Deep Learning and Applications

DSA 5204 • Lecture 1
Dr Low Yi Rui (Aaron)
Department of Mathematics

NUS
National University of Singapore

Information

Instructor

Low Yi Rui (Aaron)
Office: S17-05-19
Email: matyrl@nus.edu.sg

Slides and demo code will be available on Canvas

Reference Book

- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- https://www.deeplearningbook.org/

Hybrid Classroom

Lesson Plan

- One or two 10 min breaks
- Q&A sessions after the breaks
- Q&A at the end of class

Zoom instructions

- Join the Zoom session on Canvas
- Type questions in the chat, or ask during the Q&A sessions
- Mute your microphones, but switch them on when you'd like to ask questions
- You are encouraged to turn on your webcam when asking questions

PollEverywhere

- All polls can be accessed at

https://pollev.com/aaronyrlow

Attendance for SSG funded students

Attendance will be taken via QR code during the first break

Assessment Policies

Assessment

- Test (30%)
- Graded Homework ($10\% \times 2$)
- Course Project (Report + Presentation: 50%)

Late submission policy for homework and report

- Please drop me an email if you are going to submit late
- Late submission penalty: -20% of the total grade per day late

  e.g. Actual score = 7/10. Submitted 1 day late.

  Adjusted score = 5/10.

Consultation and Forum

Consultation slots

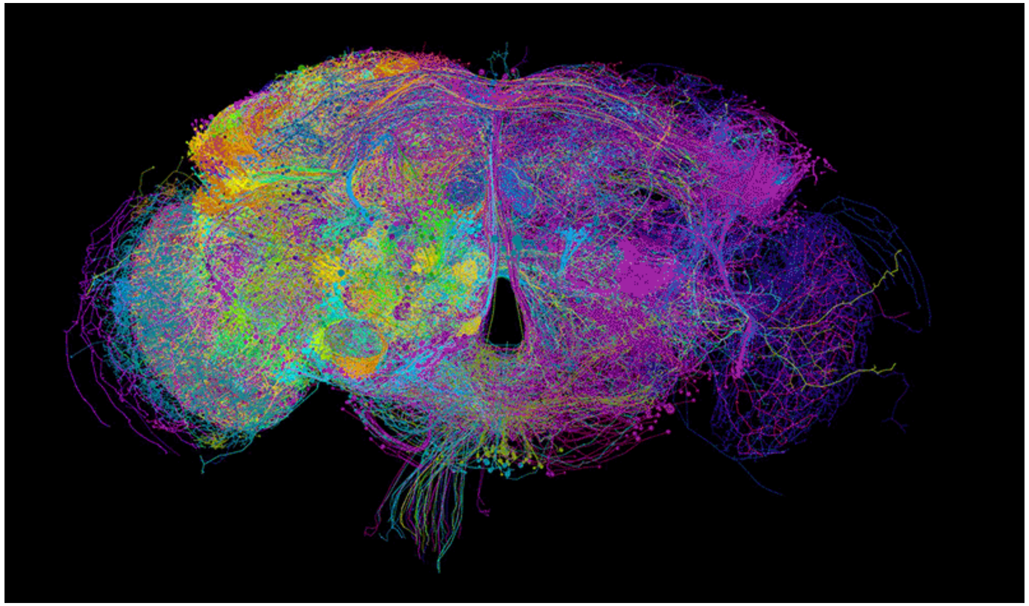
- available on Canvas (If filled, additional ones will be created)
- Currently set to 30min per session, Saturday 2-3pm

Discussion forum on Canvas

AI, Machine Learning & Deep Learning

The goal of the study of AI

To **understand** and **replicate** intelligence

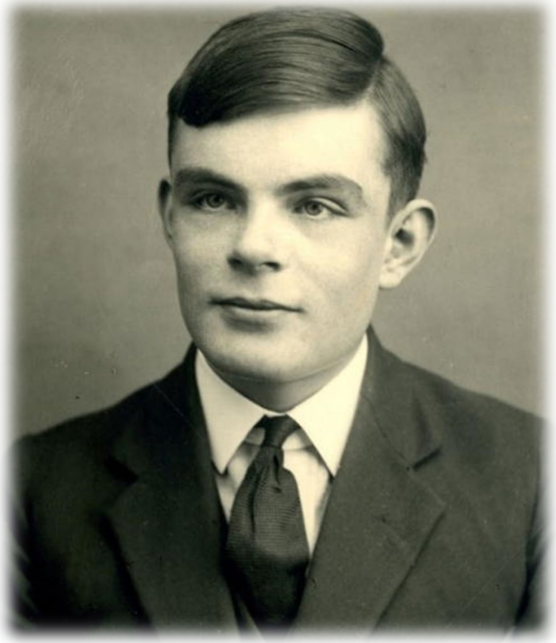


Neurons in a fruitfly's brain
Zheng et al. Cell 2018, 174 (3), 730-743.e22.

SCHWARZENEGGER
TERMINATOR
Disney·PIXAR
WALL·E
JUNE 27
her

Evolution of the concept of replication

Can machines think?

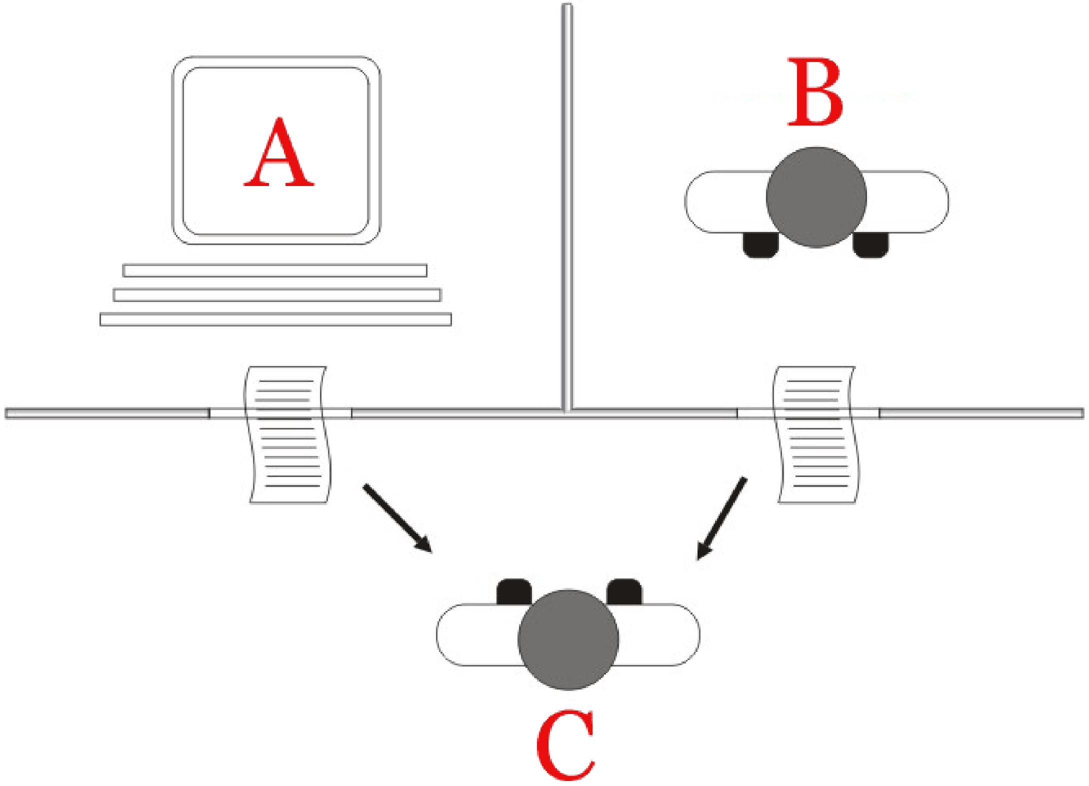


I propose to consider the question, "Can machines think?". This should begin with definitions of the meaning of the terms "machine" and "think". The definitions might be framed so as to reflect so far as possible the normal use of the words, but this attitude is dangerous…

Alan Turing, 1950

Turing, A. M. "Computing machinery and intelligence". Mind 49 433-460.

The Imitation Game

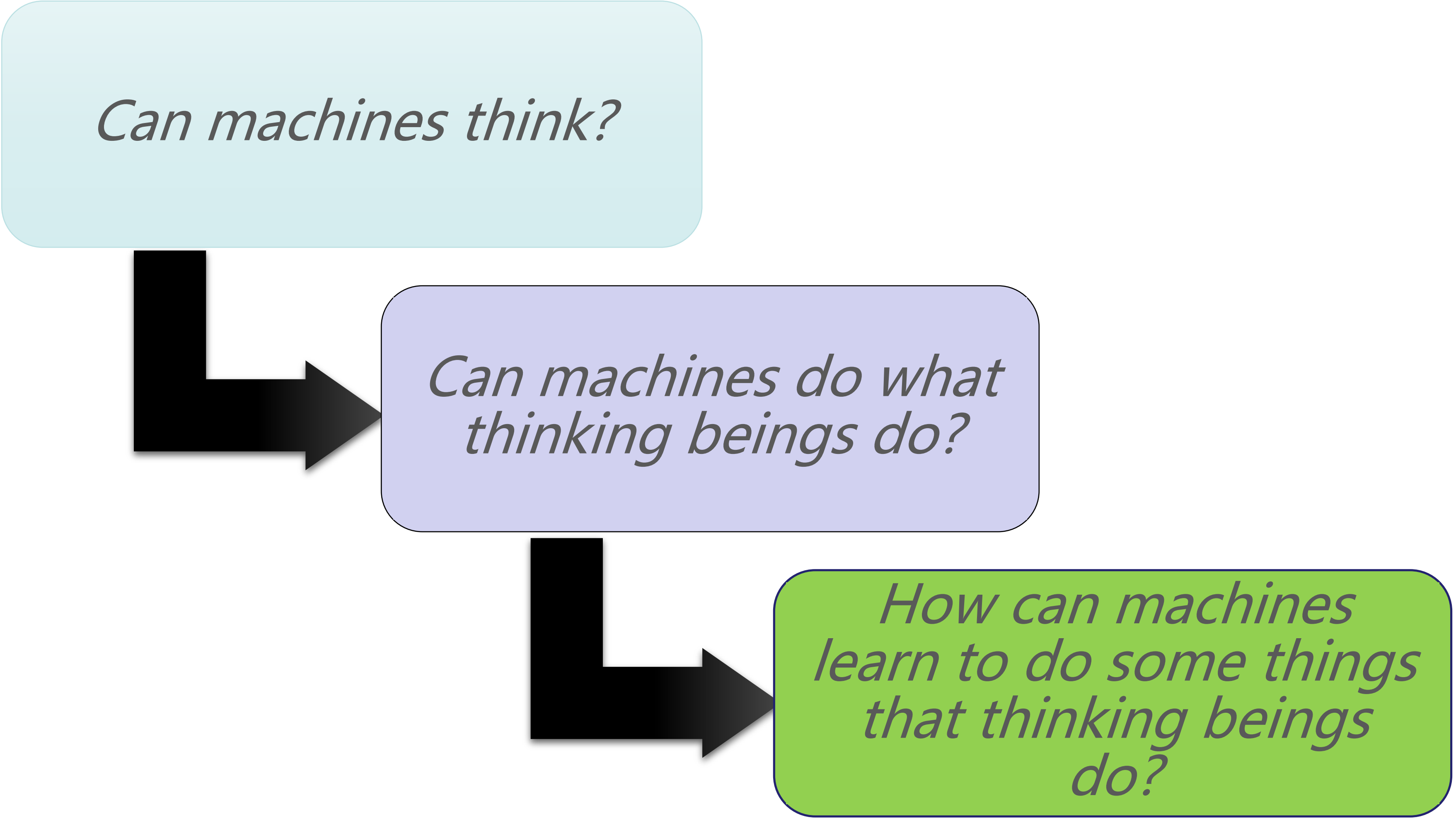


I believe that in about fifty years' time it will be possible, to programme computers … to make them play the imitation game so well that an average interrogator will not have more than 70 percent chance of making the right identification after five minutes of questioning.

