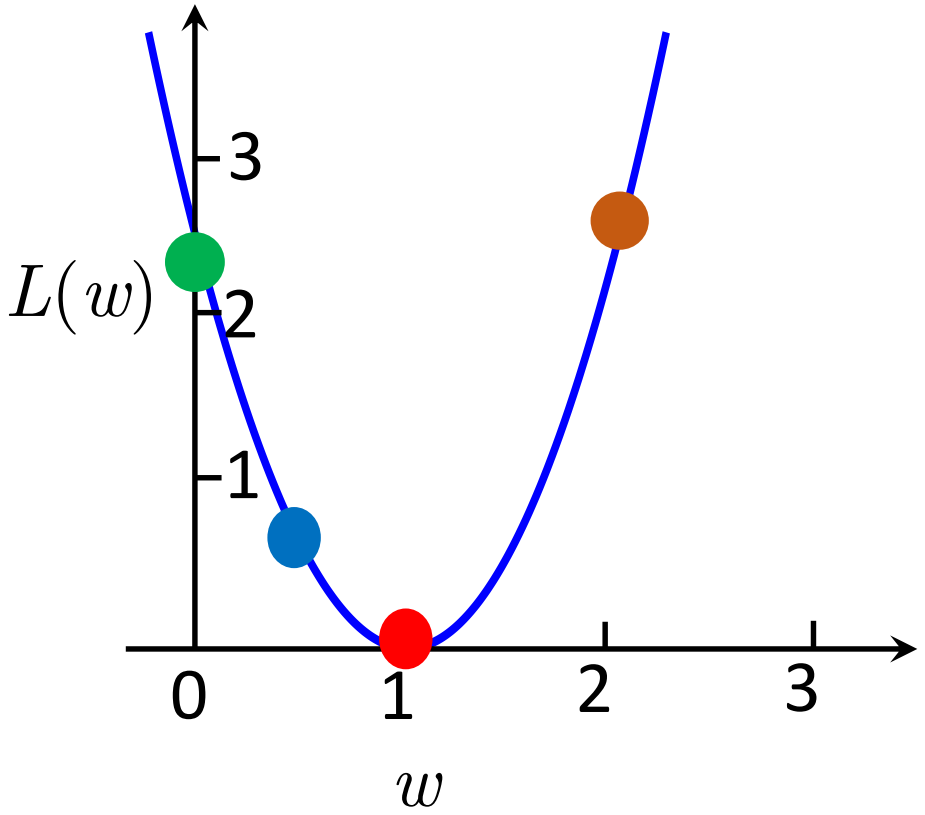
$f_w(x)$

(For fixed $w$, this is a function of $x$)




$L(w)$

(Function of the parameter $w$)




$$\min_w L(w)$$

# Step 3: Minimize Cost (Find Best Function)

$y = b + \boldsymbol{w}^T \boldsymbol{x}$

A set of functions

Model

$f_1,\ f_2 \ldots$

$$L(\boldsymbol{w}, b) = \frac{1}{2N} \sum_{i=1}^{N} \left(y - (b + \boldsymbol{w}^T \boldsymbol{x}_i)\right)^2$$

Goodness of function $f$

Pick the "Best" Function

How?
Gradient Descent

$$\boldsymbol{w}^*, b^* = \underset{\boldsymbol{w}, b}{\operatorname{argmin}} L(\boldsymbol{w}, b)$$

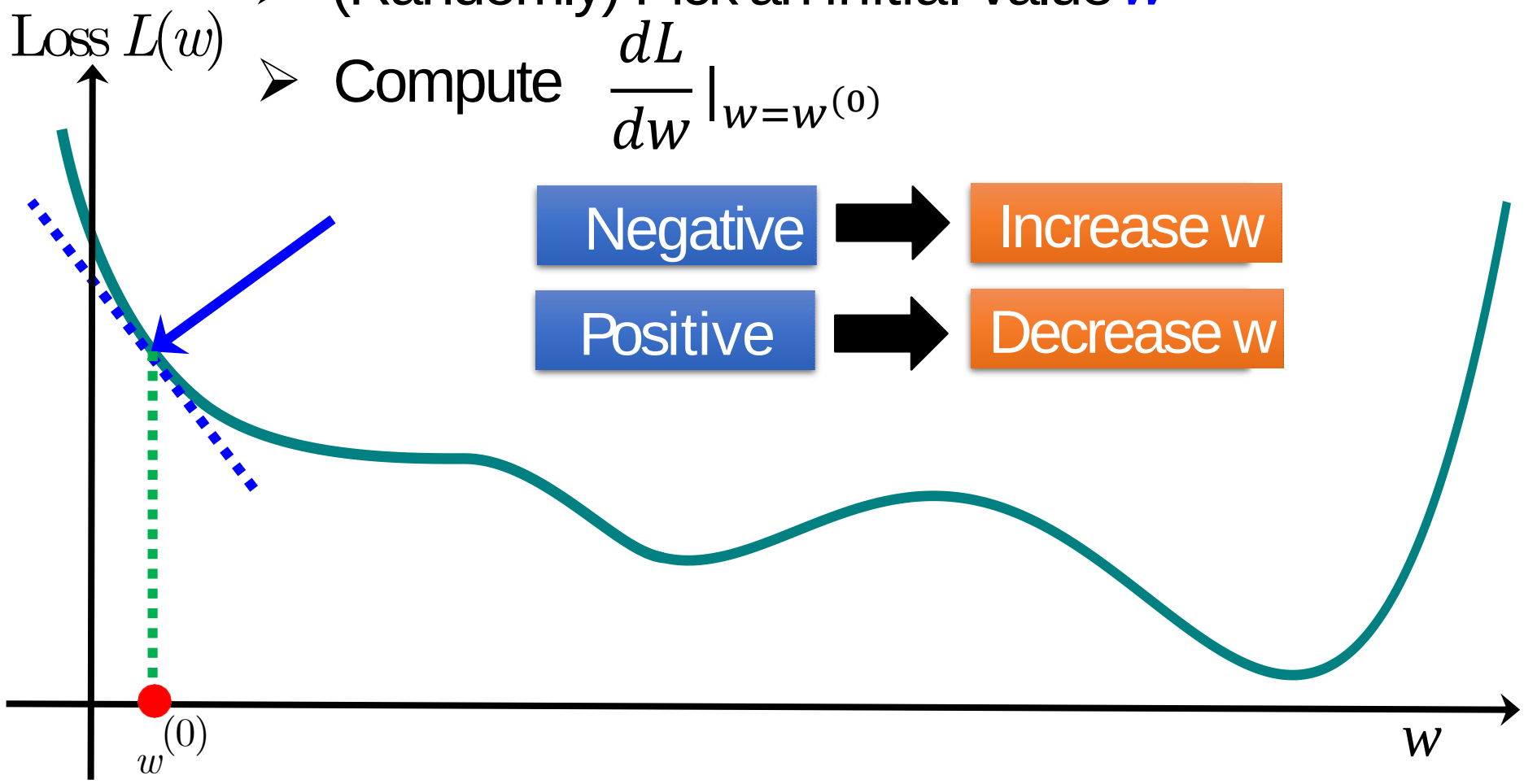
$$= \underset{\boldsymbol{w}, b}{\operatorname{argmin}} \frac{1}{2N} \sum_{i=1}^{N} \left(y - (b + \boldsymbol{w}^T \boldsymbol{x}_i)\right)^2$$

Training Data

# Gradient Descent

$$w^* = arg\min_w L(w)$$

- Consider loss function $L(w)$ with one parameter $w$:
  - (Randomly) Pick an initial value $w^{(0)}$
  - Compute $\left.\frac{dL}{dw}\right|_{w=w^{(0)}}$



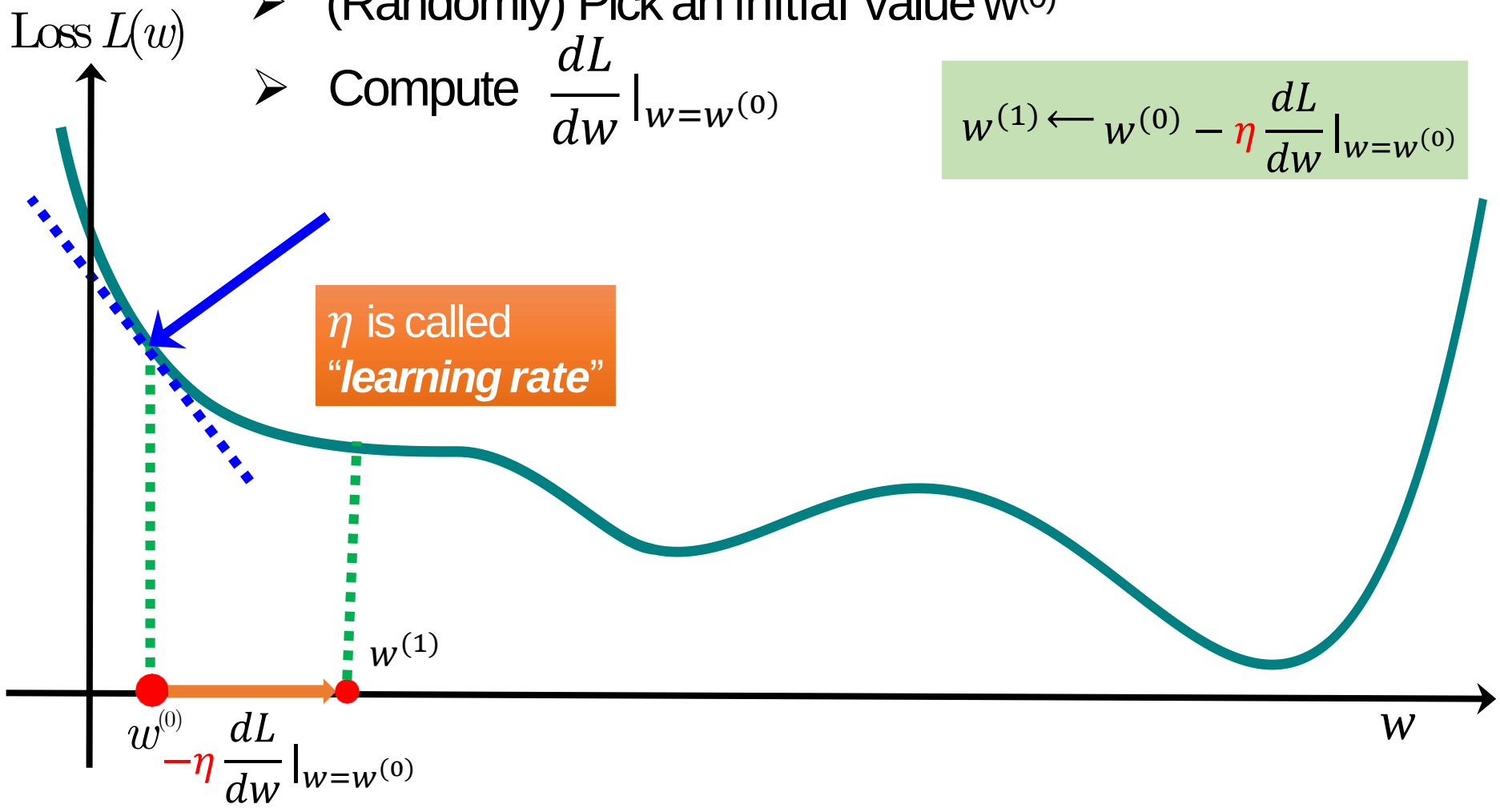

Quite a few slides adapted from Hungyi Lee

# Gradient Descent

$$w^* = arg \min_w L(w)$$

- Consider loss function $J(w)$ with one parameter w:
  - (Randomly) Pick an initial value $w^{(0)}$
  - Compute $\frac{dL}{dw}|_{w=w^{(0)}}$

