Deep Learning

Li Liu
Center for Machine Vision and Signal analysis
2019 October 28

Outline

- Course Information
- Introduction to Deep Learning
- Deep Learning Basics (Linear Regression, Loss Function)

Course Information

- Signup link:
  - Please register in weboodi first if you want to obtain credits
- Course webpage:
  - https://moodle.oulu.fi/course/view.php?id=2379
  - Lecture slides, assignments, project, grades

Course Information

- Lecturer: Li Liu
    - Li Liu <Li dat Liu et oulu dat fi>
- Teaching assistant
    - Lam Huynh (Lam dat Huynh et oulu dat fi)
    - Zhuo Su (Zhuo dat Su et oulu dat fi)
    - Yawen Cui

Course Information

- 8 lectures + 1 computer class exercise

- 4 assignments + 1 final project

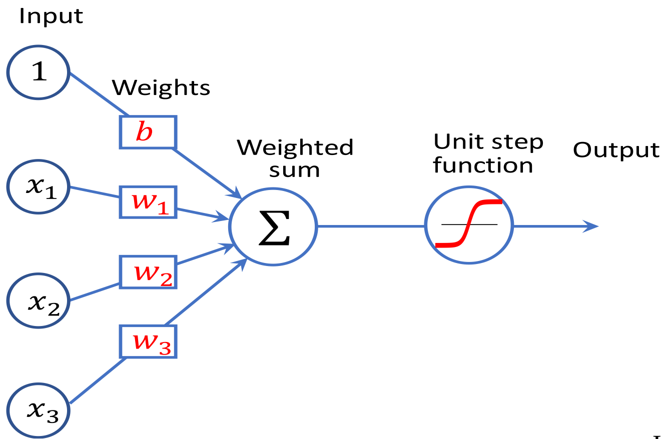
- no exam.

Assignments for registered students will be emailed later on in the week.

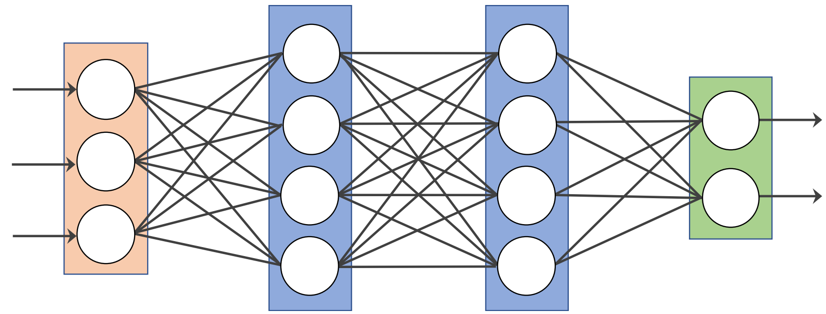
- PyTorch for assignments and final project.

In this Course

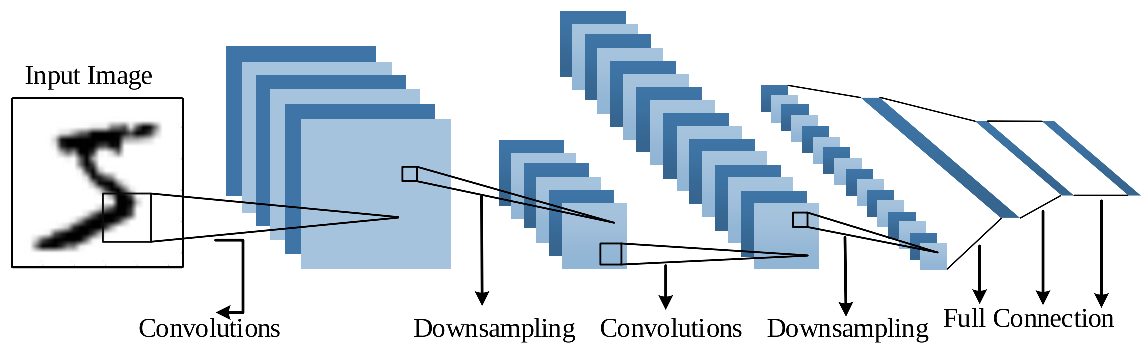
1. DL basics, linear regression, logistic regression etc.



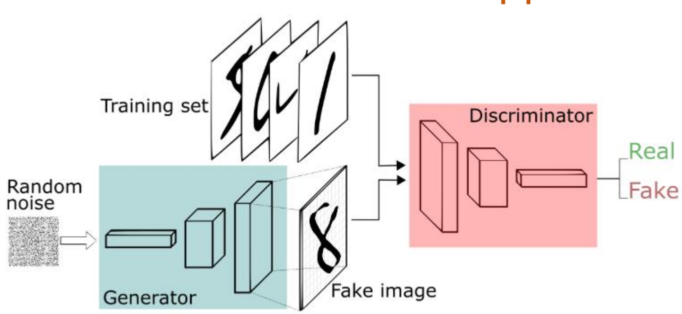
2. Multilayer neural networks, backpropagation



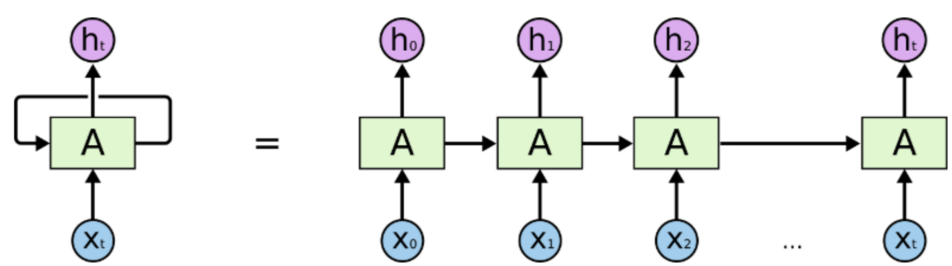
3. Convolutional Neural Networks and Applications



4. Recurrent networks and applications



5. Generative Adversarial Networks



Course Info (tentative)

Week 1

- Lecture 1: Course Overview, Introduction to Deep Learning, Deep Learning Basics (Linear Regression, Loss Function) (**Li Liu**) Today
- Lecture 2: Deep Learning Basics continued: Gradient Descent, Stochastic Gradient Descent, and Logistic Regression (**Li Liu**) This Friday

Week 2

- Tutorial on Pytorch (Groups) (**Zhuo Su**)

Week 3

- Lecture 3: Neural Networks, Back Propagation, Deep Neural Networks (**Li Liu**)

Course Info (tentative)

Week 4:

- Lecture 4: Convolutional Neural Networks (**Li Liu**)
- Lecture 5: Generative Adversarial Networks (GANs) **(Lam Huynh)**

Week 5:

- Lecture 6: Network Compression (**Zhuo Su**)
- Lecture 7: State of the art and Applications of CNN in computer vision problems (Classification, Object Detection) (**Li Liu**)
- Lecture 8: Deep Models for Text and Sequences (LSTM, RNN), with applications in Lip Reading (**Changchong Sheng**)

Course Schedule

Date	Time	Classroom	Teacher	Contents
28.10.19	Mon 14.15–16.00	TS101	Li	Lecture1
01.11.19	Fri 10.15-12.00	TS101	Li	Lecture2
04.11.19	Mon 10.15–12.00	TS137	Zhuo/Lam	Laboratory exercise
06.11.19	Wed 12.15–14.00	TS137	Zhuo/Lam	Laboratory exercise
07.11.19	Thu 10.15–12.00	TS137	Zhuo/Lam	Laboratory exercise
11.11.19	Tue 14.15–16.00	TS101	Li	Lecture3
18.11.19	Mon 12.15–14.00	TS101	Li	Lecture4
19.11.19	Tue 14.15–16.00	TS101	Lam	Lecture5
25.11.19	Mon 12.15–14.00	TS101	Zhuo	Lecture6
26.11.19	Tue 14.15–16.00	TS101	Li	Lecture7
27.11.19	Wed 12.15–14.00	TS101	Changchong	Lecture8

Course Information

- Lecturer: Li Liu
  - Li Liu <Li dat Liu et oulu dat fi>
  - Lecture 1, 2, 3, 4 7
- Teaching assistant
  - Lam Huynh (lam dat huynh et oulu dat fi)
    - Exercise
    - Project
    - Pytorch: install and familiar (QA)
    - Lecture 5
  - Zhuo Su (zhuo dat su et oulu dat fi)
    - Exercise
    - Project
    - Website management (Moodle/webwoodi/ Registration)
    - QA: answer questions
    - Lecture 6

Course Information

Important Notice!

- Estimated number of students is about 80
- For the lecture Pytorch tutorial, we have to reserve a computer room (such as TS 137), which has only 25 computers.
  - Divide students into several groups.
  - We require that you fill your name and email on this sheet here or tell us via email (Zhuo dat Su et oulu dat fi) after this lecture to decide if you will attend the PyTorch tutorial.
  - We will post group information on course webpage.
  - If the time of PyTorch tutorial is conflict to your other course, please let us know (email to Zhuo dat Su et oulu dat fi). We will make changes.

04.11.19	Mon 10.15–12.00	TS137	Zhuo/Lam	Laboratory exercise
06.11.19	Wed 12.15–14.00	TS137	Zhuo/Lam	Laboratory exercise
07.11.19	Thu 10.15–12.00	TS137	Zhuo/Lam	Laboratory exercise

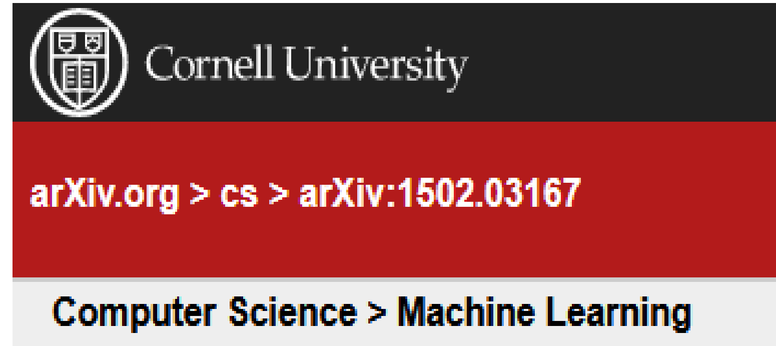
PyTorch

- CMVS
  - use our own servers and the virtual environment :
  - This environment supports Jupyter notebooks.

- CSC (Finnish Center for Scientific Computing)

- PyTorch has been installed on TS137.
  - on your computer,
    - https://pytorch.org/
    - Zhuo Su (Zhuo dat Su et oulu dat fi)
    - laptop (Ubuntu 16.04).

Selflearning



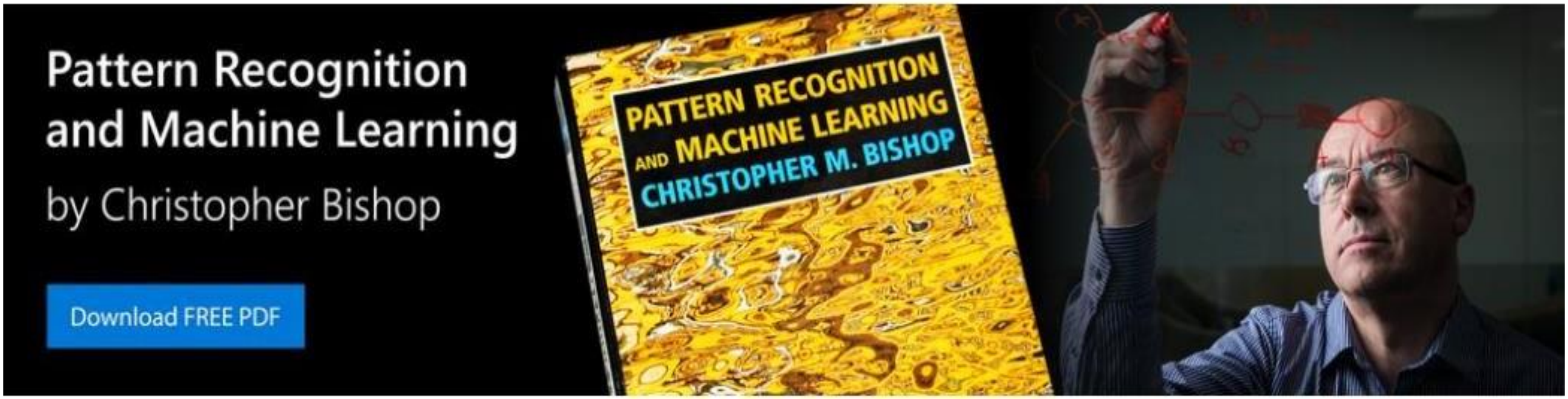
- Find on social media (Twitter, Facebook …)
- Read latest papers in a subarea
- Write code to reinforce understanding of concepts

Books

http://www.deeplearningbook.org/

Form a group to read books and discuss each chapter.

PRML Book https://aka.ms/prml



Some good blogs

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

…and many others …

Some good tutorials

Ryota Tomioka MSR Summer School 2018
https://notebooks.azure.com/ryotat/projects/DLTutorial
Pure numpy tutorial on Perceptron, MLPs, MLPs with autodiff

Kai Arulkumaran MLSS 2019
https://github.com/mlss-2019/tutorials/tree/master/deep_learning
Basics of CNNs, RNNs, VAE, GANs (PyTorch)

Sebastian Raschka
https://github.com/rasbt/deeplearning-models
Collection of different architectures/models (TensorFlow/PyTorch)

PyTorch/TensorFlow framework tutorials
Demos: https://cs.stanford.edu/people/karpathy/convnetjs/

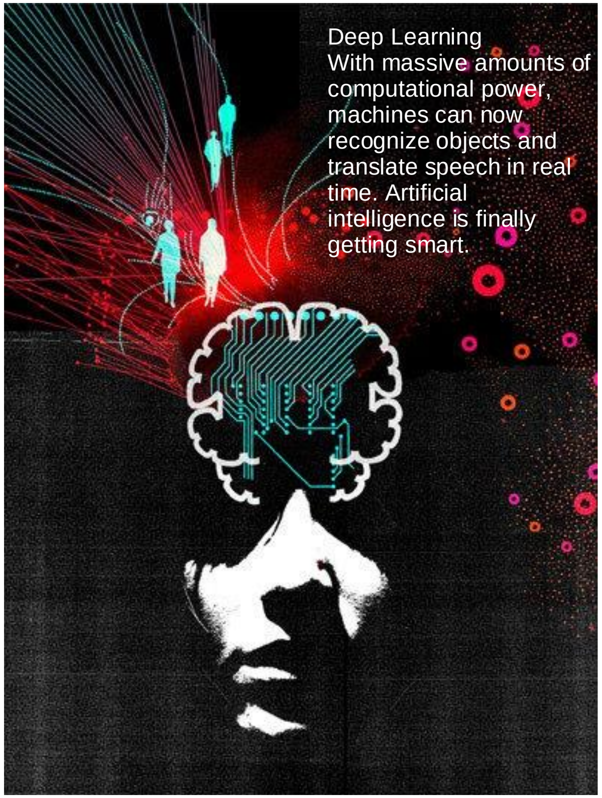
…and many others …

Outline

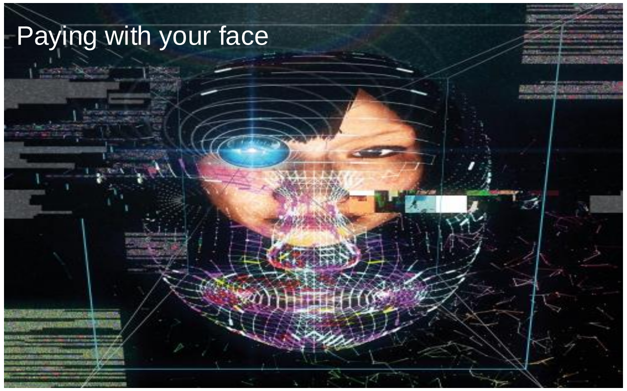
- Course Information
- Introduction to Deep Learning
- Deep Learning Basics (Linear Regression, Loss Function)

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

This course focuses on the basic techniques.

