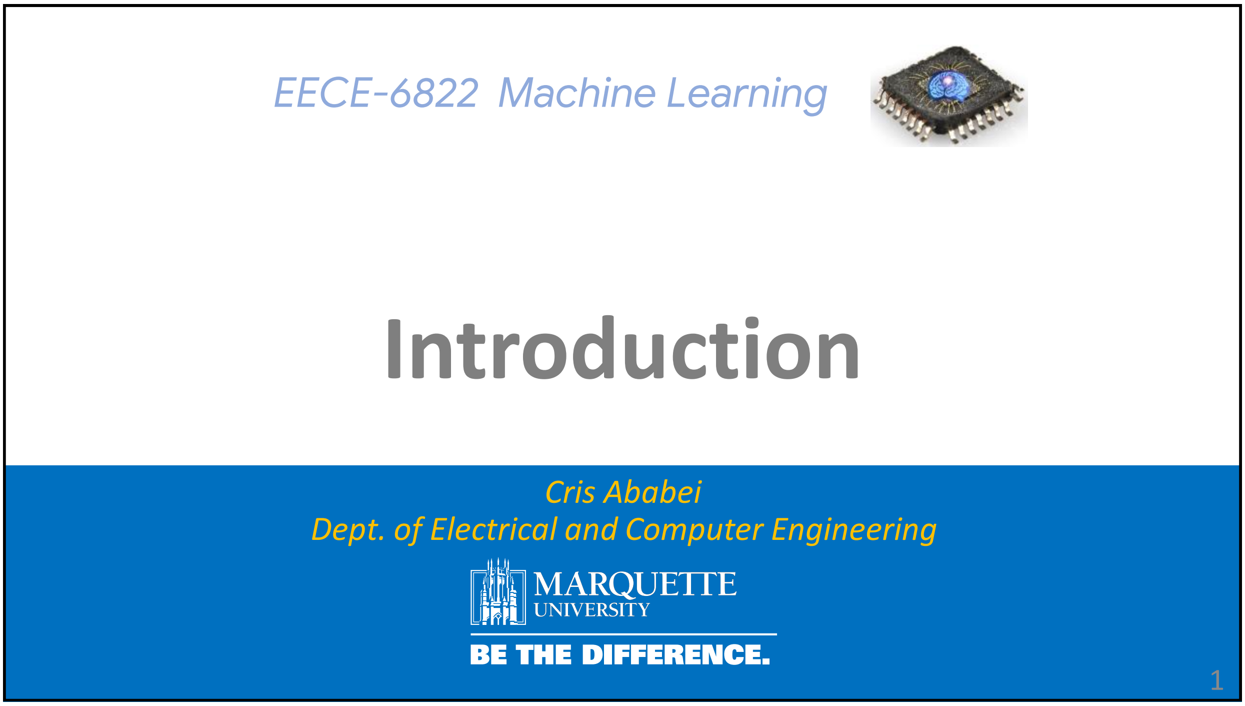
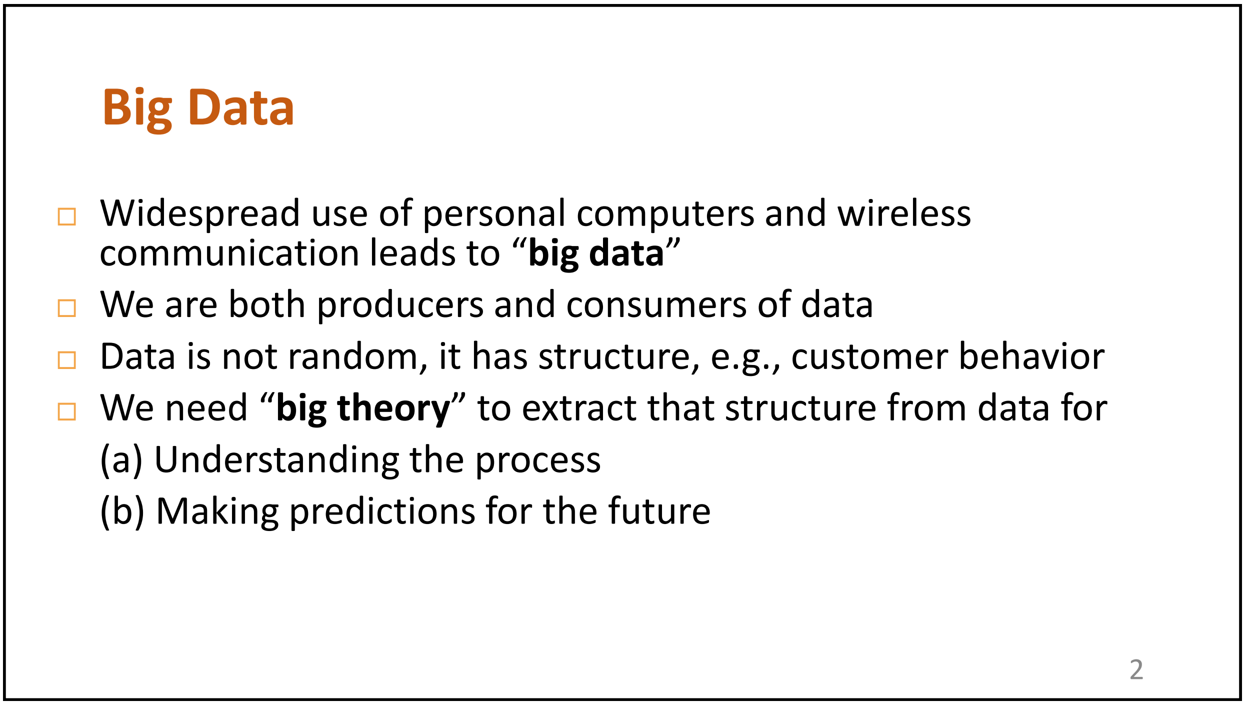


EECE-6822 Machine Learning

Introduction

Cris Ababei
Dept. of Electrical and Computer Engineering

MARQUETTE UNIVERSITY
BE THE DIFFERENCE.



Big Data

- □ Widespread use of personal computers and wireless communication leads to “**big data**”
- □ We are both producers and consumers of data
- □ Data is not random, it has structure, e.g., customer behavior
- □ We need “**big theory**” to extract that structure from data for

  (a) Understanding the process

  (b) Making predictions for the future

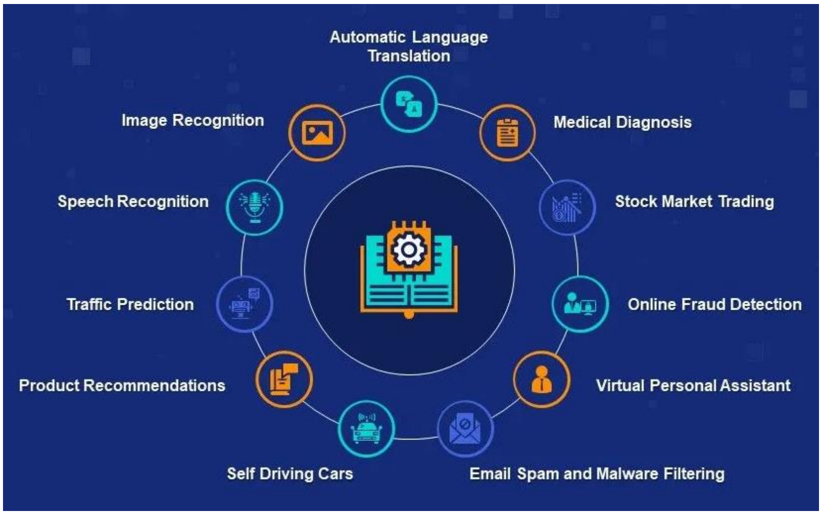
PART 1

Introduction to Machine Learning (ML)

3

3

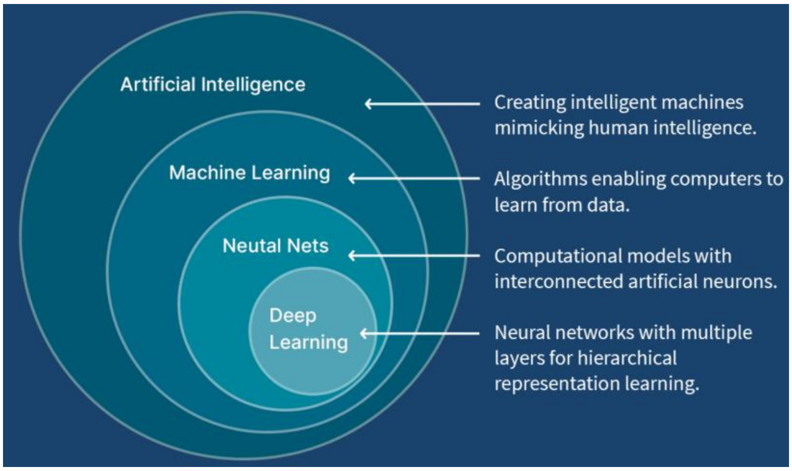
Applications of Machine Learning



Source: https://swisscognitive.ch/2021/03/18/applications-of-machine-learning

4

4



Creating intelligent machines mimicking human intelligence.

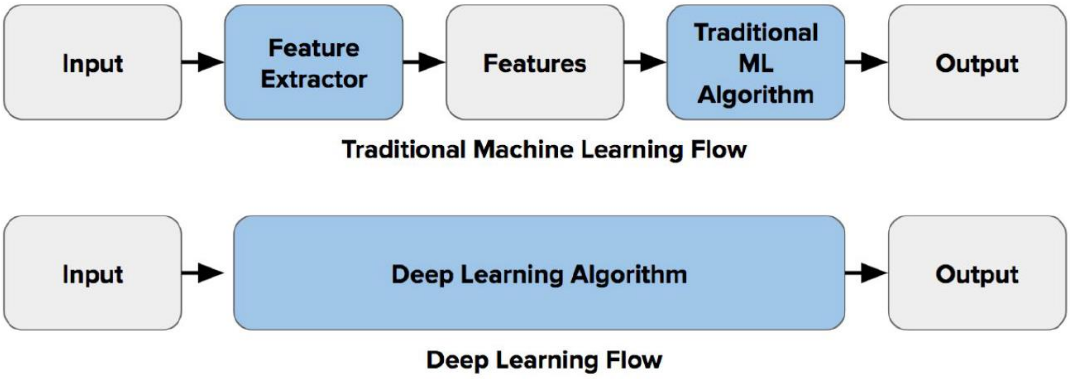
Algorithms enabling computers to learn from data.