$$w^{(1)} \leftarrow w^{(0)} - \eta \frac{dL}{dw}|_{w=w^{(0)}}$$



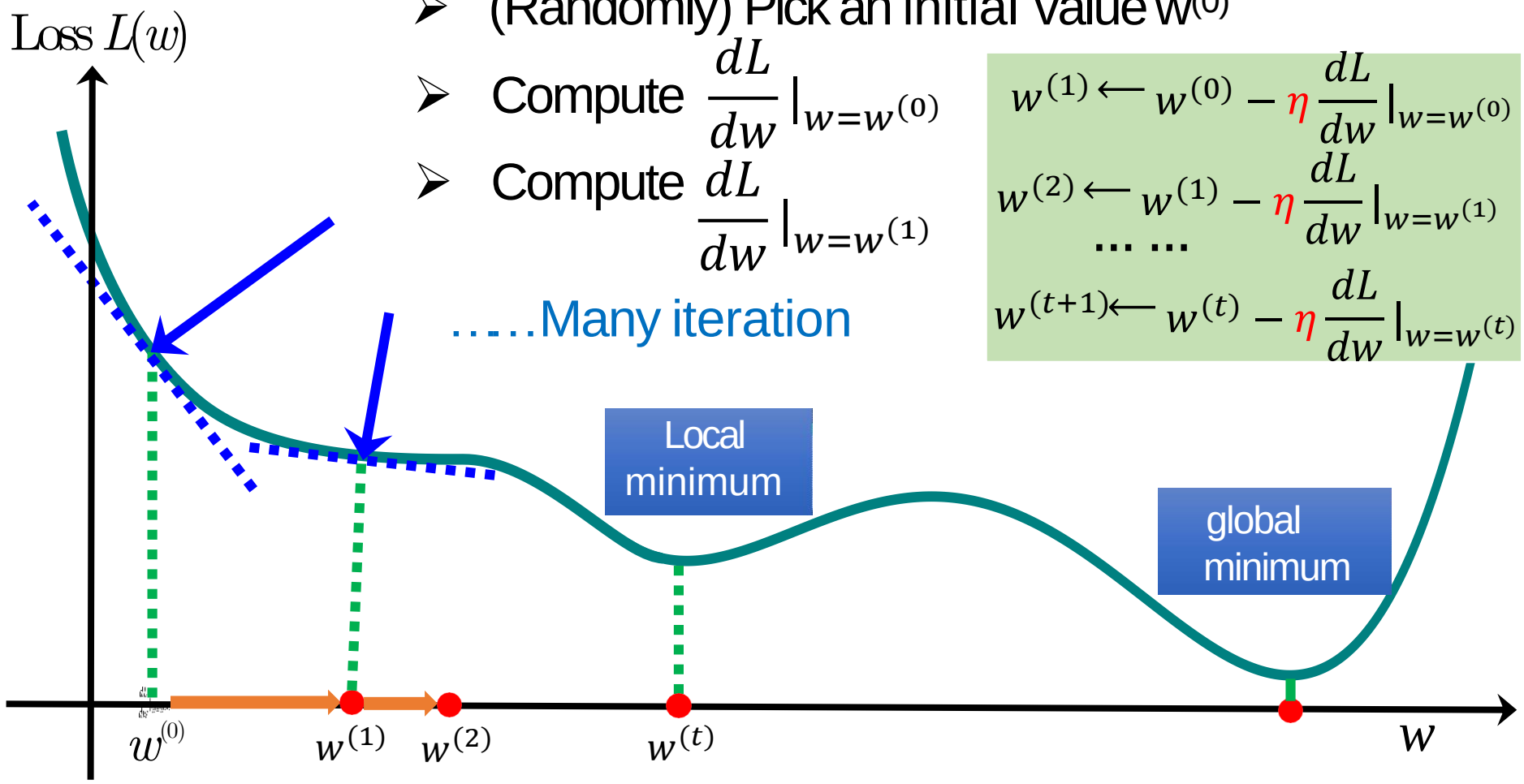
# Gradient Descent

$$w^* = arg\min_w L(w)$$

- Consider loss function $J(w)$ with one parameter w:
  - (Randomly) Pick an initial value $w^{(0)}$
  - Compute $\frac{dL}{dw}|_{w=w^{(0)}}$
  - Compute $\frac{dL}{dw}|_{w=w^{(1)}}$

……Many iteration

$$w^{(1)} \leftarrow w^{(0)} - \eta \frac{dL}{dw}|_{w=w^{(0)}}$$
$$w^{(2)} \leftarrow w^{(1)} - \eta \frac{dL}{dw}|_{w=w^{(1)}}$$
$$\dots\dots$$
$$w^{(t+1)} \leftarrow w^{(t)} - \eta \frac{dL}{dw}|_{w=w^{(t)}}$$



# Gradient Descent

- How about two parameters? $w^*, b^* = arg \min_{w,b} L(w,b)$
  - (Randomly) Pick an initial value $w^{(0)}$ ,$b^{(0)}$
  - Compute partial derivative

$$\frac{\partial L}{\partial w}\Big|_{w=w^{(0)},b=b^{(0)}} \quad , \quad \frac{\partial L}{\partial b}\Big|_{w=w^{(0)},b=b^{(0)}}$$

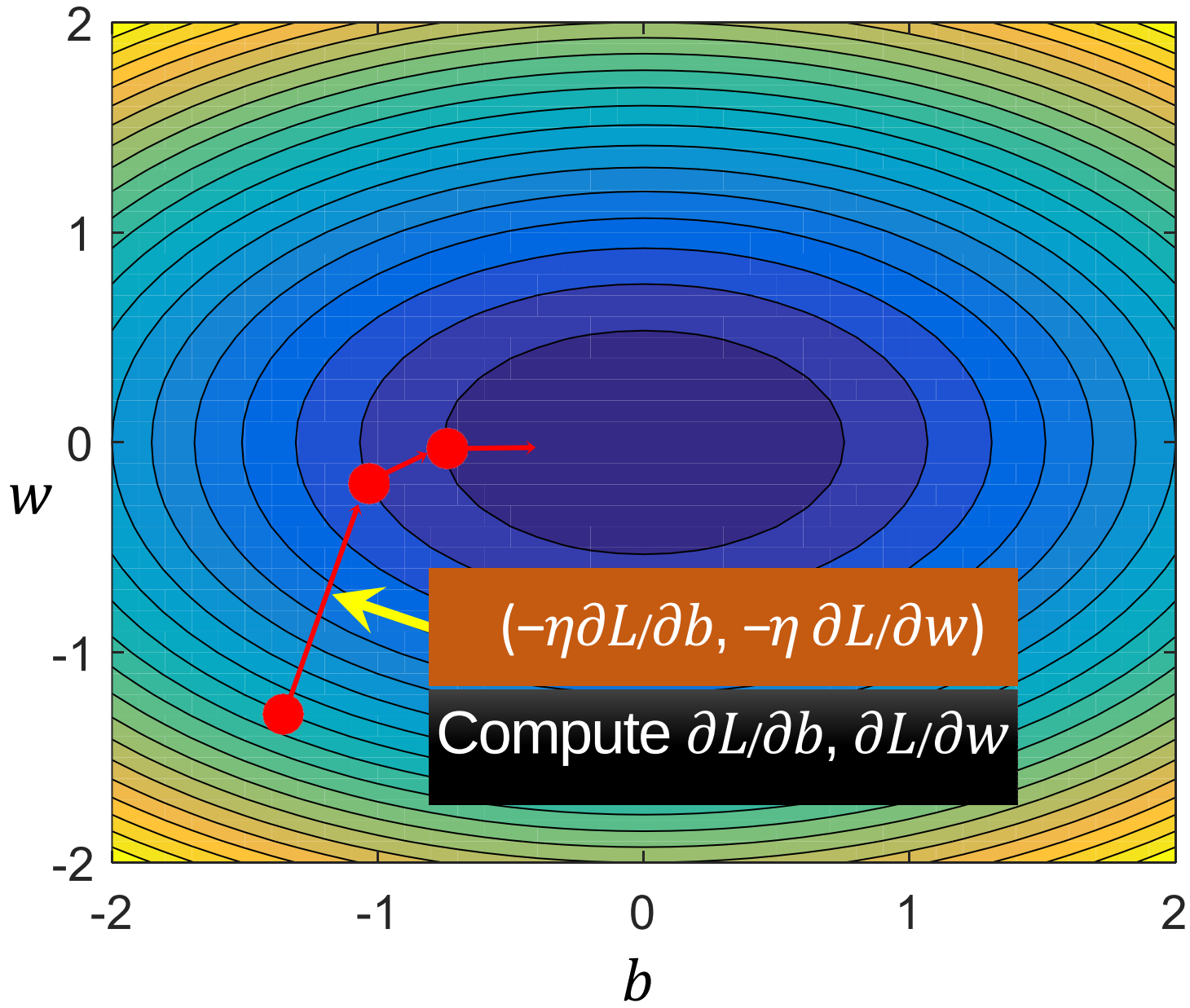
$$\nabla L = \begin{bmatrix} \frac{\partial L}{\partial w} \\ \frac{\partial L}{\partial b} \end{bmatrix} \text{gradient}$$

$$w^{(1)} \leftarrow w^{(0)} - \eta \frac{\partial L}{\partial w}\Big|_{w=w^{(0)},b=b^{(0)}} \quad , \quad b^{1} \leftarrow b^{(0)} - \eta \frac{\partial L}{\partial b}\Big|_{w=w^{(0)},b=b^{(0)}}$$

  - Compute $\frac{\partial L}{\partial w}\Big|_{w=w^{(1)},b=b^{(1)}} \quad , \quad \frac{\partial L}{\partial b}\Big|_{w=w^{(1)},b=b^{(1)}}$

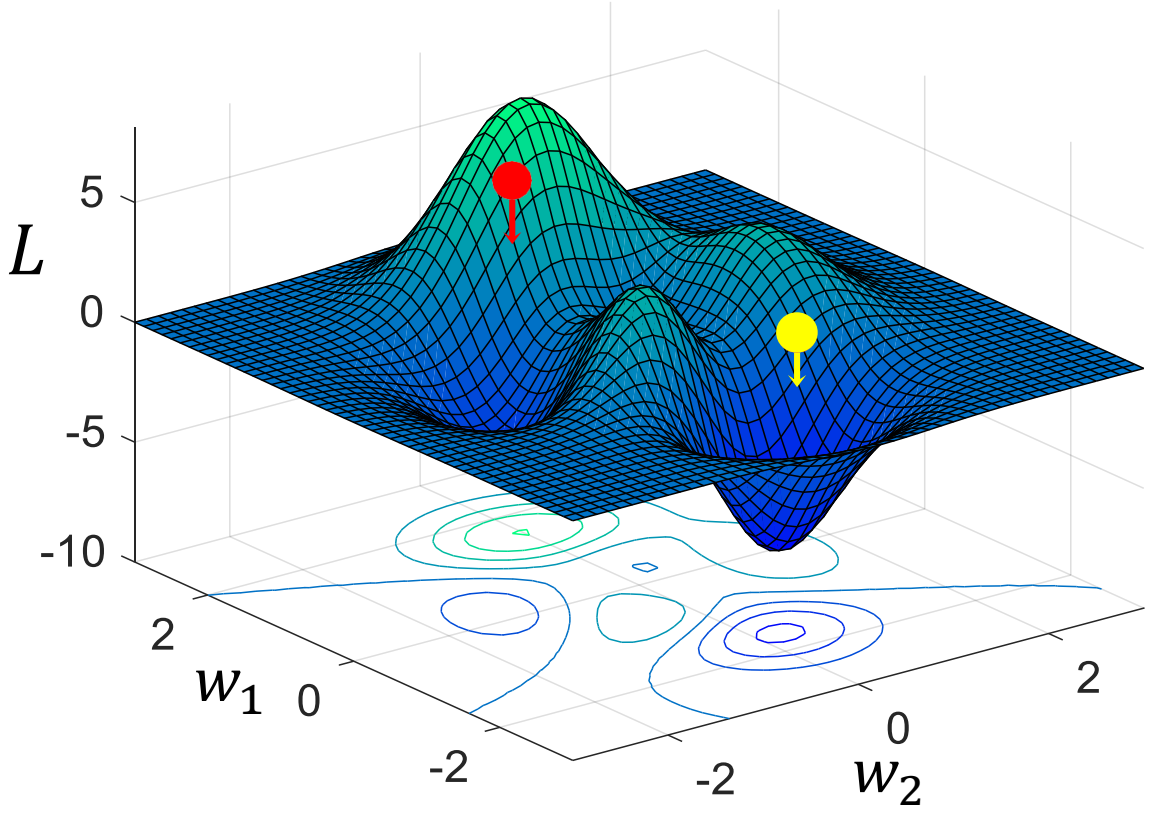
$$w^{(2)} \leftarrow w^{(1)} - \eta \frac{\partial L}{\partial w}\Big|_{w=w^{(1)},b=b^{(1)}} \quad , \quad b^{(2)} \leftarrow b^{(1)} - \eta \frac{\partial L}{\partial b}\Big|_{w=w^{(1)},b=b^{(1)}}$$