Deep Learning in **One** Slide

- **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

Unlimited Photos, Movies, Home Videos, Medical Images…

Tens of Billions of Photos on Facebook

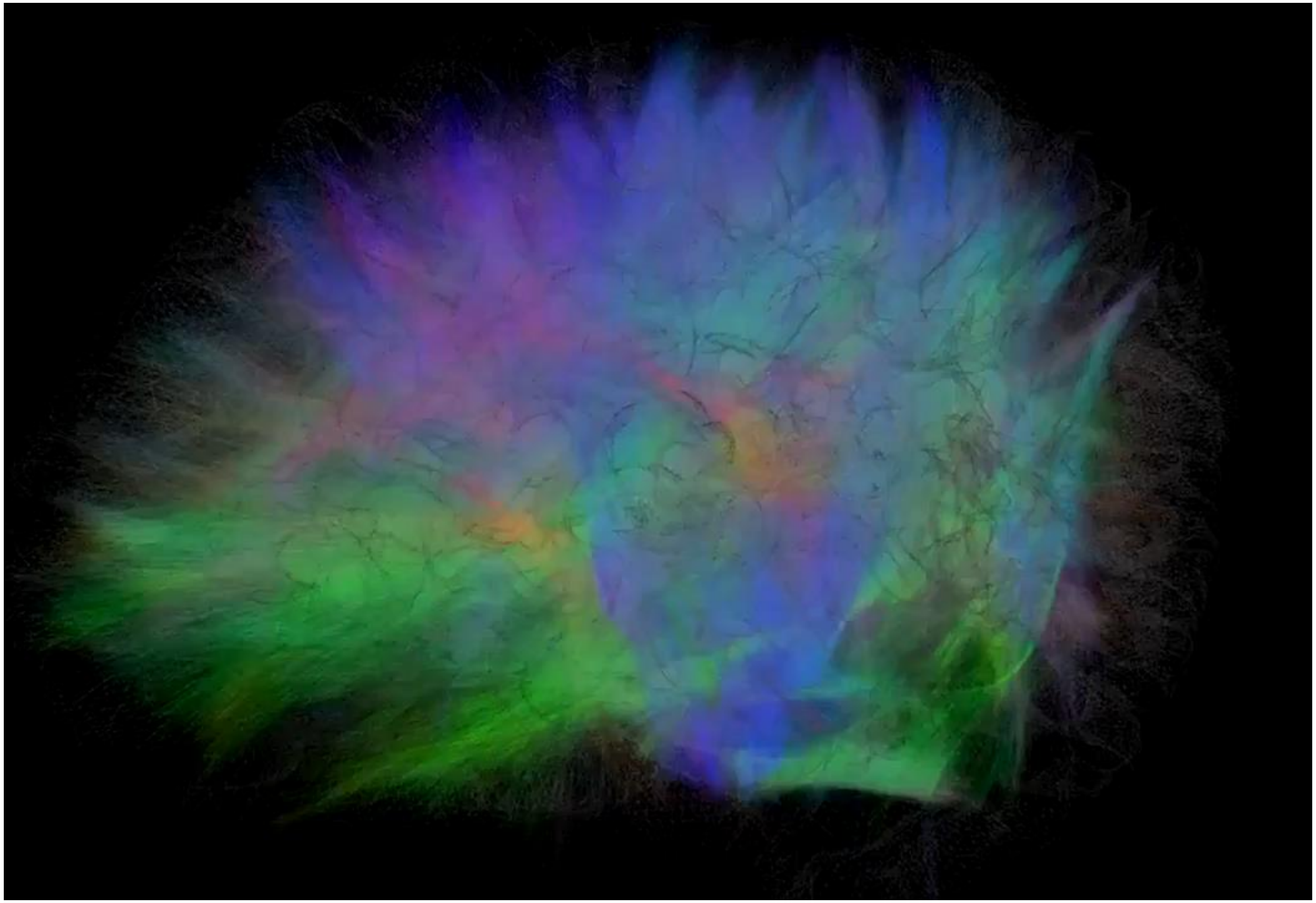
Hundreds of Millions of Videos on YouTube

Large Amount of Surveillance Videos

Object Recognition

"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

A Incomplete History of Deep Learning

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

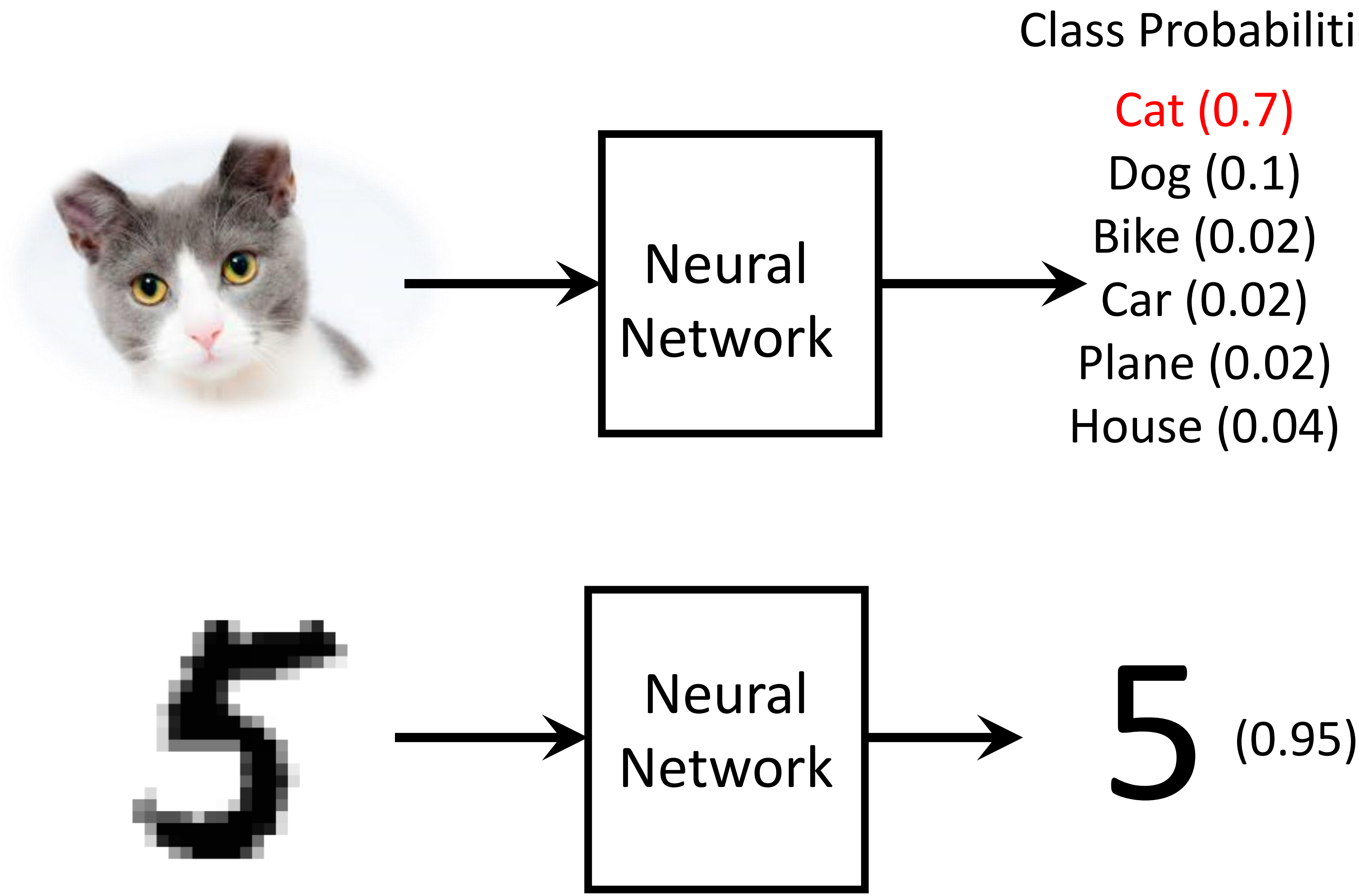
For much, much, *much* more detail, see Schmidhuber's historical overview (Neural Networks, 2015)

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

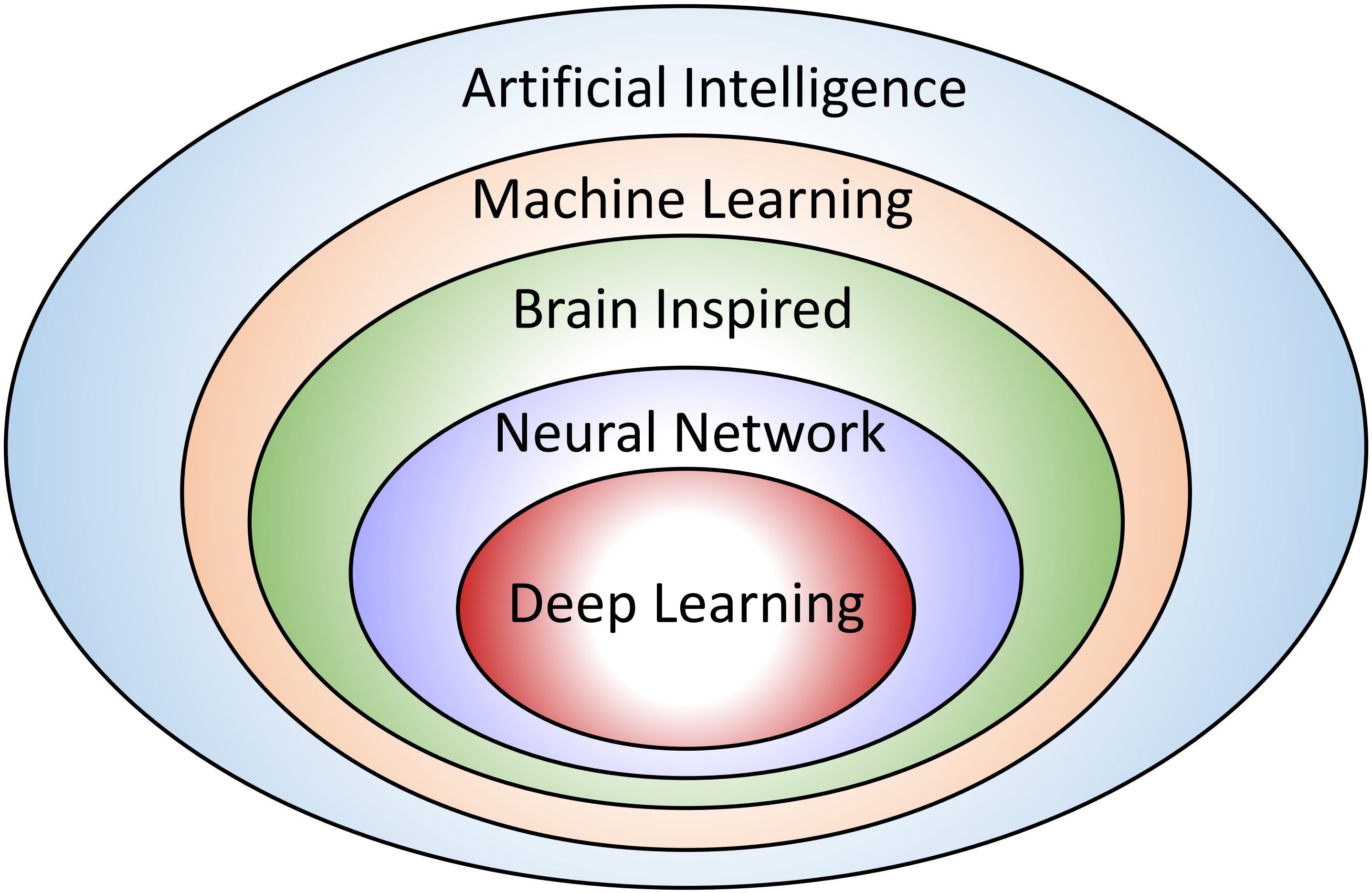
This course: PyTorch the most popular today

A Simple Image Classification Example



Deep learning is representation learning

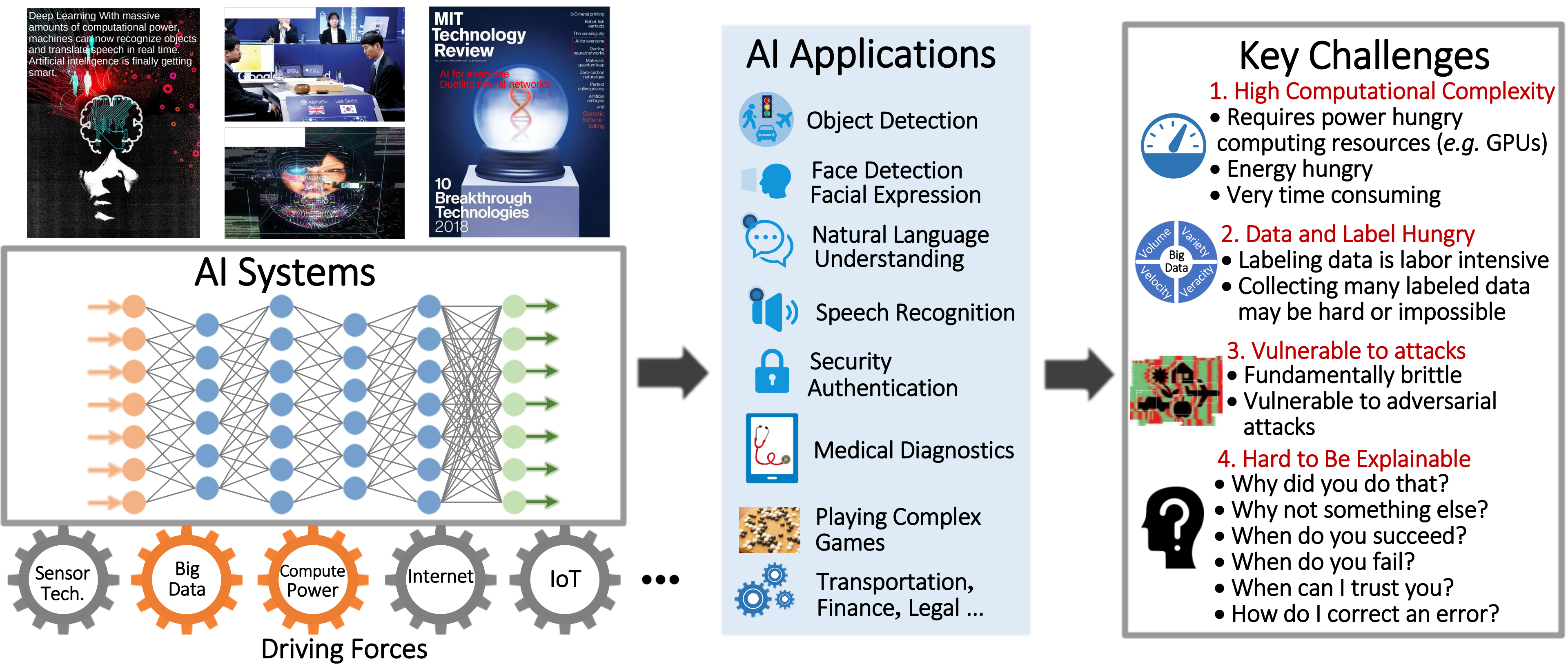
(aka feature learning)



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

Here the data consists of input and output pairs.

Challenges of Deep Learning



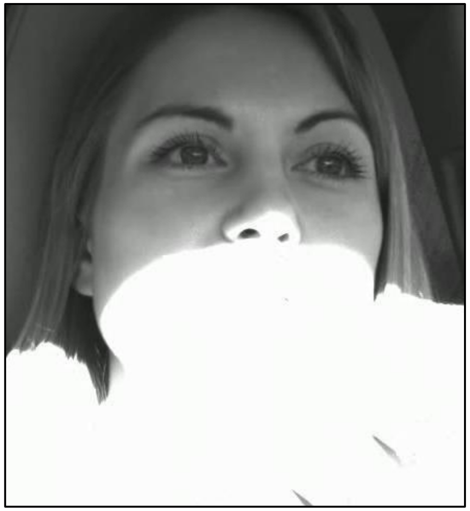
The Challenge of Deep Learning

- Ask the right question and know what the answer means:

image classification ≠ scene understanding

- Select, collect, and organize the right data to train on:

photos ≠ synthetic ≠ real world video frames



Visual Understanding is Harder

Examples of what we can’t do well:

- Who are they?
- What are they doing?
- Where are they?
- Mirrors
- 3D Structure
- What happens next?
- Emotions
- Intentions
- ……

Deep Learning
Deep Trouble
(Nature, Oct. 2019)