Computational models with interconnected artificial neurons.

Neural networks with multiple layers for hierarchical representation learning.

- **Artificial Intelligence (AI)** encompasses any technique that enables computers to mimic human behavior.
- **Machine Learning (ML)** is a subfield of AI focused on developing algorithms that learn to solve problems by analyzing data for patterns.
- **Deep Learning (DL)** is a type of ML that leverages **Neural Networks (NN)** and Big Data.

Deep vs Classical Learning



Traditional Machine Learning Flow

Deep Learning Flow

https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

"Deep Learning automates the design and interactions of features by constructing deep networks of nonlinearly activated, connected neurons"

Machine Learning

- Machine learning can be broadly defined as **computational methods** using **experience** to improve/**optimize performance** **or** to make **accurate predictions**.
- **Experience** refers to the past information available to the learner, which typically takes the form of **electronic data** collected and made available for analysis.
- **Data** could be in the form of digitized human-labeled training sets, or other types of information obtained via interaction with the environment.
- Build a **model** that is **a good and useful approximation** to the data.

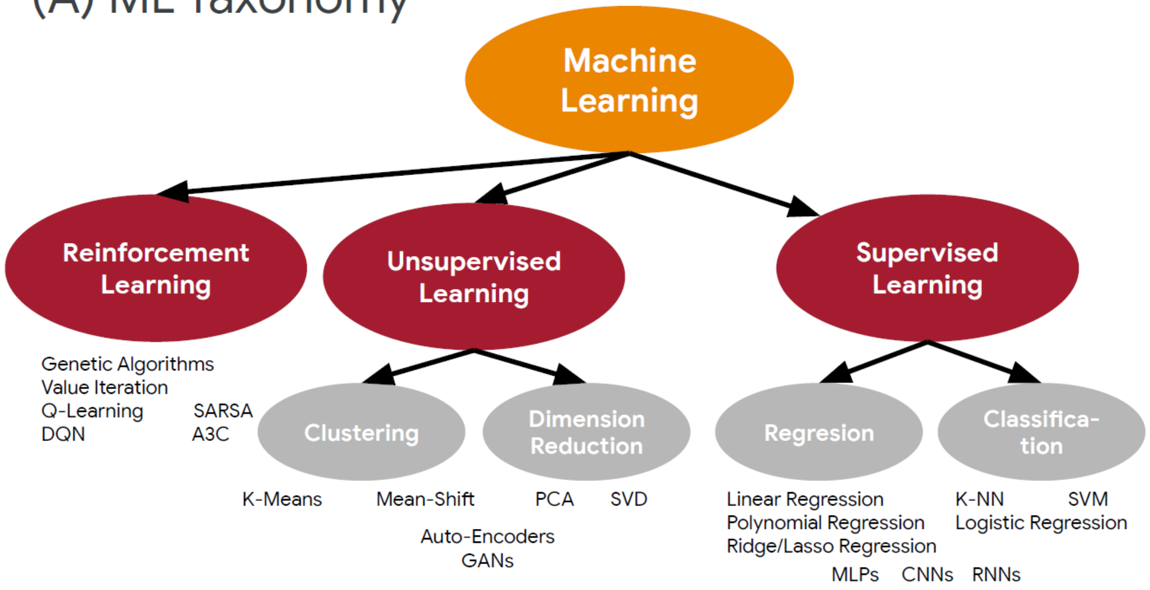
Machine Learning Ingredients

- Data: past observations
- Hypotheses/Models: devised to capture the patterns in data
- Algorithm: method to find the hypothesis that best captures the pattern/target $\boldsymbol{f}$ in data
- Prediction: apply model to forecast future observations
- Role of Statistics:
  - Inference from a sample
- Role of Computer Science: Efficient algorithms to
  - Solve the optimization problem
  - Representing and evaluating the model for inference

Types of Learning Problems

- Supervised
  - Given training samples, which are input-output pairs, find a deterministic function that maps any input to an output such that disagreement with future input-output observations is minimized
- Unsupervised
  - Given training samples (objects), but without associated outputs, find some structure in the training samples
- Reinforcement
  - Similar to supervised learning, but the associated output, also called reward or reinforcement, is not necessarily given immediately. Goal is to maximize the reward.

(A) ML Taxonomy



Machine Learning

Reinforcement Learning

Genetic Algorithms
Value Iteration
Q-Learning SARSA
DQN A3C

Unsupervised Learning

Clustering

K-Means Mean-Shift

Dimension Reduction

PCA SVD

Auto-Encoders
GANs

Supervised Learning

Regresion

Linear Regression
Polynomial Regression
Ridge/Lasso Regression

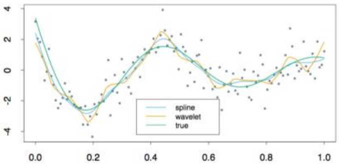
Classifica-tion

K-NN SVM
Logistic Regression

MLPs CNNs RNNs

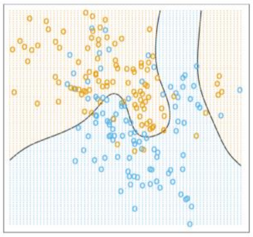
For detailed discussion of this (with examples) see: https://a2r-lab.org/files/CS249_F20_Slides.pdf

Flavors of ML



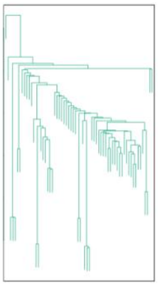
Regression

Predict continuous value:
ex: stock market, credit score, temperature, Netflix rating



Classification

Predict categorical value:
loan or not? spam or not? what disease is this?



Unsupervised Learning

Predict structure:
tree of life from DNA, find similar images, community detection

Self-supervised:
Model distribution of large-scale data, like text or images (large vision and language models)

Mix of statistics (conceptual) and algorithms (programming)

Example: Credit Approval

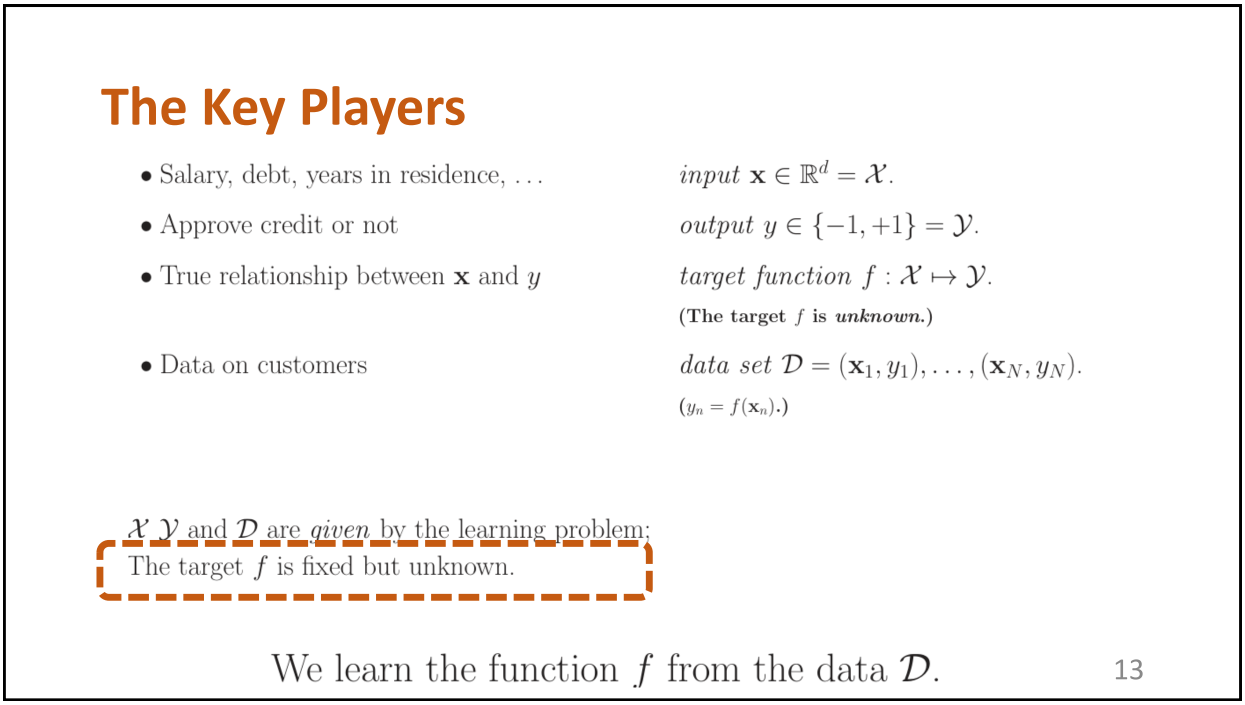
Let's use a conceptual example to crystallize the issues.

- Using salary, debt, years in residence, etc., approve for credit or not.
- No magic credit approval formula.
- Banks have lots of data.
  - customer information: salary, debt, etc.
  - whether or not they defaulted on their credit.

age	32 years
gender	male
salary	40,000
debt	26,000
years in job	1 year
years at home	3 years
. . .	. . .

Approve for credit?

A pattern exists. We don't know it. We have data to learn it.