# Gradient Descent



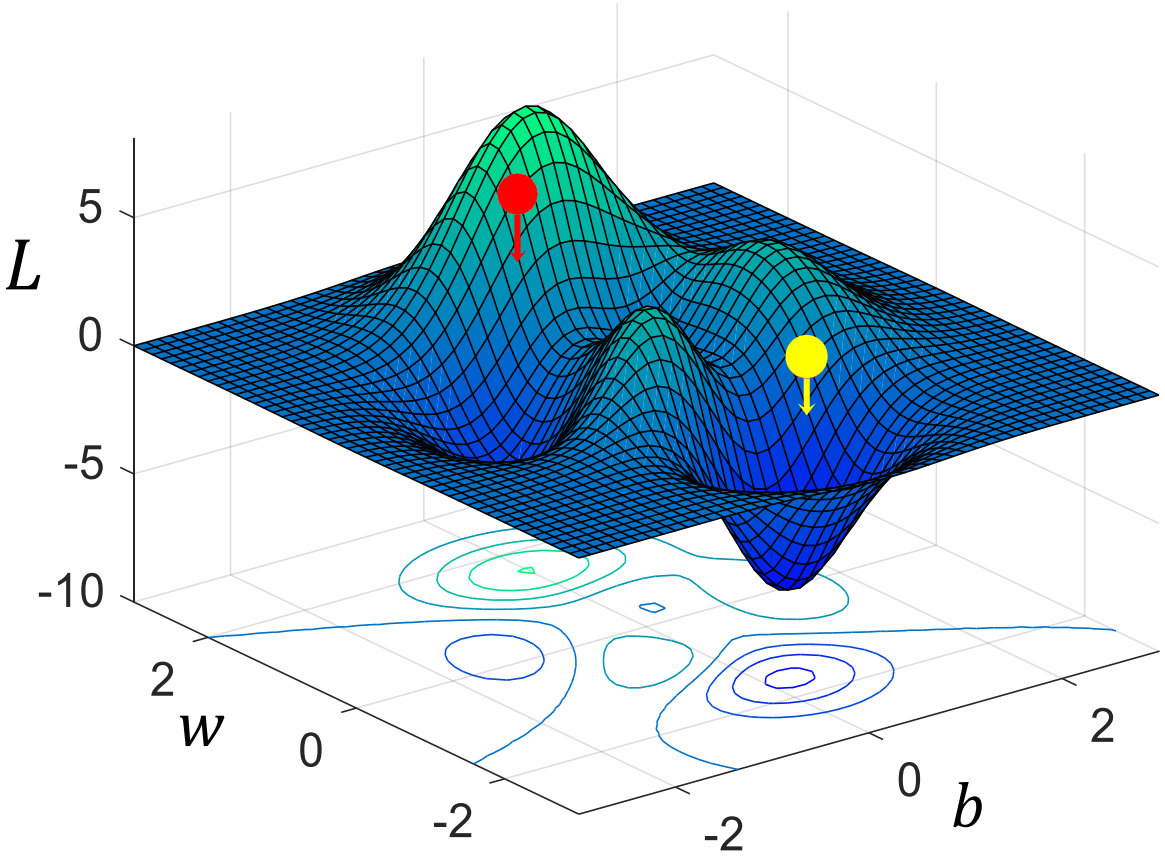
# Step 3: Gradient Descent for Linear Regression

- Worry?

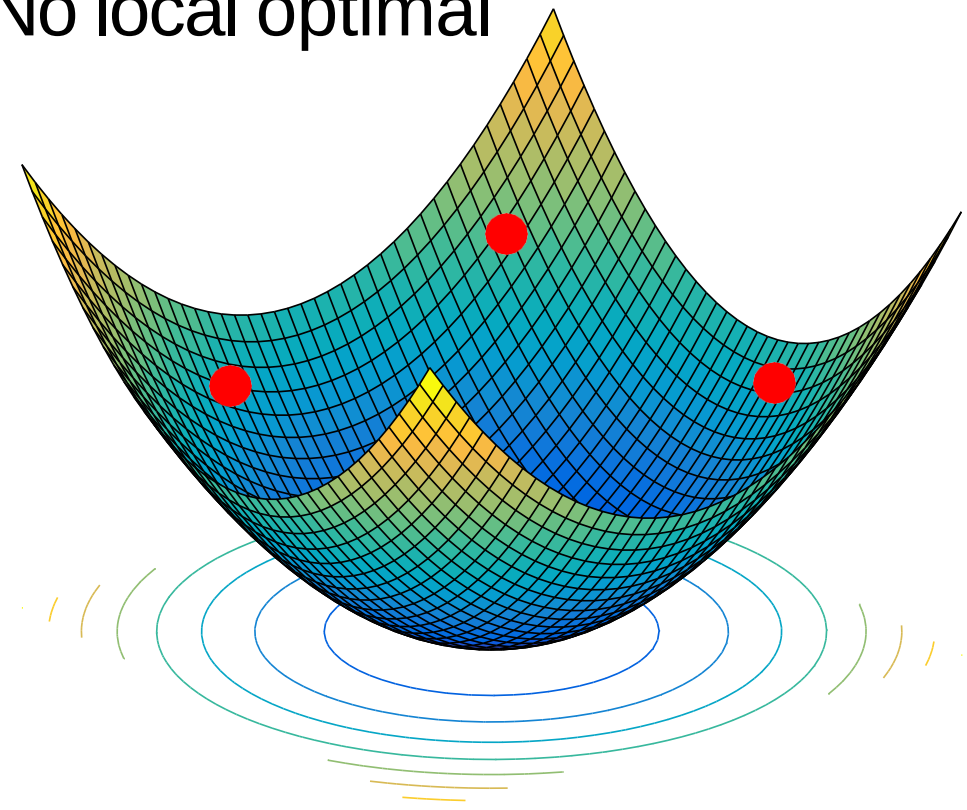


# Step 3: Gradient Descent for Linear Regression

- Worry?




Don't worry. In linear regression, the loss function L is convex.
No local optimal



# Step 3: Gradient Descent for Linear Regression

- Formulation of $\frac{\partial L}{\partial w}$ and $\frac{\partial L}{\partial b}$

$$L(w,b) = \sum_{i=1}^{N} \left( y^{(i)} - \left( b + w \cdot x^{(i)} \right) \right)^2$$

$$\frac{\partial L}{\partial w} = \sum_{i=1}^{N} 2 \left( y^{(i)} - \left( b + w \cdot x^{(i)} \right) \right) \left( -x^{(i)} \right)$$

$$\frac{\partial L}{\partial b} = ?$$

# Step 3: Gradient Descent for Linear Regression

- Formulation of $\partial L/\partial w$ and $\partial L/\partial b$

$$L(w,b) = \sum_{i=1}^{N} \left( y^{(i)} - \left( b + w \cdot x^{(i)} \right) \right)^2$$

$$\frac{\partial L}{\partial w} = \sum_{i=1}^{N} 2 \left( y^{(i)} - \left( b + w \cdot x^{(i)} \right) \right) \left( -x^{(i)} \right)$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{N} 2 \left( y^{(i)} - \left( b + w \cdot x^{(i)} \right) \right) (-1)$$

# How are the results?

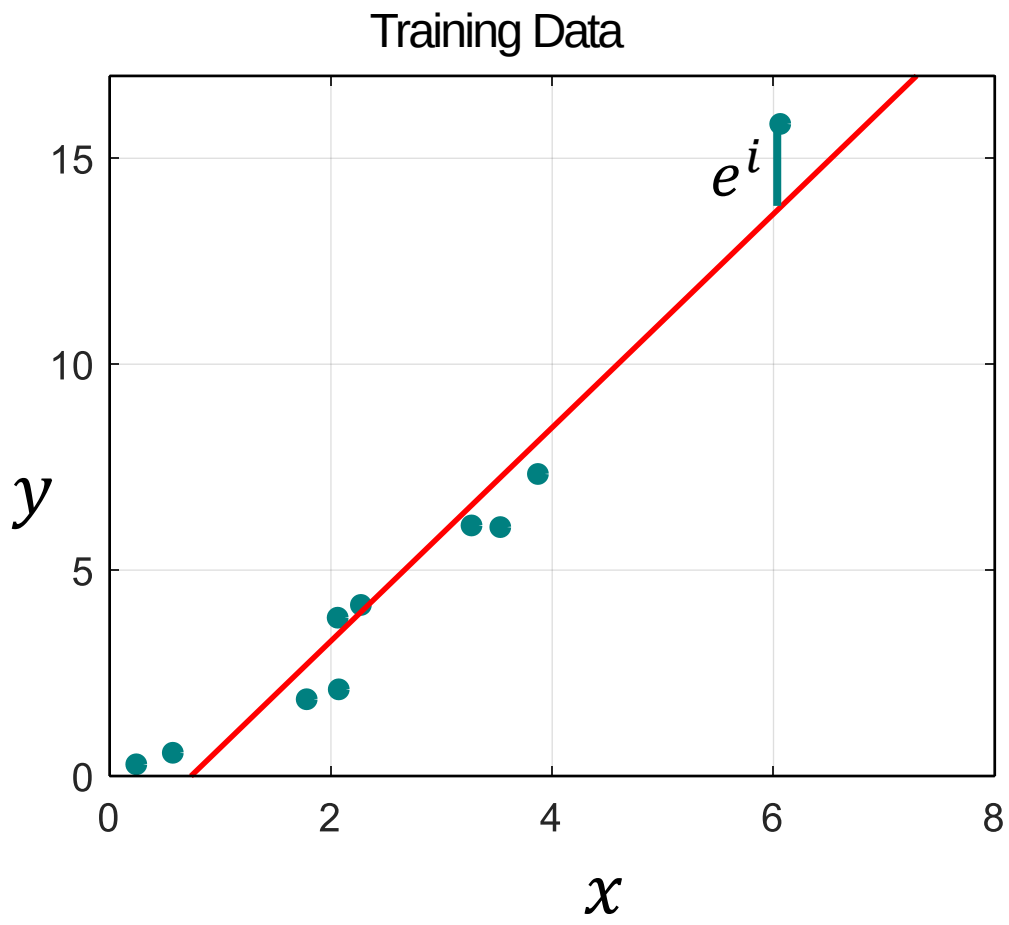
$$y = b + w \cdot x$$

$b$ = -1.91

$w$ = 2.59

Average Error on Training Data

$$= \sum_{i=1}^{10} e^i = 1.12$$



# How are the results? - Generalization

What we really care about is the error on new data (testing data)