Alan M Turing, 1950

https://en.wikipedia.org/wiki/Turing_test#/media/File:Turing_test_diagram.png

From artificial intelligence to machine learning

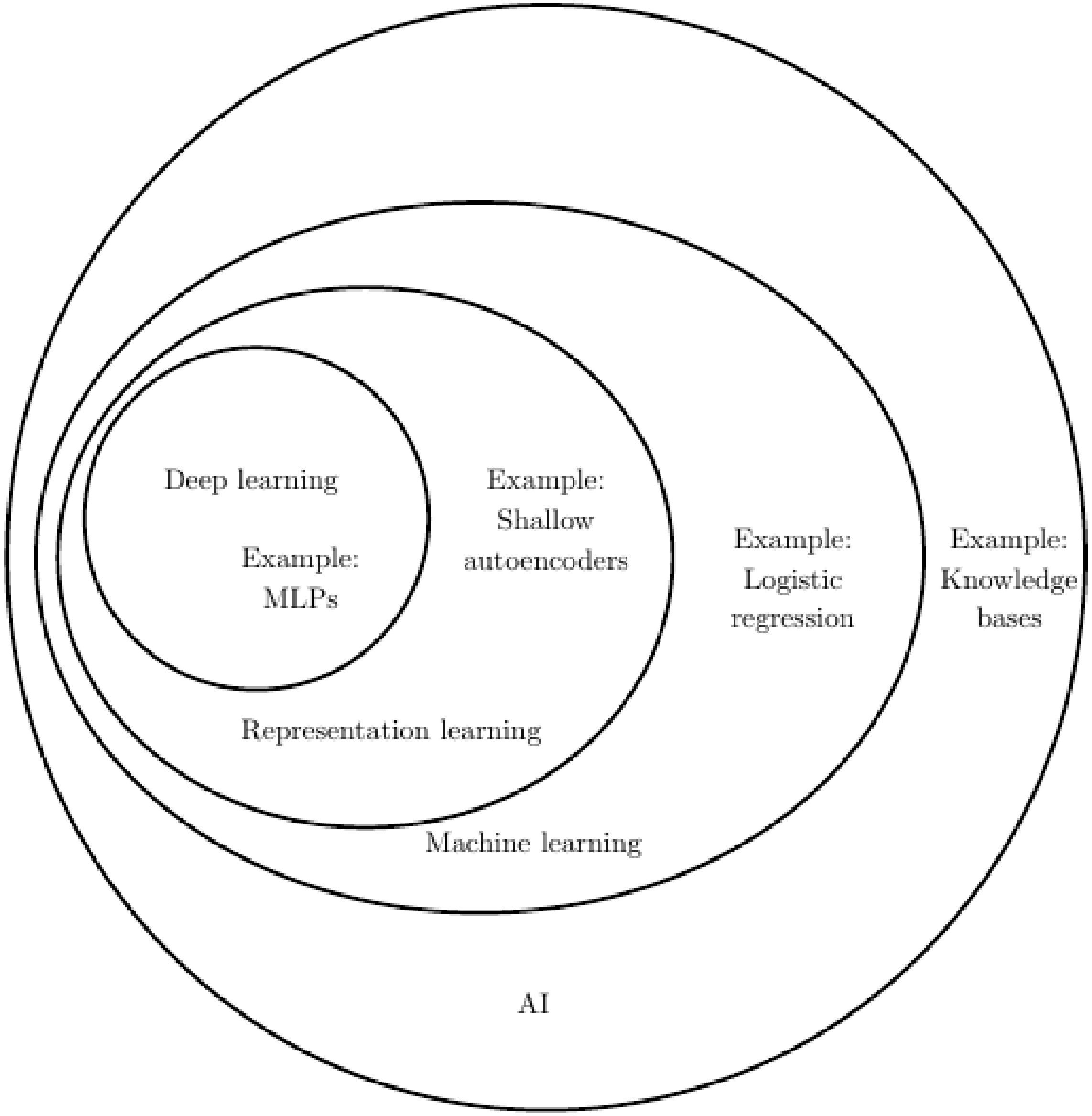


Can machines think?

Can machines do what thinking beings do?

How can machines learn to do some things that thinking beings do?

Deep Learning

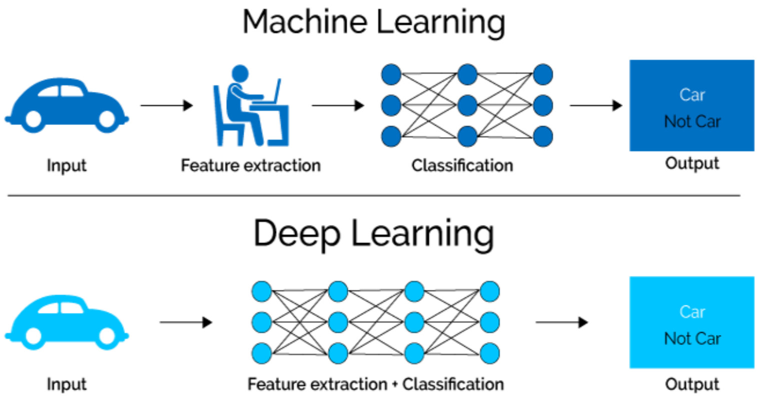


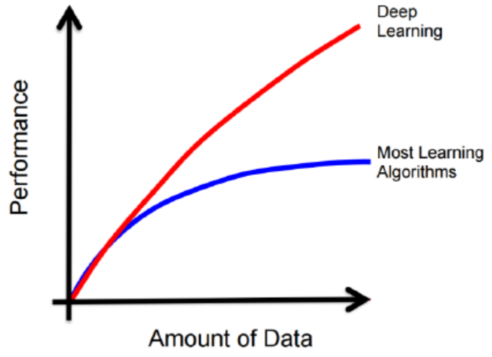
A Brief History of Deep Learning

- 1940s: Artificial neural networks
- 1950s: Roots of the backpropagation algorithm
- 1950s-80s: Perceptron, RNNs
- 1980s-90s: CNN, LSTM, Bidirectional RNNs
- 2000s-Present: deep learning has become the state of the art for a variety of applications
    - AlexNet (2012)
    - DeepFace (2014)
    - Generative Adversarial networks (GANs) (2014)
    - AlphaGo (2016)
    - BERT (2018)
    - Molecular Dynamics, Protein folding, etc (2020-21)

Why deep learning now?