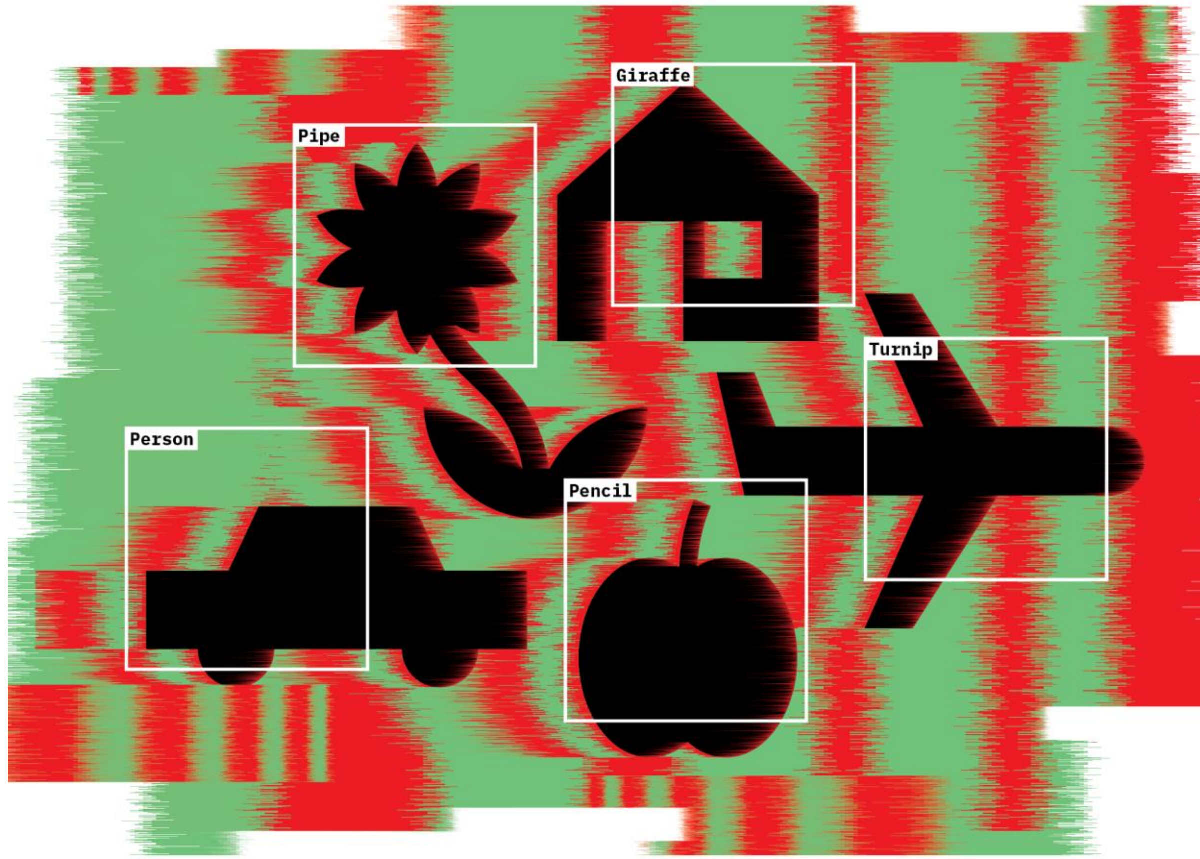


ILLUSTRATION BY EDGAR BĄK

DEEP TROUBLE FOR DEEP LEARNING

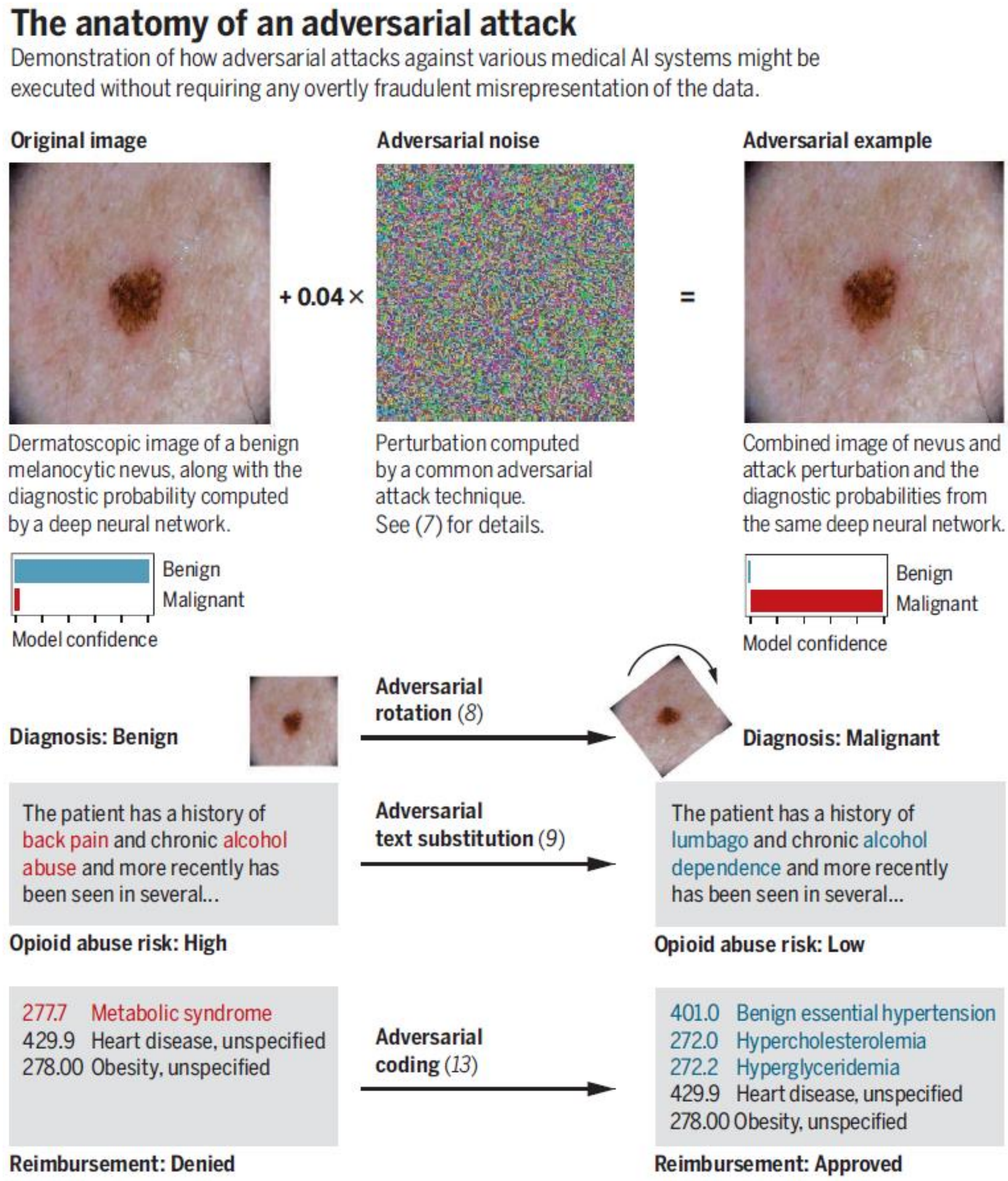
BY DOUGLAS HEAVEN

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 Challenge of Deep Learning:

Efficient Teaching + Efficient Learning

- Humans can learn from very few examples
- Machines (in most cases) need thousands/millions of examples

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

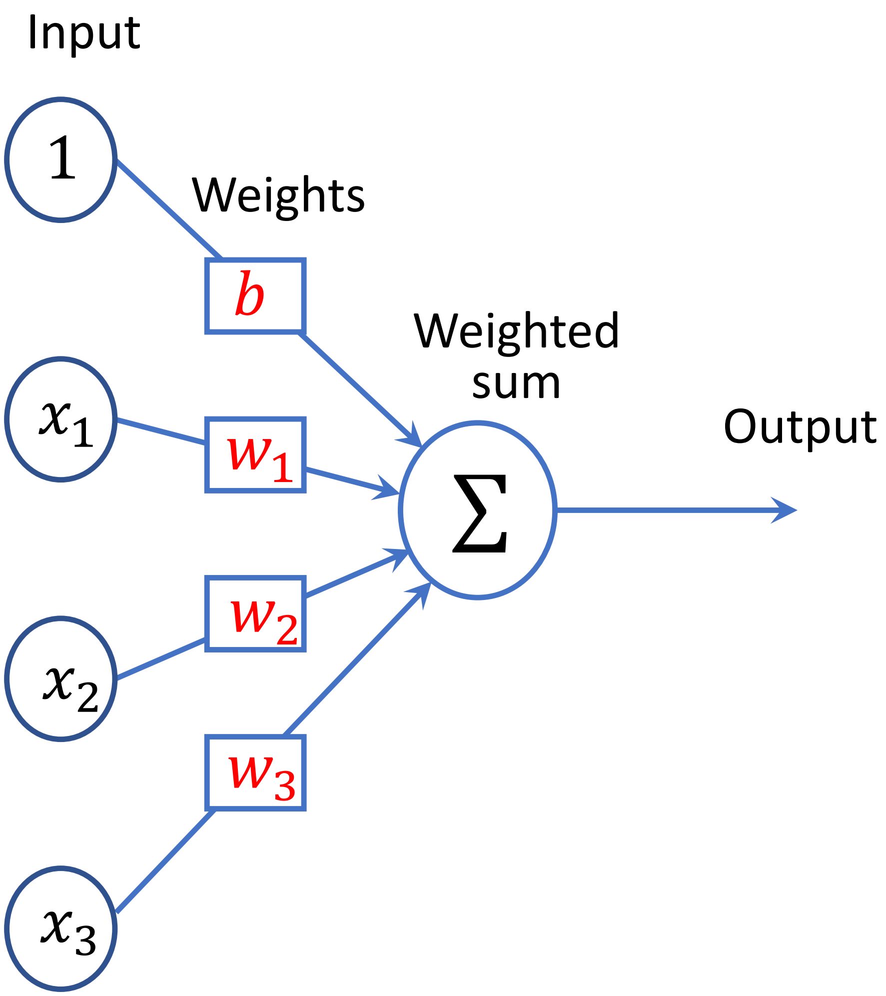
Outline

- Course Information
- Introduction to Deep Learning
- Deep Learning Basics (Linear Regression, Loss Function)

Deep Learning Basics

- Linear regression
- Model complexity
- Regularization
- Gradient descent
- 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.
- 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 $y^{(i)} \in \mathbb{R}$

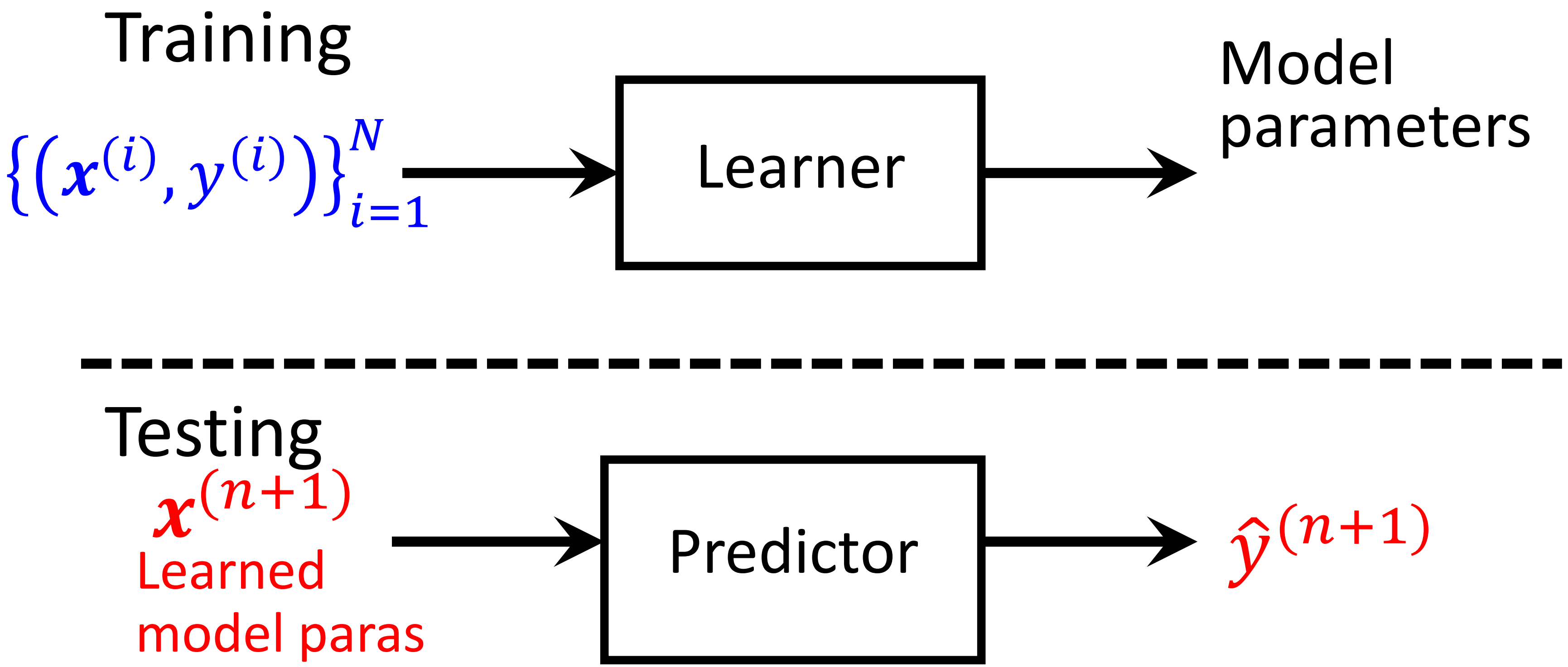
- A typical dataset with N=4 instances and n=4 variables (features)

	Size (m²)	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:

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

inner product

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

Model Representation

In matrix form, the expression for the linear model is:

$$\hat{\boldsymbol{y}} = \mathbf{X}\boldsymbol{w} + b \cdot \mathbf{1}_{N\times 1}$$

With $\hat{\boldsymbol{y}} \in \mathbb{R}^n$, $\mathbf{X} \in \mathbb{R}^{N\times n}$, $\boldsymbol{w} \in \mathbb{R}^n$ and b is the bias term, (for the given dataset)

$$\begin{bmatrix} \hat{y}^{(1)} \\ \vdots \\ \hat{y}^{(N)} \end{bmatrix} = \begin{bmatrix} x_1^{(1)} & \cdots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(N)} & \cdots & x_n^{(N)} \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} + \begin{bmatrix} b \\ \vdots \\ b \end{bmatrix}$$

$$\mathbf{1}_{N\times 1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{N\times 1}$$

Example: N=4 instances, n=4 features

$$\hat{\boldsymbol{y}} = \mathbf{X}\boldsymbol{w} + b \cdot \mathbf{1}_{N\times 1}$$

Size (m^2)	Number of bedrooms	Number of floors	Age of home (years)	Price (*1000$)
180	4	2	45	460
140	3	1	40	232
120	3	1	30	315
90	2	1	36	178

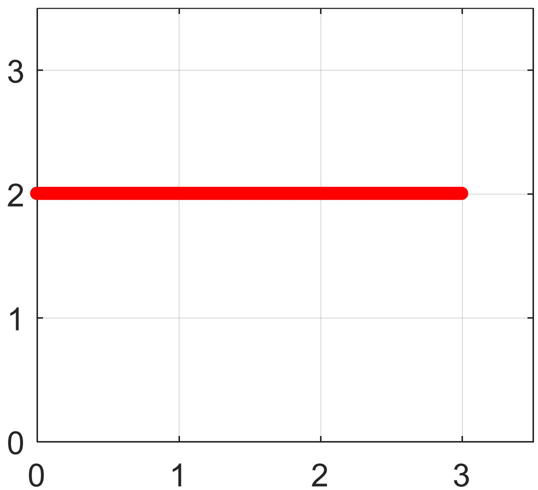
$$\mathbf{X} = \begin{bmatrix} 180 & 4 & 2 & 45 \\ 140 & 3 & 1 & 40 \\ 120 & 3 & 1 & 30 \\ 90 & 2 & 1 & 36 \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

Linear Hypothesis

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

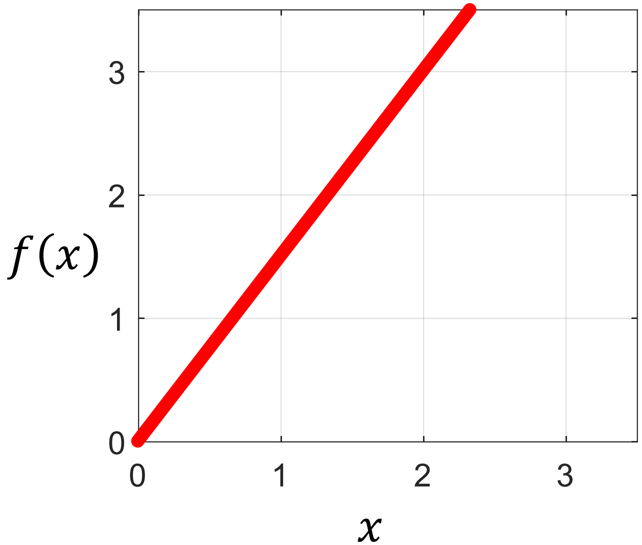
$f(x) = 2 + 0 \cdot x$



$b = 2$

$w = 0$

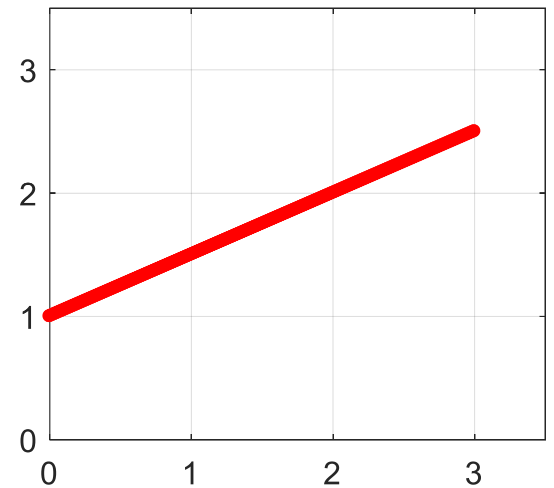
$f(x) = 0 + 1.5 \cdot x$



$b = 0$

$w = 1.5$

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



$b = 1$

$w = 0.5$

Linear Regression: Formulation

- Given: a training dataset of *N* instances of (input, output) pairs $\left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$, 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

One variable example

$\boldsymbol{w}$ and b are parameters

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

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

A set of functions

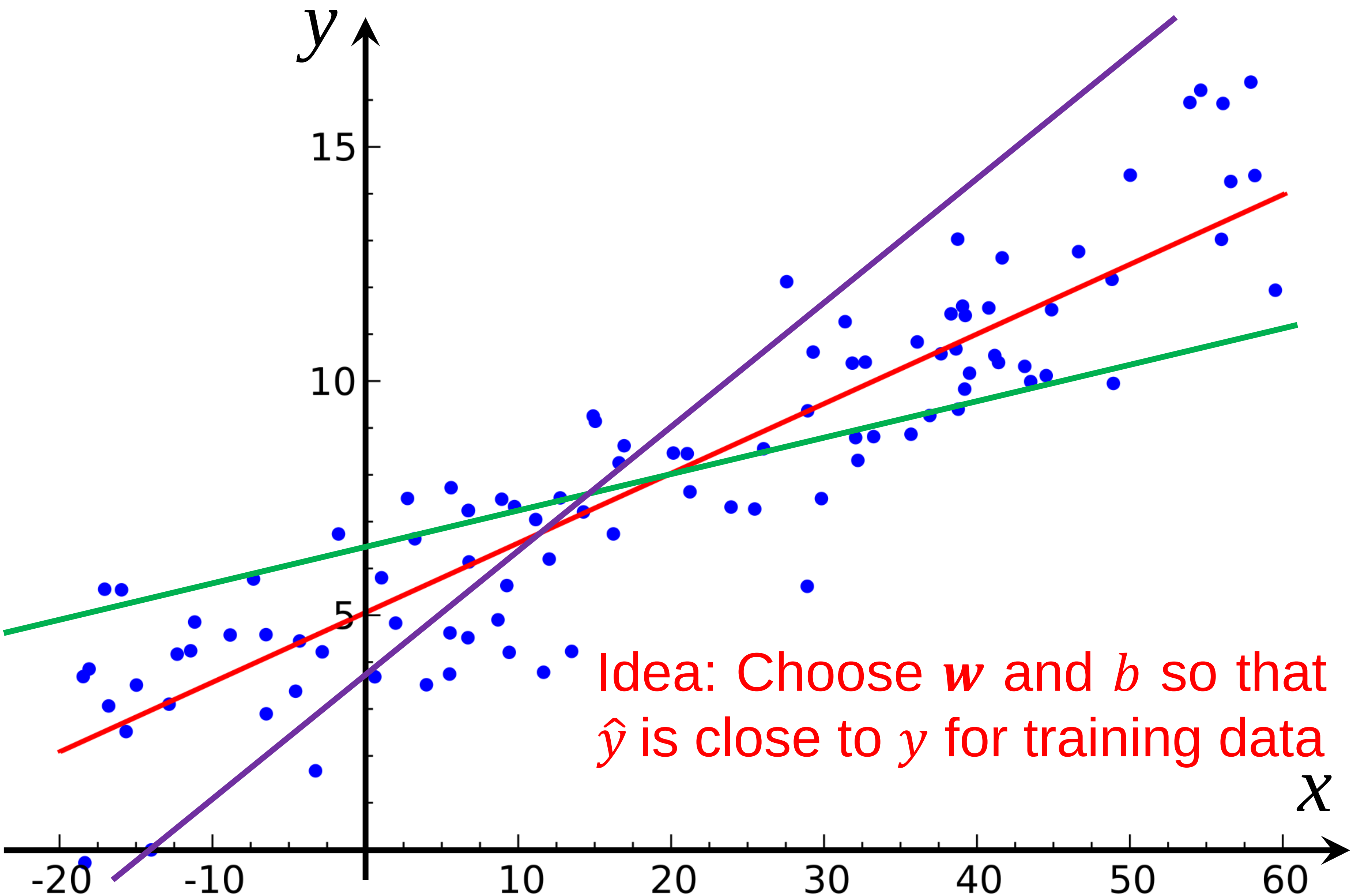
Model

$f_1, f_2 \cdots$

Linear Model:

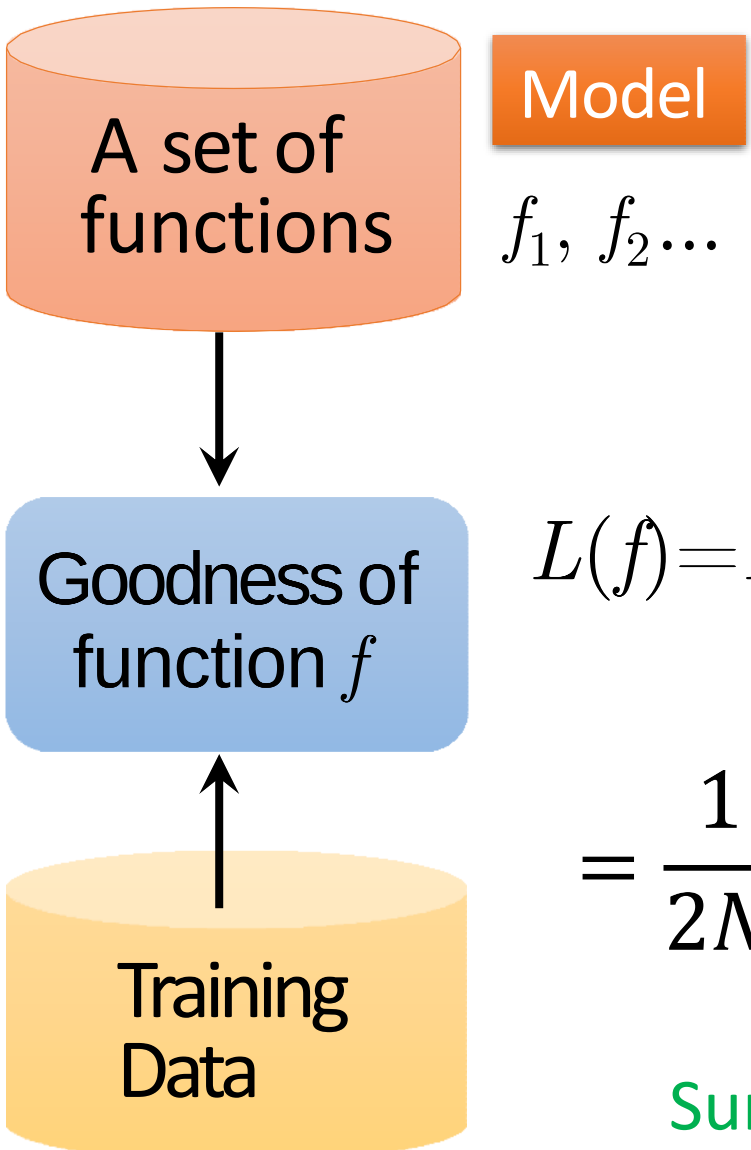
$$\hat{y} = f(\boldsymbol{x}) = b + \boldsymbol{w}^T \boldsymbol{x}$$

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


y
x
15
10
5
-20
-10
10
20
30
40
50
60
Idea: Choose $\boldsymbol{w}$ and b so that
$\hat{y}$ is close to y for training data

Step 2: Cost Function (Measure Goodness of Function)

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



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

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

$$= \underbrace{\frac{1}{2N} \sum_{i=1}^{N}}_{\text{Sum over training data}} \underbrace{\boxed{\left(y^{(i)} - \left(b + \boldsymbol{w}^T \boldsymbol{x}^{(i)}\right)\right)^2}}_{\text{Estimated error}}$$

Estimated error

Sum over training data

Now we have...

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

Parameters: $\boldsymbol{w},\ b$

Cost function:

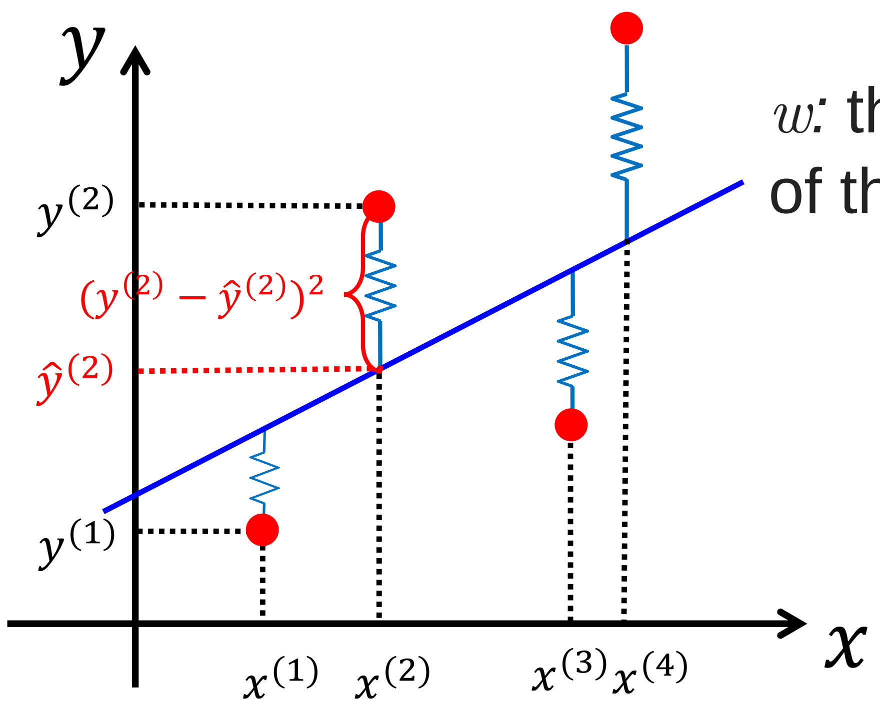
$$L(\boldsymbol{w}, b) = \frac{1}{2N} \sum_{i=1}^{N} \left(y^{(i)} - \left(b + \boldsymbol{w}^T \boldsymbol{x}^{(i)} \right) \right)^2$$

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

Cost Function: Intuitive

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

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

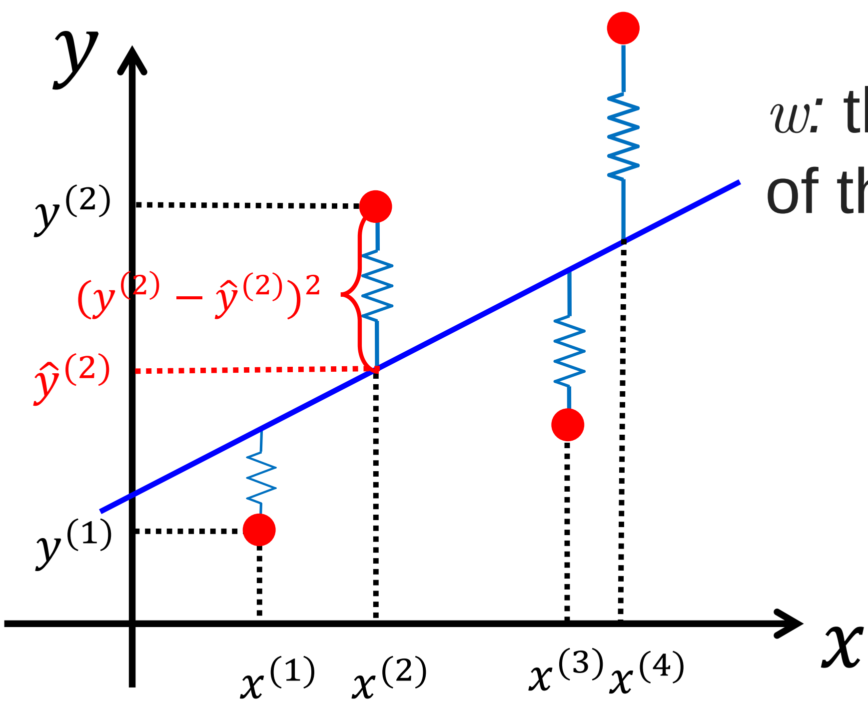


w: the slope of the fit line

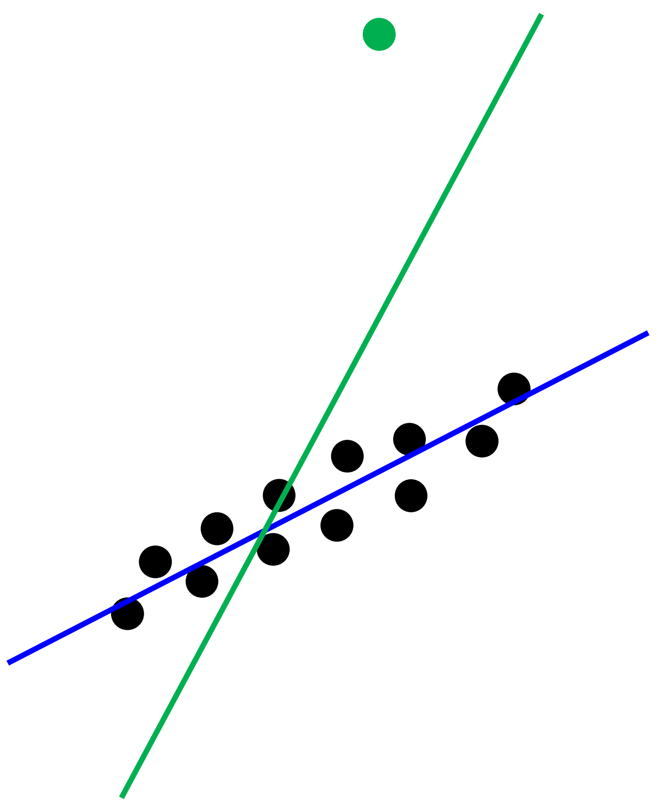
Cost Function: Intuitive

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

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



w: the slope of the fit line

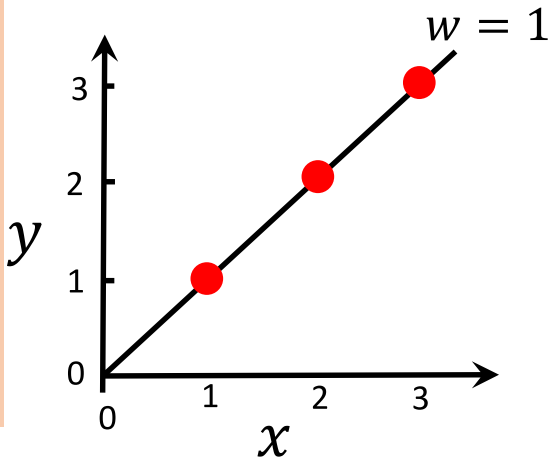


Cost Function: Intuitive

$f_w(x) = wx$

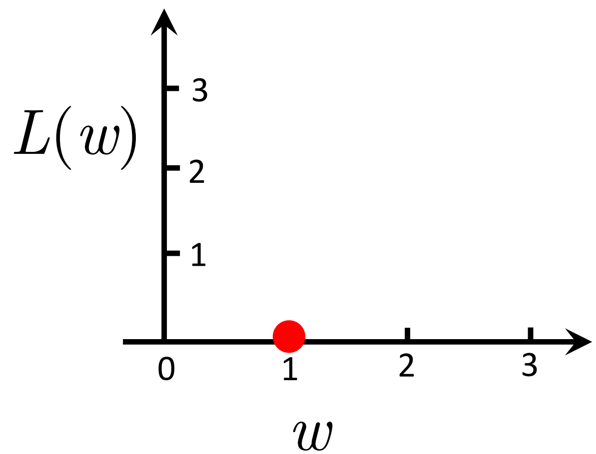
(For fixed w, this is a function of x)

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



$L(w)$

(Function of the parameter w)



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

$\boldsymbol{J(0.5) = ?}$

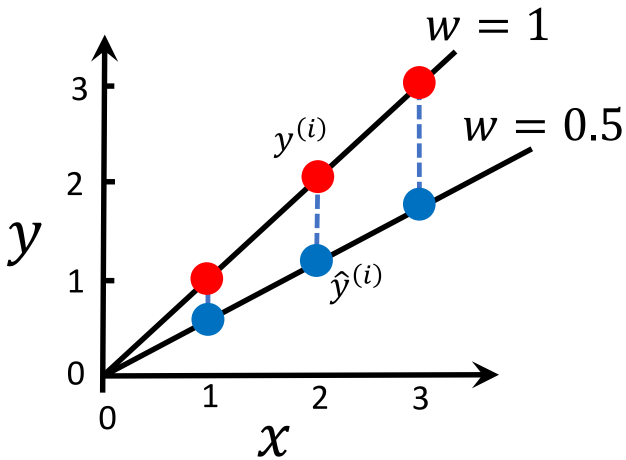
$$= \frac{1}{2N}\sum_{i=1}^{N}\left(wx^{(i)} - y^{(i)}\right)^2 = \frac{1}{2m}(0^2 + 0^2 + 0^2) = 0$$

Adapted from Andrew Ng

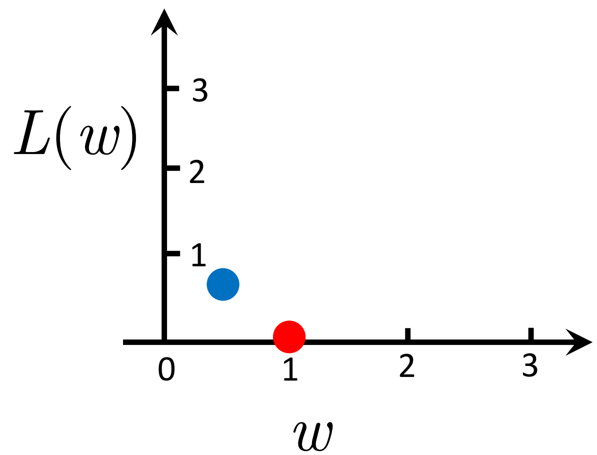
Cost Function: Intuitive

$f_w(x)$ $L(w)$

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



(Function of the parameter w)



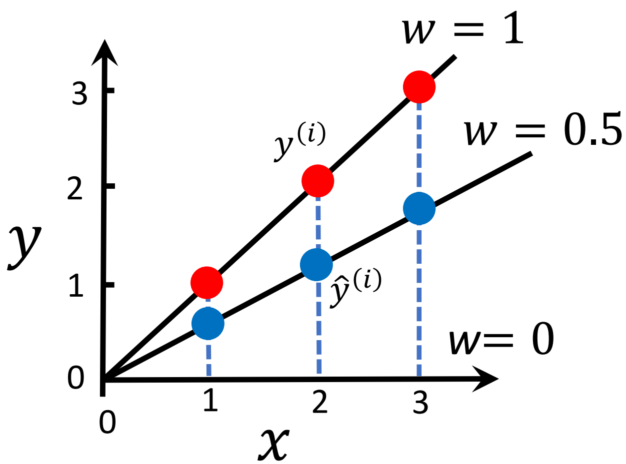
$$L(0.5) = \frac{1}{2N}\left((0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right)$$

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

Cost Function: Intuitive

$f_w(x)$

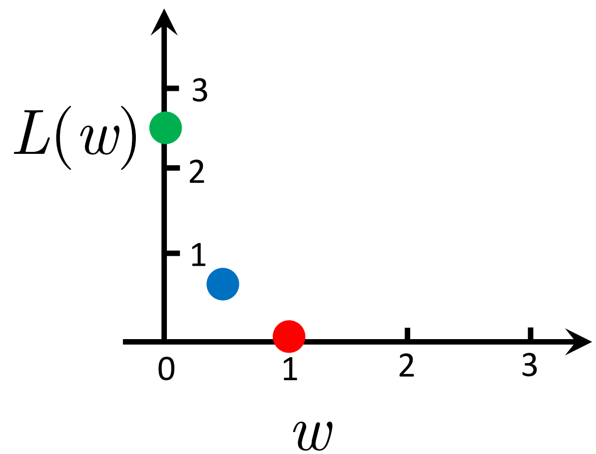
(For fixed w, this is a function of x)



$$L(0) = \frac{1}{2N}\left((1)^2 + (2)^2 + (3)^2\right)$$
$$= \frac{1}{2 \times 3}(14) \approx 2.3$$

$L(w)$

(Function of the parameter w)