$$y = b + w \cdot x$$

$b$ = -1.91

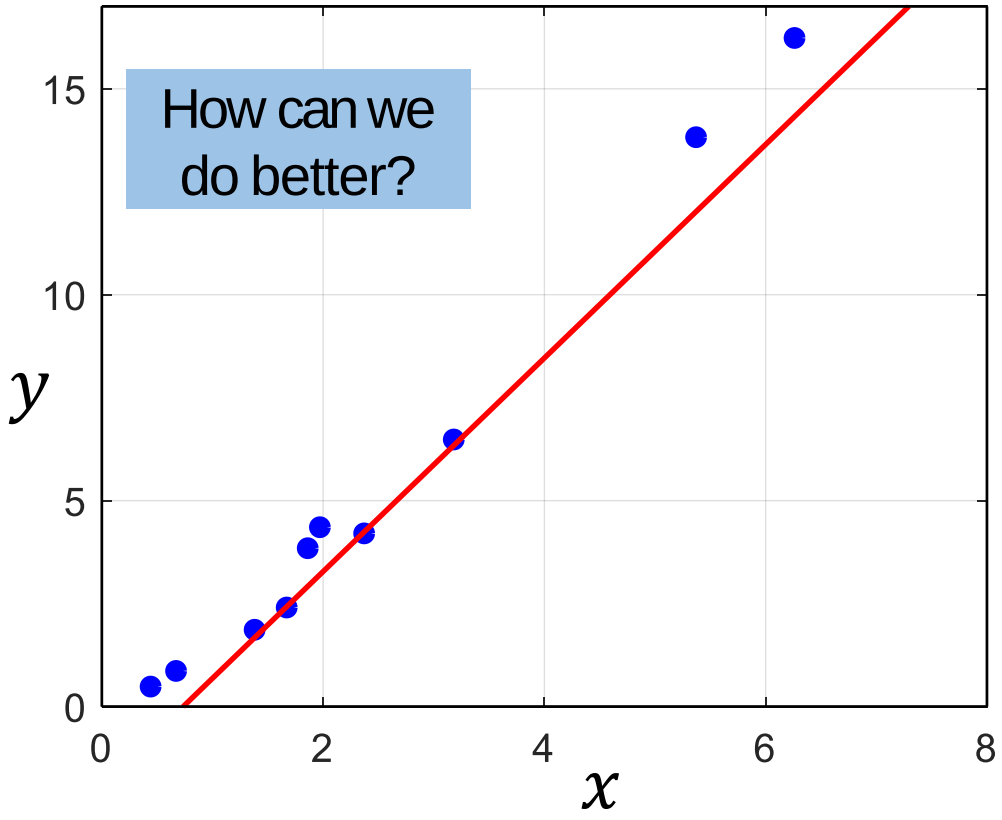
$w$ = 2.59

Average Error on Training Data

$$= \sum_{i=1}^{10} e^i = 1.12$$

Average Error on Training Data (1.24)

Another 10 points as testing data



# Selecting another Model

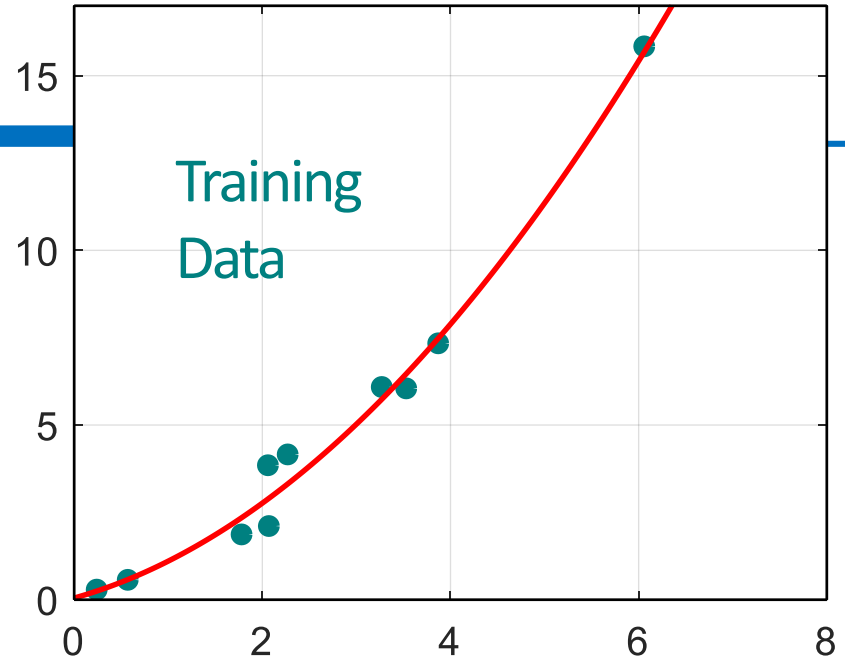
$$y = b + w_1 \cdot x + w_2 \cdot x^2$$

## Best Function

$b = 0.05$

$w_1 = 0.76$, $w_2 = 0.30$

Average Error = 0.53

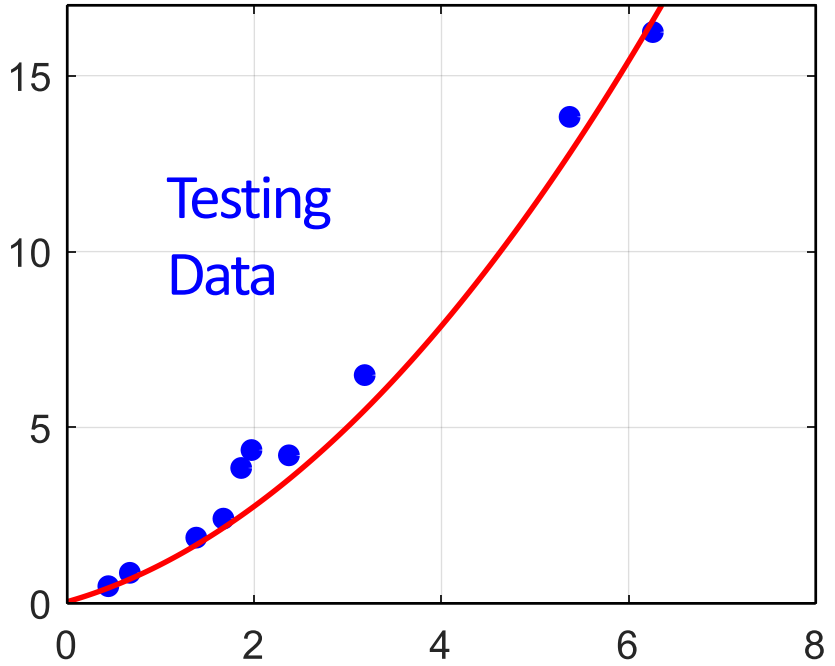



## Testing Results

Average Error = 0.82

Better!

Could it be even better?



# Selecting another Model

$$y = b + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3$$

## Best Function

$b = -0.13$

$w_1 = 1.15$, $w_2 = 0.13$

$w_3 = 0.018$

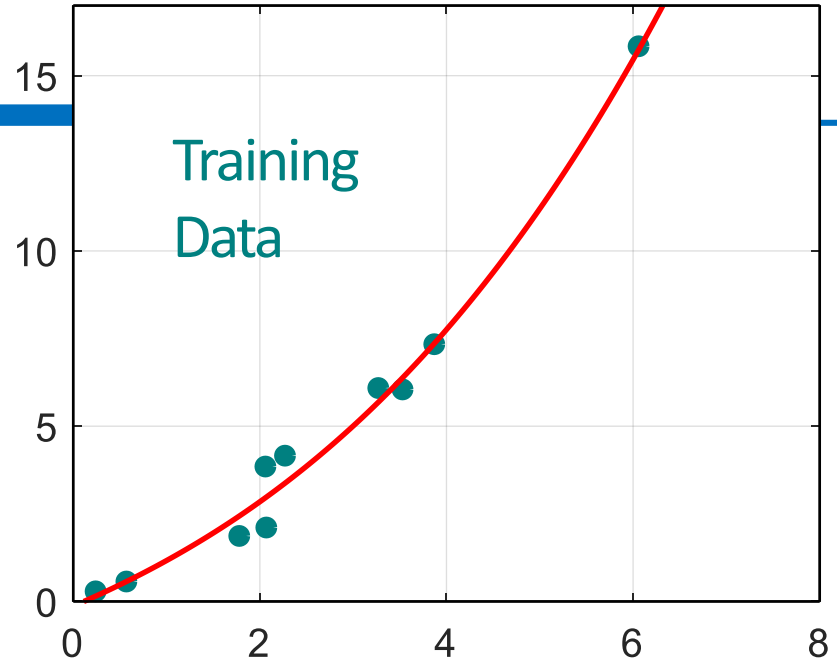
Average Error = 0.52

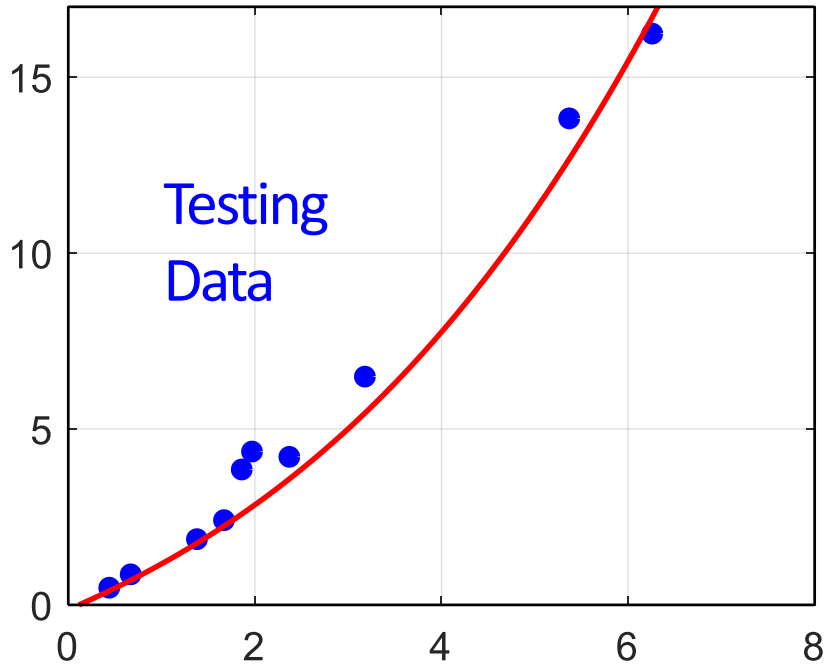
## Testing Results

Average Error = 0.81

Slightly better.

How about more complex model?



# Selecting another Model

$$y = b + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3 + w_4 \cdot x^4$$