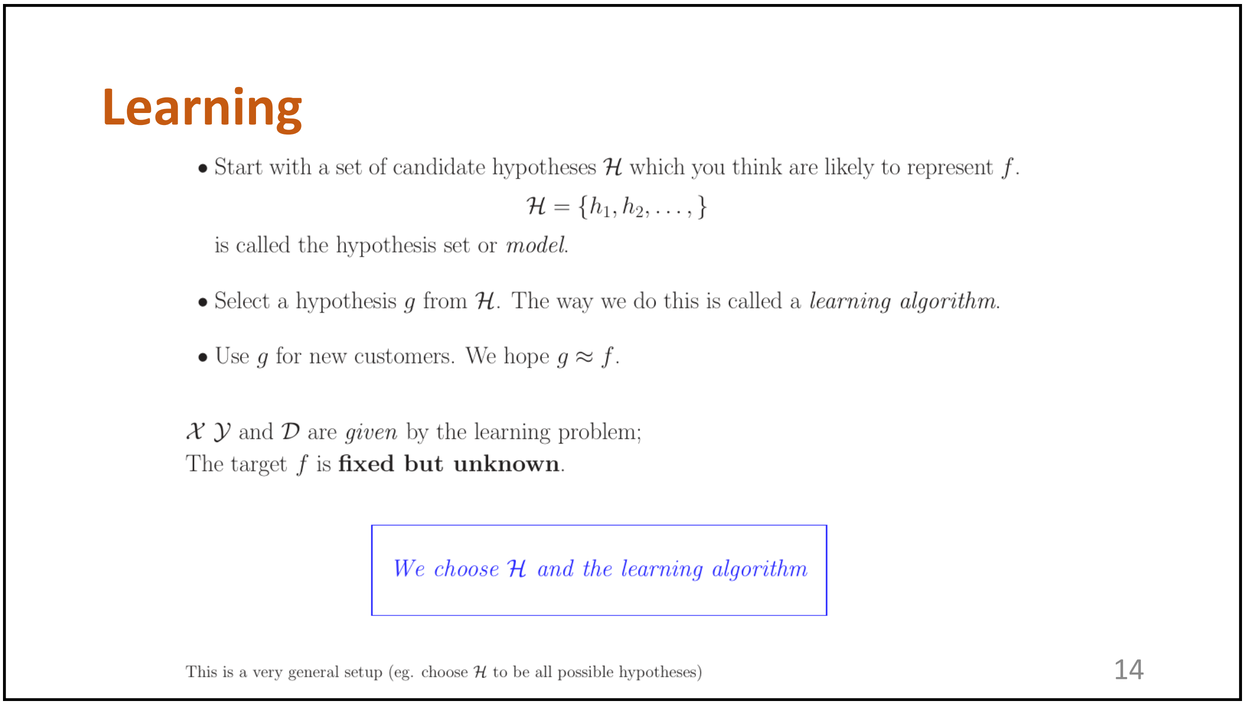


The Key Players

• Salary, debt, years in residence, ...	*input* $\mathbf{x} \in \mathbb{R}^d = \mathcal{X}$.
• Approve credit or not	*output* $y \in \{-1, +1\} = \mathcal{Y}$.
• True relationship between $\mathbf{x}$ and y	*target function* $f : \mathcal{X} \mapsto \mathcal{Y}$. **(The target f is *unknown*.)**
• Data on customers	*data set* $\mathcal{D} = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$. ($y_n = f(\mathbf{x}_n)$.)

$\mathcal{X}$ $\mathcal{Y}$ and $\mathcal{D}$ are *given* by the learning problem;
The target f is fixed but unknown.

We learn the function f from the data $\mathcal{D}$.



Learning

- Start with a set of candidate hypotheses $\mathcal{H}$ which you think are likely to represent f.

$$\mathcal{H} = \{h_1, h_2, \ldots, \}$$

is called the hypothesis set or *model.*

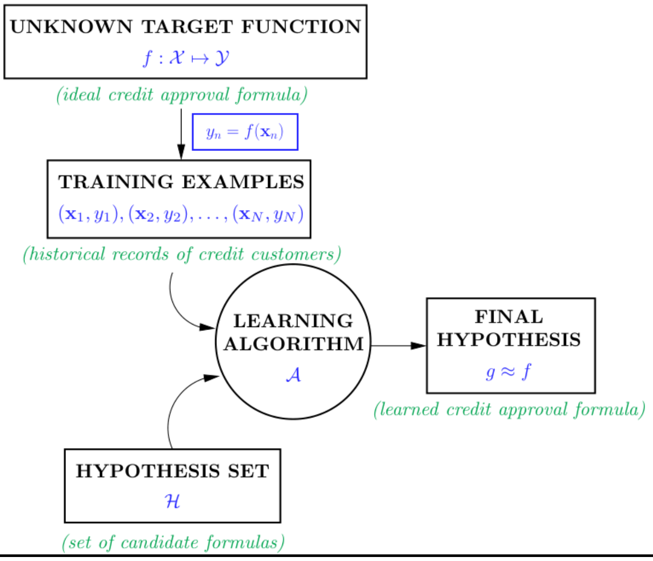
- Select a hypothesis g from $\mathcal{H}$. The way we do this is called a *learning algorithm.*
- Use g for new customers. We hope $g \approx f$.

$\mathcal{X}$ $\mathcal{Y}$ and $\mathcal{D}$ are *given* by the learning problem;
The target f is **fixed but unknown**.

We choose $\mathcal{H}$ and the learning algorithm

This is a very general setup (eg. choose $\mathcal{H}$ to be all possible hypotheses)

Summary of the Learning Setup



UNKNOWN TARGET FUNCTION

$f : \mathcal{X} \mapsto \mathcal{Y}$

(ideal credit approval formula)

$y_n = f(\mathbf{x}_n)$

TRAINING EXAMPLES

$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$

(historical records of credit customers)

LEARNING ALGORITHM

$\mathcal{A}$

FINAL HYPOTHESIS

$g \approx f$

(learned credit approval formula)

HYPOTHESIS SET

$\mathcal{H}$

(set of candidate formulas)

Solution Components

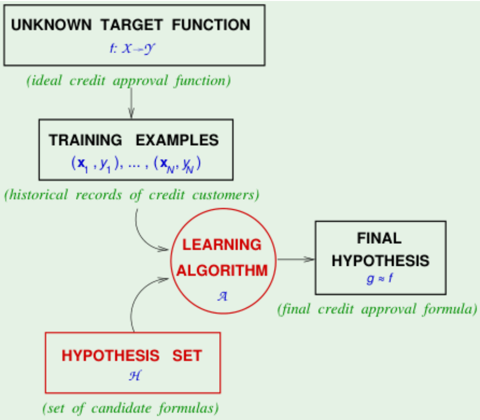
The 2 solution components of the learning problem:

- The Hypothesis Set

  $\mathcal{H} = \{h\} \qquad g \in \mathcal{H}$

- The Learning Algorithm

Together, they are referred to as the *learning model*.



UNKNOWN TARGET FUNCTION

$f: \mathcal{X} \to \mathcal{Y}$

(ideal credit approval function)

TRAINING EXAMPLES

$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

(historical records of credit customers)

LEARNING ALGORITHM

$\mathcal{A}$

FINAL HYPOTHESIS

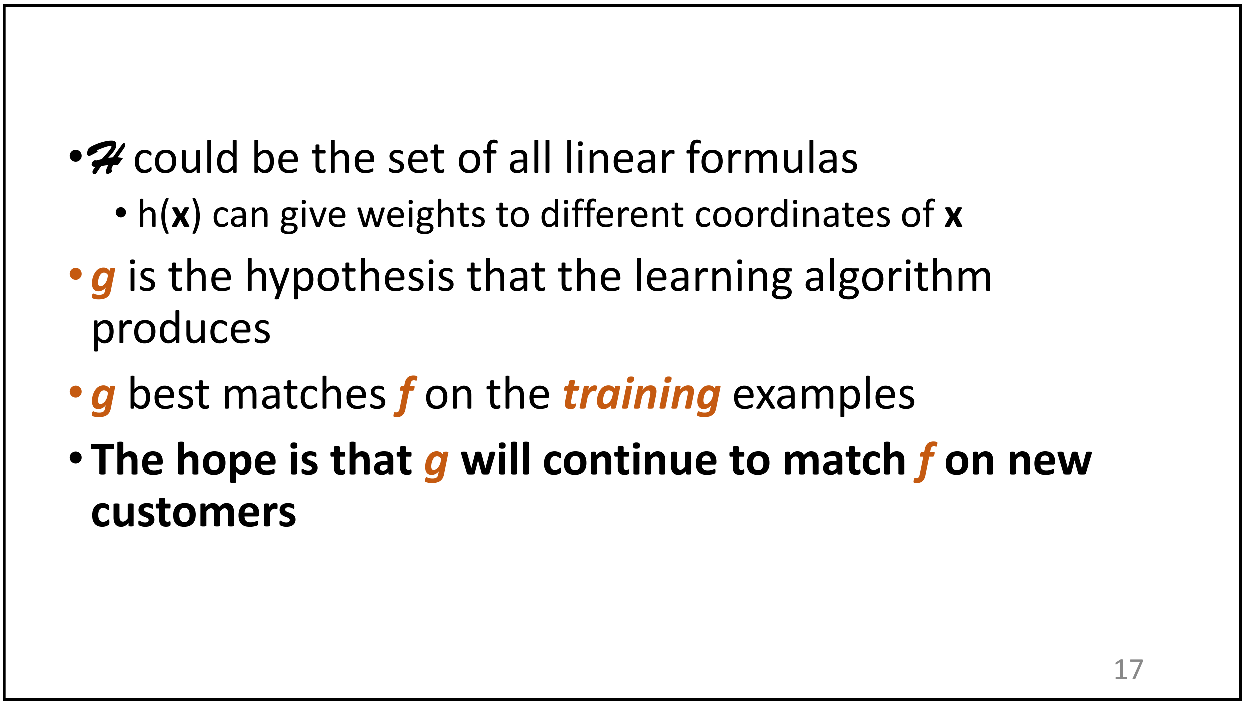
$g \approx f$

(final credit approval formula)