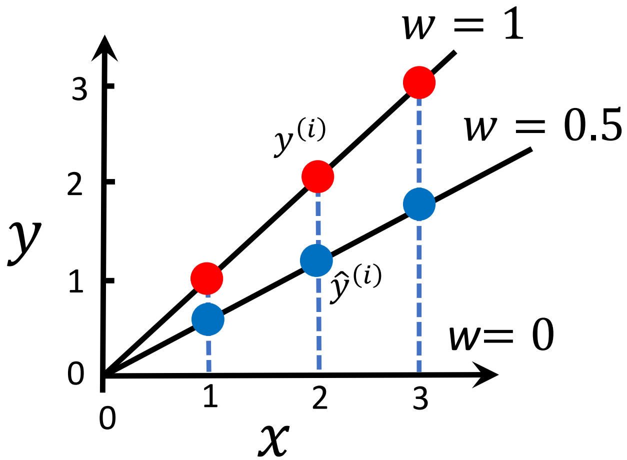


$$\min_{w} L(w)$$

Cost Function: Intuitive

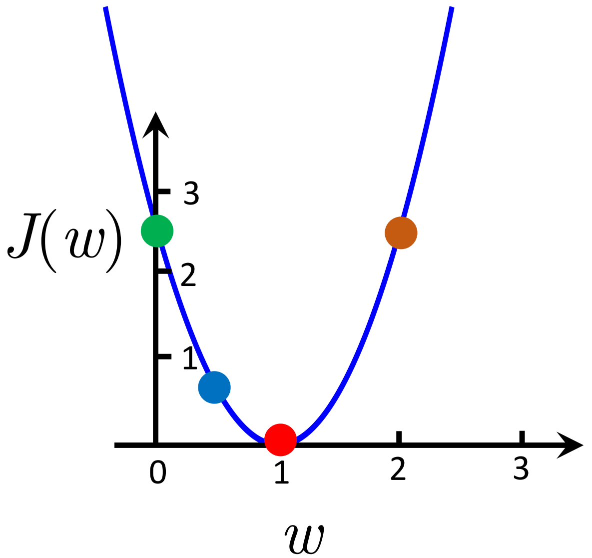
$f_w(x)$

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



$L(w)$

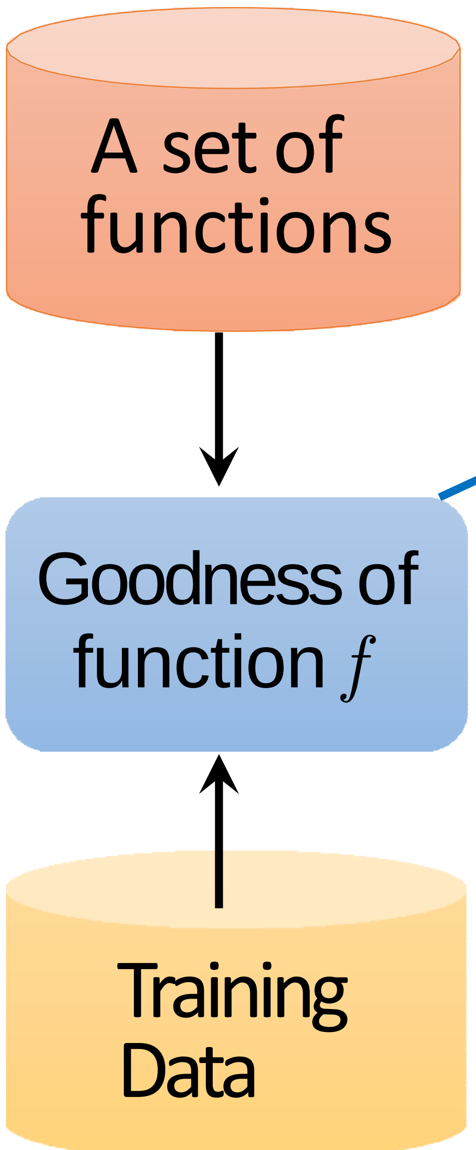
(Function of the parameter w)



$$\min_{w} L(w)$$

Step 3: Minimize Cost (Find Best Function)

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



Model

$f_1,\ f_2 \ldots$

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

Pick the "Best" Function

$$\boldsymbol{w}^*, b^* = \underset{\boldsymbol{w}, b}{\arg\min}\, L(\boldsymbol{w}, b)$$

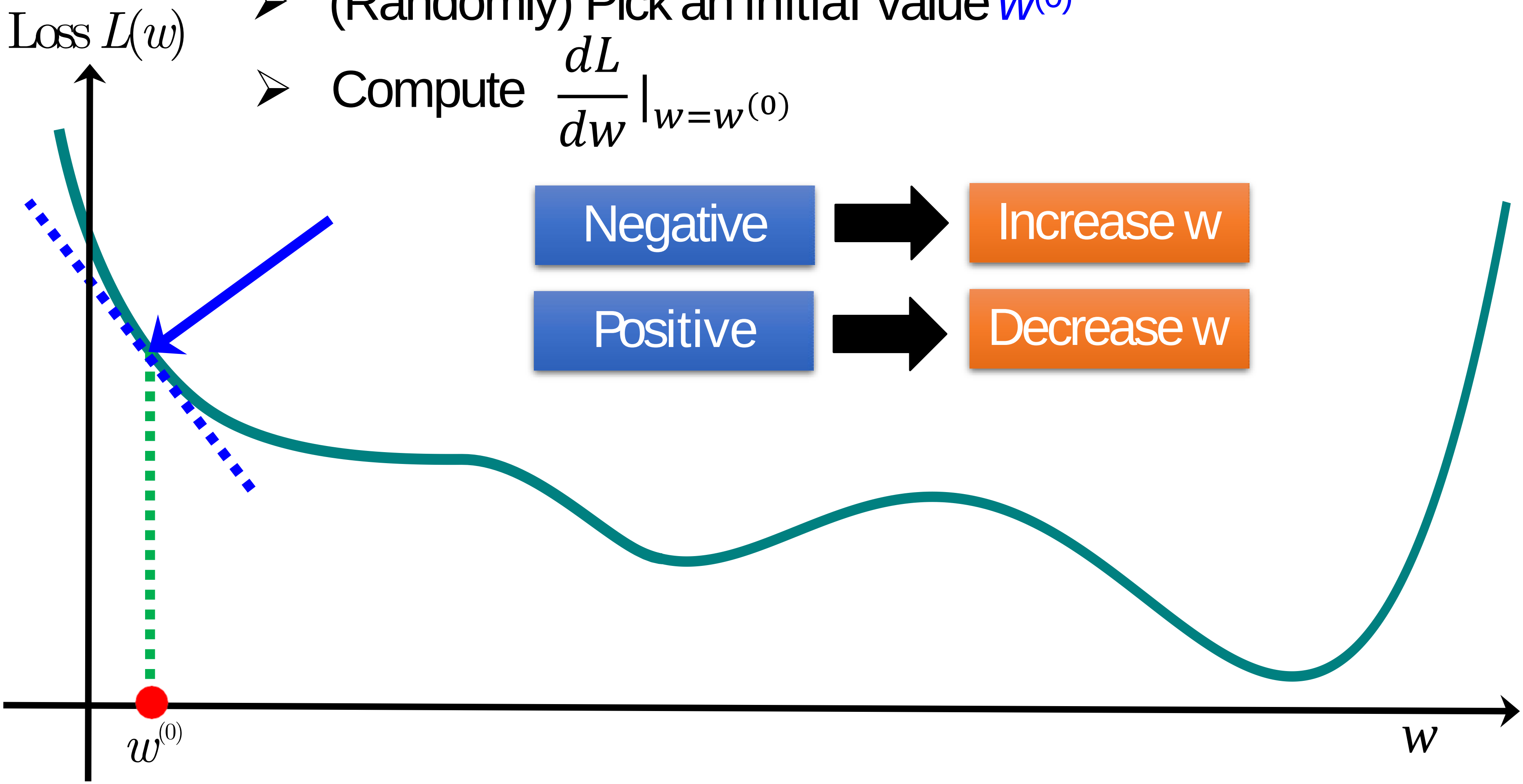
$$= \underset{\boldsymbol{w}, b}{\arg\min}\, \frac{1}{2N} \sum_{i=1}^{N} \left(y - (b + \boldsymbol{w}^T \boldsymbol{x}_i)\right)^2$$

How? Gradient Descent

Gradient Descent

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

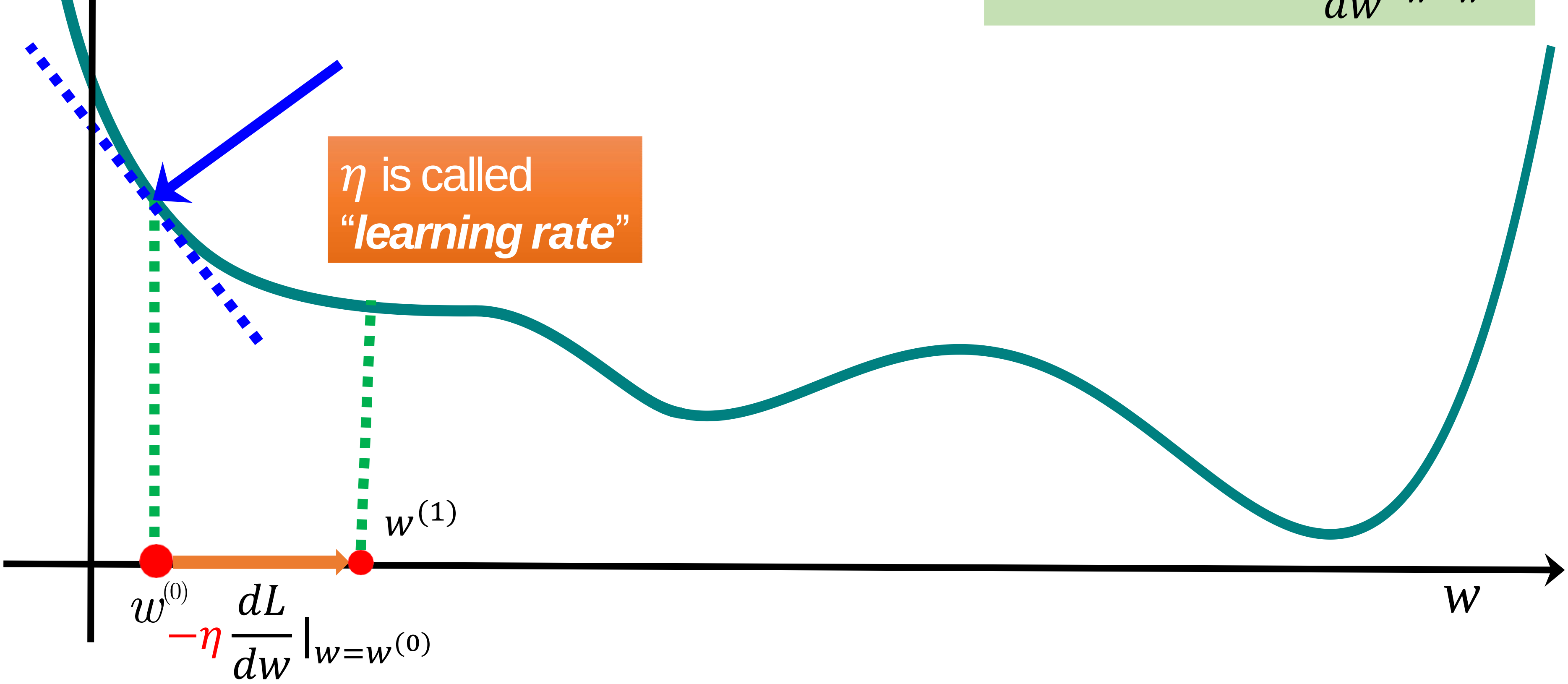
- Consider loss function $L(w)$ with one parameter w:
  - (Randomly) Pick an initial value $w^{(0)}$
  - Compute $\frac{dL}{dw}|_{w=w^{(0)}}$



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 $\frac{dL}{dw}|_{w=w^{(0)}}$



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 $\frac{dL}{dw}|_{w=w^{(1)}}$

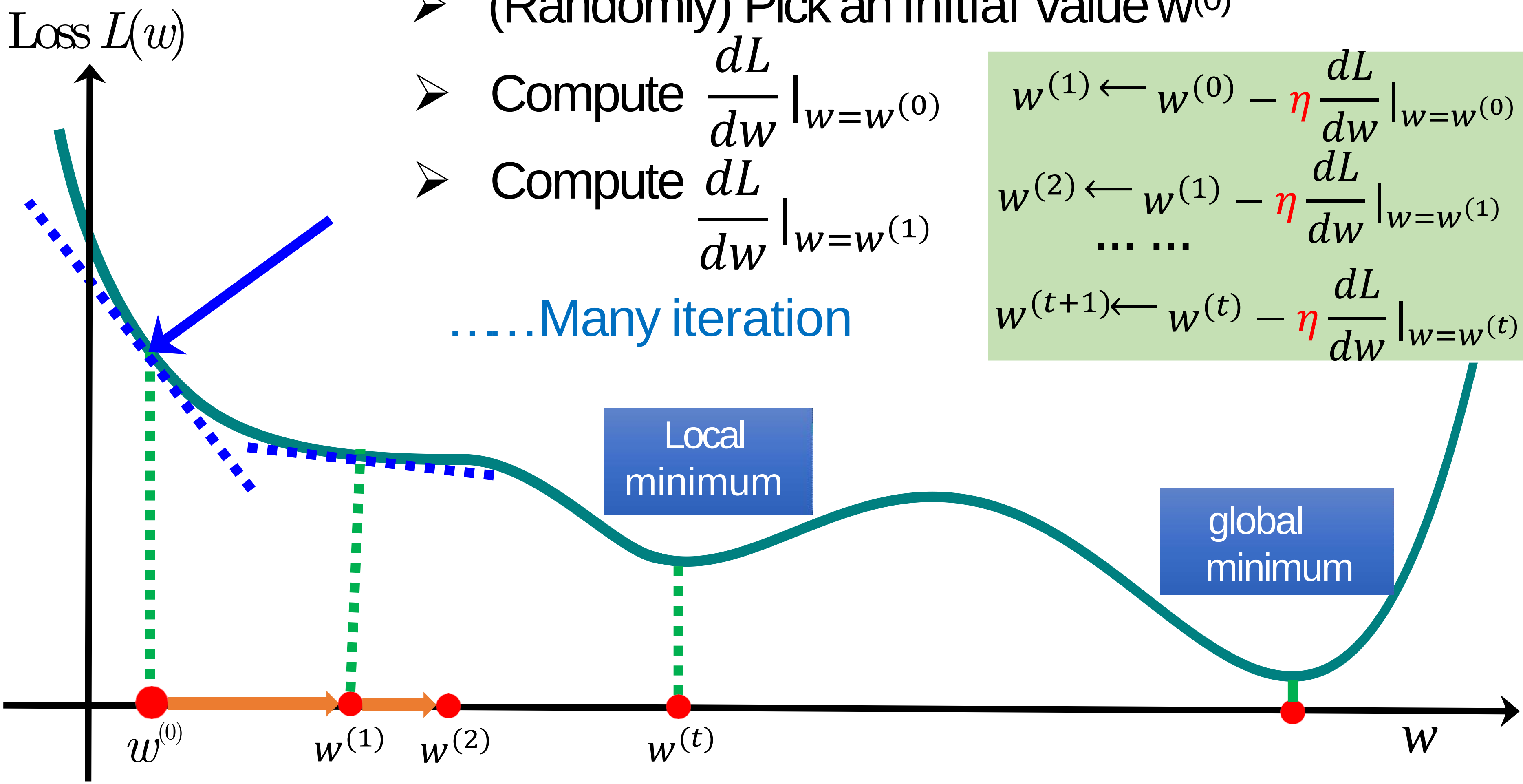
……Many iteration

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

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

… …

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



Gradient Descent

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

- 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

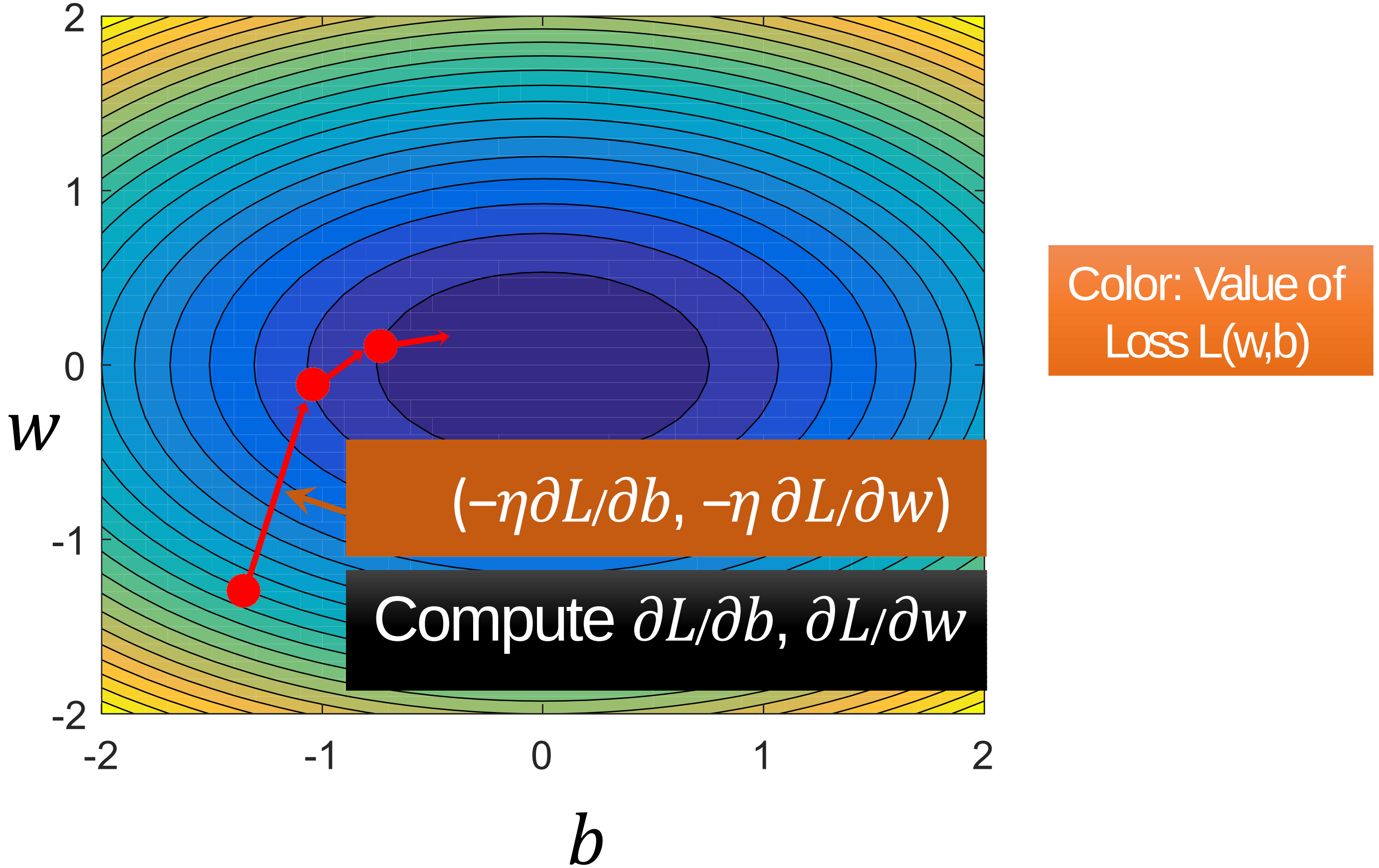
$$\frac{\partial L}{\partial w}|_{w=w^{(0)},b=b^{(0)}} \quad , \quad \frac{\partial L}{\partial b}|_{w=w^{(0)},b=b^{(0)}}$$