## Best Function

$b = 0.85$

$w_1 = -2.45$, $w_2 = 2.91$

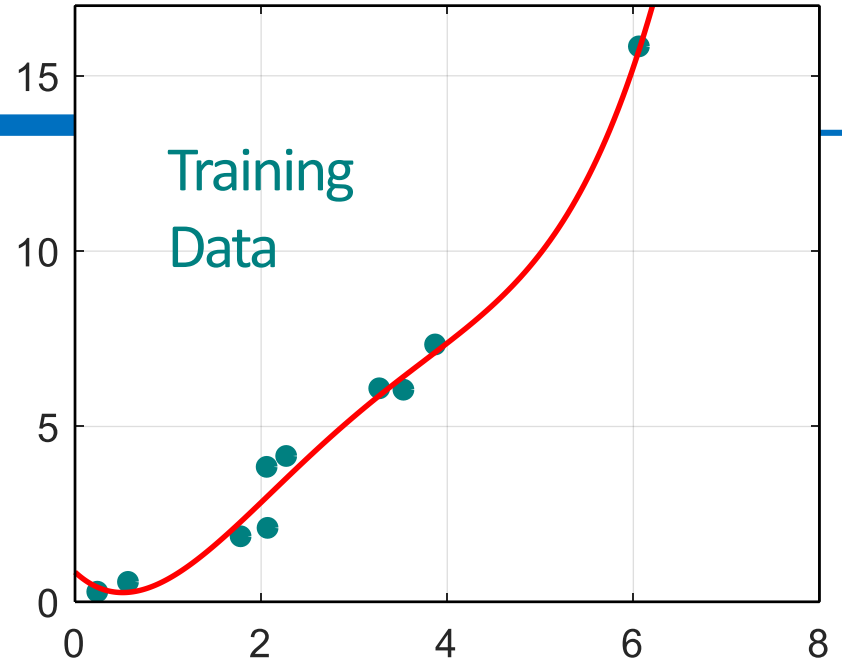
$w_3 = 0.72$, $w_4 = 0.06$

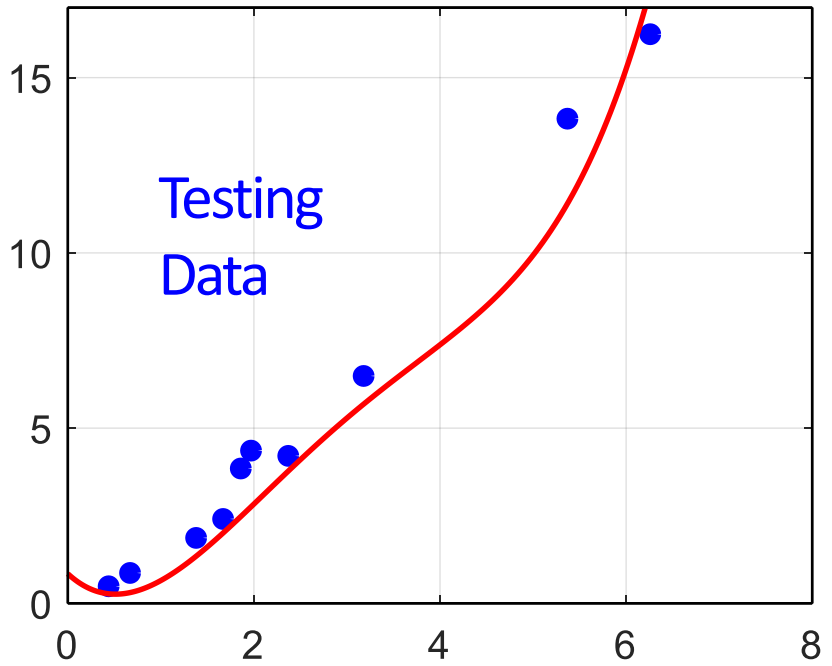
Average Error = 0.50

## Testing Results

Average Error = 1.14

Results become worse.



# Selecting another Model

$$y = b + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3 + w_4 \cdot x^4 + w_5 \cdot x^5$$

## Best Function

$b = 0.52$

$w_1 = -0.93$, $w_2 = 1.06$

$w_3 = 0.18$, $w_4 = -0.13$

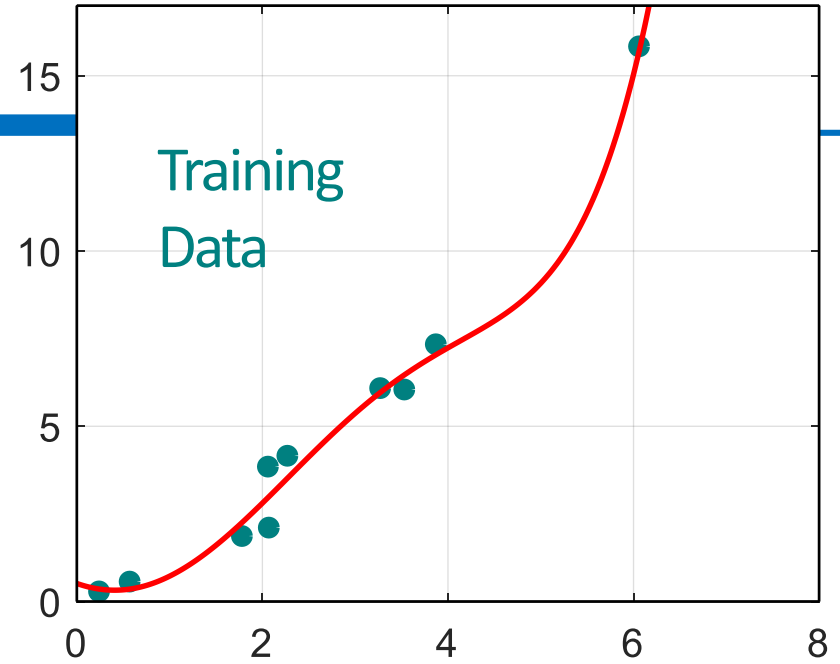
$w_5 = -0.01$

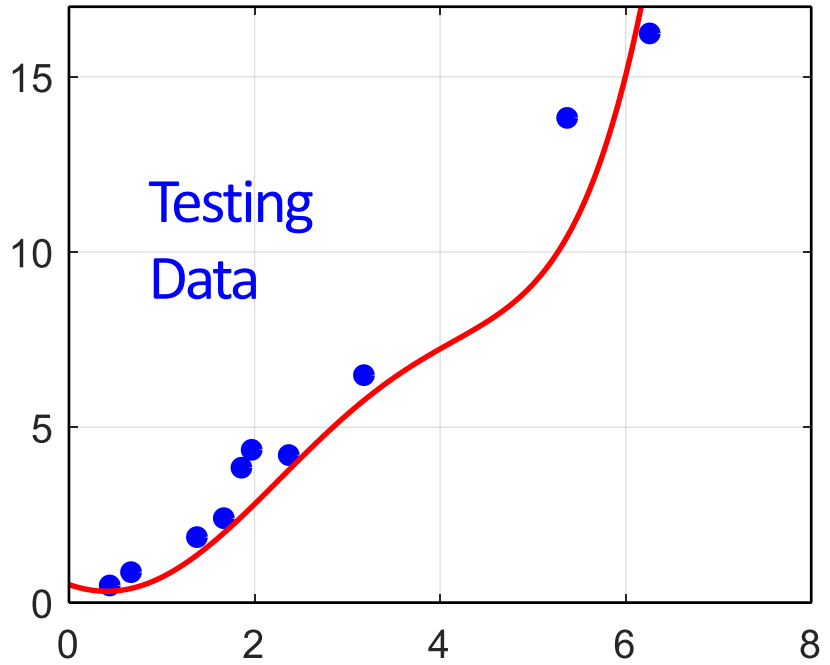
Average Error = 0.49

## Testing Results

Average Error = 1.47

Results become even worse.



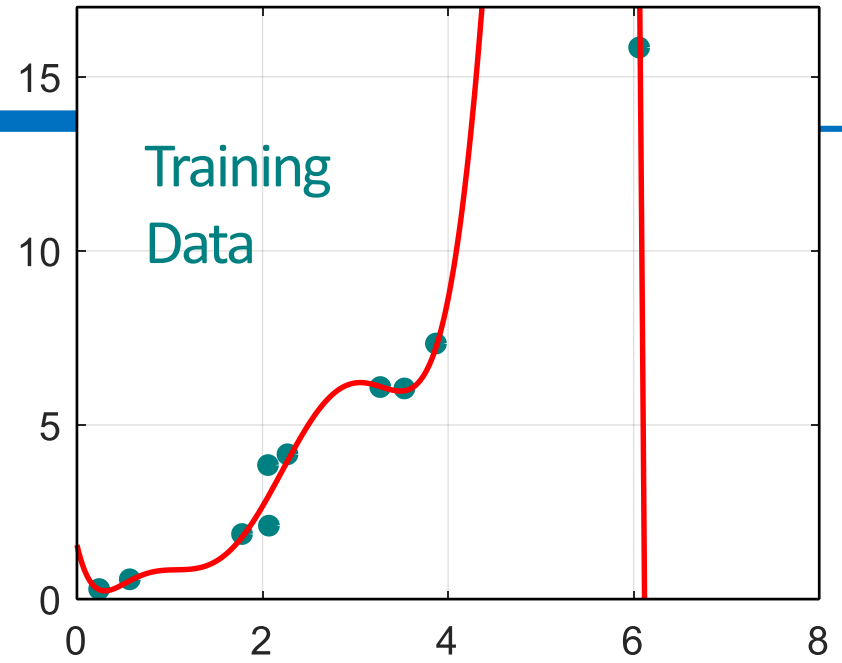



# Selecting another Model

$$y = b + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3 + w_4 \cdot x^4 + w_5 \cdot x^5 + w_6 \cdot x^6 + w_7 \cdot x^7$$

## Best Function

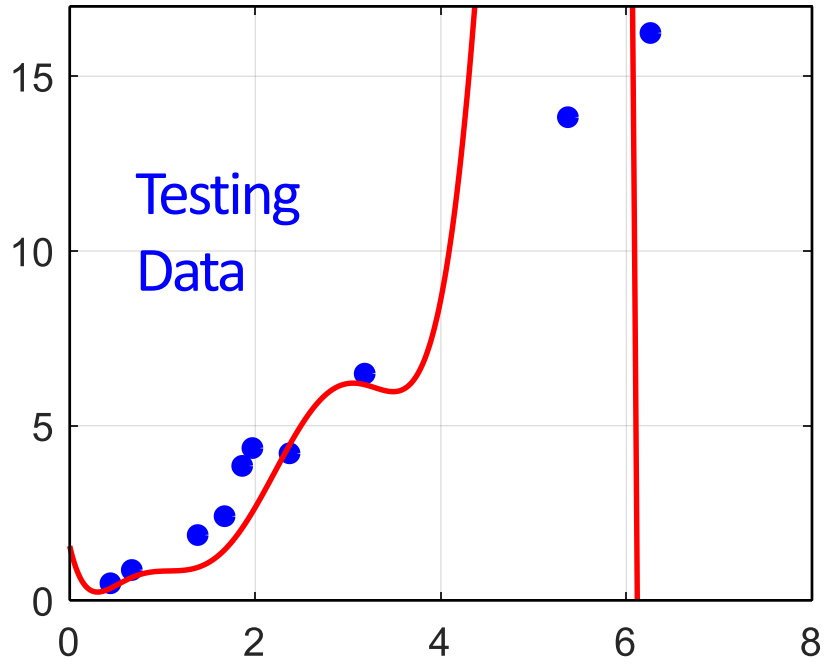
Average Error = 0.40




## Testing Results

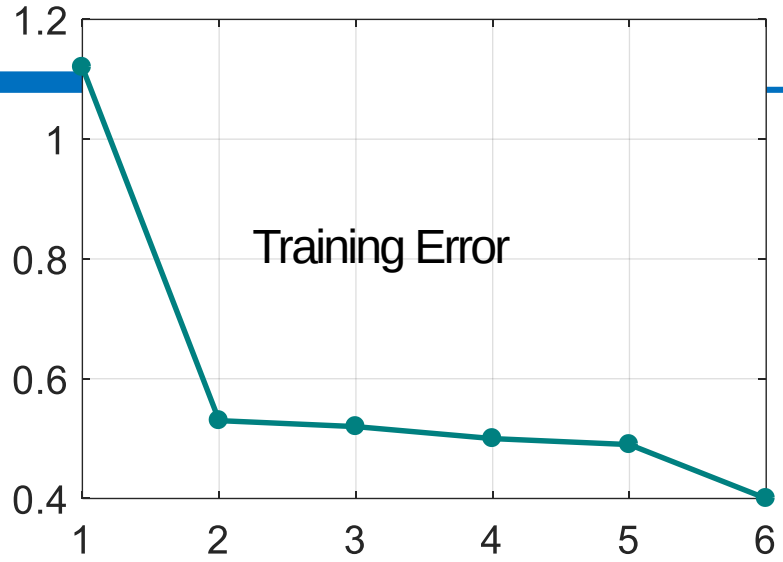
Average Error = 33.05

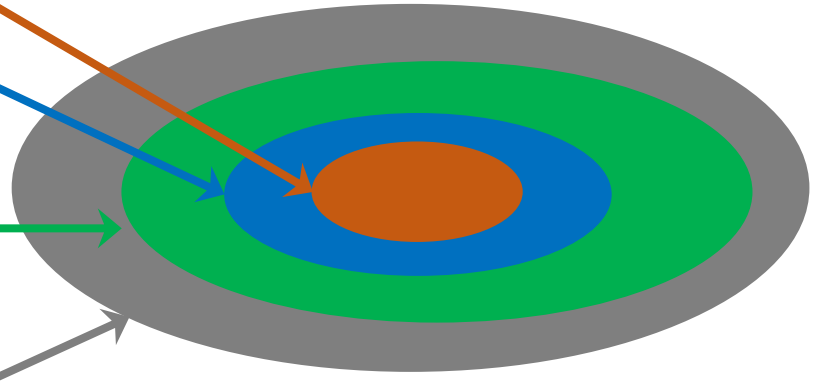
Results become so bad.