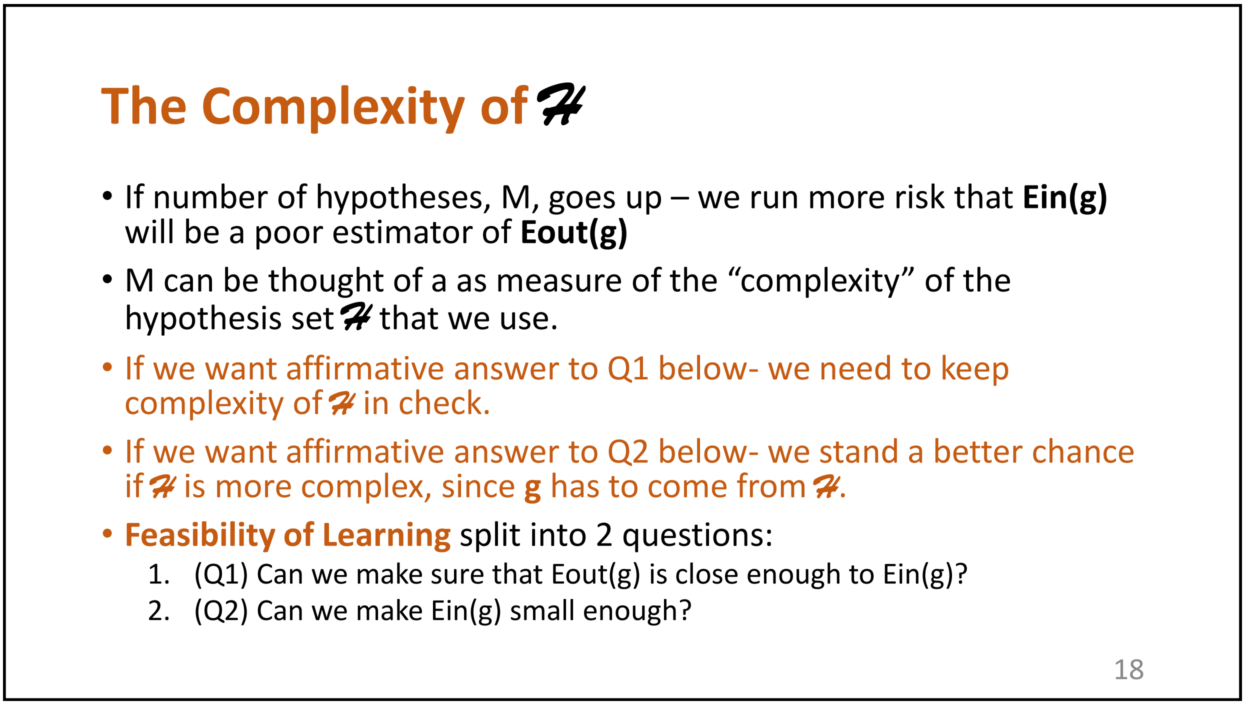
HYPOTHESIS SET

$\mathcal{H}$

(set of candidate formulas)



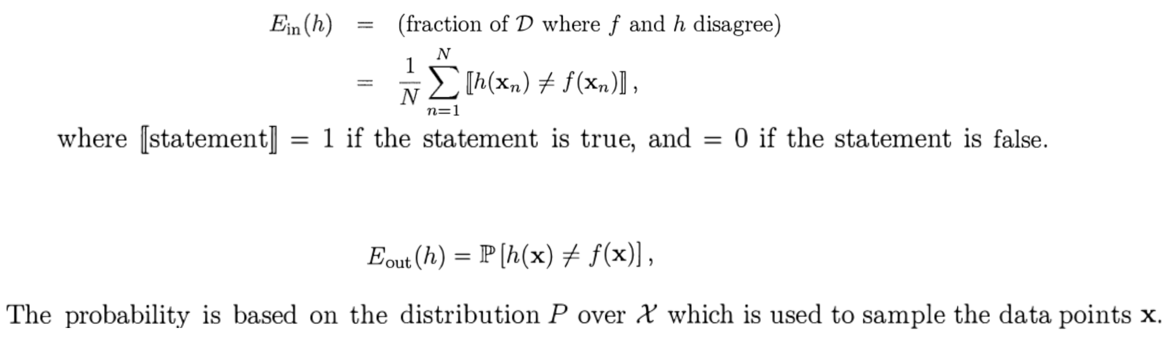
- $\mathcal{H}$ could be the set of all linear formulas
  - h(**x**) can give weights to different coordinates of **x**
- ***g*** is the hypothesis that the learning algorithm produces
- ***g*** best matches ***f*** on the ***training*** examples
- **The hope is that *g* will continue to match *f* on new customers**



The Complexity of $\mathcal{H}$

- If number of hypotheses, M, goes up – we run more risk that **Ein(g)** will be a poor estimator of **Eout(g)**
- M can be thought of a as measure of the “complexity” of the hypothesis set $\mathcal{H}$ that we use.
- If we want affirmative answer to Q1 below- we need to keep complexity of $\mathcal{H}$ in check.
- If we want affirmative answer to Q2 below- we stand a better chance if $\mathcal{H}$ is more complex, since **g** has to come from $\mathcal{H}$.
- **Feasibility of Learning** split into 2 questions:
  1. (Q1) Can we make sure that Eout(g) is close enough to Ein(g)?
  2. (Q2) Can we make Ein(g) small enough?

In-sample Error **Ein(g)**
Out-of-sample Error **Eout(g)**



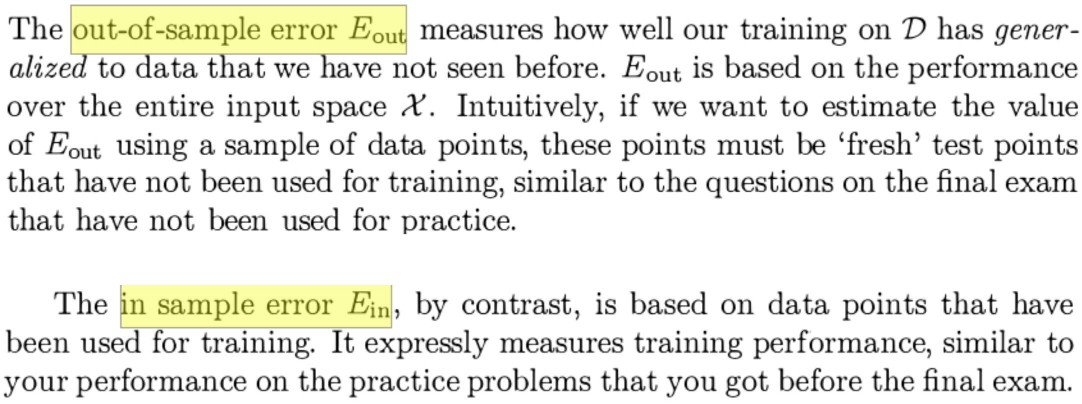
$$E_{\text{in}}(h) = (\text{fraction of } \mathcal{D} \text{ where } f \text{ and } h \text{ disagree}) = \frac{1}{N}\sum_{n=1}^{N} [\![h(\mathbf{x}_n) \neq f(\mathbf{x}_n)]\!] ,$$

where $[\![\text{statement}]\!] = 1$ if the statement is true, and $= 0$ if the statement is false.

$$E_{\text{out}}(h) = \mathbb{P}\left[h(\mathbf{x}) \neq f(\mathbf{x})\right],$$

The probability is based on the distribution P over $\mathcal{X}$ which is used to sample the data points $\mathbf{x}$.

In-sample Error **Ein(g)**
Out-of-sample Error **Eout(g)**



The out-of-sample error E_{out} measures how well our training on $\mathcal{D}$ has *generalized* to data that we have not seen before. E_{out} is based on the performance over the entire input space $\mathcal{X}$. Intuitively, if we want to estimate the value of E_{out} using a sample of data points, these points must be 'fresh' test points that have not been used for training, similar to the questions on the final exam that have not been used for practice.

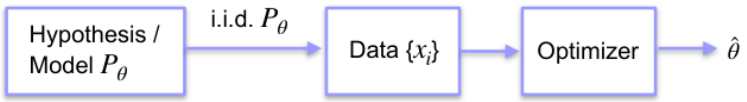
The in sample error E_{in}, by contrast, is based on data points that have been used for training. It expressly measures training performance, similar to your performance on the practice problems that you got before the final exam.

Views of Learning

- Difference between **Learning approach** and **Design approach**
  - Data plays different roles
  - Design approach: problem is well specified and one can analytically derive ***f*** without the need for data
  - Learning approach: problem is less specified, and one needs data to pin down what ***f*** is
- **Statistics** shares the basic premise of learning from data
  - Focuses somewhat on idealized models and analyzes them in great detail
- **Data mining** focuses on finding patterns, correlations or anomalies in large relational databases
  - Is the same as learning from data, where more emphasis is on data analysis than on prediction

Another Way of Looking at Learning

- Learning is…
  - Collect some data
    - E.g., coin flips
  - Choose a hypothesis class or model
    - E.g., Bernoulli
  - Choose a loss function
    - E.g., data likelihood
  - Choose an optimization procedure
    - E.g., set derivative to zero to obtain MLE