Data





Computing

NVIDIA CUDA

TensorFlow

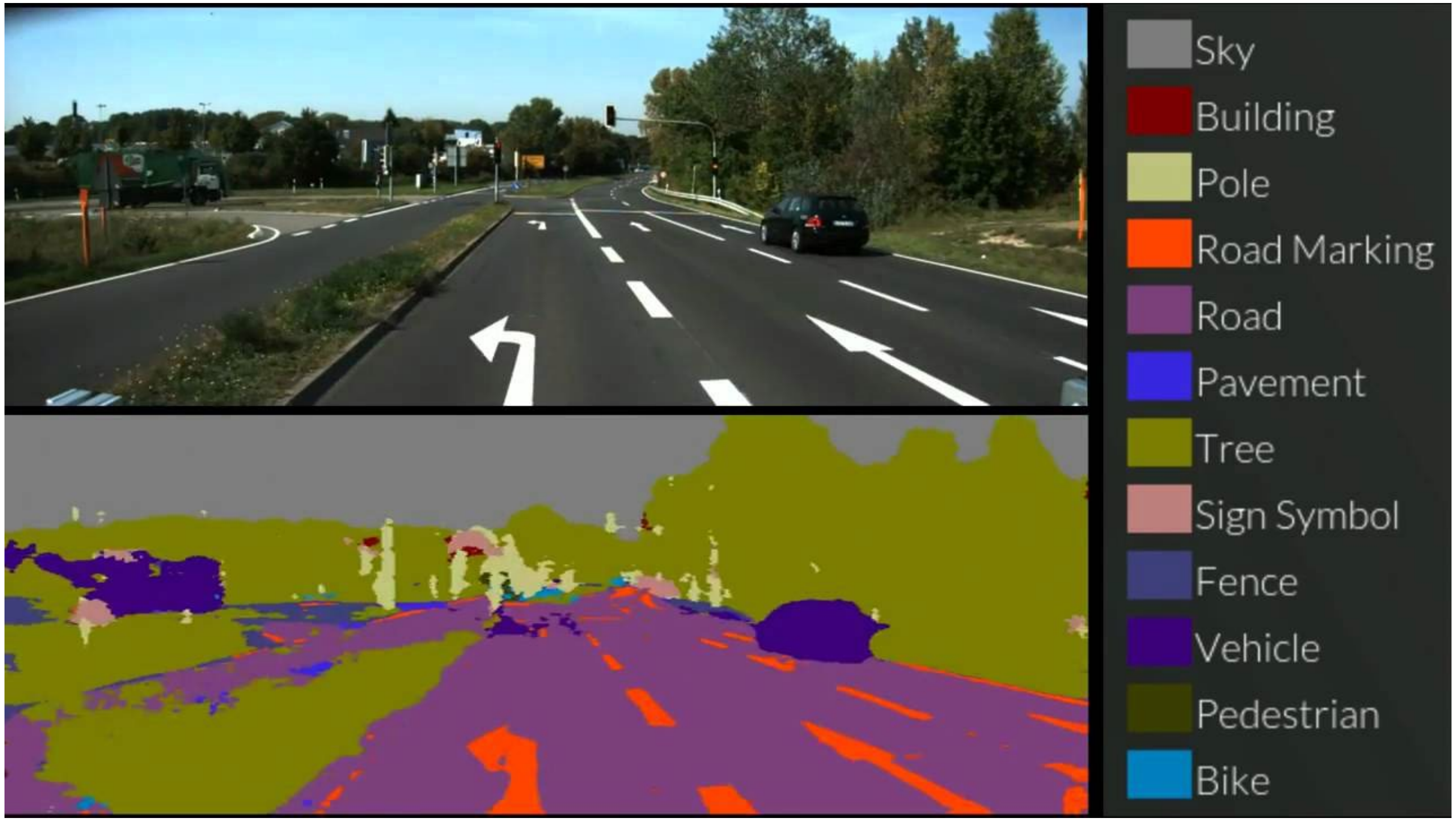
PYTORCH

theano

mxnet

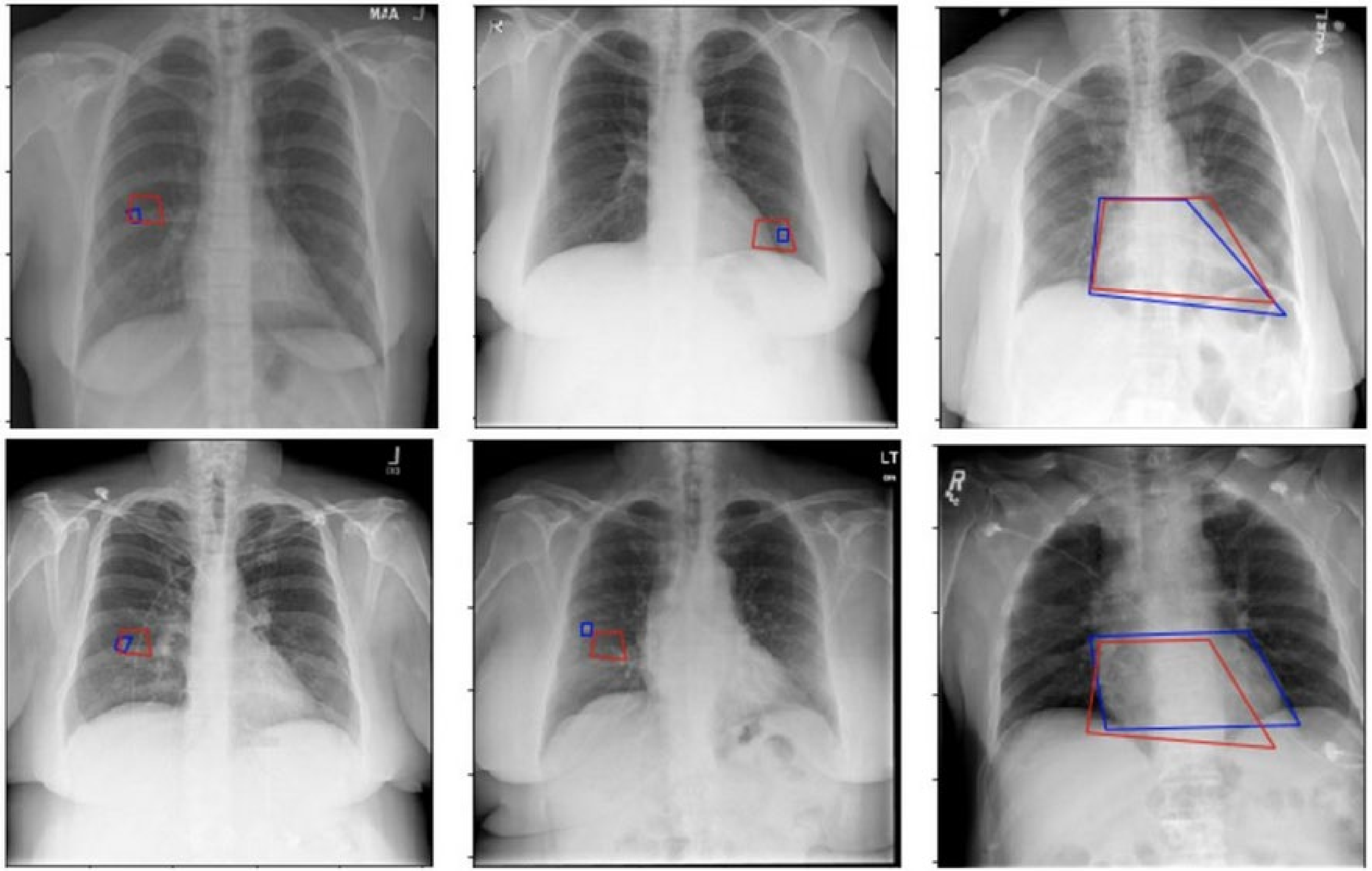
K Keras

Deep Learning in Autonomous Driving



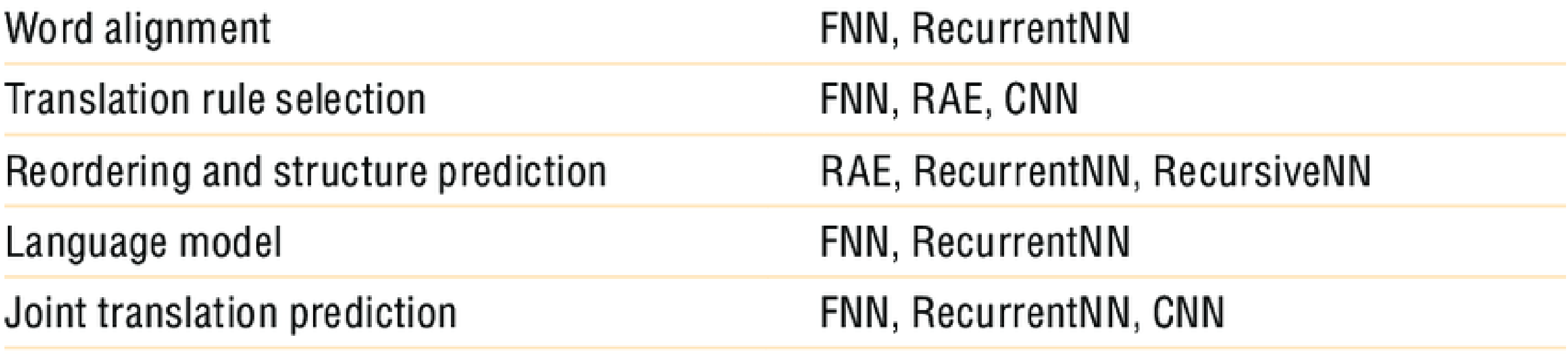
https://www.youtube.com/watch?v=kMMbW96nMW8

Deep Learning in Medical Imaging

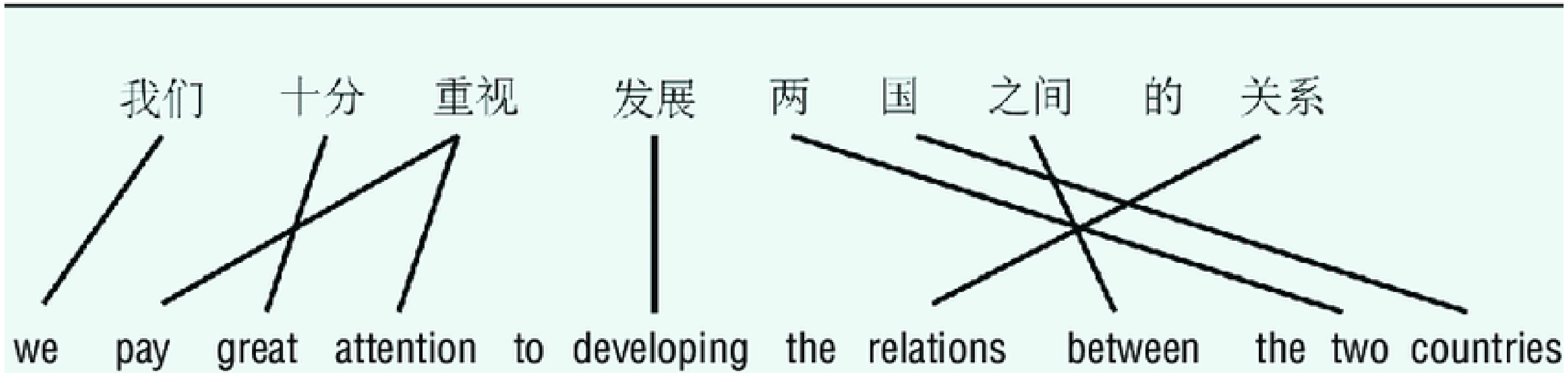


Moradi, Mehdi, et al. International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, Cham, 2018.

Deep Learning in Machine Translation



Word alignment	FNN, RecurrentNN
Translation rule selection	FNN, RAE, CNN
Reordering and structure prediction	RAE, RecurrentNN, RecursiveNN
Language model	FNN, RecurrentNN
Joint translation prediction	FNN, RecurrentNN, CNN

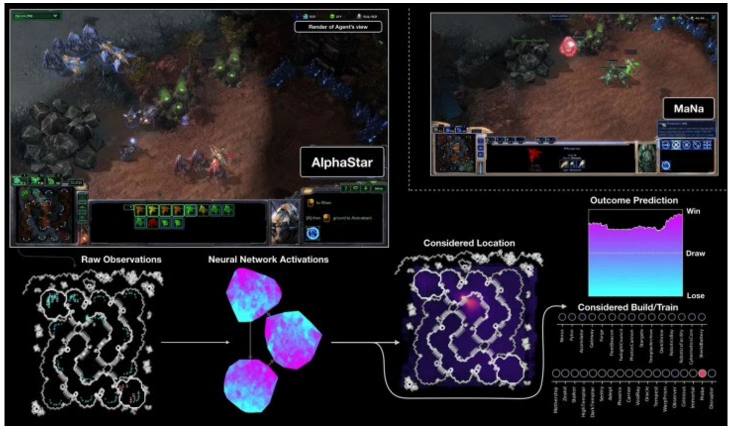

我们 十分 重视 发展 两 国 之间 的 关系
we pay great attention to developing the relations between the two countries

Zhang, Jiajun, and Chengqing Zong. "Deep neural networks in machine translation: An overview." (2015).

Deep RL: towards AGI

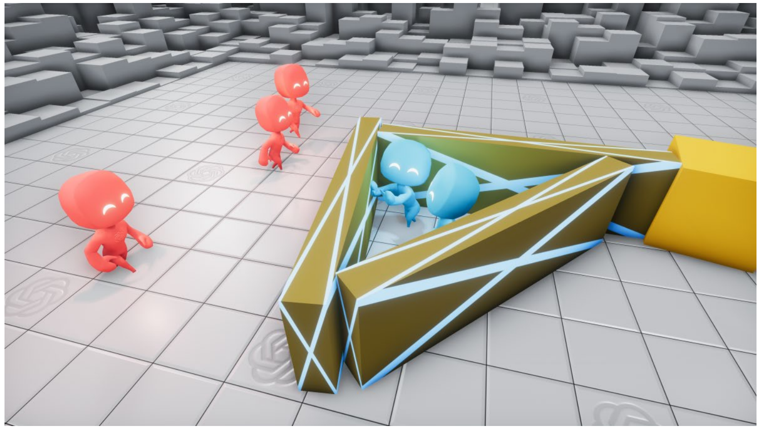
AlphaGo/AlphaZero

AlphaStar

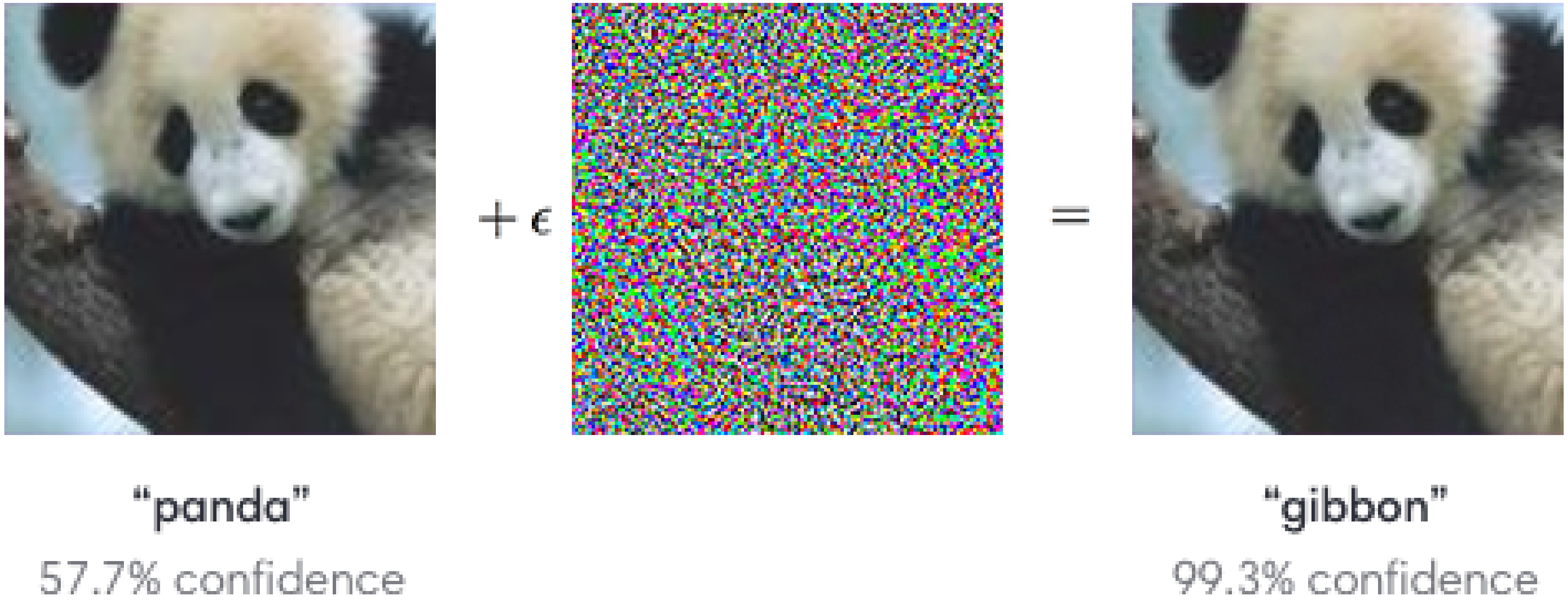


OpenAI Five

OpenAI Hide-and-Seek



Limitations of Deep Learning



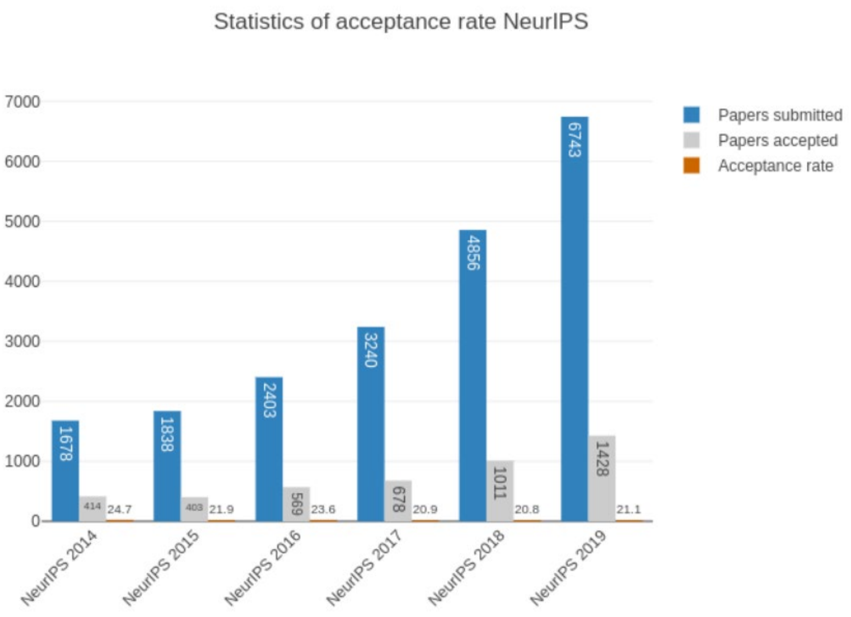
This Class

Modern introduction to deep learning and its applications

- Architectures
- Learning algorithms
- Practical techniques
- Applications via demos/homework/projects
- An overview of active areas of DL research

Deep Learning Research

- Deep learning research is very active!
- Learning how to learn is very important
- The course project serves this purpose (more info later)