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

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

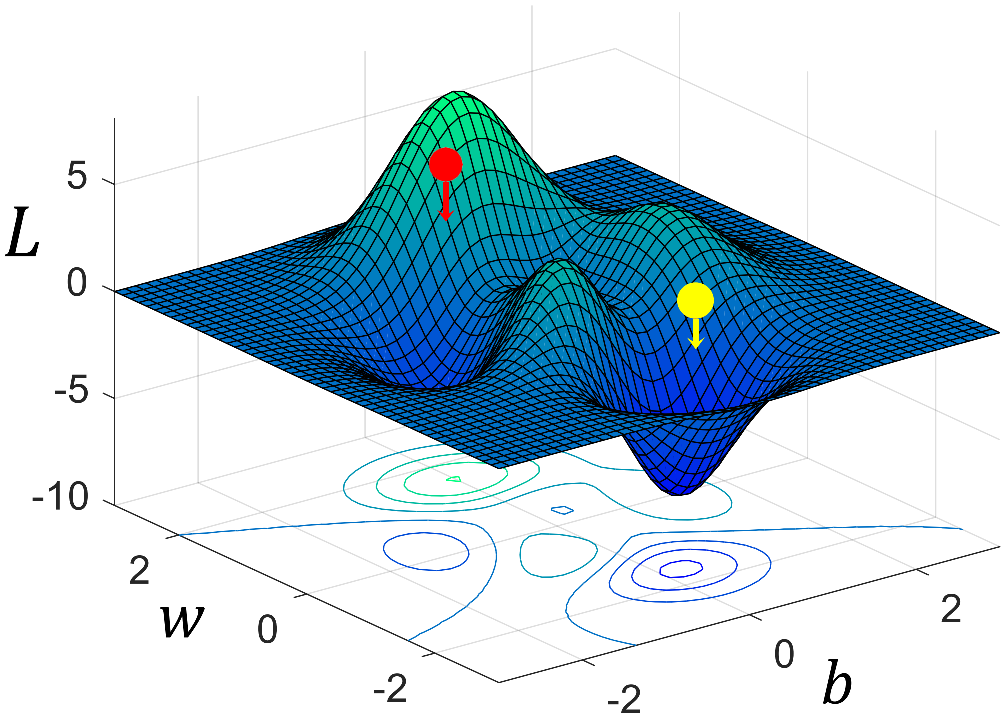
$$w^{(2)} \leftarrow w^{(1)} - \eta \frac{\partial L}{\partial w}|_{w=w^{(1)},b=b^{(1)}} \quad , \quad b^{(2)} \leftarrow b^{(1)} - \eta \frac{\partial L}{\partial b}|_{w=w^{(1)},b=b^{(1)}}$$

Gradient Descent



Step 3: Gradient Descent for Linear Regression

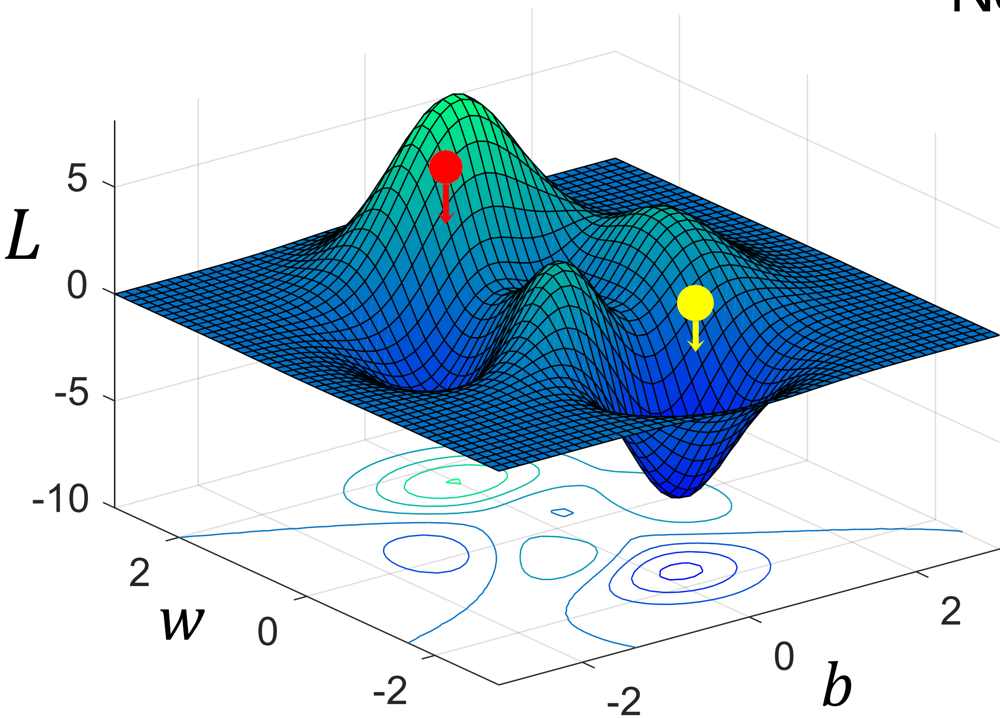
- Worry?

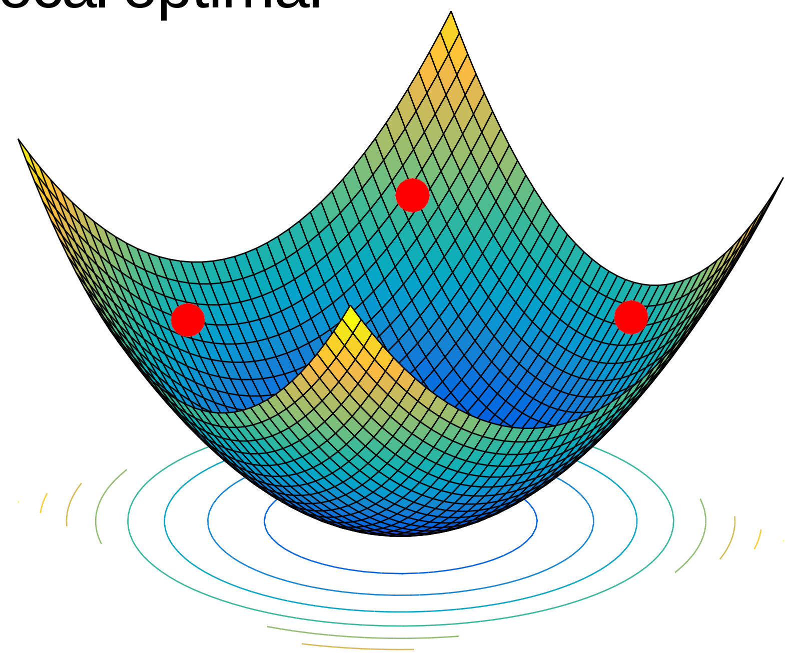


Step 3: Gradient Descent for Linear Regression

- 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 $\frac{\partial L}{\partial b}$

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

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

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

Step 3: Gradient Descent for Linear Regression

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

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

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

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

How are the results?

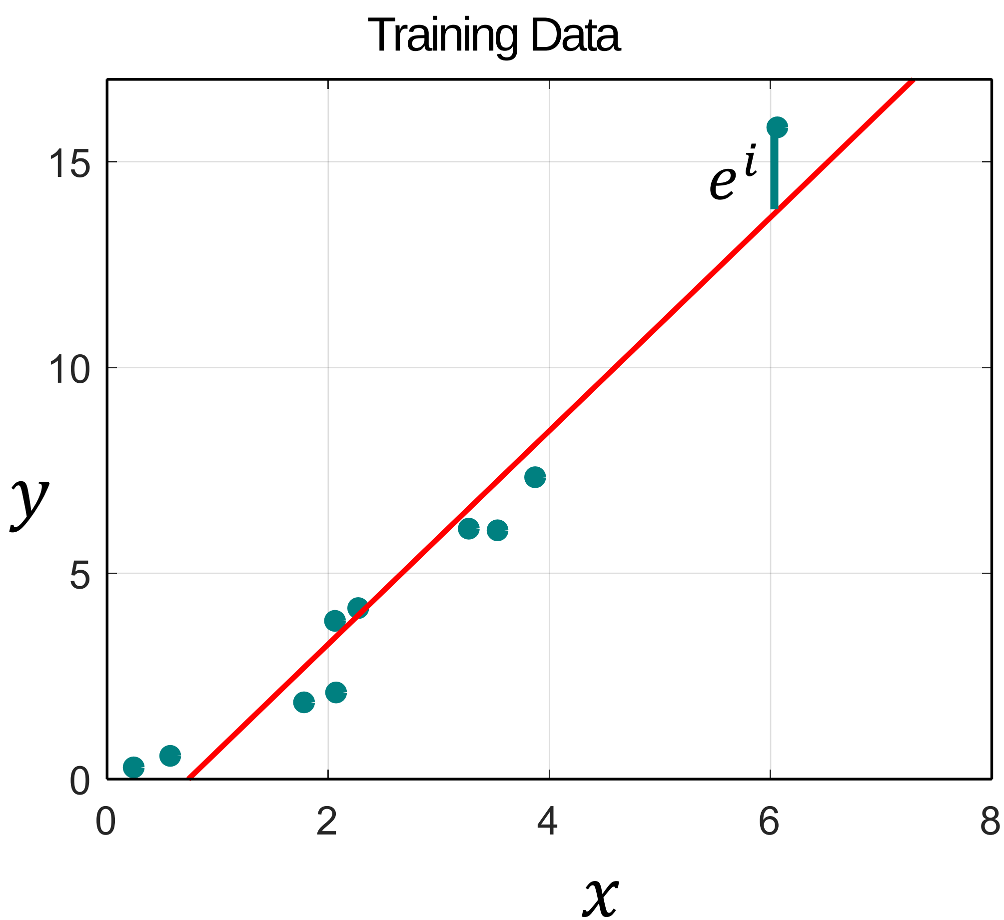
$$y = b + w \cdot x$$

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

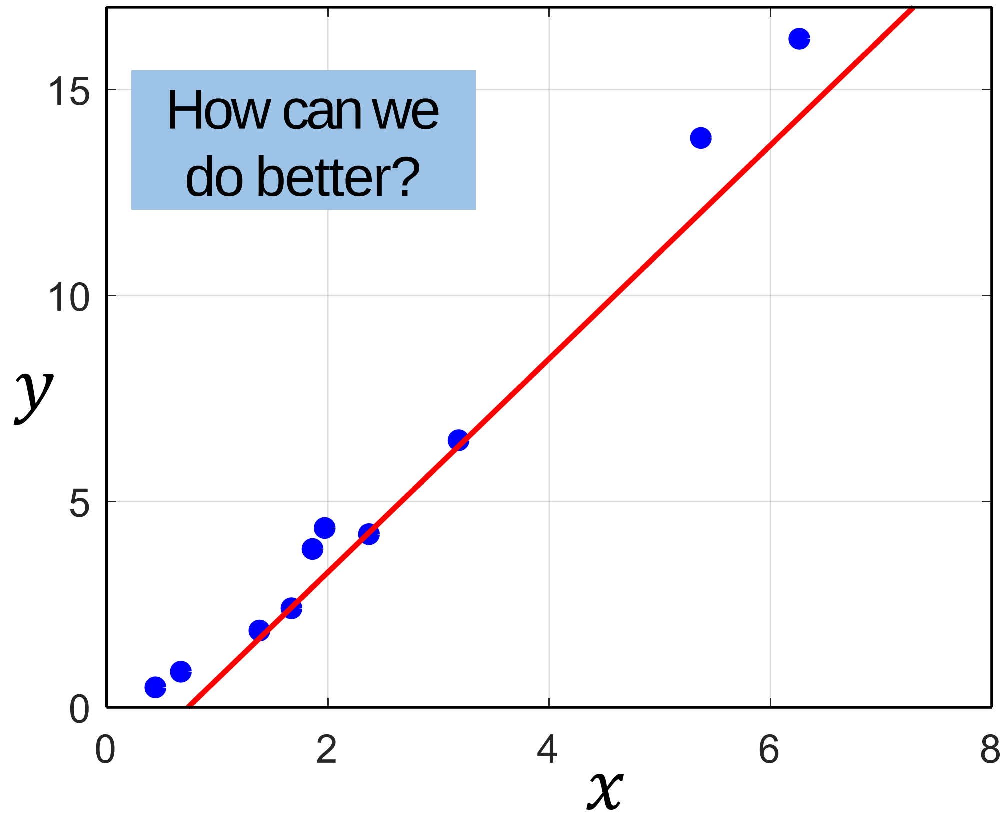
w = 2.59

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

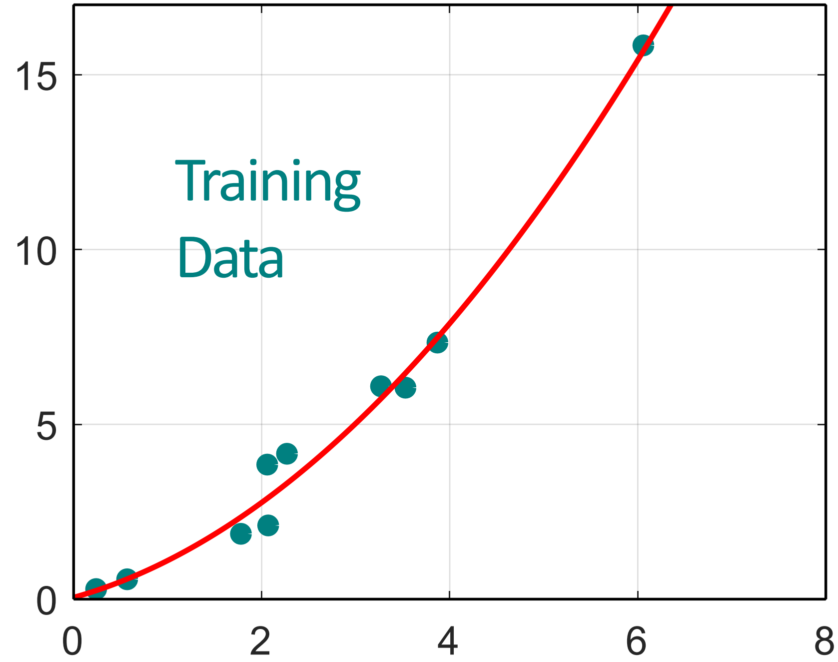
$$y = b + w_1 \cdot x + w_2 \cdot x^2$$

Best Function

$b = 0.05$

$w_1 = 0.76$, $w_2 = 0.30$

Average Error = 0.53

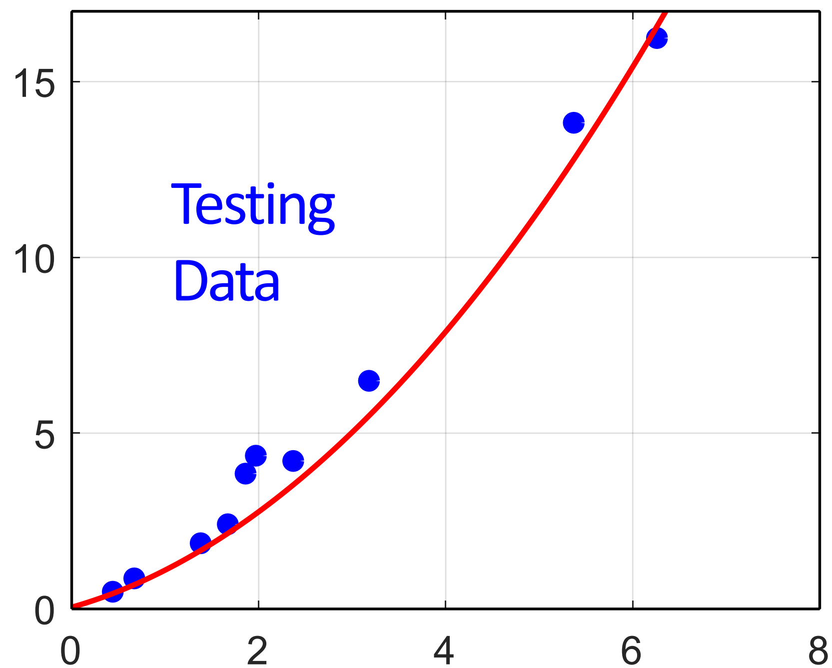


Testing Results

Average Error = 0.82

Better!

Could it be even better?