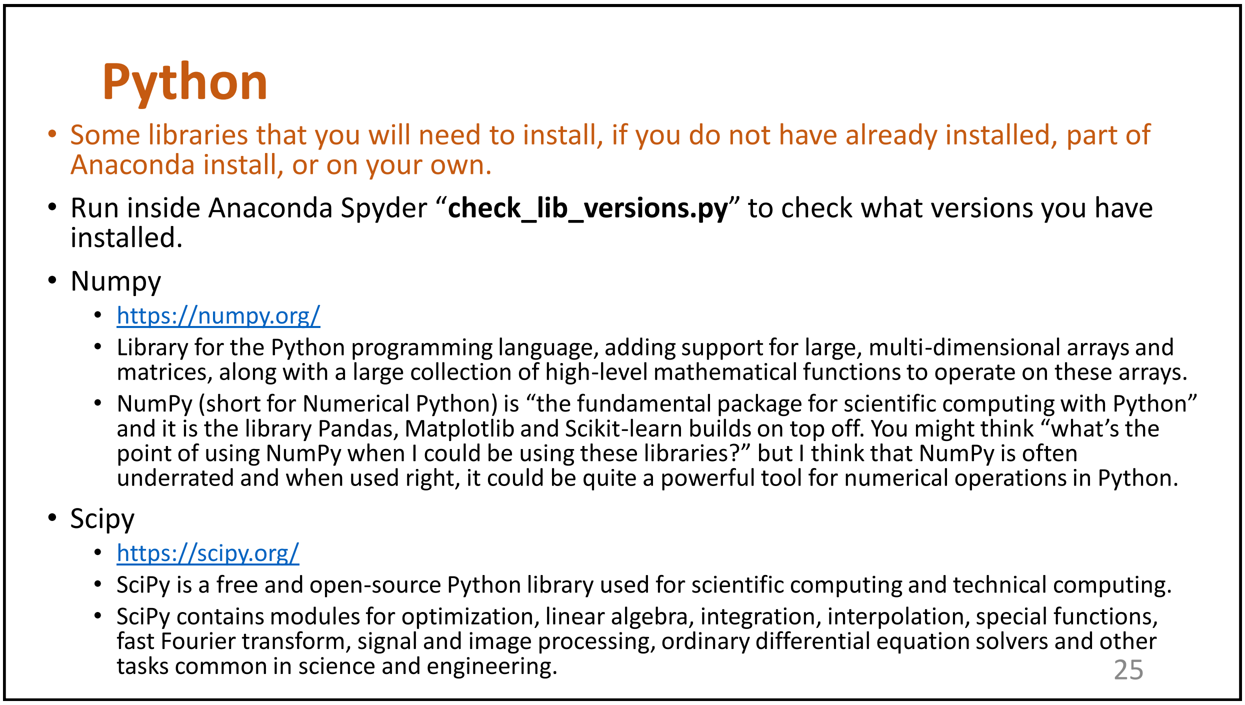


Coding Applications

Python and More

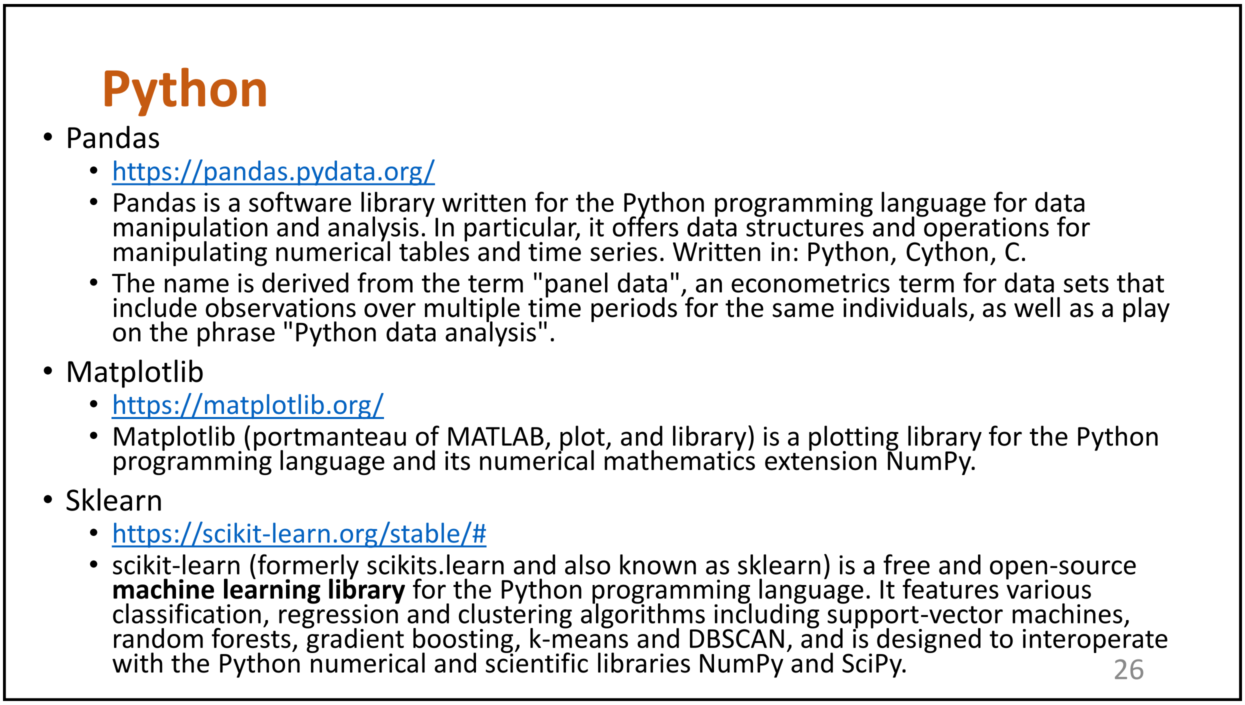
Useful Libraries and Tools

- Jupyter Notebook
  - Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.
- Google Colab
  - Google Colab is a free, cloud-based service that allows users to write and execute Python code in a browser-based environment, essentially a hosted Jupyter Notebook. It provides free access to computing resources, including GPUs and TPUs, and is well-suited for machine learning, data science, and educational purposes.
- Anaconda Spyder
  - Spyder is a powerful scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It features a unique combination of the advanced editing, analysis, debugging and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection and beautiful visualization capabilities of a scientific package.
- VS Code
  - You are free to use this or any other IDE for programming in Python



Python

- Some libraries that you will need to install, if you do not have already installed, part of Anaconda install, or on your own.
- Run inside Anaconda Spyder “**check_lib_versions.py**” to check what versions you have installed.
- Numpy
  - https://numpy.org/
  - Library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
  - NumPy (short for Numerical Python) is “the fundamental package for scientific computing with Python” and it is the library Pandas, Matplotlib and Scikit-learn builds on top off. You might think “what’s the point of using NumPy when I could be using these libraries?” but I think that NumPy is often underrated and when used right, it could be quite a powerful tool for numerical operations in Python.
- Scipy
  - https://scipy.org/
  - SciPy is a free and open-source Python library used for scientific computing and technical computing.
  - SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, fast Fourier transform, signal and image processing, ordinary differential equation solvers and other tasks common in science and engineering.



Python

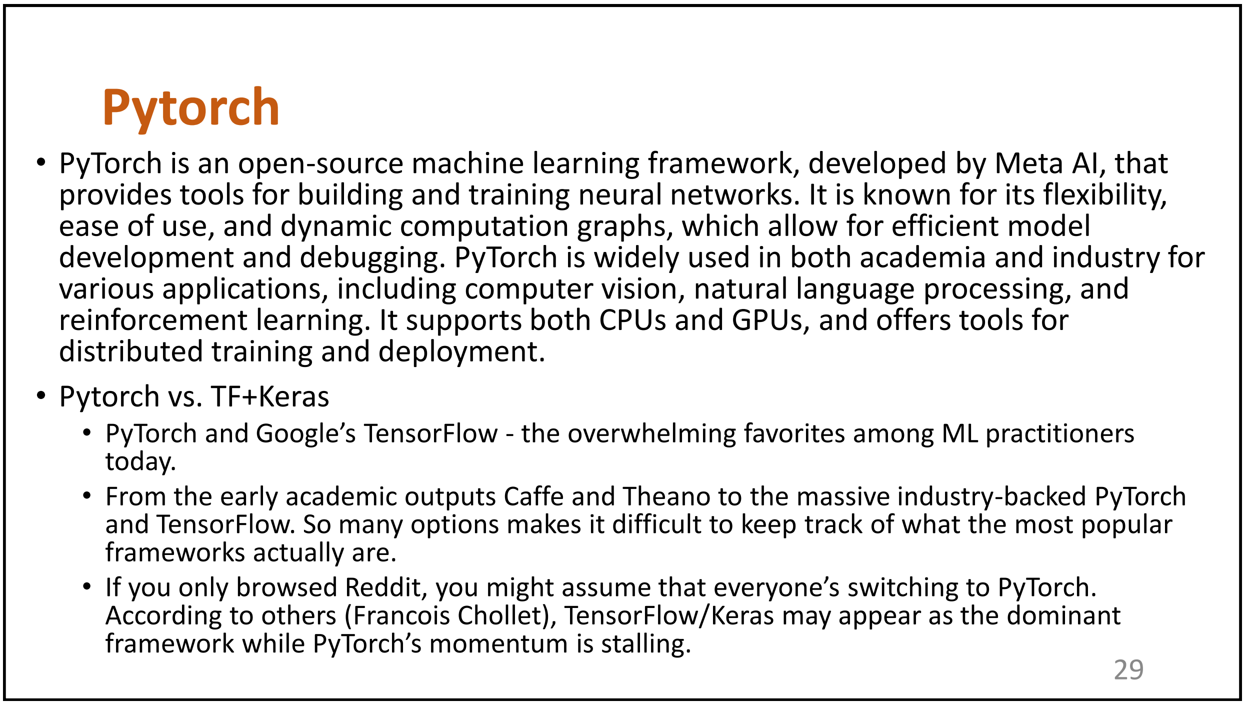
- Pandas
  - https://pandas.pydata.org/
  - Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. Written in: Python, Cython, C.
  - The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals, as well as a play on the phrase "Python data analysis".
- Matplotlib
  - https://matplotlib.org/
  - Matplotlib (portmanteau of MATLAB, plot, and library) is a plotting library for the Python programming language and its numerical mathematics extension NumPy.
- Sklearn
  - https://scikit-learn.org/stable/#
  - scikit-learn (formerly scikits.learn and also known as sklearn) is a free and open-source **machine learning library** for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

TensorFlow (TF) 2.0

- Free and open-source library for numerical computation and large-scale machine learning, particularly deep learning. It allows developers to build and deploy machine learning models across various platforms, including servers, edge devices, and the web. TensorFlow is widely used for tasks like image recognition, natural language processing, and computational simulations.
- On open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them.
- The name "TensorFlow" is derived from the operations which neural networks perform on multidimensional data arrays or tensors! It's literally a flow of tensors.
- A tensor, then, is the mathematical representation of a physical entity that may be characterized by magnitude and multiple directions.
- TensorFlow provides APIs for Python, C++, Haskell, Java, Go, Rust.