Mathematics and Programming

Mathematics

- Linear algebra
- Probability
- *Deep learning Book* (Chapters 2-3)

Programming

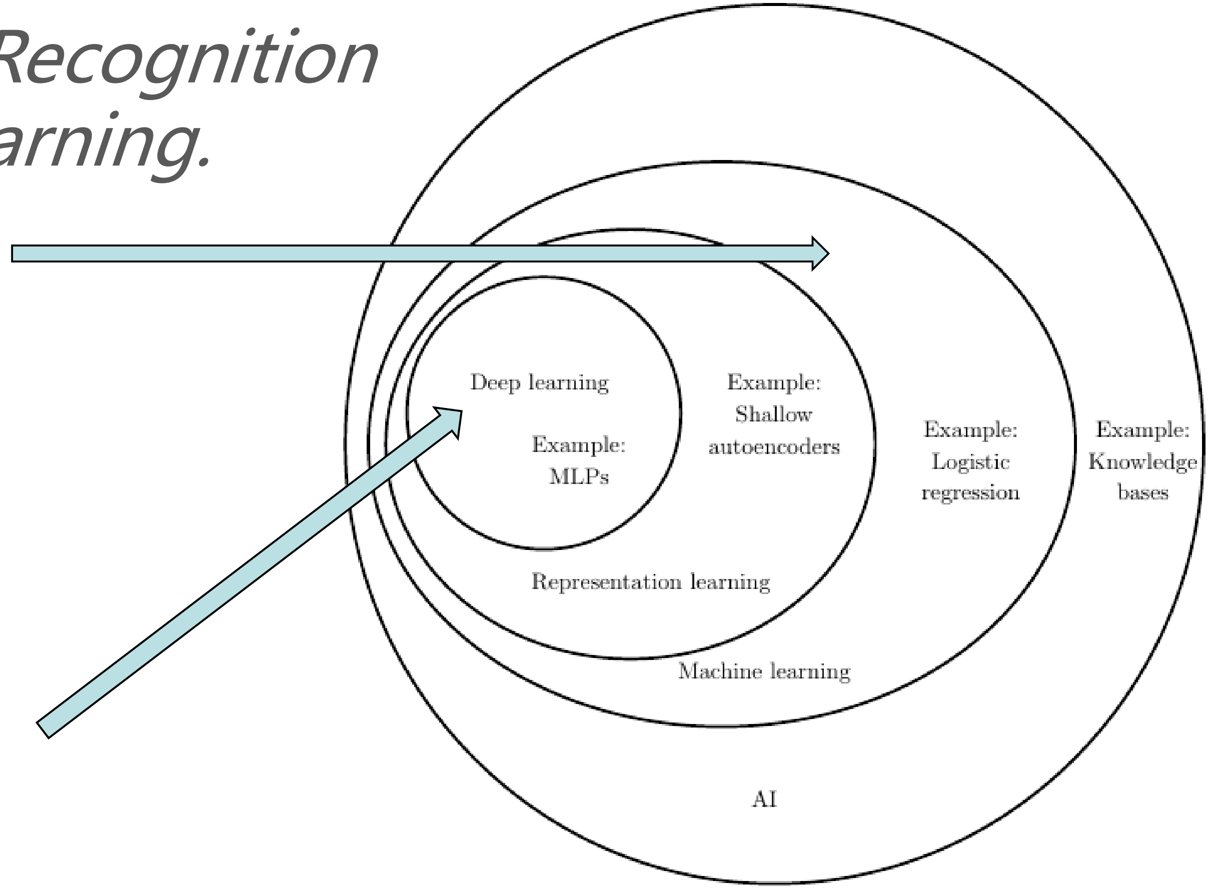
- Basic Python
- Useful libraries to know: Numpy, Scipy, Sklearn, Pandas, Seaborn/Matplotlib
- Deep learning libraries: Tensorflow(Keras)/Pytorch
- Tutorials are widely available online

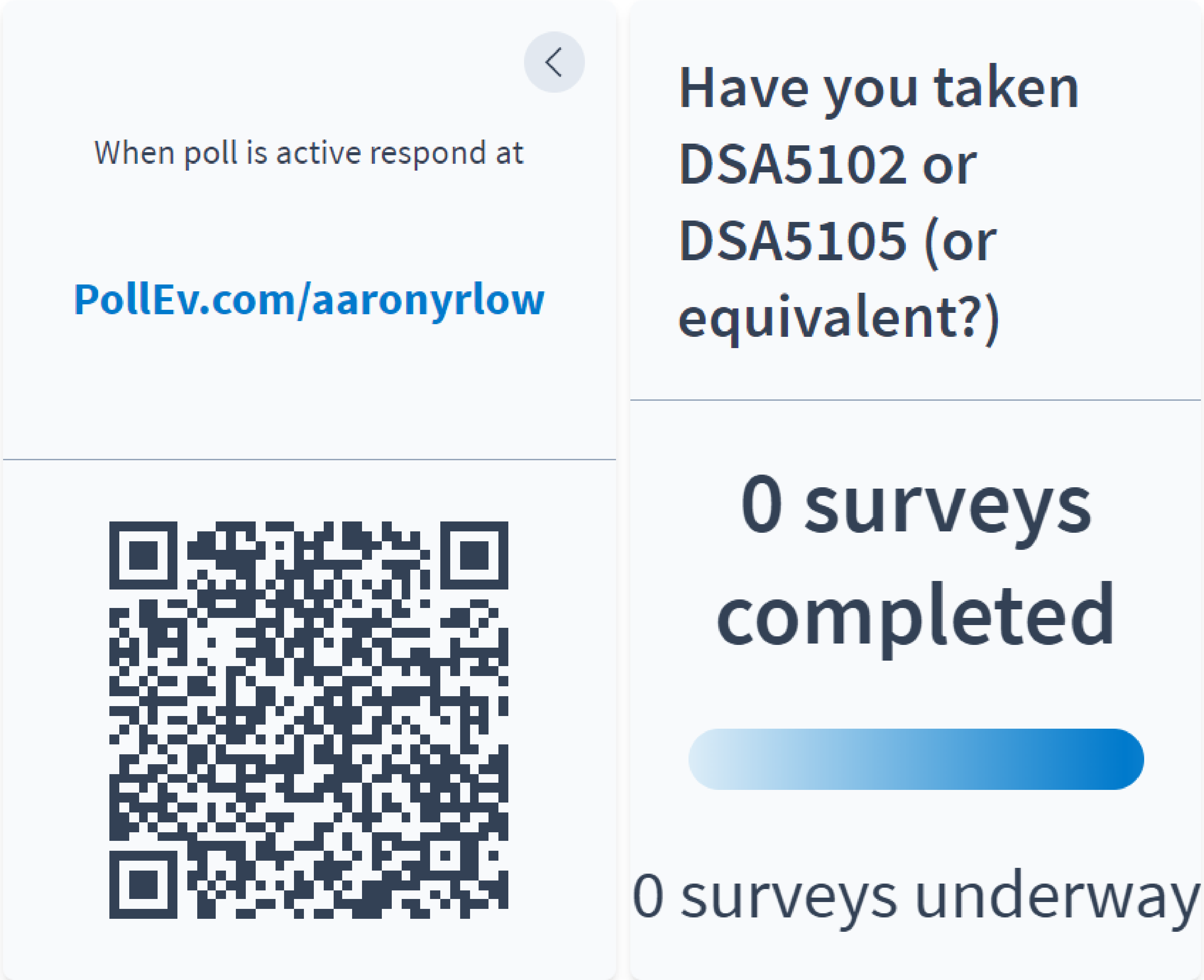
DSA5102 vs DSA5204

Bishop, *Pattern Recognition and Machine Learning.*

DSA5102/
DSA5105

DSA5204





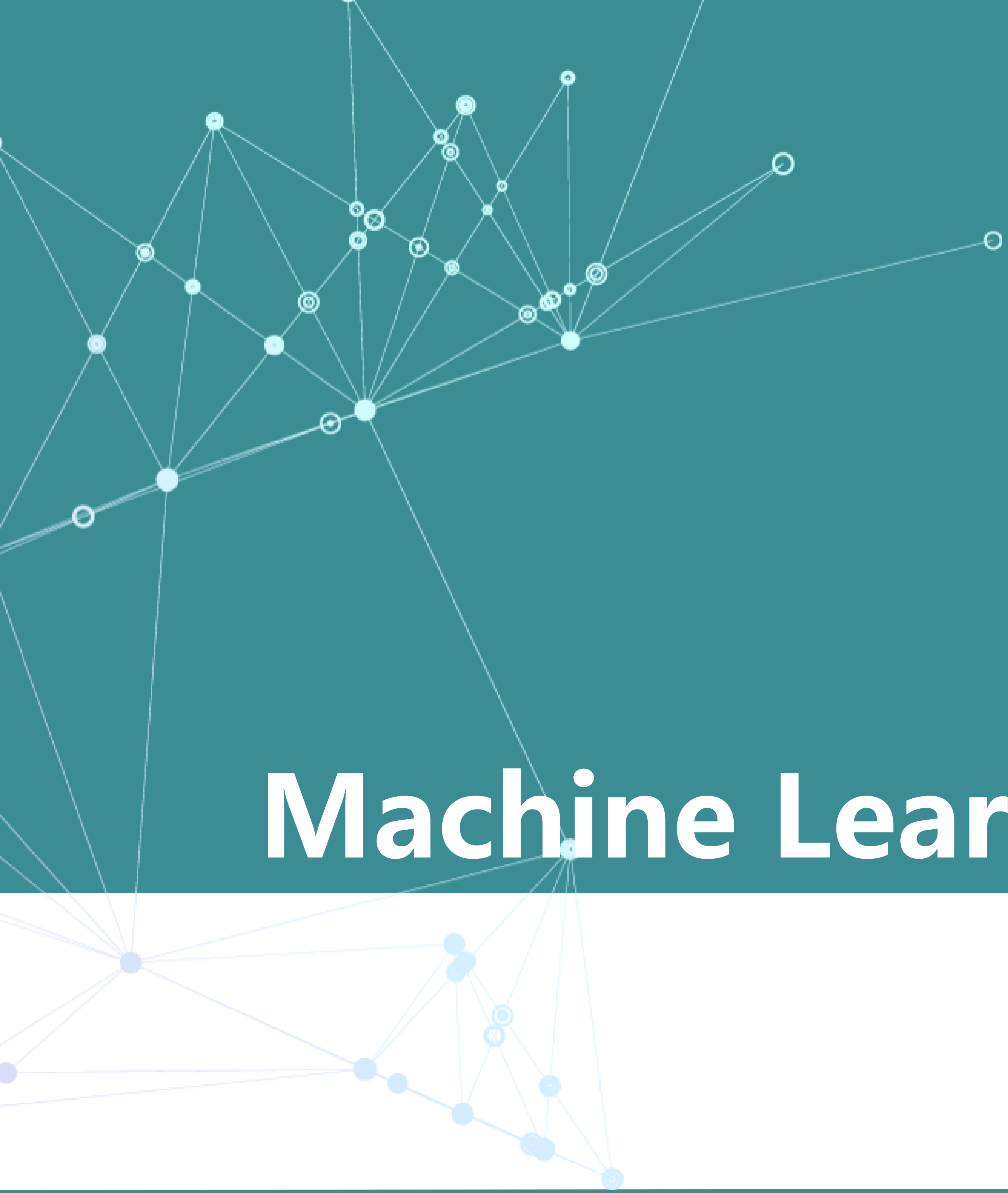
When poll is active respond at

PollEv.com/aaronyrlow

Have you taken DSA5102 or DSA5105 (or equivalent?)