Selecting another Model

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

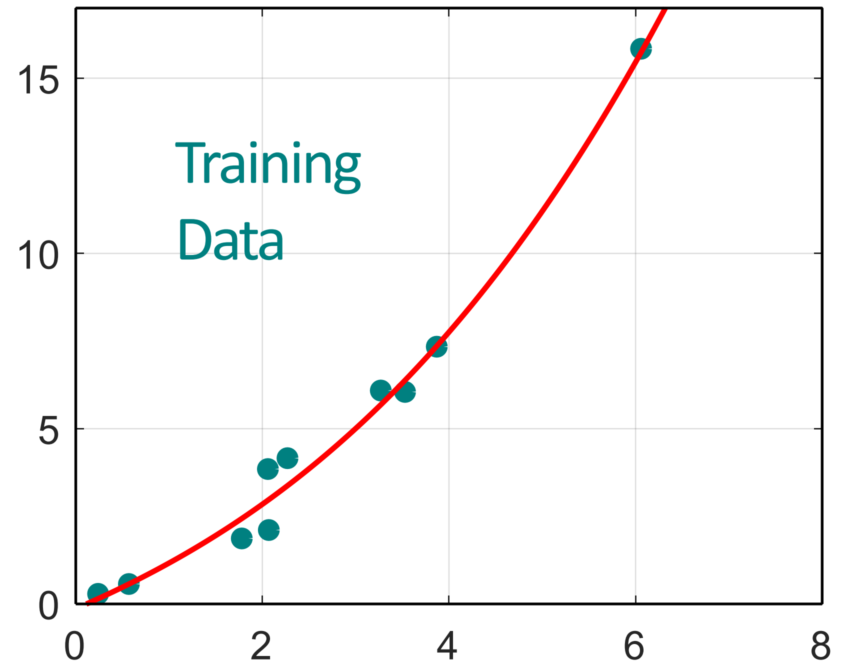
Best Function

b = -0.13

w_1 = 1.15, w_2 = 0.13

w_3 = 0.018

Average Error = 0.52

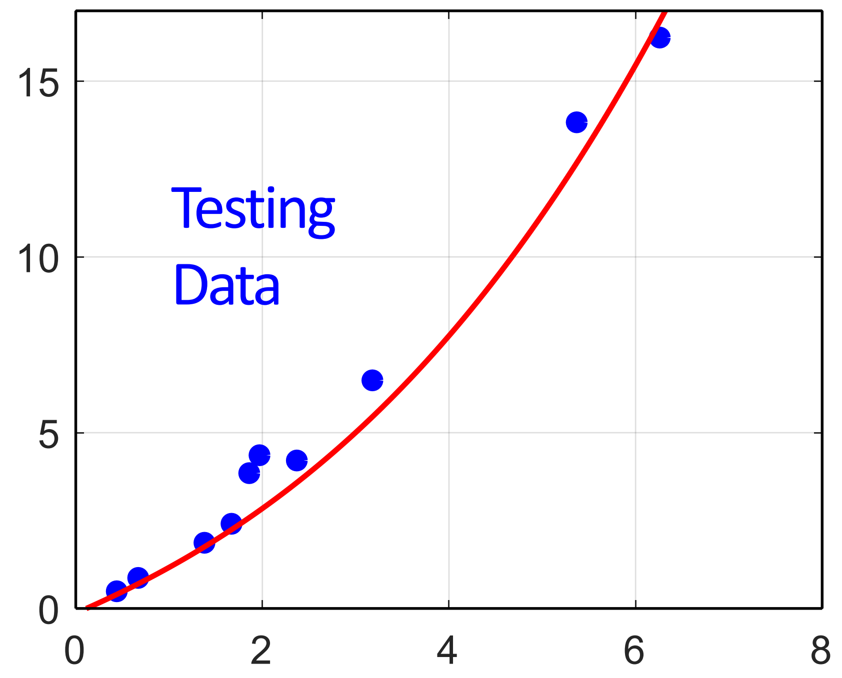


Testing Results

Average Error = 0.81

Slightly better.

How about more complex model?

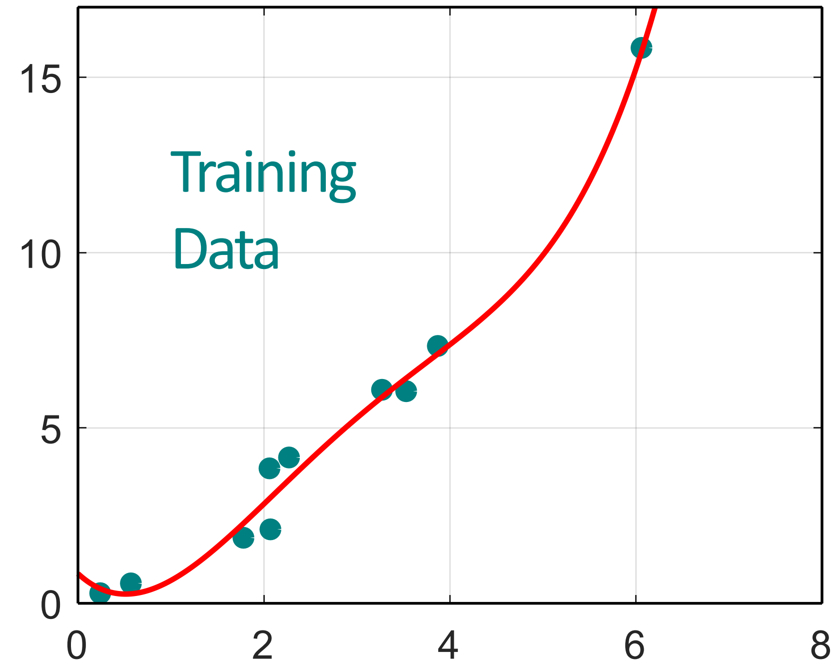


Selecting another Model

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

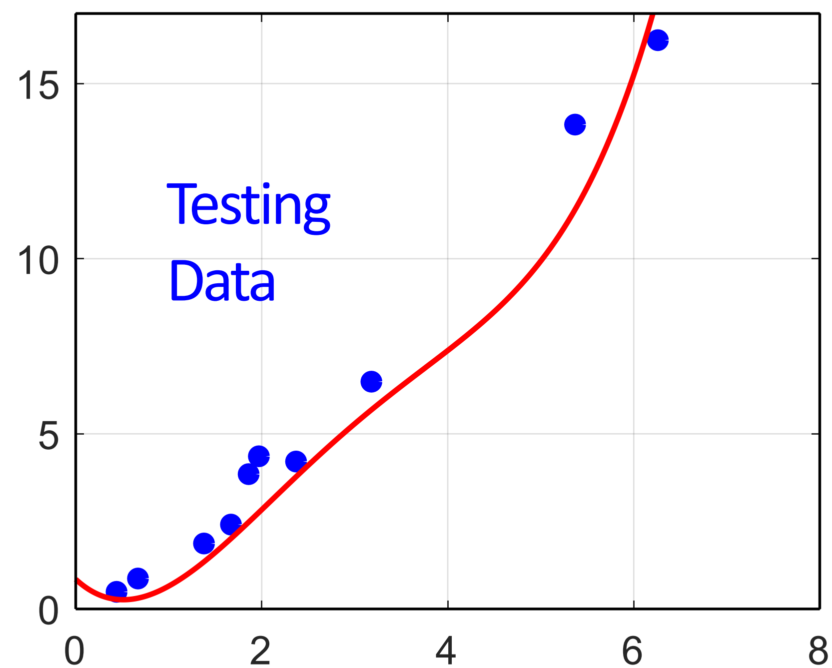
Best Function

$b = 0.85$
$w_1 = -2.45$, $w_2 = 2.91$
$w_3 = 0.72$, $w_4 = 0.06$
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

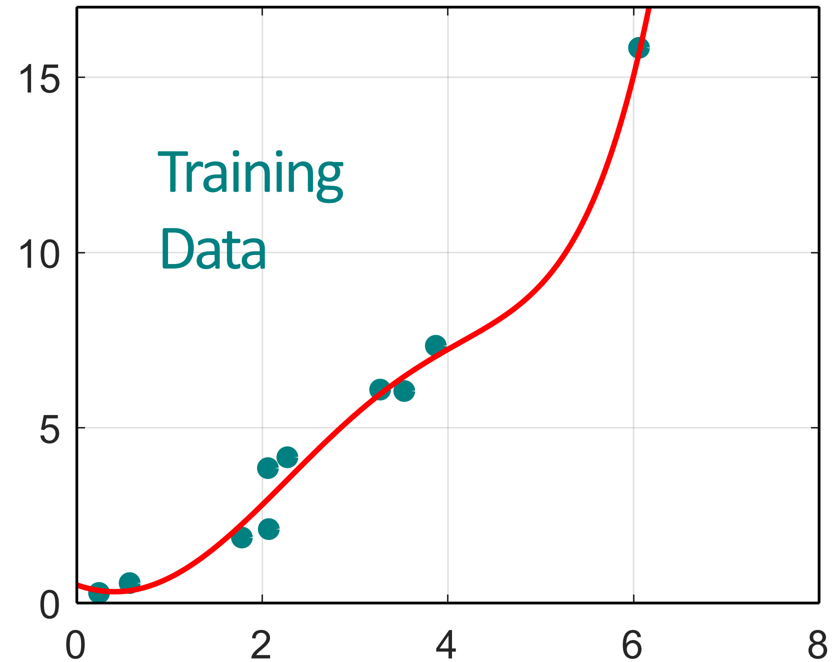
$b = 0.52$

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

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

$w_5 = -0.01$

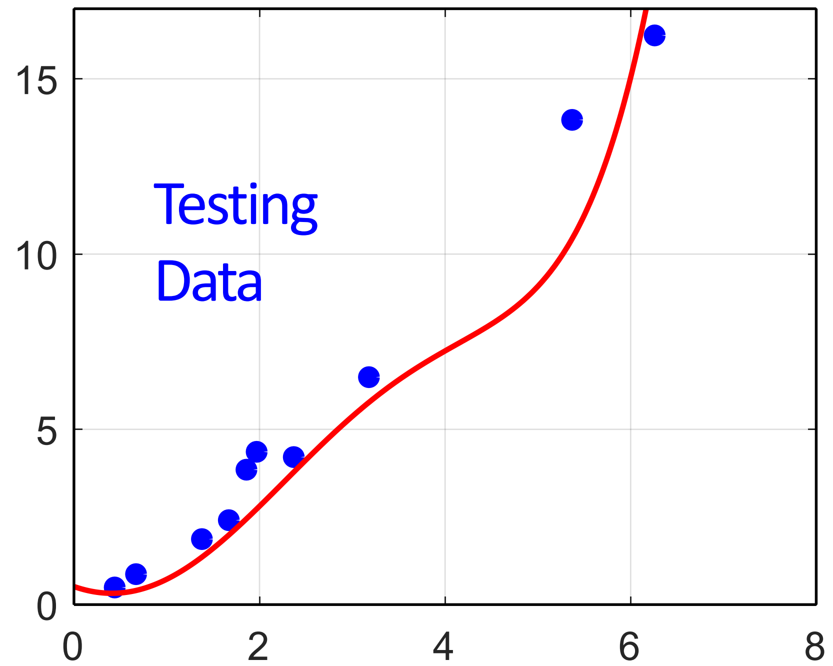
Average Error = 0.49



Testing Results

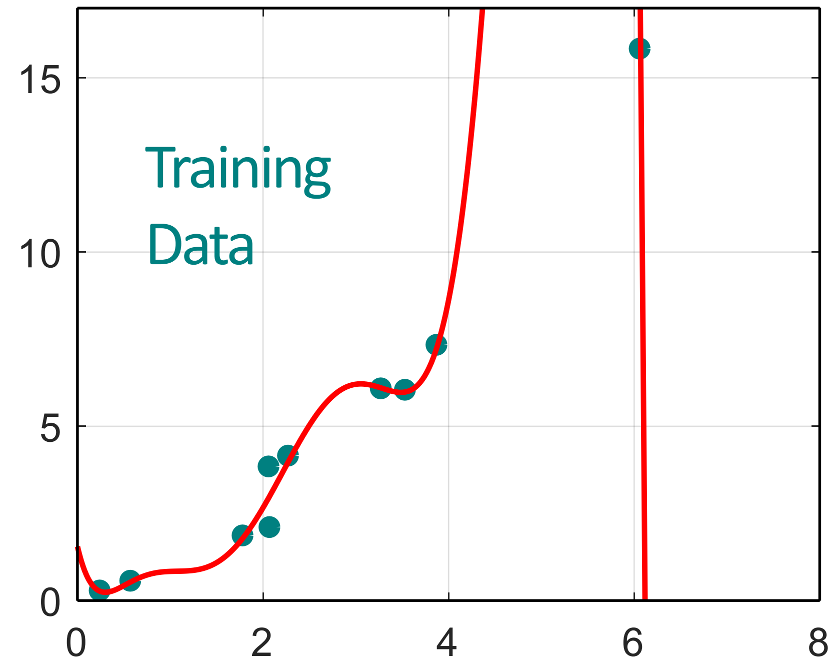
Average Error = 1.47

Results become even worse.



Selecting another Model

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



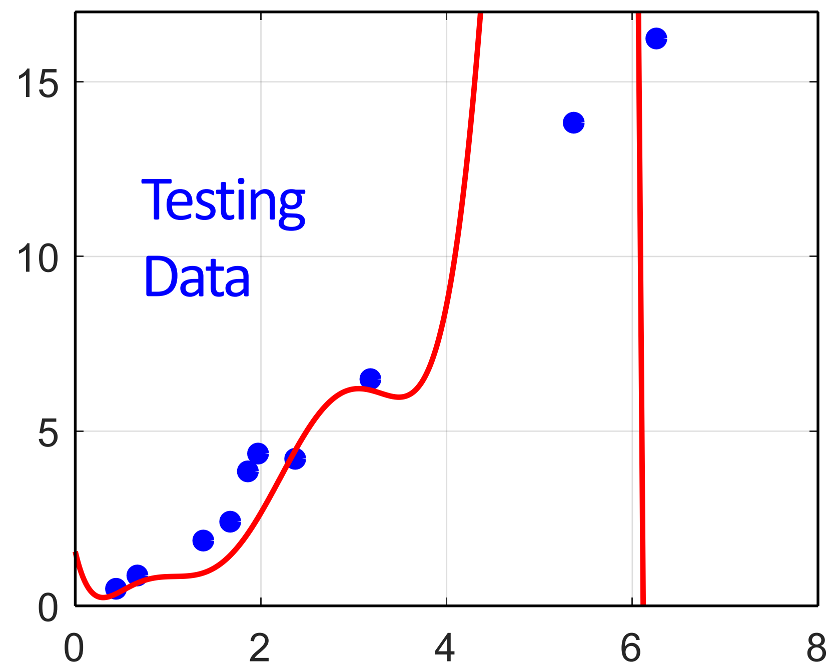
Best Function

Average Error = 0.40

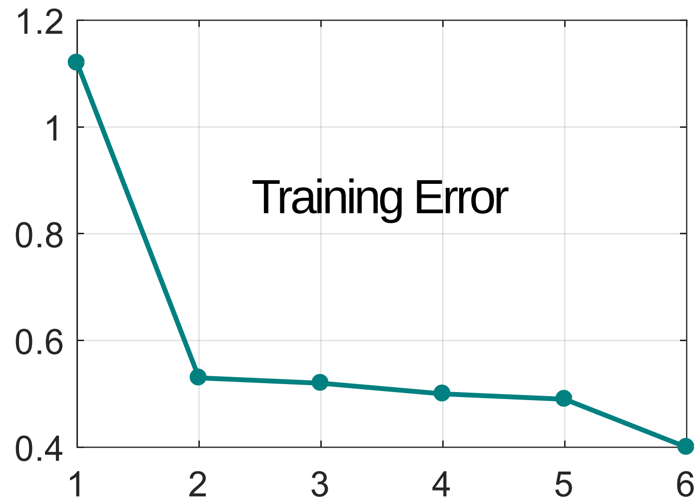
Testing Results

Average Error = 33.05

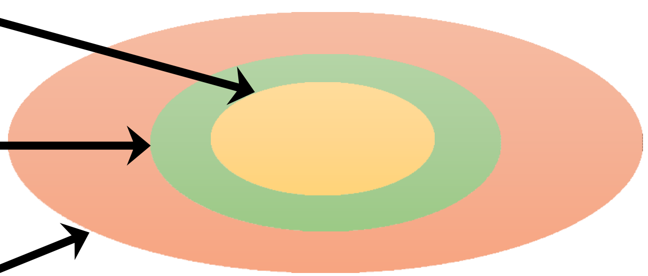
Results become so bad.



Model Selection



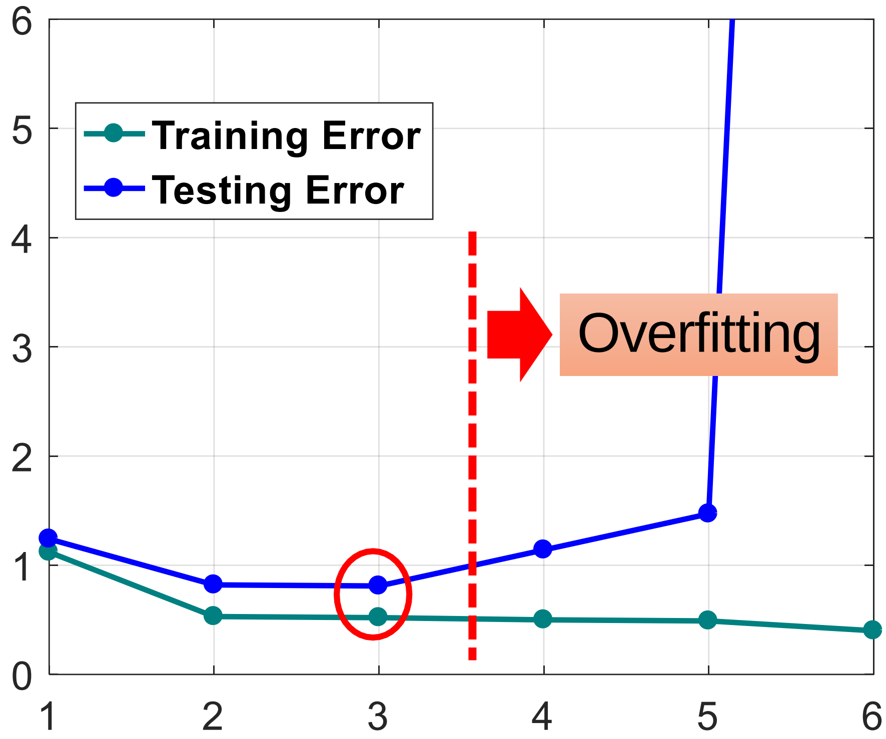
1. $y = b + w \cdot x$
2. $y = b + w_1 \cdot x + w_2 \cdot x^2$
3. $y = b + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3$
4. $y = b + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3 + w_4 \cdot x^4$
5. $y = b + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3 + w_4 \cdot x^4 + w_5 \cdot x^5$
6. $y = b + w_1 \cdot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



A more complex model yields lower error on training data.