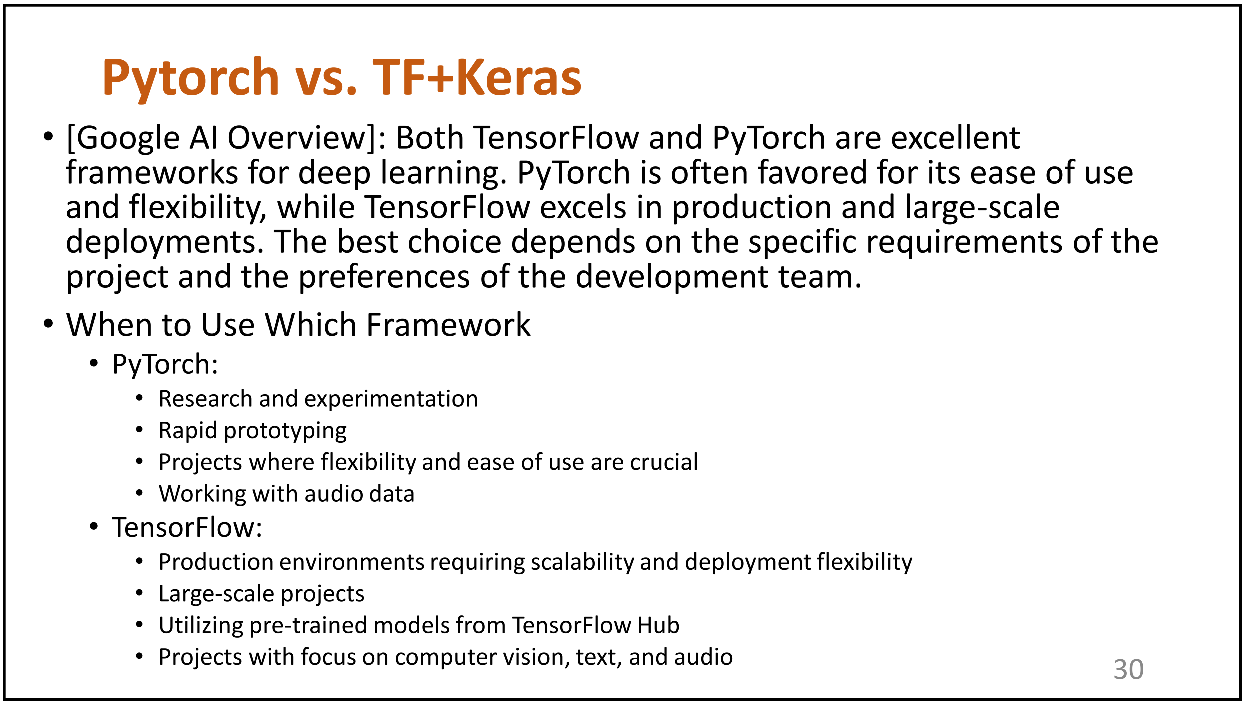
Keras → TensorFlow 2.0

- Keras is an open-source deep learning library written in Python. Keras was initially an independent library, it was later incorporated into TensorFlow 2.0.
- The project was started in 2015 by Francois Chollet. It quickly became a popular framework for developers, becoming one of, if not the most, popular deep learning libraries.
- Between 2015 and 2019, developing deep learning models using mathematical libraries like TensorFlow, Theano, and PyTorch was cumbersome, requiring tens or even hundreds of lines of code to achieve the simplest tasks. The focus of these libraries was on research, flexibility, and speed, not ease of use.
- This integration is commonly referred to as the tf.keras interface or API ("tf" is short for "TensorFlow"). This is to distinguish it from the so-called standalone Keras open source project.
  - Standalone Keras: The standalone open source project that supports TensorFlow, Theano, and JAX backends
  - tf.keras: The Keras API integrated into TensorFlow 2
- Nowadays, since the features of other backends are dwarfed by TensorFlow 2, the latest Keras library supports only TensorFlow, and these two are the same.



Pytorch

- PyTorch is an open-source machine learning framework, developed by Meta AI, that provides tools for building and training neural networks. It is known for its flexibility, ease of use, and dynamic computation graphs, which allow for efficient model development and debugging. PyTorch is widely used in both academia and industry for various applications, including computer vision, natural language processing, and reinforcement learning. It supports both CPUs and GPUs, and offers tools for distributed training and deployment.
- Pytorch vs. TF+Keras
  - PyTorch and Google's TensorFlow - the overwhelming favorites among ML practitioners today.
  - From the early academic outputs Caffe and Theano to the massive industry-backed PyTorch and TensorFlow. So many options makes it difficult to keep track of what the most popular frameworks actually are.
  - If you only browsed Reddit, you might assume that everyone's switching to PyTorch. According to others (Francois Chollet), TensorFlow/Keras may appear as the dominant framework while PyTorch's momentum is stalling.



Pytorch vs. TF+Keras

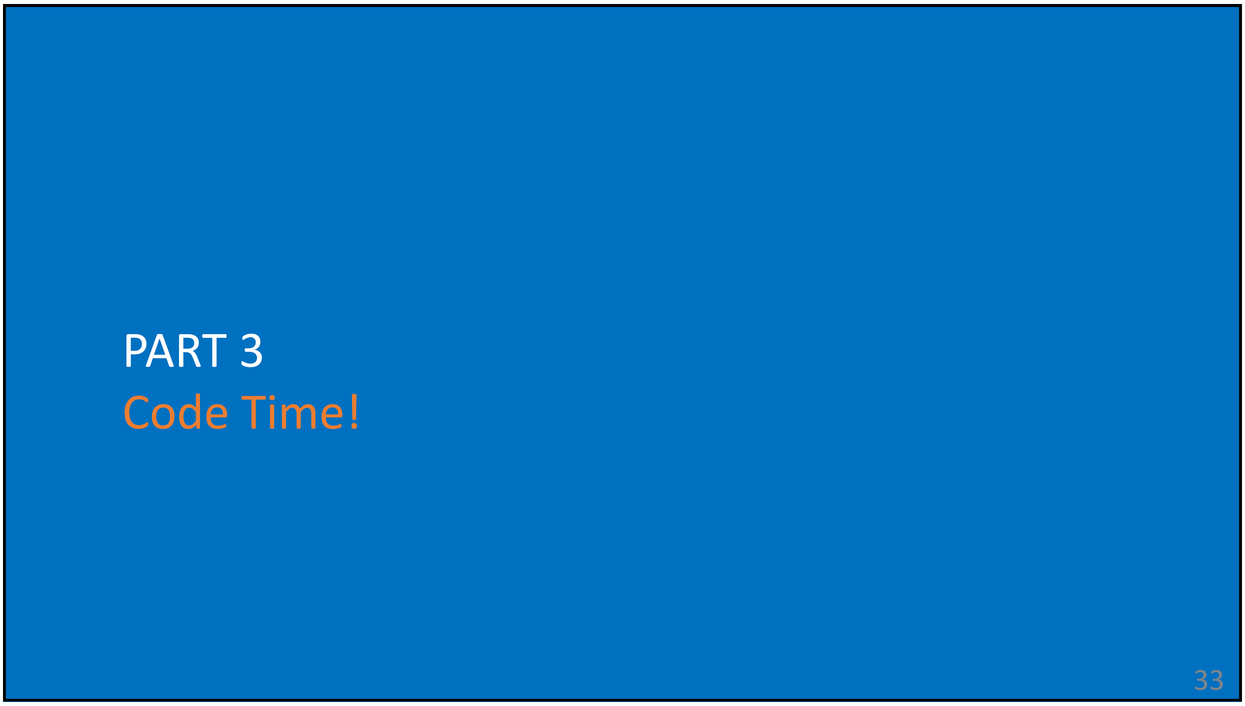
- [Google AI Overview]: Both TensorFlow and PyTorch are excellent frameworks for deep learning. PyTorch is often favored for its ease of use and flexibility, while TensorFlow excels in production and large-scale deployments. The best choice depends on the specific requirements of the project and the preferences of the development team.
- When to Use Which Framework
  - PyTorch:
    - Research and experimentation
    - Rapid prototyping
    - Projects where flexibility and ease of use are crucial
    - Working with audio data
  - TensorFlow:
    - Production environments requiring scalability and deployment flexibility
    - Large-scale projects
    - Utilizing pre-trained models from TensorFlow Hub
    - Projects with focus on computer vision, text, and audio