# Model Selection

1. $y = b + w \cdot x$
2. $y = b + w_1 \cdot x + w_2 \cdot x^2$
3. $y = b + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3$
4. $y = b + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3 + w_4 \cdot x^4$
5. $y = b + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3 + w_4 \cdot x^4 + w_5 \cdot x^5$
6. $y = b + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3 + w_4 \cdot x^4 + w_5 \cdot x^5 + w_6 \cdot x^6 + w_7 \cdot x^7$

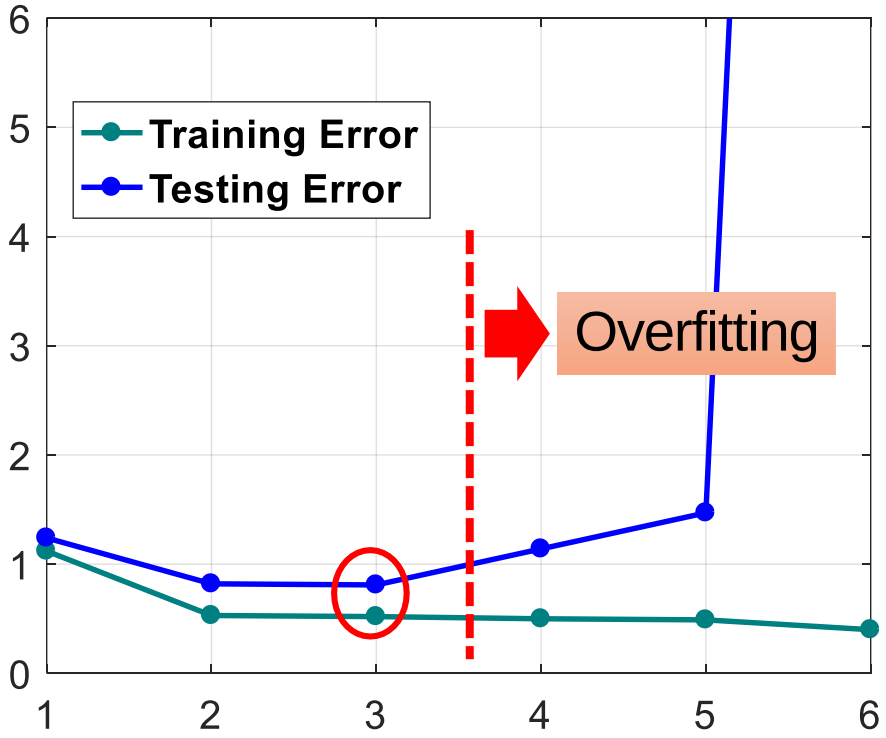


A more complex model yields lower error on training data.

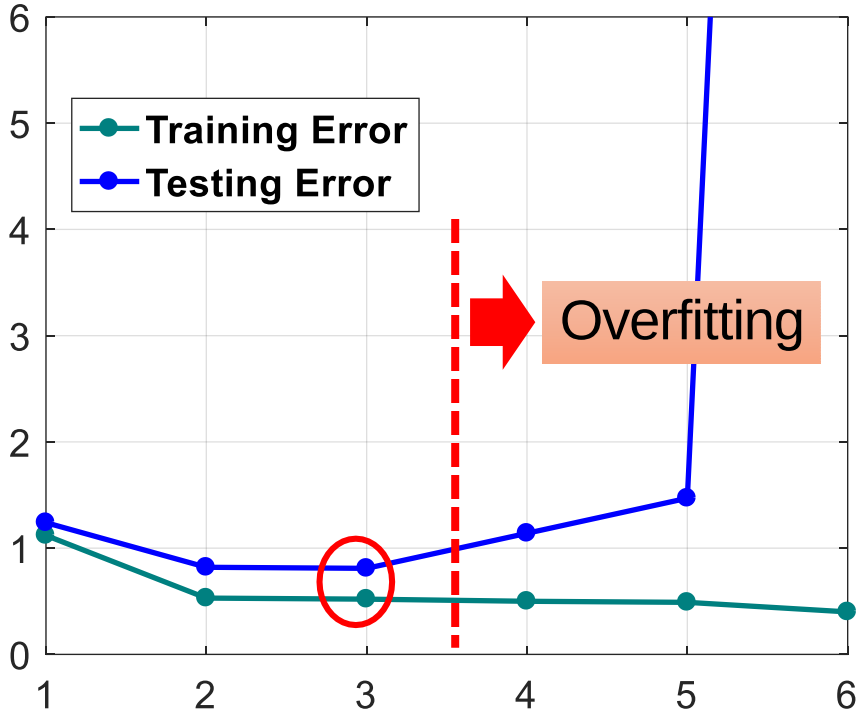
If we can truly find the best function

# Model Selection




| | Training Error | Testing Error |
|---|---|---|
| 1 | 1.12 | 1.24 |
| 2 | 0.53 | 0.82 |
| 3 | 0.52 | 0.81 |
| 4 | 0.50 | 1.14 |
| 5 | 0.49 | 1.47 |
| 6 | 0.40 | 33.05 |

# Model Selection

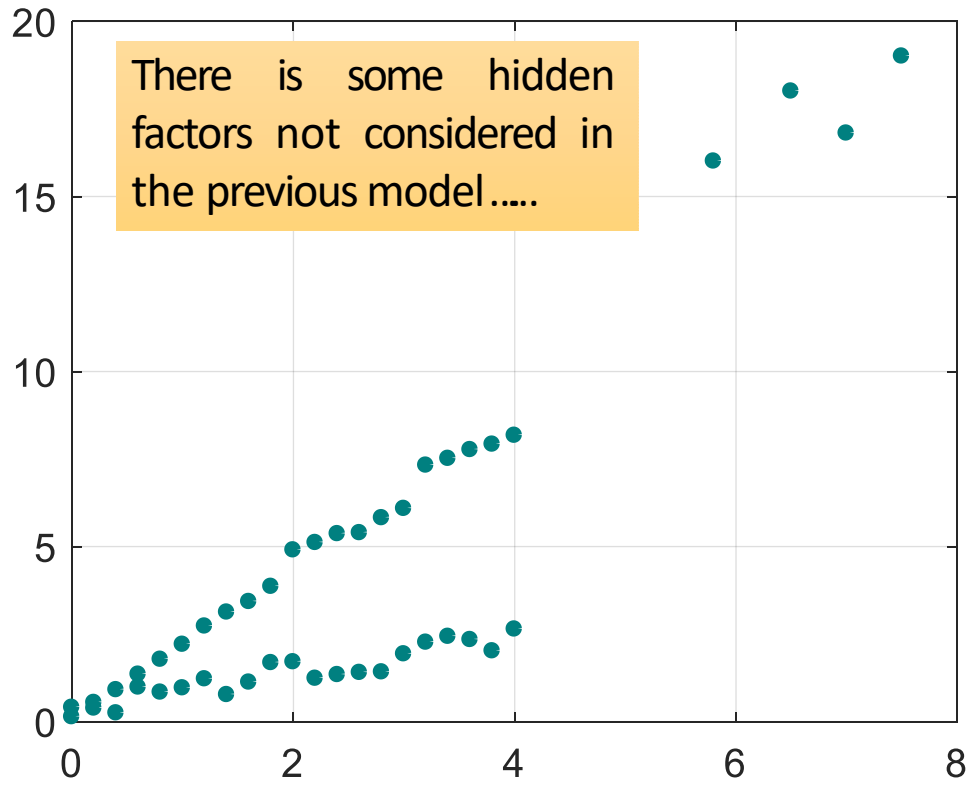



| | Training Error | Testing Error |
|---|---|---|
| 1 | 1.12 | 1.24 |
| 2 | 0.53 | 0.82 |
| 3 | 0.52 | 0.81 |
| 4 | 0.50 | 1.14 |
| 5 | 0.49 | 1.47 |
| 6 | 0.40 | 33.05 |

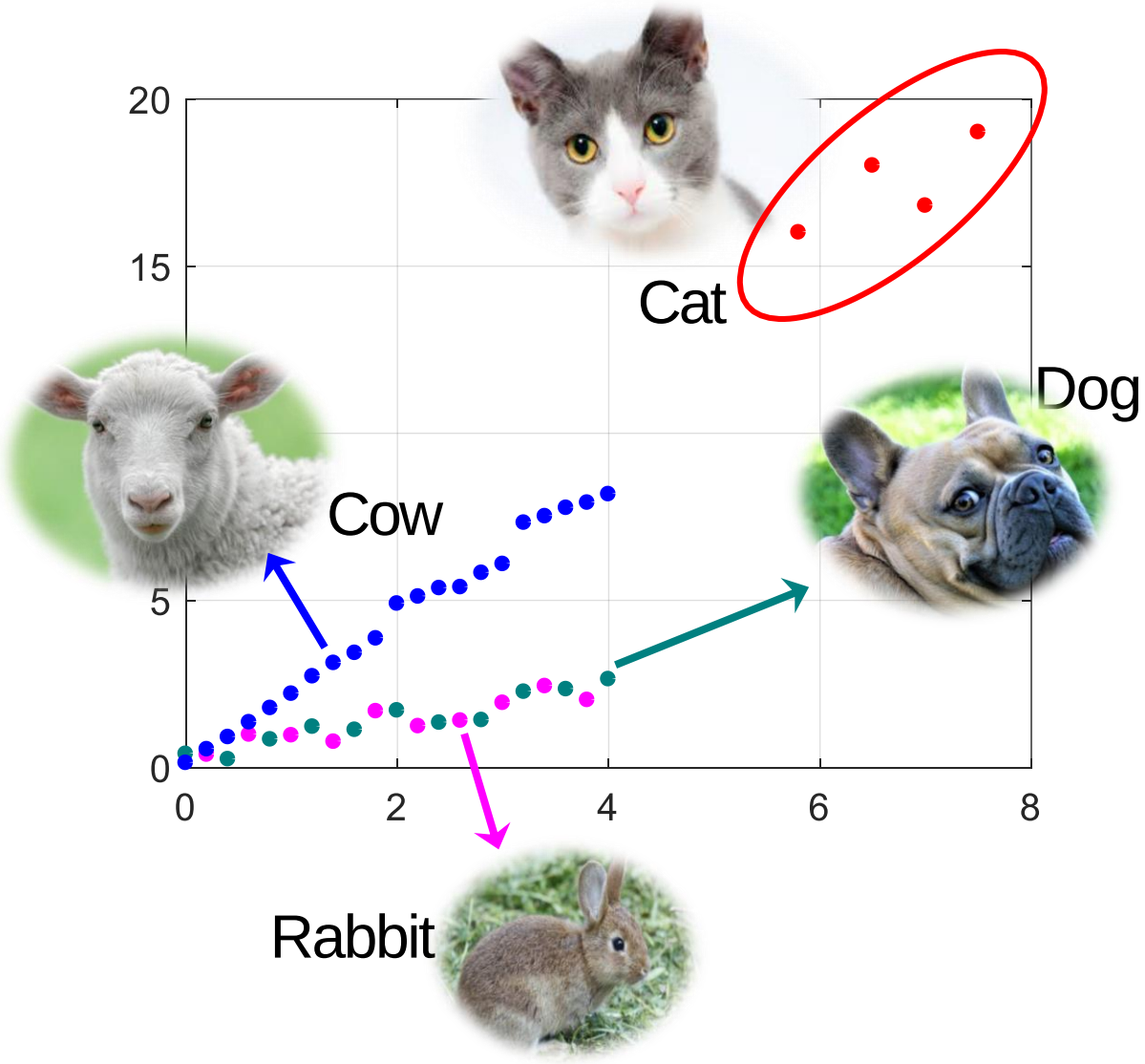
A more complex model does not always lead to better performance on **testing data**.