0 surveys completed

0 surveys underway

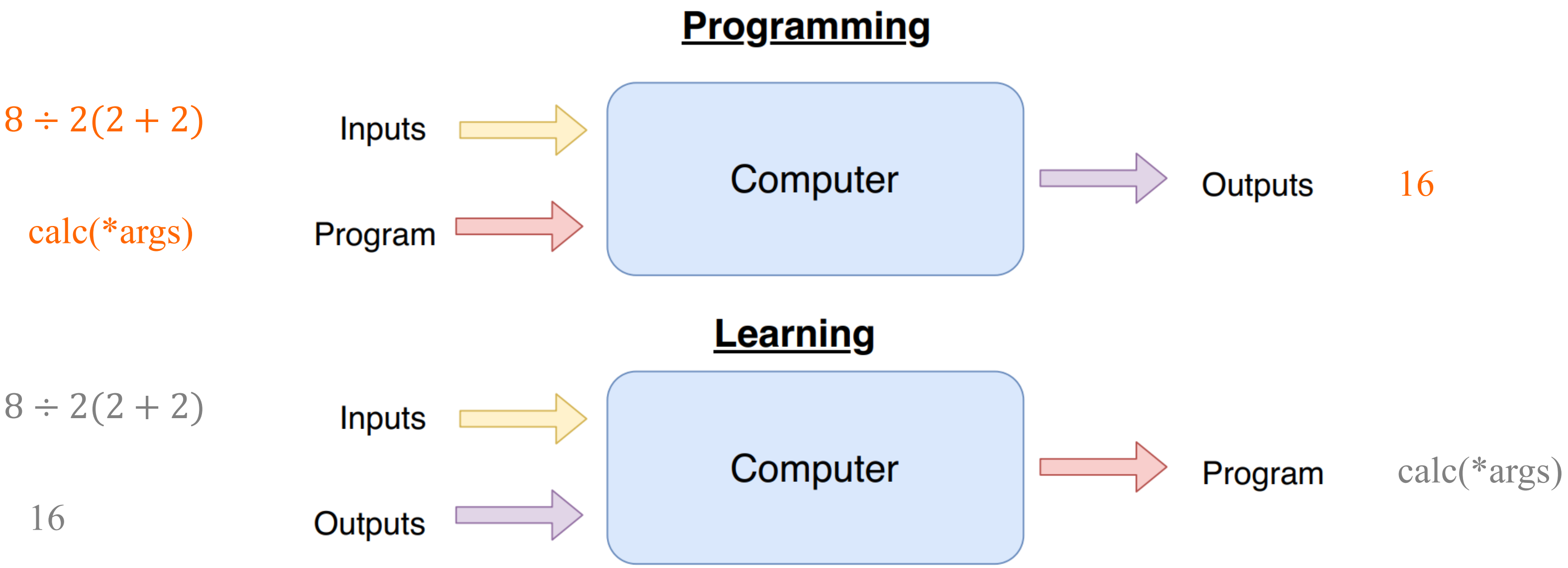


Machine Learning Basics

A concrete definition of learning

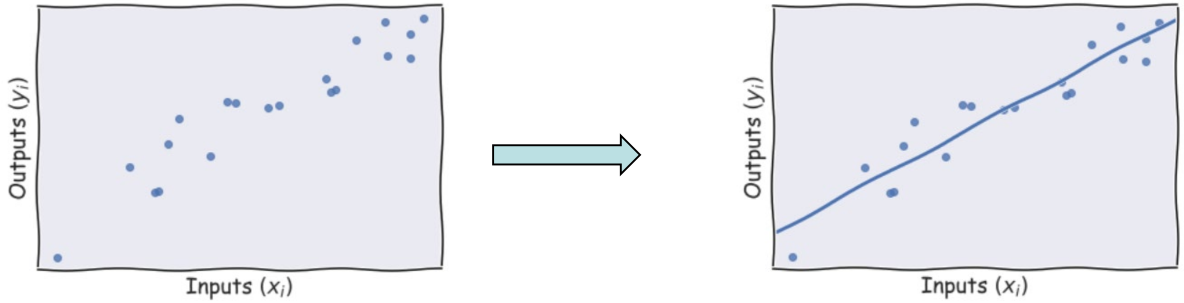
A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E.

Tom Mitchell

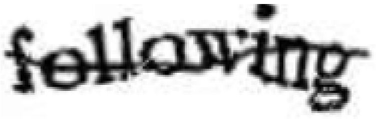


Tasks

Prediction

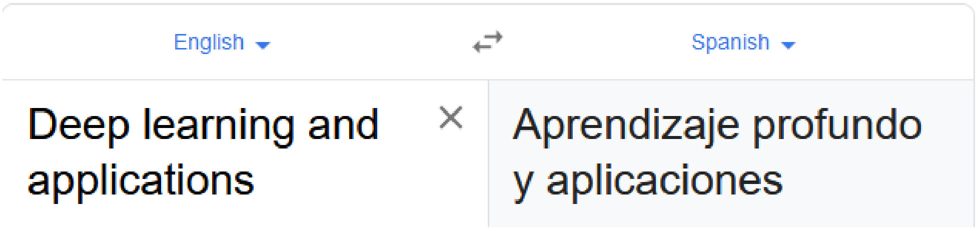


Transcription



following

Translation

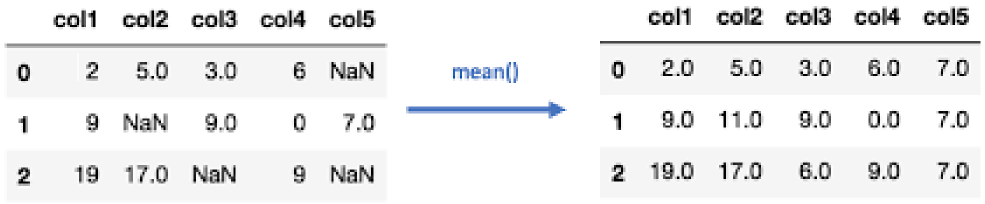


Deep learning and applications

Aprendizaje profundo y aplicaciones

Tasks

Imputation



	col1	col2	col3	col4	col5
0	2	5.0	3.0	6	NaN
1	9	NaN	9.0	0	7.0
2	19	17.0	NaN	9	NaN

mean()

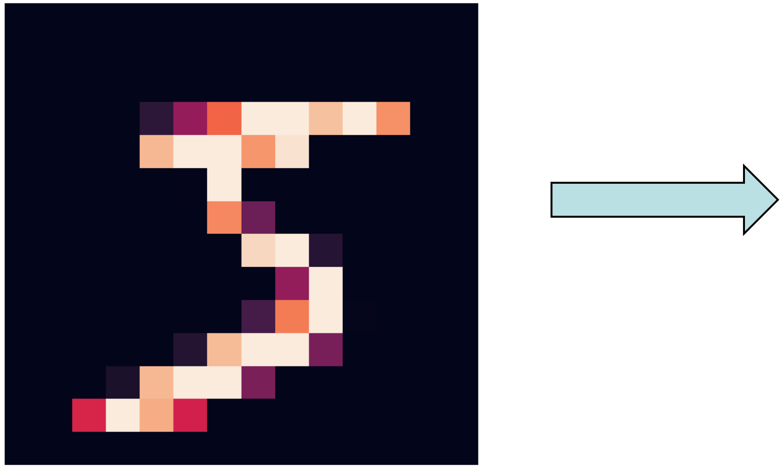
	col1	col2	col3	col4	col5
0	2.0	5.0	3.0	6.0	7.0
1	9.0	11.0	9.0	0.0	7.0
2	19.0	17.0	6.0	9.0	7.0

Generation

Experience

Experience = Data

Usually represented by vectors/matrices/tensors



0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	30	94	170	253	253	225	253	195	0	0
0	0	0	0	219	253	253	198	247	0	0	0	0	0
0	0	0	0	0	1	253	0	0	0	0	0	0	0
0	0	0	0	0	0	190	70	0	0	0	0	0	0
0	0	0	0	0	0	0	240	253	25	0	0	0	0
0	0	0	0	0	0	0	0	93	253	0	0	0	0
0	0	0	0	0	0	0	46	183	253	2	0	0	0
0	0	0	0	0	24	221	253	253	78	0	0	0	0
0	0	0	18	219	253	253	80	0	0	0	0	0	0
0	0	136	253	212	132	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

Performance

Performance measures how well a machine learning model does in a given task.

Examples:

- Regression: mean squared error of predictions
- Classification: accuracy of classification
- Image generation?

Other issues

- Learning algorithm: how do we improve P on task T with experience E?
- What is an objective measure of performance P?

Linear Regression as a Canonical Example

The T, E, P of Regression

(T) Task:

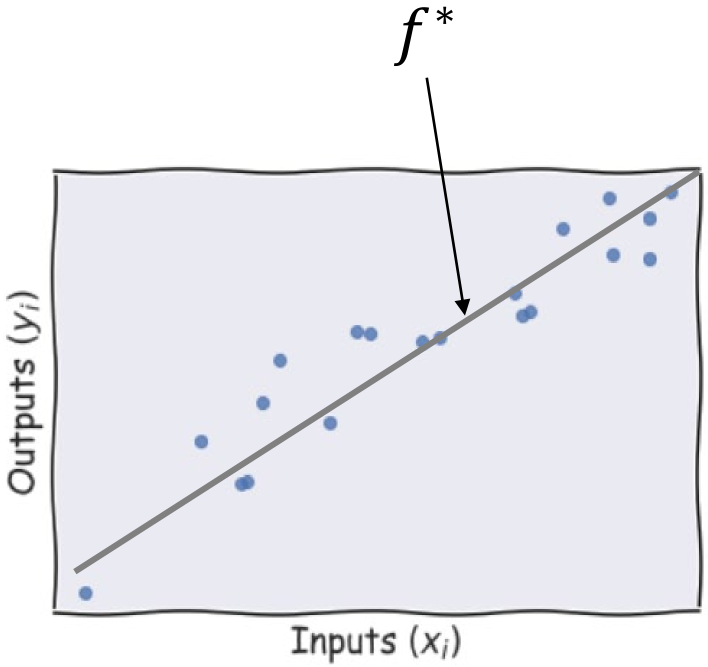
- Inputs: $\boldsymbol{x} \in \mathbb{R}^d$ Outputs: $y \in \mathbb{R}$
- Underlying relationship: $y = f^*(\boldsymbol{x})$
- We want to predict outputs given inputs

(E) Experience:

$$\mathcal{D} = \{\boldsymbol{x}_i, y_i = f^*(\boldsymbol{x})\}_{i=1}^N$$

or a noisy version

$$\mathcal{D} = \{\boldsymbol{x}_i, y_i = f^*(\boldsymbol{x}) + \epsilon_i\}_{i=1}^N$$



The T, E, P of Regression

Before discussing performance, we need to define a **hypothesis space**

$$\mathcal{H} = \{\text{collection of candidates } f\}$$

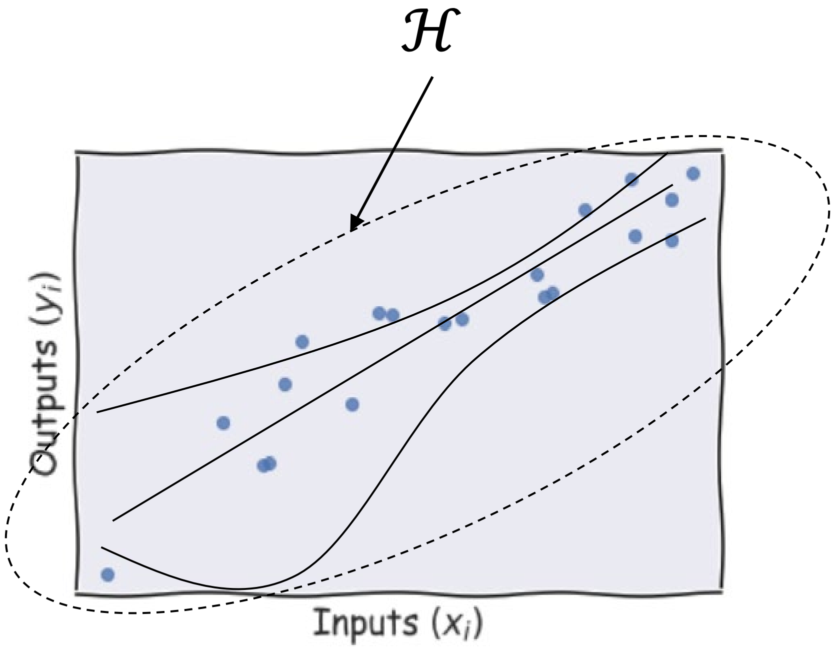
(P) Performance of a $f \in \mathcal{H}$:

$$\text{Distance}(f, f^*)$$

Learning:

Find a $\hat{f}$ that is closest to f^*

$$\hat{f} = \arg\min_{f \in \mathcal{H}} \text{Distance}(f^*, f)$$



How is distance measured?