PART 2

Review of basics in probability, random variables, statistics

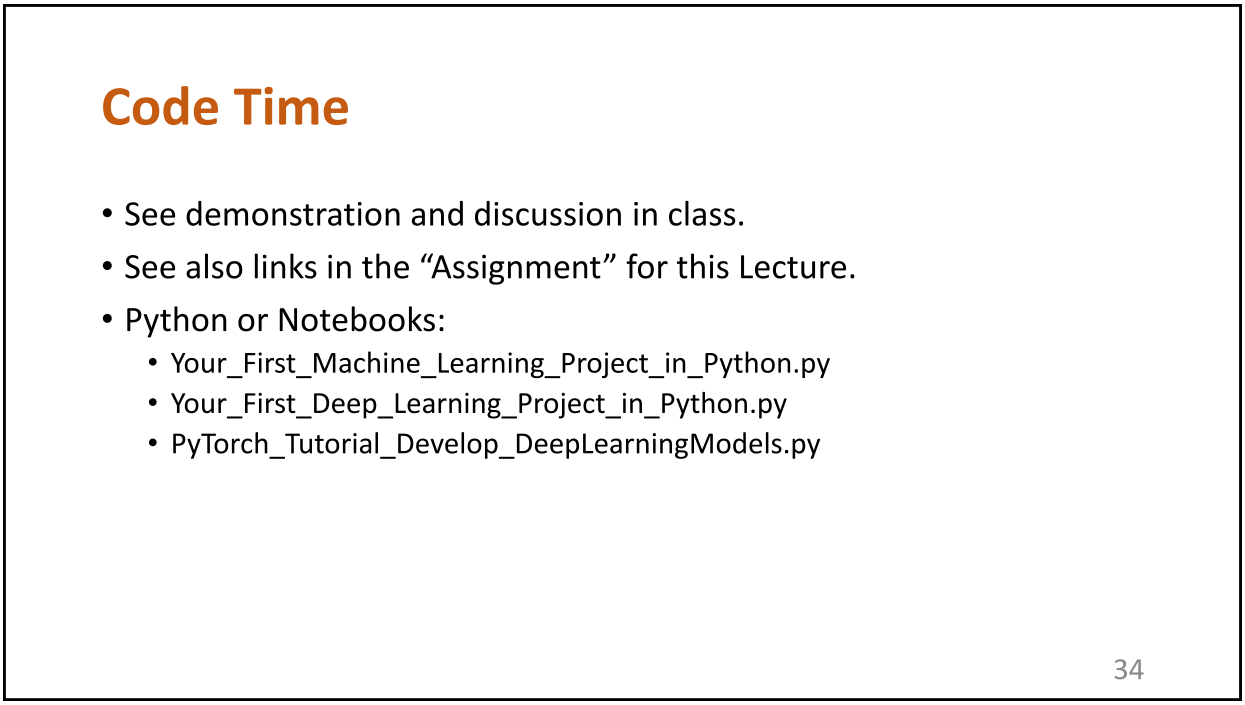
Review

- See hand-written notes and presentation in class.
- See also links in the “Assignment” for this Lecture.
- Python or Notebooks:
  - check_lib_versions.py (mlm)
  - Python_Intro.ipynb (uwash)
  - Linear_Algebra_Review.ipynb (uwash)
  - math_linear_algebra.ipynb (geron)
  - math_differential_calculus.ipynb (geron)



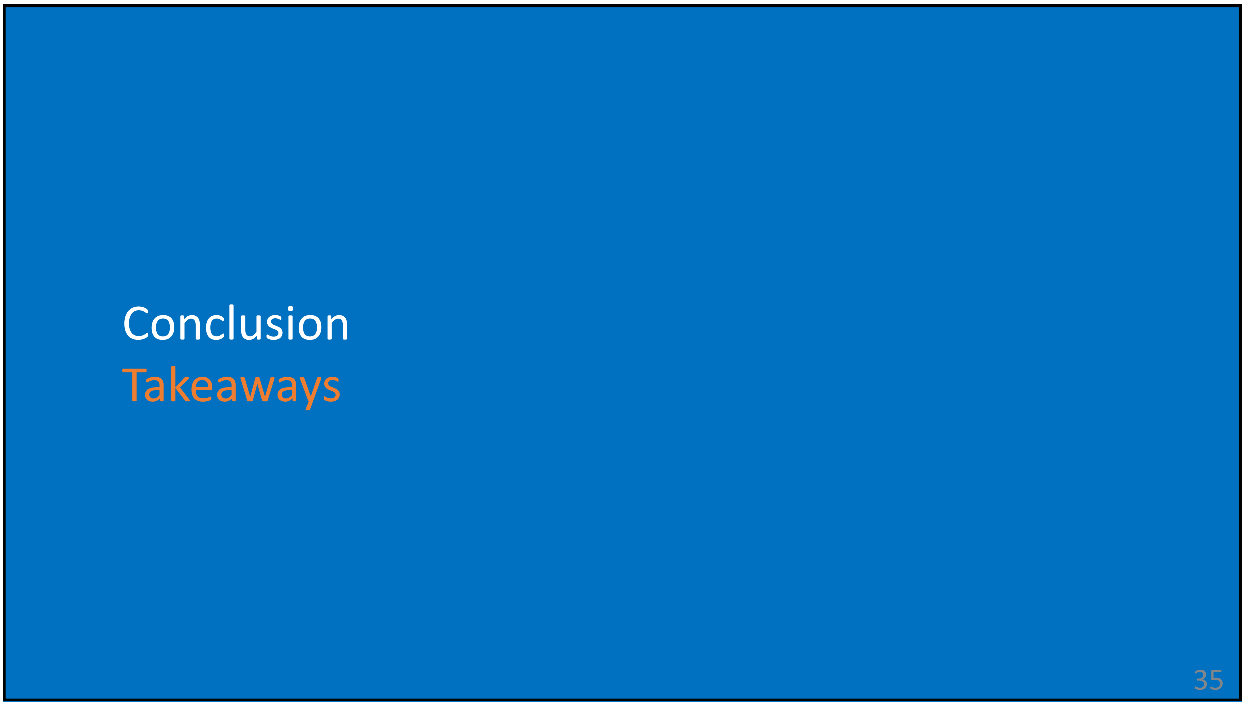
PART 3

Code Time!



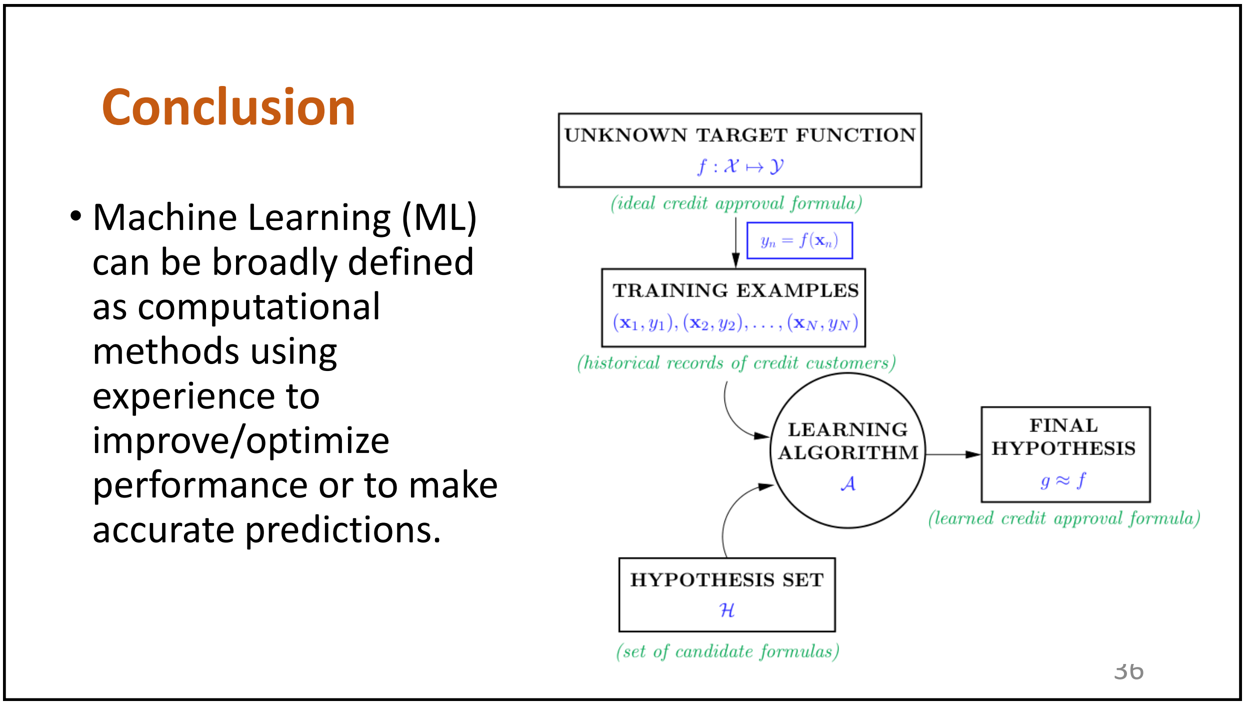
Code Time

- See demonstration and discussion in class.
- See also links in the “Assignment” for this Lecture.
- Python or Notebooks:
  - Your_First_Machine_Learning_Project_in_Python.py
  - Your_First_Deep_Learning_Project_in_Python.py
  - PyTorch_Tutorial_Develop_DeepLearningModels.py



Conclusion

Takeaways



Conclusion

- Machine Learning (ML) can be broadly defined as computational methods using experience to improve/optimize performance or to make accurate predictions.

UNKNOWN TARGET FUNCTION

$f : \mathcal{X} \mapsto \mathcal{Y}$

(ideal credit approval formula)

$y_n = f(\mathbf{x}_n)$

TRAINING EXAMPLES

$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$

(historical records of credit customers)

LEARNING ALGORITHM

$\mathcal{A}$

FINAL HYPOTHESIS

$g \approx f$

(learned credit approval formula)

HYPOTHESIS SET

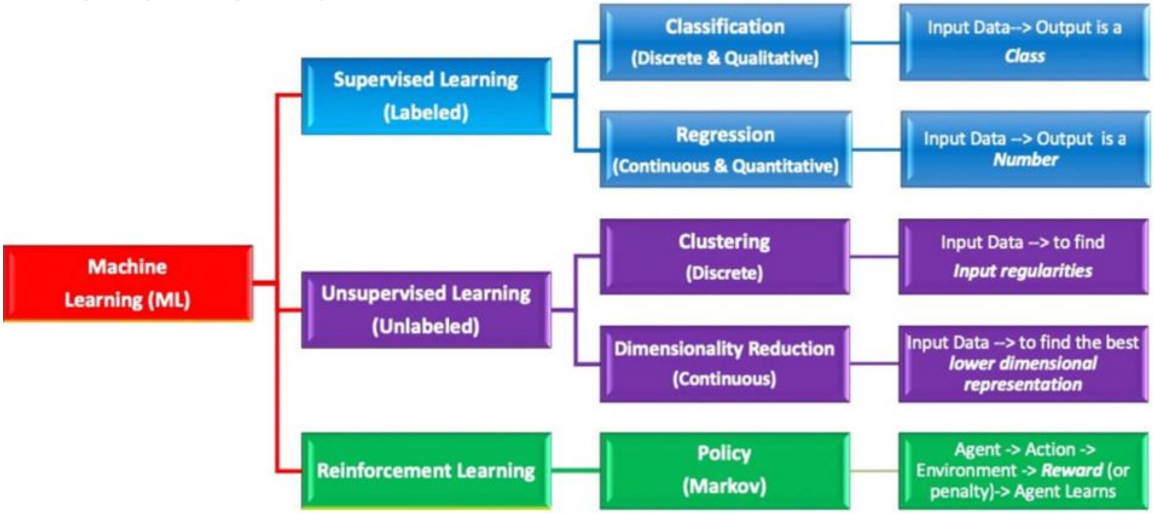
$\mathcal{H}$

(set of candidate formulas)

Takeaways

- One aspect that differentiates **learning** from the rest (e.g., data science, data mining) is that the ultimate goal is to use the learned/fitted model (i.e., **g**) to make predictions.
- Feasibility of Learning is split into two questions:
  1. Can we make sure that **Eout(g)** is close enough to **Ein(g)?**
  2. Can we make sure **Ein(g)** is small enough?