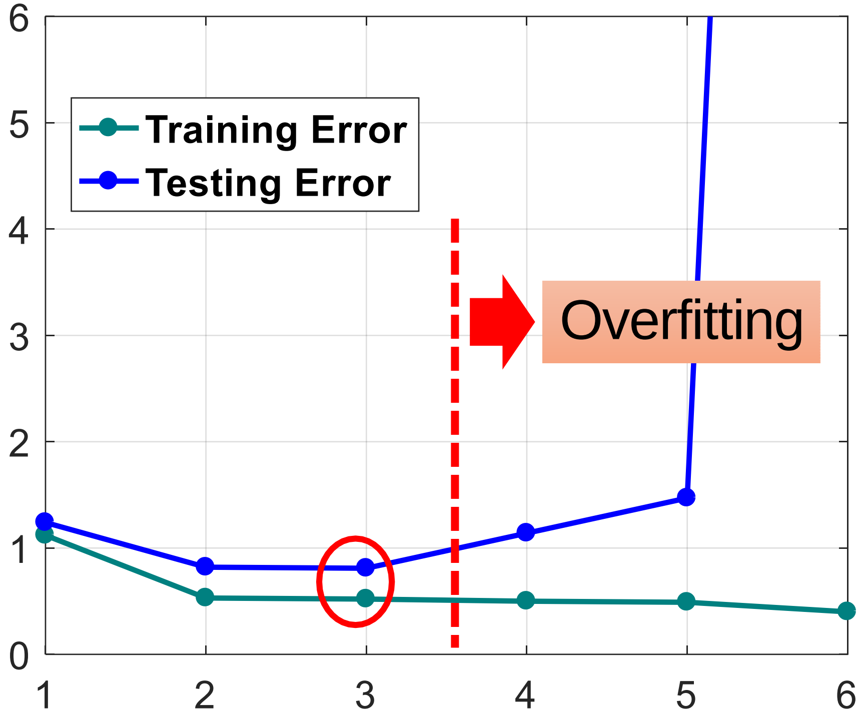
If we can truly find the best function

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

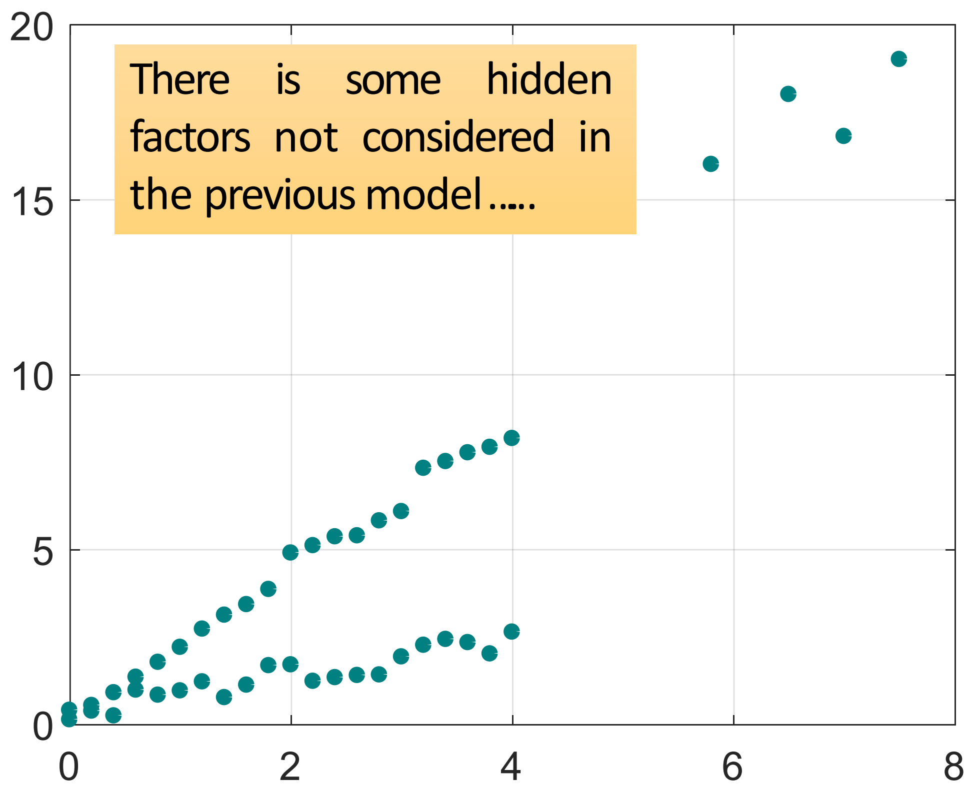


	Training Error	Testing Error
1	1.12	1.24
2	0.53	0.82
3	0.52	0.81
4	0.50	1.14
5	0.49	1.47
6	0.40	33.05

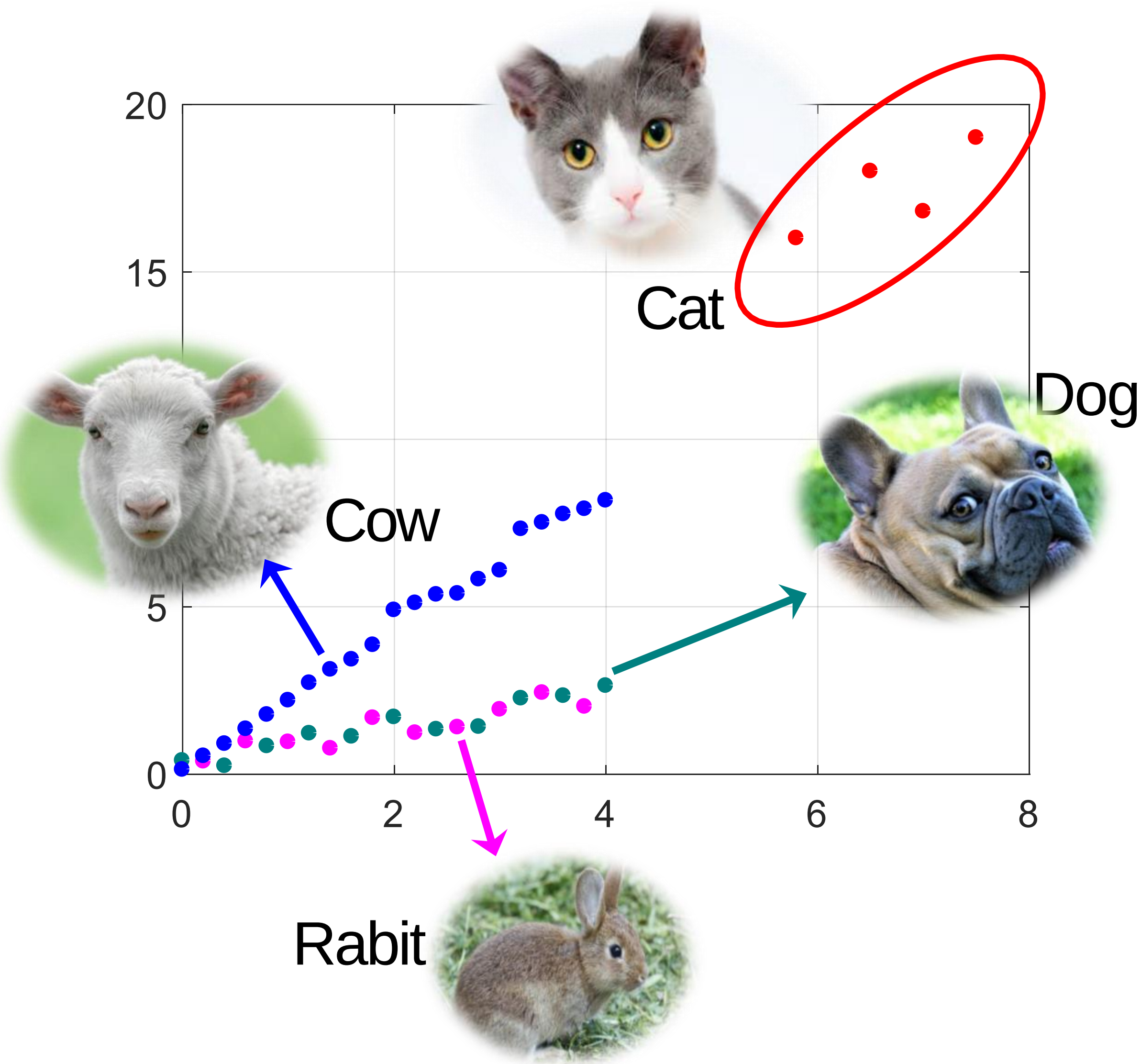
A more complex model does not always lead to better performance on **testing data**.

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 = species of x

x

If x_s = Cow: $y = b_1 + w_1 x$

If x_s = Cat: $y = b_2 + w_2 x$

If x_s = Rabit: $y = b_3 + w_3 x$

If x_s = Dog: $y = b_4 + w_4 x$

y

Back to step 1: Redesign the Model

$$y = b + \sum w_i \boxed{x_i}$$

Linear model?

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

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

Back to step 1: Redesign the Model

$$
\begin{aligned}
y =\ & b_1 \cdot 1 \\
& + w_1 \cdot 1 \\
& + b_2 \cdot 0 \\
& + w_2 \cdot 0 \\
& + b_3 \cdot 0 \\
& + w_3 \cdot 0 \\
& + b_4 \cdot 0 \\
& + w_4 \cdot 0
\end{aligned}
$$

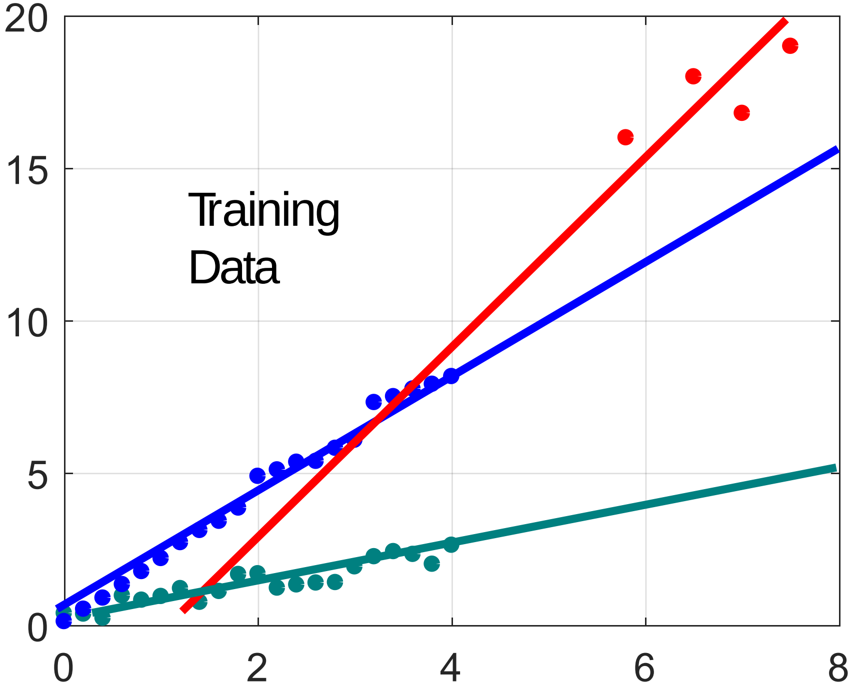
$$y = b + \sum w_i \boxed{x_i}$$

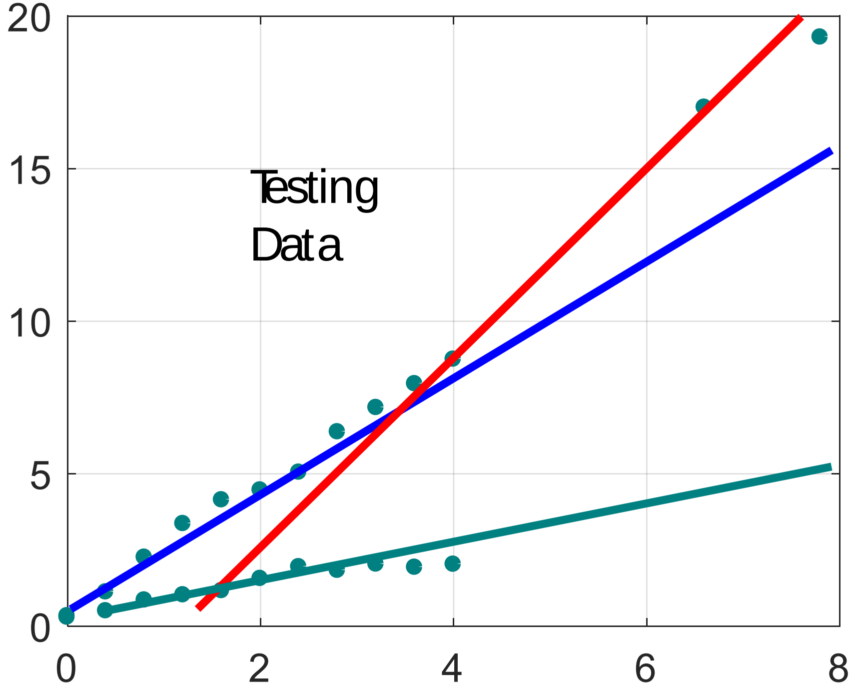
Linear model?

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

If x_s=Cow,

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





Are there any other
hidden factors?

Back to step 1: Redesign the Model Again

x

If x_s = Cow: $$y' = b_1 + w_1 \cdot x_1 + w_5 \cdot x_1^2$$

If x_s = Dog: $$y' = b_2 + w_2 \cdot x_1 + w_6 \cdot x_1^2$$

If x_s = Rabit: $$y' = b_3 + w_3 \cdot x_1 + w_7 \cdot x_1^2$$

If x_s = Cat: $$y' = b_4 + w_4 \cdot x_1 + w_8 \cdot x_1^2$$

$$y = y' + w_9 \cdot x_2^2 + w_{10} \cdot x_2^2$$
$$+ w_{11} \cdot x_3^2 + w_{12} \cdot x_3^2 + w_{13} \cdot x_4^2 + w_{14} \cdot x_4^2$$

y

Training Error = 1.9

Testing Error = 102.3

Overfitting!

Back to step 2: Regularization

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

The functions with smaller w_j are better

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

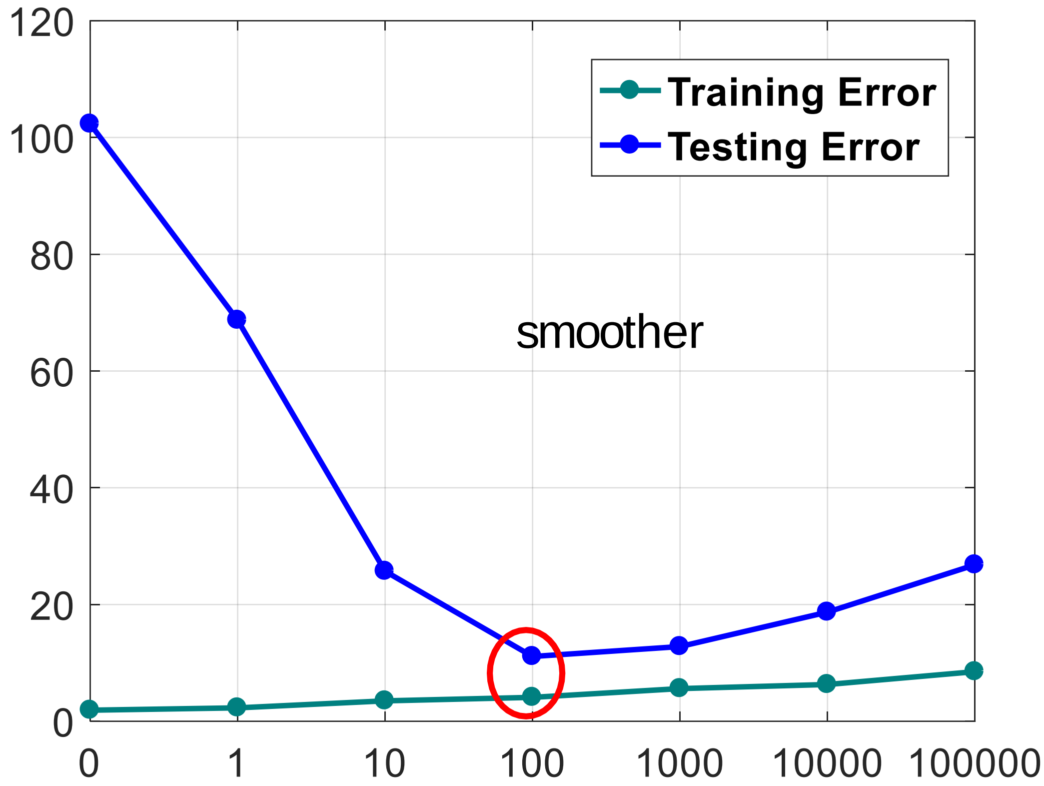
- Why smooth functions are preferred?

$$\underset{+w_j \Delta x_j}{y} = b + \sum_j w_j \underset{+\Delta x_j}{x_j}$$

- If some noises corrupt input x_j when testing

A smoother function has less influence.

Regularization



λ	Training	Testing
0	1.9	102.3
1	2.3	68.7
10	3.5	25.7
100	4.1	11.1
1000	5.6	12.8
10000	6.3	18.7
100000	8.5	26.8

How smooth?
Select λ obtaining the best model

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

Back to step 2: Regularization

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

The functions with smaller w_j are better

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

- Why smooth functions are preferred?

$$\underset{+w_i \Delta x_i}{y} = b + \sum_j w_j \underset{+\Delta x_i}{x_j}$$

- If some noises corrupt input x_j when testing

A smoother function has less influence.

Next Lecture:
Gradient Descent: theory and tips
Logistic Regression
(This Friday)