A notion of distance can be defined by a **loss function** L evaluated over the dataset

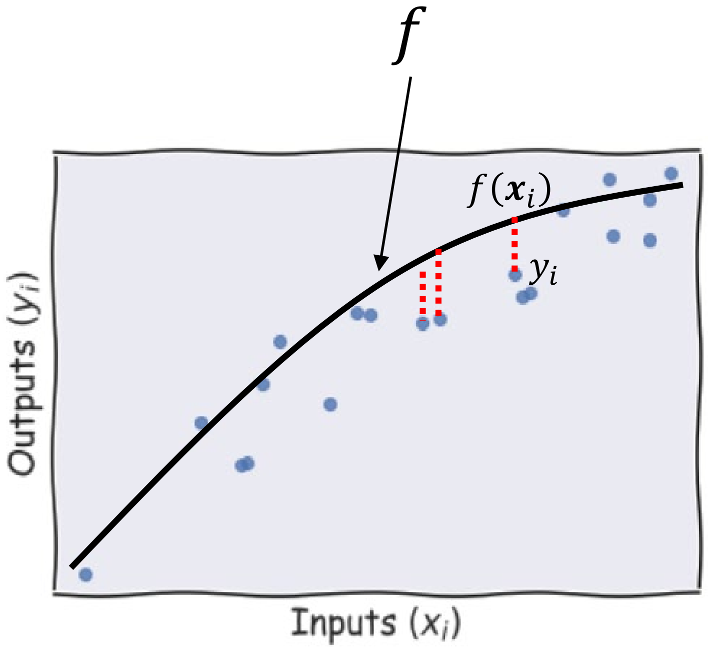
$$\text{Distance} = \frac{1}{N}\sum_{i=1}^{N} L(f(\boldsymbol{x}_i), y_i)$$

Example: square loss

$$L(y', y) = \frac{1}{2}(y' - y)^2$$

We will encounter many other loss functions throughout this class!

This distance is called the **empirical risk**, and we denote it by $R_{\text{emp}}(f, \mathcal{D})$



Empirical Risk Minimization

With the empirical risk at hand, we can concretely formulate the sentence

Increase performance P with experience E on task T

as an optimization problem

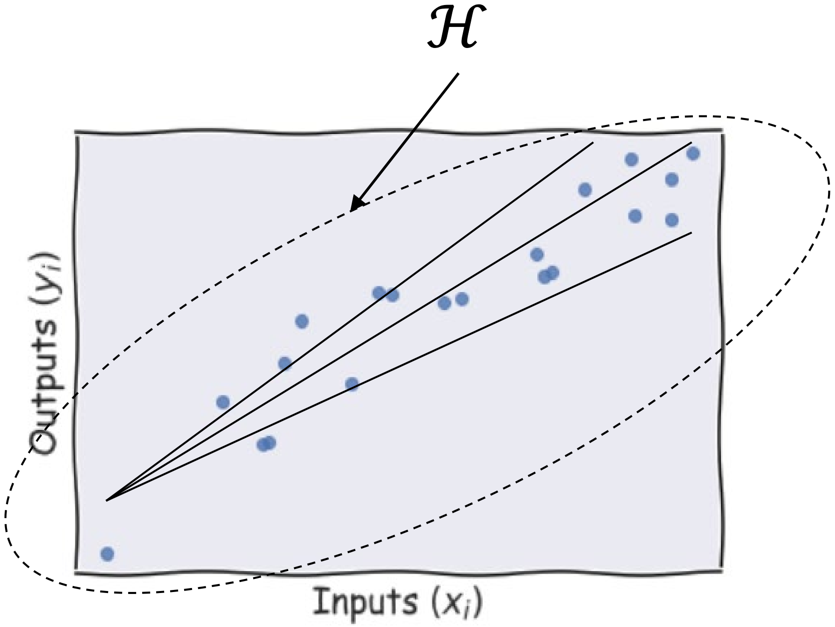
$$\hat{f} = \operatorname{argmin}_{f \in \mathcal{H}} R_{\mathrm{emp}}(f, \mathcal{D})$$

This is known as **empirical risk minimization** (ERM)

Example: Linear Regression

Linear regression is a particular choice of $\mathcal{H}$ -- linear functions

$$\mathcal{H} = \{f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} \ : \ \boldsymbol{w} \in \mathbb{R}^d\}$$



Example: Linear Regression

Empirical risk minimization under square loss:

$$\min_{w} R_{\text{emp}}(\boldsymbol{w}) = \frac{1}{2N}\sum_{i=1}^{N}(\boldsymbol{w}^T\boldsymbol{x}_i - y_i)^2$$

How do we solve this?

The Least Squares Formula

Define the design matrix and output vector

$$X = \begin{pmatrix} - & \boldsymbol{x}_1 & - \\ - & \boldsymbol{x}_2 & - \\ \vdots & \vdots & \vdots \\ - & \boldsymbol{x}_N & - \end{pmatrix} \qquad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

Then, the ERM can be rewritten as

$$\min_{\boldsymbol{w}} R_{\text{emp}}(\boldsymbol{w}) = \frac{1}{2N} ||X\boldsymbol{w} - \boldsymbol{y}||^2$$

The Least Squares Formula

The minimum of $R_{\text{emp}}(\boldsymbol{w})$ can be found by

$$\nabla_{\boldsymbol{w}} R_{\text{emp}}(\boldsymbol{w}) = 0$$

This gives the equation

$$X^T(X\boldsymbol{w} - \boldsymbol{y}) = 0$$

which yields

$$\boxed{\begin{gathered}\boldsymbol{w} = \widehat{\boldsymbol{w}} = (X^T X)^{-1} X^T \boldsymbol{y} \\ \hat{f}(\boldsymbol{x}) = \widehat{\boldsymbol{w}}^T \boldsymbol{x}\end{gathered}}$$

Ordinary Least Squares Formula

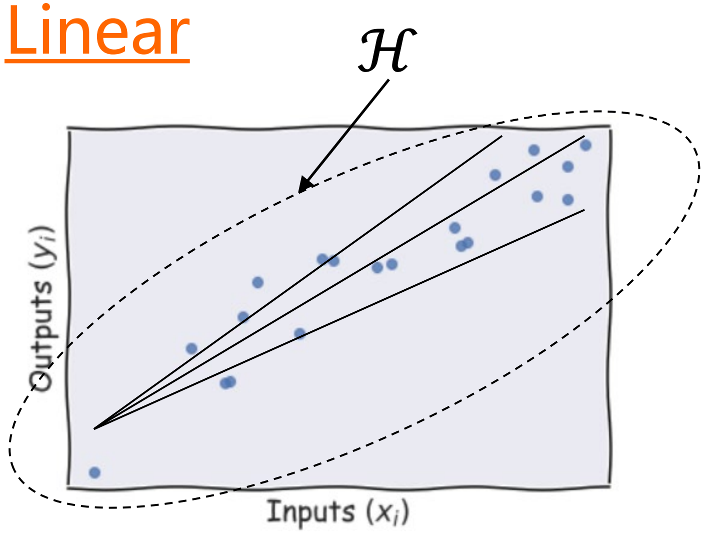
Extension to Affine Functions

Very often, linear functions are not rich enough to model our data.

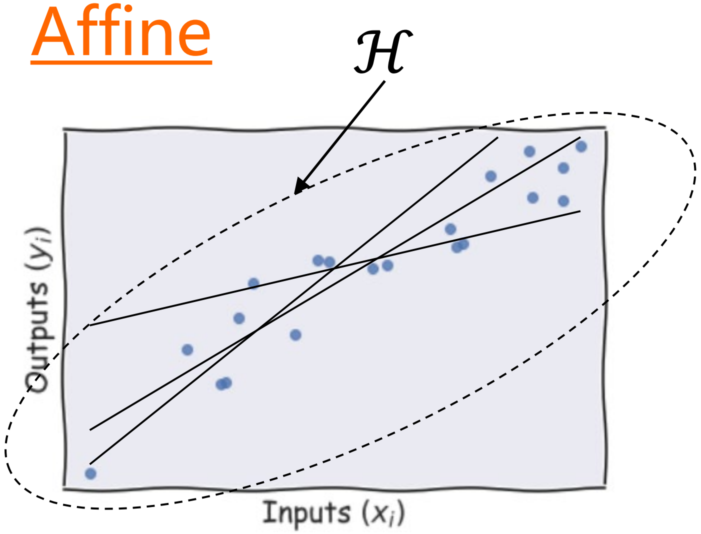
In the simplest case, we can consider **affine functions**

$$\mathcal{H} = \{f(x) = \boldsymbol{w}^T x + b \; : \; \boldsymbol{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$$

Linear



Affine



Extension to Affine Functions

The least squares formula is the same, except we change the definition of the design matrix

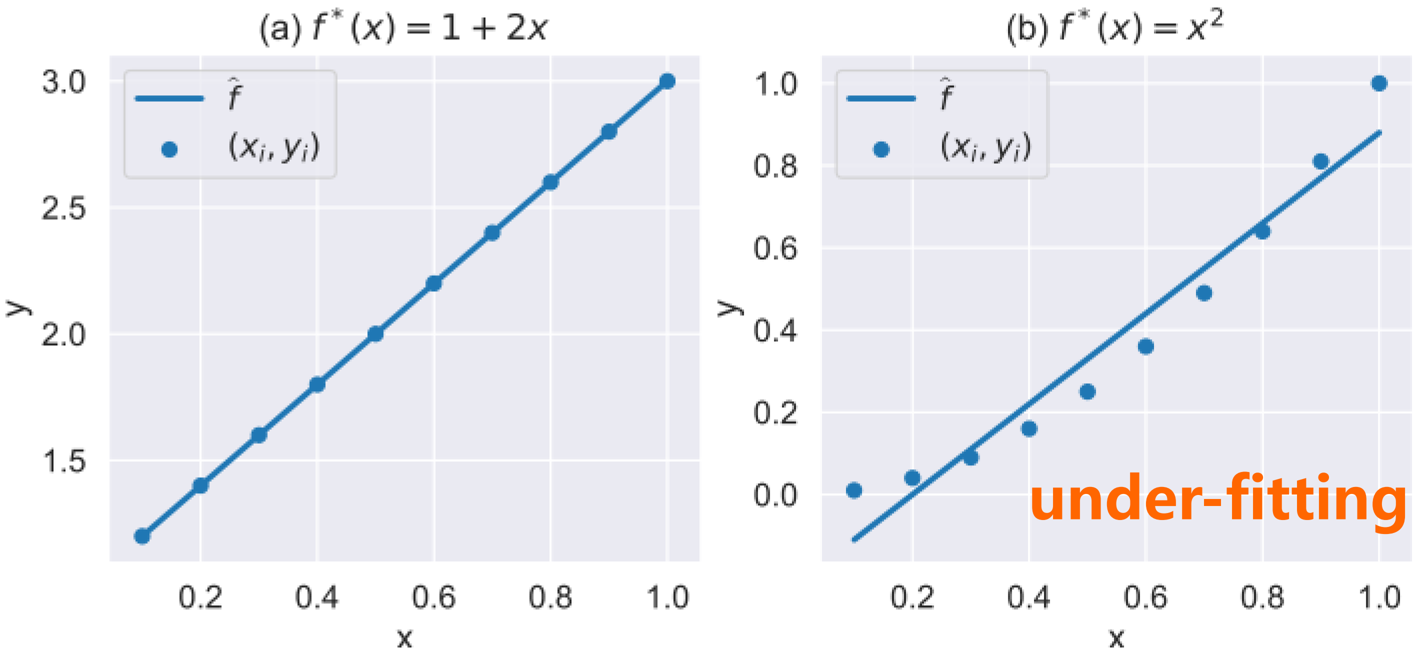
$$X = \begin{pmatrix} - & \boldsymbol{x}_1 & - \\ - & \boldsymbol{x}_2 & - \\ \vdots & \vdots & \vdots \\ - & \boldsymbol{x}_N & - \end{pmatrix} \quad \Rightarrow \quad X = \begin{pmatrix} - & \boldsymbol{x}_1 & - & 1 \\ - & \boldsymbol{x}_2 & - & 1 \\ \vdots & \vdots & \vdots & \vdots \\ - & \boldsymbol{x}_N & - & 1 \end{pmatrix}$$

$$w = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{pmatrix} \quad \Rightarrow \quad w = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \\ b \end{pmatrix}$$

Model Capacity and Underfitting

With every $\mathcal{H}$ comes the concept of **capacity**:

How large is the hypothesis space $\mathcal{H}$?