This is **Overfitting**. ➡ Select suitable model

# Let's collect more data



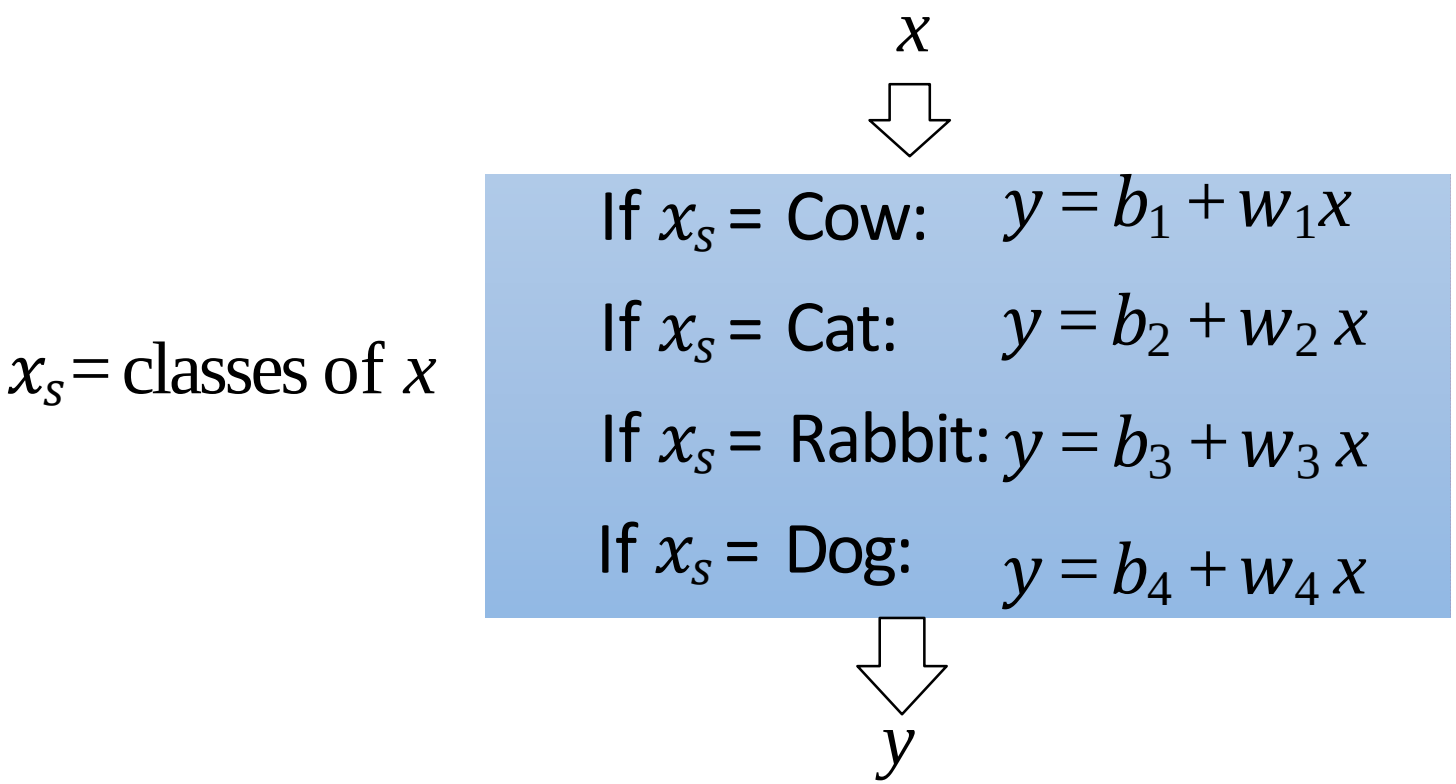
# What are the hidden factors?



# Back to step 1: Redesign the Model

$$y = b + \sum w_i x_i$$

Linear model?




$x_s$ = classes of $x$

# Back to step 1: Redesign the Model

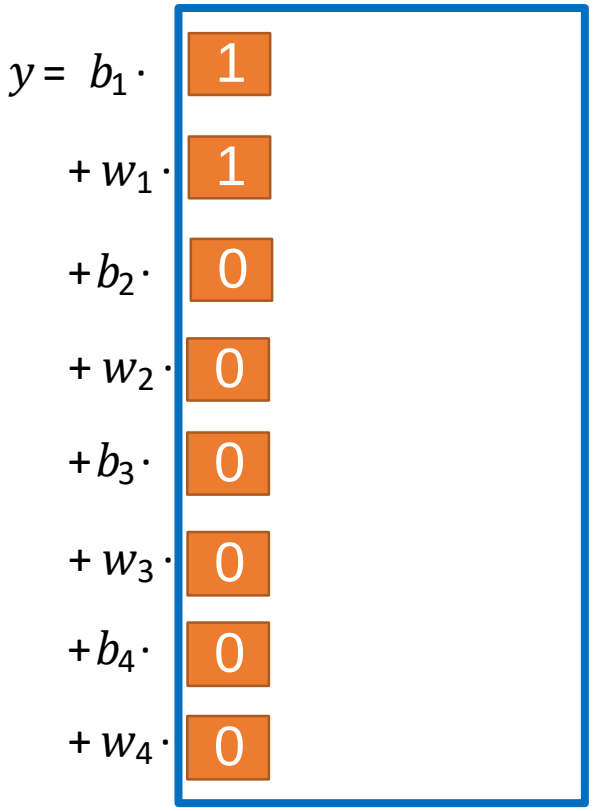
$$y = b_1 \cdot \delta(x_s = \text{Cow}) + w_1 \cdot \delta(x_s = \text{Cow}) \cdot x + b_2 \cdot \delta(x_s = \text{Dog}) + w_2 \cdot \delta(x_s = \text{Dog}) \cdot x + b_3 \cdot \delta(x_s = \text{Rabbit}) + w_3 \cdot \delta(x_s = \text{Rabit}) \cdot x + b_4 \cdot \delta(x_s = \text{Cat}) + w_4 \cdot \delta(x_s = \text{Cat}) \cdot x$$

$$y = b + \sum w_i x_i$$

Linear model?

$$\delta(x_s = \text{Cow}) = \begin{cases} = 1, \text{If } x_s = \text{Cow} \\ = 0, \text{otherwise} \end{cases}$$

# Back to step 1: Redesign the Model




$$y = b + \sum w_i x_i$$

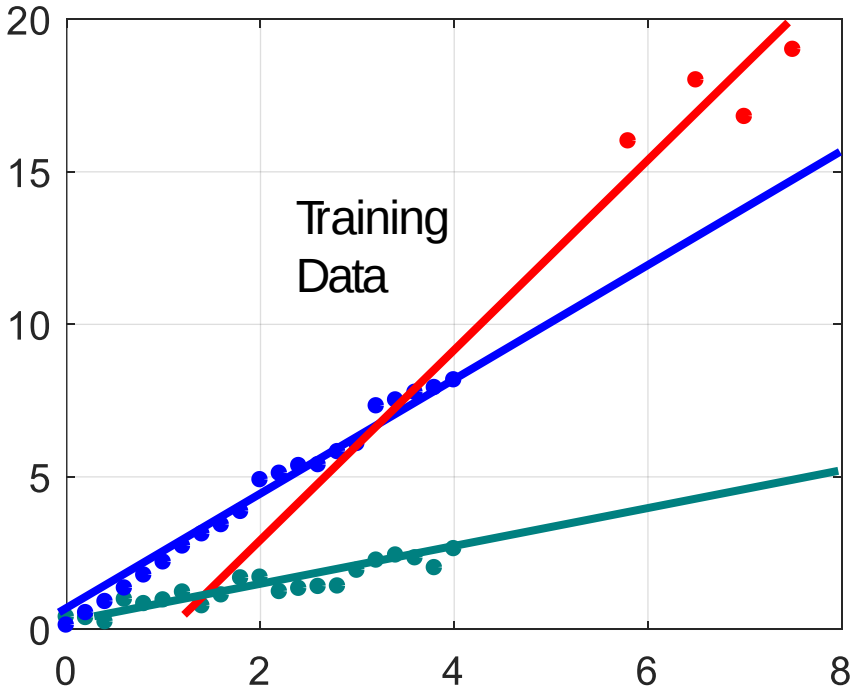
Linear model?

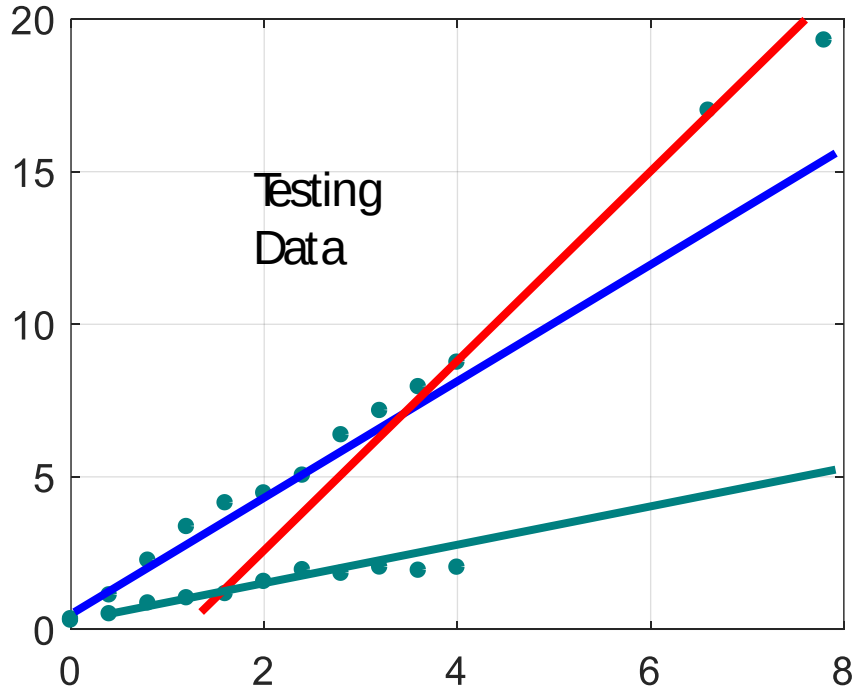
$$\delta(x_s = \text{Cow}) = \begin{cases} = 1, \text{If } x_s = \text{Cow} \\ = 0, \text{otherwise} \end{cases}$$