Extension to Linear Basis Models

Clearly, linear models cannot fit functions like

$$f^*(x) = x^2 \qquad f^*(x) = x^3 \qquad f^*(x) = \cos x$$

This motivates **linear basis models** in the form

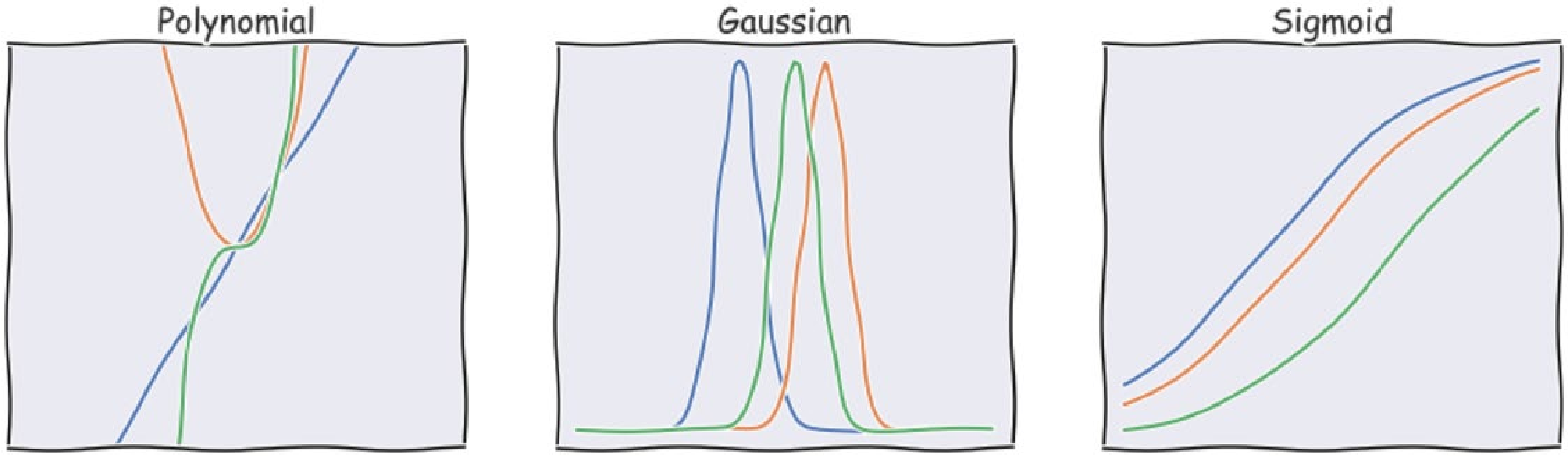
$$\mathcal{H} = \left\{ f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) = \sum_{i=1}^{m} w_i \phi_i(\boldsymbol{x}) \ : \ \boldsymbol{w} \in \mathbb{R}^m \right\}$$

The functions $\boldsymbol{\phi} = (\phi_1, \dots, \phi_m)$ are called **basis functions** and are chosen *a priori.* They are also called **feature maps.**

Examples of basis functions

Some choices of basis functions in 1D

- Polynomial basis: $\phi_j(x) = x^j$
- Gaussian basis: $\phi_j(x) = \exp\left(-\frac{\left(x-m_j\right)^2}{2s^2}\right)$
- Sigmoid basis: $\phi_j(x) = \sigma\left(\frac{x-m_j}{s}\right)$ with $\sigma(b) = \frac{1}{1+e^{-b}}$



Capacity of Linear Basis Models

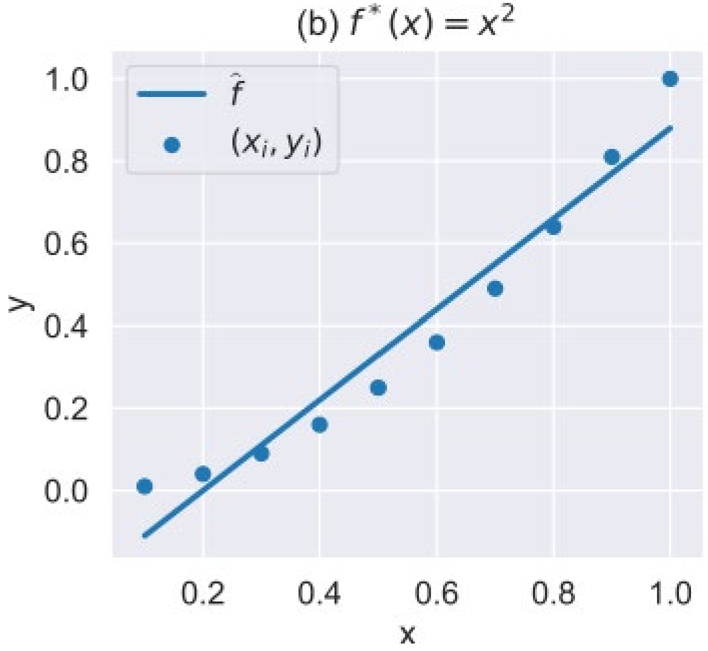
Note that linear basis models strictly generalizes linear/affine models!

- Set $\phi_i(\boldsymbol{x}) = x_i$ for $i = 1, \dots, d$ and $\phi_{d+1}(\boldsymbol{x}) = 1$

Using the basis

$$\phi_i(x) = x^{i-1} \text{ for } i = 1,2,3$$

We can easily fit this with a linear model



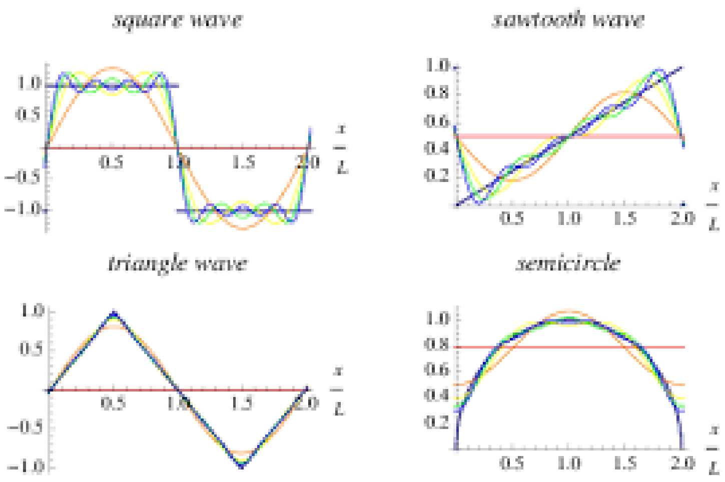
(b) $f^*(x) = x^2$

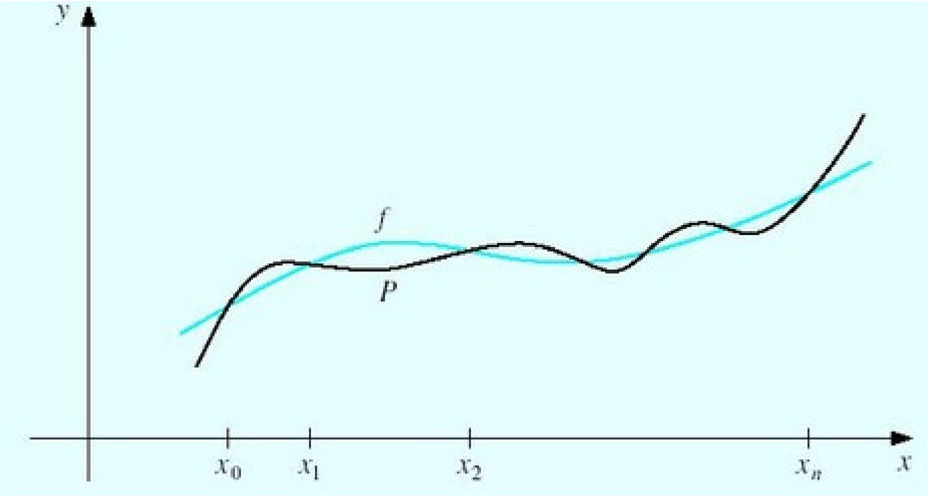
Universality

So, how big is the capacity of linear basis models?

With appropriate choices of basis functions and $m \to \infty$, they are **universal**, in the sense that they can approximate* any* function/relationship!

Examples: Fourier series, Weierstrass approximation





Least Squares Formula for Linear Basis Models

Least square formula obtained by changing the design matrix

$$X = \begin{pmatrix} - & \boldsymbol{x}_1 & - \\ - & \boldsymbol{x}_2 & - \\ \vdots & \vdots & \vdots \\ - & \boldsymbol{x}_N & - \end{pmatrix} \Rightarrow \quad \Phi = \begin{pmatrix} - & \boldsymbol{\phi}(\boldsymbol{x}_1) & - \\ - & \boldsymbol{\phi}(\boldsymbol{x}_2) & - \\ \vdots & \vdots & \vdots \\ - & \boldsymbol{\phi}(\boldsymbol{x}_N) & - \end{pmatrix}$$

The new empirical risk is

$$R_{\text{emp}}(\boldsymbol{w}) = \frac{1}{2N} ||\Phi\boldsymbol{w} - \boldsymbol{y}||^2$$

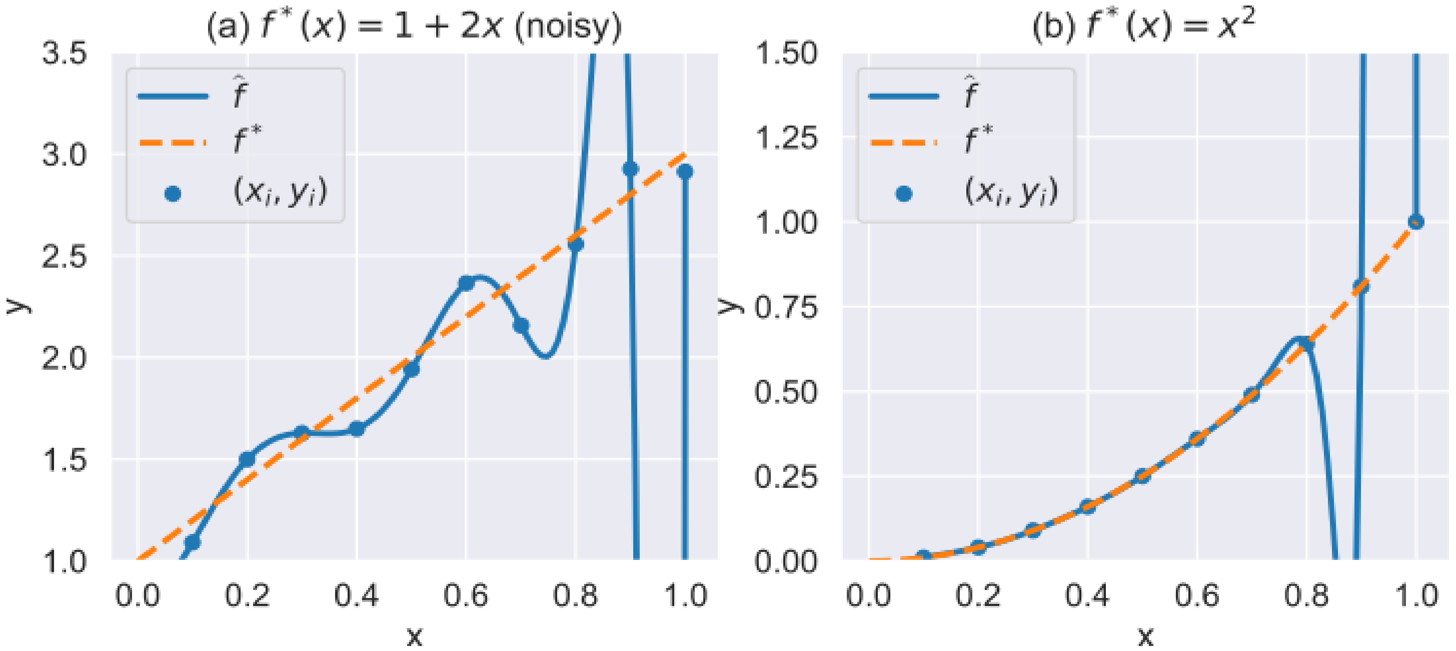
Solution: $\hat{\boldsymbol{w}} = (\Phi^T\Phi)^{-1}\Phi^T\boldsymbol{y}$

So, why not use many ϕ_i's?