If $x_s$ = Cow,

$$y = b_1 + w_1 \cdot x$$

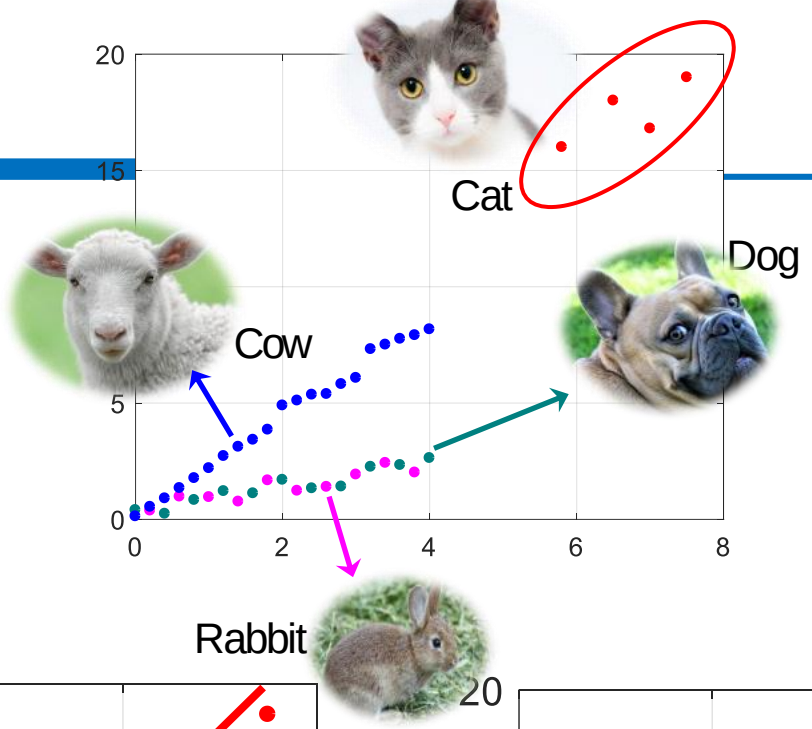
# Results

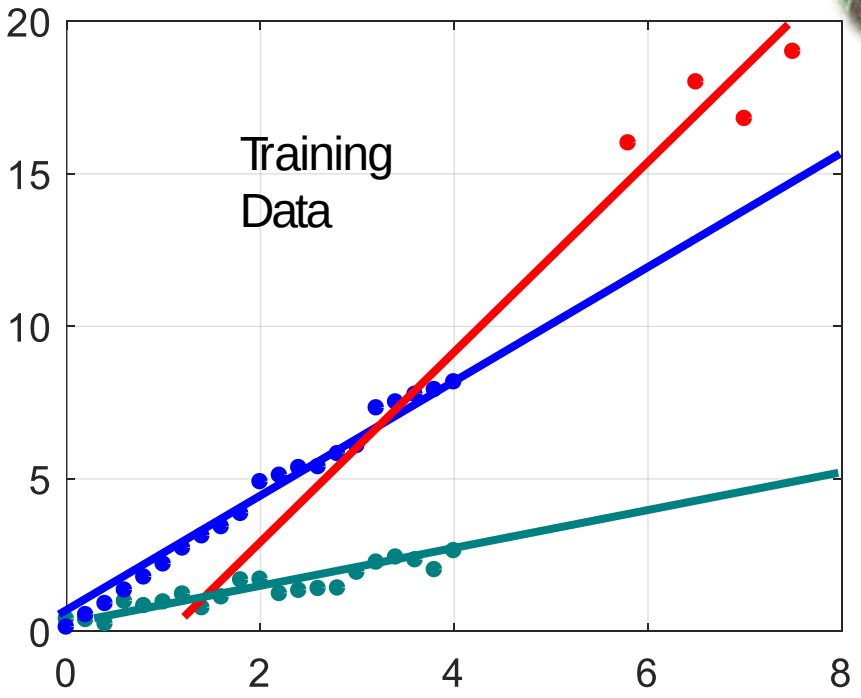


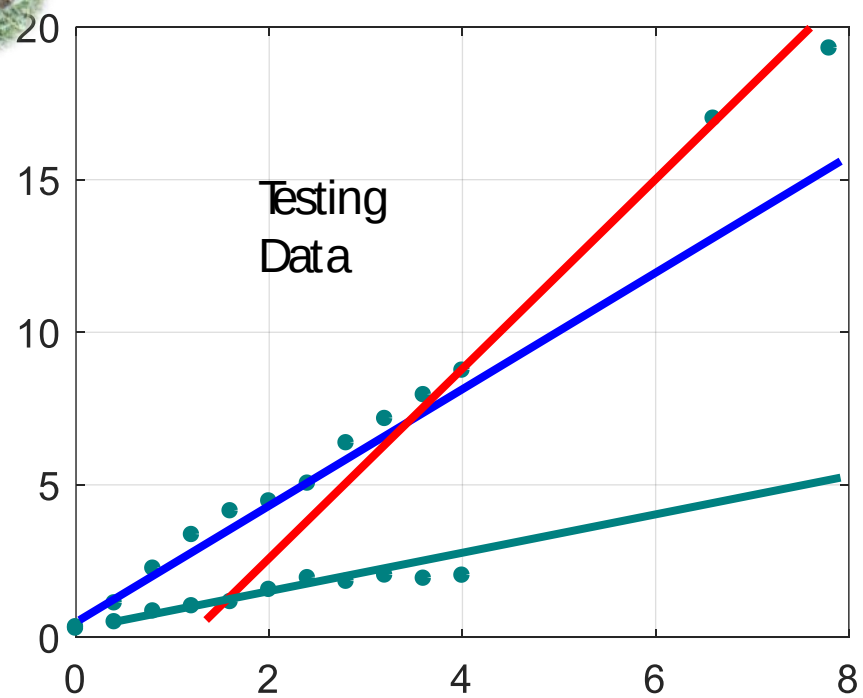




Are there any other hidden factors?

# Results

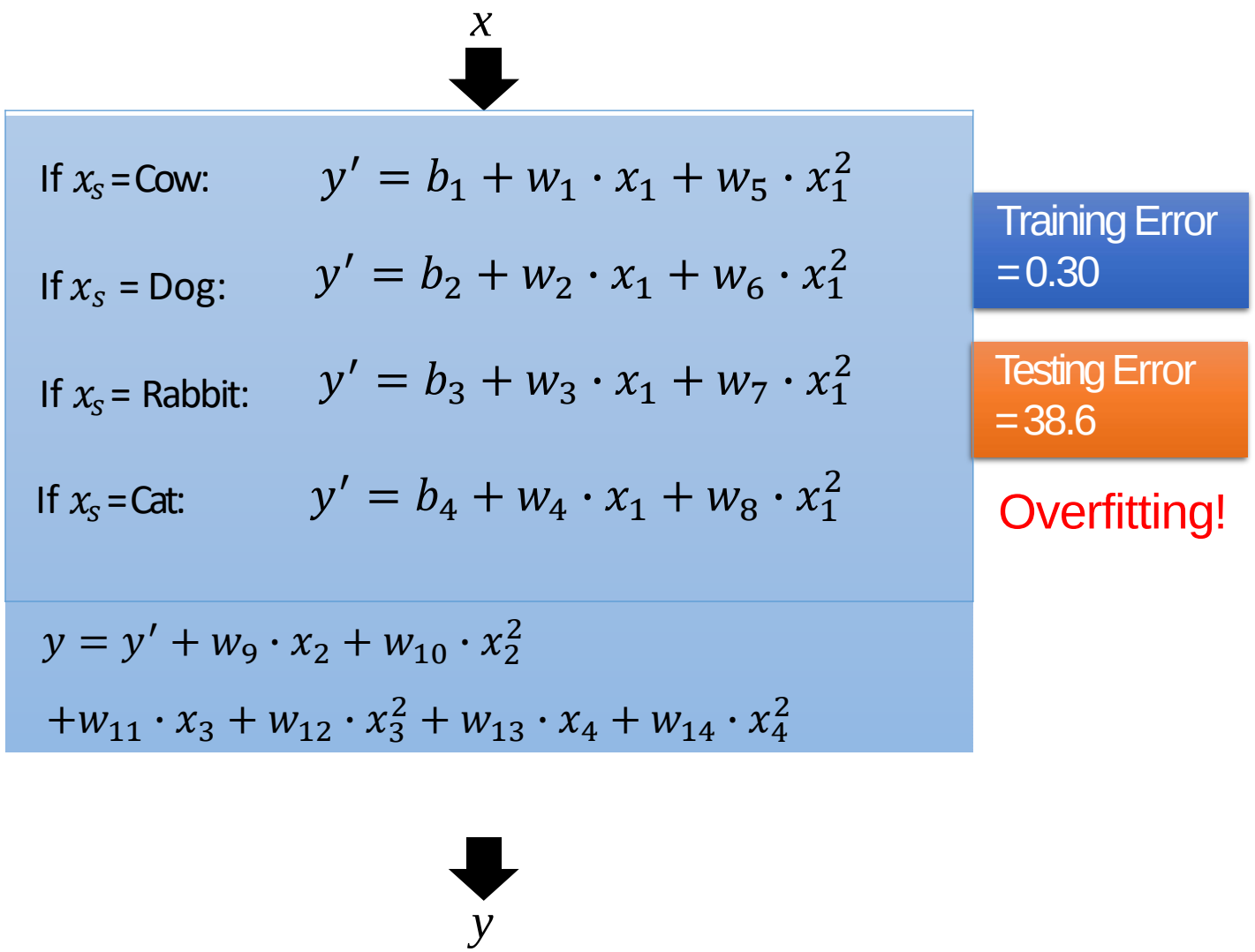



Are there any other hidden factors?

# Back to step 1: Redesign the Model Again




Training Error = 0.30

Testing Error = 38.6

Overfitting!

# Back to step 2: Regularization

$$y = b + \sum w_j x_j$$

The functions with smaller $w_j$ are better

$$L = \sum_i \left( y^{(i)} - \left( b + \sum w_j x_j^{(i)} \right) \right)^2 + \lambda \sum (w_j)^2$$

- Why smooth functions are preferred?

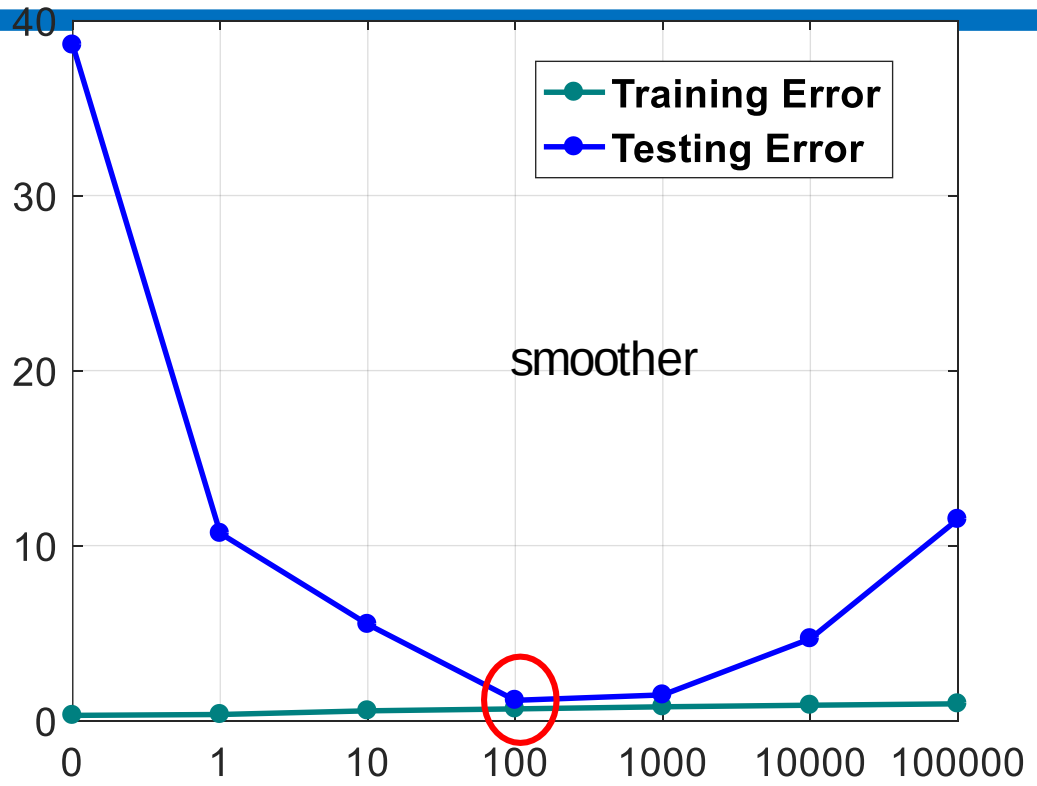
$$y = b + \sum_j w_j x_j$$

$+w_j \Delta x_j$ $+\Delta x_j$

- If some noises corrupt input $x_j$ when testing

A smoother function (slow changing, low frequency) has less influence.

# Regularization




| $\lambda$ | Training | Testing |
|---|---|---|
| 0 | 0.30 | 38.6 |
| 1 | 0.35 | 10.7 |
| 10 | 0.56 | 5.5 |
| 100 | 0.67 | 1.16 |
| 1000 | 0.79 | 1.47 |
| 10000 | 0.88 | 4.68 |
| 100000 | 0.96 | 11.5 |

How smooth?
Select $\lambda$ obtaining the best model

- Training error: larger$\lambda$, considering the training error less
- We prefer smooth function, but don't be too smooth.

# Conclusion

- Course Information
- Introduction to Deep Learning
- Deep Learning Basics
  - Linear Regression
  - Loss Function
  - Gradient Descent
  - Regularization

# Next Lecture:
# Loss Functions
# Gradient Descent: theory and tips
# Logistic Regression