Let us use

$$\mathcal{H} = \left\{f(x) = \sum_{j=0}^{99} w_j x^j : \boldsymbol{w} \in \mathbb{R}^{100}\right\}$$

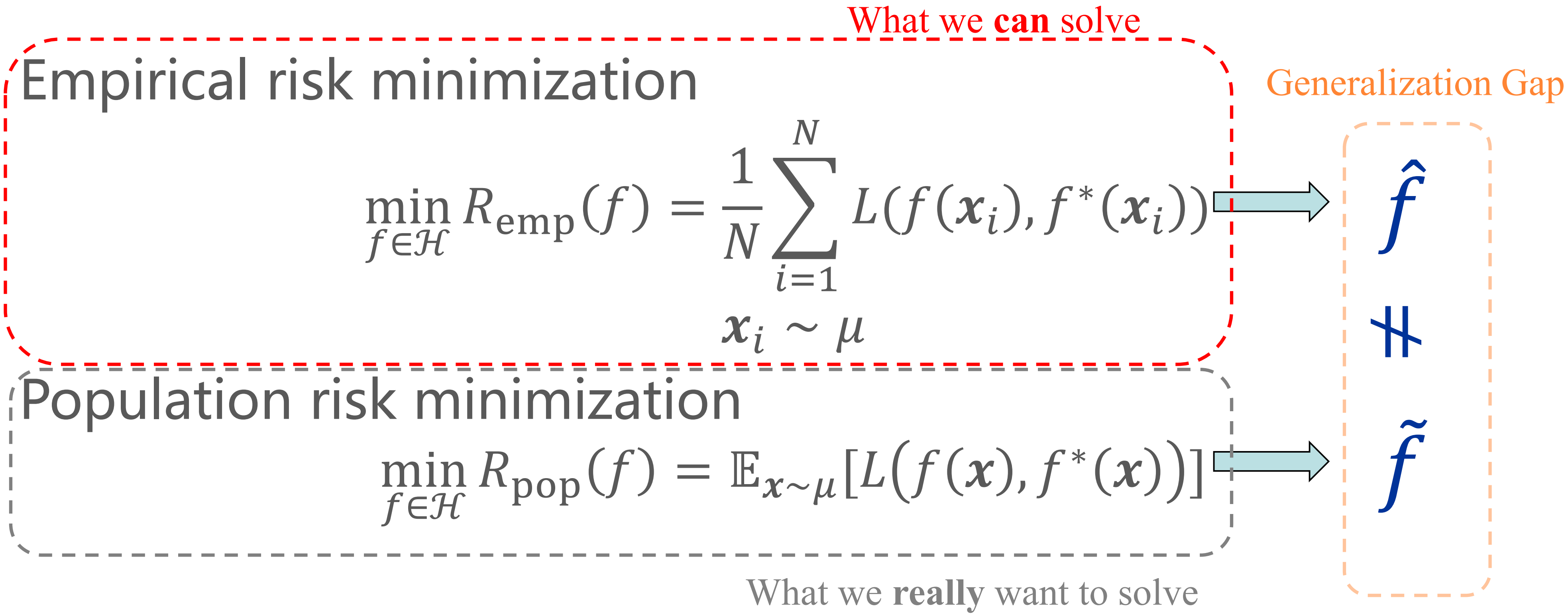
to fit our linear and quadratic examples



This is called over-fitting. Why are these bad?

Learning ≠ Optimization/ERM

We want to do well on **unseen** data! In other words, our model must **generalize.**



What we **can** solve

Empirical risk minimization

$$\min_{f\in\mathcal{H}} R_{\text{emp}}(f) = \frac{1}{N}\sum_{i=1}^{N} L(f(\boldsymbol{x}_i), f^*(\boldsymbol{x}_i))$$

$$\boldsymbol{x}_i \sim \mu$$

Population risk minimization

$$\min_{f\in\mathcal{H}} R_{\text{pop}}(f) = \mathbb{E}_{\boldsymbol{x}\sim\mu}[L\big(f(\boldsymbol{x}), f^*(\boldsymbol{x})\big)]$$

What we **really** want to solve

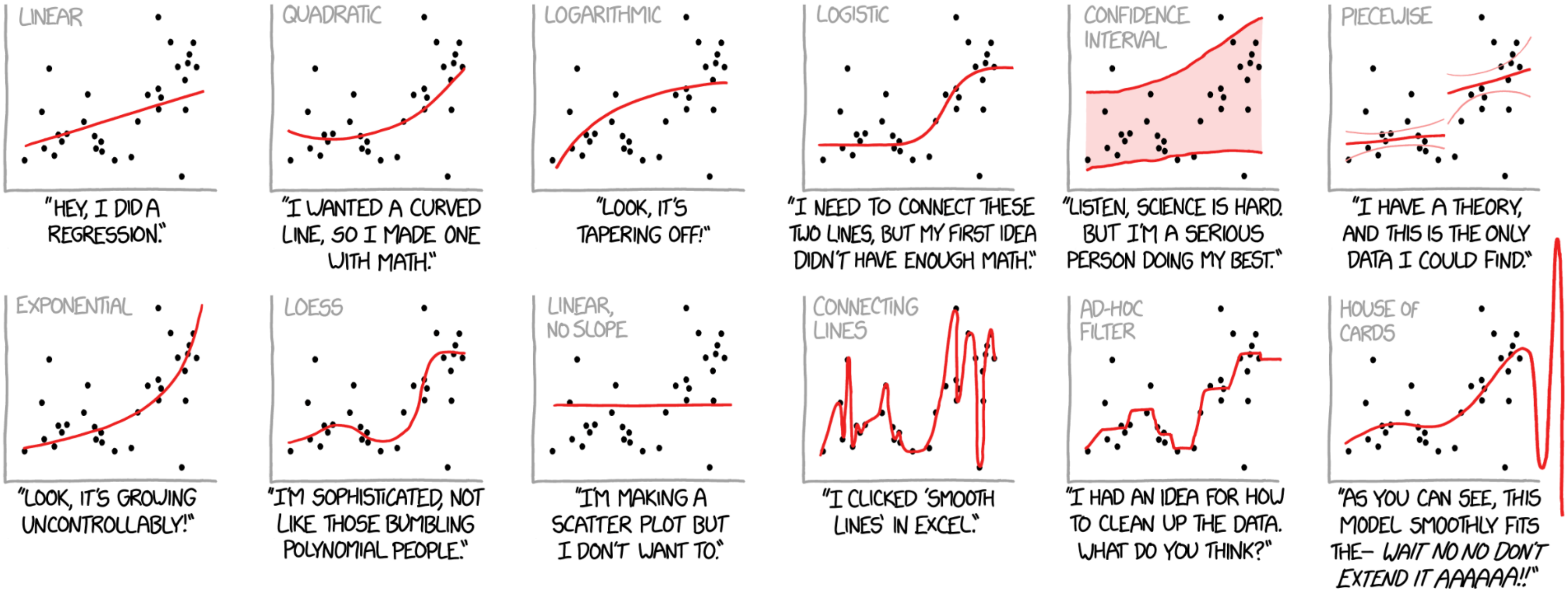
Generalization Gap

$$\hat{f}$$

$$\nparallel$$

$$\tilde{f}$$

Hypothesis Space and Generalization


LINEAR
"HEY, I DID A REGRESSION."
QUADRATIC
"I WANTED A CURVED LINE, SO I MADE ONE WITH MATH."
LOGARITHMIC
"LOOK, IT'S TAPERING OFF!"
LOGISTIC
"I NEED TO CONNECT THESE TWO LINES, BUT MY FIRST IDEA DIDN'T HAVE ENOUGH MATH."
CONFIDENCE INTERVAL
"LISTEN, SCIENCE IS HARD. BUT I'M A SERIOUS PERSON DOING MY BEST."
PIECEWISE
"I HAVE A THEORY, AND THIS IS THE ONLY DATA I COULD FIND."
EXPONENTIAL
"LOOK, IT'S GROWING UNCONTROLLABLY!"
LOESS
"I'M SOPHISTICATED, NOT LIKE THOSE BUMBLING POLYNOMIAL PEOPLE."
LINEAR, NO SLOPE
"I'M MAKING A SCATTER PLOT BUT I DON'T WANT TO."
CONNECTING LINES
"I CLICKED 'SMOOTH LINES' IN EXCEL."
AD-HOC FILTER
"I HAD AN IDEA FOR HOW TO CLEAN UP THE DATA. WHAT DO YOU THINK?"
HOUSE OF CARDS
"AS YOU CAN SEE, THIS MODEL SMOOTHLY FITS THE- WAIT NO NO DON'T EXTEND IT AAAAAA!!"

Curve fitting methods and the message they send. https://xkcd.com/2048/

How do we estimate the population risk?

We do not know the sample distribution μ, but we already have samples from it in our dataset $\mathcal{D}$.

Therefore, we can use a train-test split

$$\mathcal{D} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{test}}$$

- The training set $\mathcal{D}_{\text{train}}$ is used for empirical risk minimization
- The testing set $\mathcal{D}_{\text{test}}$ is used as an ultimate evaluation of performance
- Important: our learning algorithm should never peek at $\mathcal{D}_{\text{test}}$!

Generalization

The problem of generalization is a central one in machine learning.

In this course, we will study a variety of ways to improve generalization, including

- Choice of architecture
- Regularization
- Data augmentation
- Optimization methods

Classification vs Regression

Let us consider a K-class classification problem

What should we understand as a linear model in this case?

The Argmax Way

Linear models for regression

$$y = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})$$

Linear models for classification

$$y_i = \boldsymbol{w}_i^T \boldsymbol{\phi}(\boldsymbol{x}) \;\; i = 1,2,\dots,K$$

$$\text{Prediction: } \underset{i=1,\dots,K}{\arg\max}\, y_i$$

Is this the same as SVM (Binary classification by hyperplanes)?

$$\mathrm{Sign}\left(\boldsymbol{w}^T \boldsymbol{x} + b\right)$$

Limitations of This Approach

Let’s write a linear K-class classifier in a more compact way

- Define a matrix $W \in \mathbb{R}^{K \times m}$
- Define an argmax function $g: \mathbb{R}^K \to \{1, \dots, K\}$ by

  $$g(\boldsymbol{z}) = i \ \text{ if } \ z_i > z_j \text{ for all } j \neq i$$

  To break ties, we can alternatively define

  $$g(\boldsymbol{z}) = \underset{i=1,\dots,K}{\arg\min} \left\{ z_i : z_i = \max_{j=1,\dots,K} z_j \right\}$$
- A linear classification model is thus

  $$f: \mathbb{R}^d \to \{1,2, \dots, K\}$$

  $$f(\boldsymbol{x}) = g\big(W\boldsymbol{\phi}(\boldsymbol{x})\big)$$

Limitations of This Approach

We can train these networks by defining a loss, say square error and minimize

$$L(W) = \frac{1}{2}\left(y_i - g\left(W\boldsymbol{\phi}(\boldsymbol{x}_{\mathbf{i}})\right)\right)^2$$

No exact solution, so we perform gradient descent

$$W_{t+1} = W_t - \eta\nabla_W L(W)$$

What’s wrong?

Alternatively: One-hot Encoding and Softmax Activation

We can use the one-hot encoding to represent each label $y_i = k$ (class k) as a vector in $\mathbb{R}^K$

$$\boldsymbol{y}_i = (0, \dots, 0, \underbrace{1,}_{\text{k}^{\text{th}}\text{ Position}} 0, \dots, 0) \in \mathbb{R}^K, \qquad i = 1, \dots, N$$

Then, our classification model can return a row of probabilities of a sample belong to each class, e.g.

$$f(\boldsymbol{x}_i) = (a_1, \dots, \underbrace{a_k,}_{\text{Probability of belong to class } k} \dots, a_K) \in \mathbb{R}^K, \qquad i = 1, \dots, N$$

This can be done by using the **softmax** function $s: \mathbb{R}^K \rightarrow \mathbb{R}^K$

$$s(\boldsymbol{z})_k = \frac{\exp(z_k)}{\sum_j \exp(z_j)}$$

What does it do?

This gives the following hypothesis space

$$\mathcal{H} = \left\{ f(\boldsymbol{x}) = s\big(W\boldsymbol{\phi}(\boldsymbol{x})\big) : W \in \mathbb{R}^{K \times m} \right\}$$

Notice that $f \in \mathcal{H}$ always outputs a vector which can be interpreted as **probabilities** over K classes

Loss Functions for Classification

By right, classification should be done with zero-one loss: 1 if incorrect, 0 if correct. This is just accuracy!

Why do we not use this?

Instead, we consider loss surrogate

$$L: \mathbb{R}^K \times \mathbb{R}^K \rightarrow \mathbb{R}_+$$

where

- If $\boldsymbol{y} = \boldsymbol{y}'$, then $L(\boldsymbol{y}, \boldsymbol{y}')$=0
- If $\boldsymbol{y} \approx \boldsymbol{y}'$, then $L(\boldsymbol{y}, \boldsymbol{y}')$ is small
- L is differentiable, with non (almost) everywhere zero gradients

Examples

We can still use the square loss

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

What could be some issues with this?

The cross-entropy loss is defined as

$$L(\boldsymbol{y}, \boldsymbol{y}') = -\sum_{k=1}^{K} y_k' \log y_k$$

Theses are not the only choices! Any differentiable "distance" between two probability distributions suffices.

Summary

- The T, E, P formulation of machine learning
- Linear regression as an illustration
  - Hypothesis space and Capacity
  - Learning algorithm
  - Underfitting and Overfitting
  - Performance metrics and test set
- Classification: use one-hot encoding, compare probabilities

Useful tools for programming

Version control with **Git**

- https://www.freecodecamp.org/news/what-is-git-and-how-to-use-it-c341b049ae61/

Interactive python with **Jupyter notebooks**

- https://www.datacamp.com/community/tutorials/tutorial-jupyter-notebook

Data visualization using **Seaborn** and **Pandas**

- https://jakevdp.github.io/PythonDataScienceHandbook/04.14-visualization-with-seaborn.html

Demo (Linear Regression)

See uploaded under folder “Demos”