Overview of ML



Machine Learning (ML)

Supervised Learning (Labeled)

Classification (Discrete & Qualitative)

Input Data--> Output is a *Class*

Regression (Continuous & Quantitative)

Input Data --> Output is a *Number*

Unsupervised Learning (Unlabeled)

Clustering (Discrete)

Input Data --> to find *Input regularities*

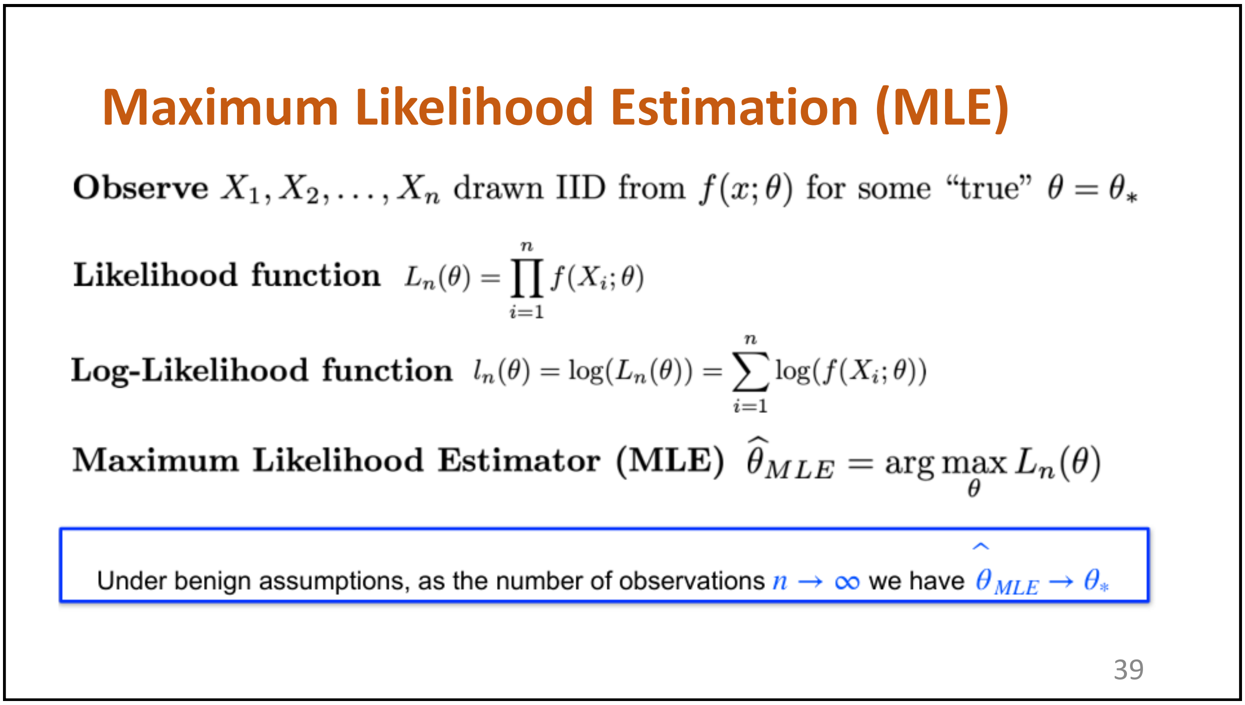
Dimensionality Reduction (Continuous)

Input Data --> to find the best *lower dimensional representation*

Reinforcement Learning

Policy (Markov)

Agent -> Action -> Environment -> *Reward* (or penalty)-> Agent Learns



Maximum Likelihood Estimation (MLE)

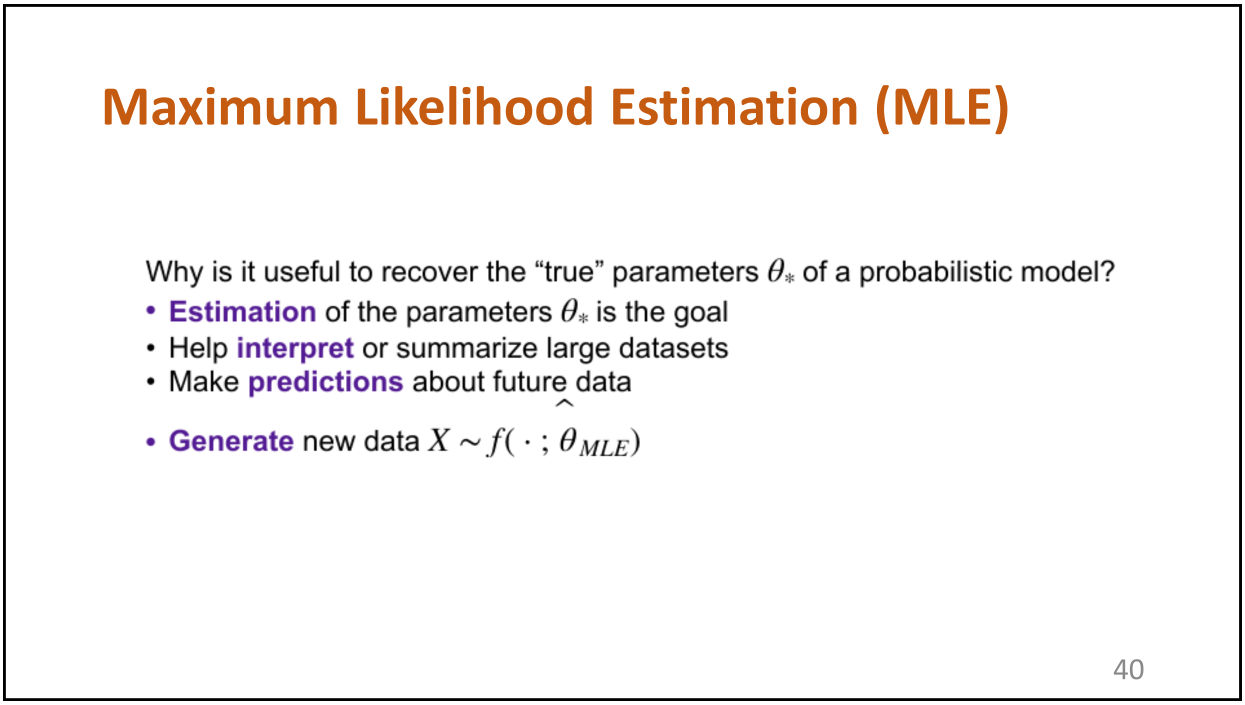
Observe $X_1, X_2, \ldots, X_n$ drawn IID from $f(x;\theta)$ for some "true" $\theta = \theta_*$

Likelihood function $L_n(\theta) = \prod_{i=1}^{n} f(X_i;\theta)$

Log-Likelihood function $l_n(\theta) = \log(L_n(\theta)) = \sum_{i=1}^{n} \log(f(X_i;\theta))$

Maximum Likelihood Estimator (MLE) $\widehat{\theta}_{MLE} = \arg\max_{\theta} L_n(\theta)$

Under benign assumptions, as the number of observations $n \to \infty$ we have $\hat{\theta}_{MLE} \to \theta_*$



Maximum Likelihood Estimation (MLE)

Why is it useful to recover the "true" parameters θ_* of a probabilistic model?

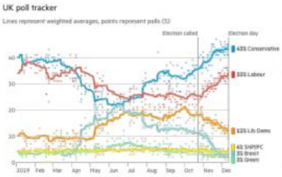
- **Estimation** of the parameters θ_* is the goal
- Help **interpret** or summarize large datasets
- Make **predictions** about future data
- **Generate** new data $X \sim f(\,\cdot\,; \hat{\theta}_{MLE})$

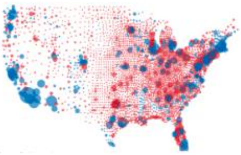
Estimation

Observe $X_1, X_2, \ldots, X_n$ drawn IID from $f(x;\theta)$ for some "true" $\theta = \theta_*$

Opinion polls
How does the greater population feel about an issue? Correct for over-sampling?

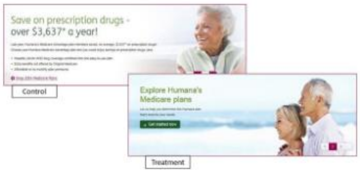
- θ_* is "true" average opinion
- $X_1, X_2, \ldots$ are sample calls





A/B testing
How do we figure out which ad results in more click-through?

- θ_* are the "true" average rates
- $X_1, X_2, \ldots$ are binary "clicks"

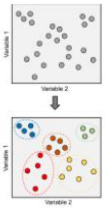

Save on prescription drugs - over $3,637* a year!
Control
Explore Humana's Medicare plans
Treatment

Interpret

Observe $X_1, X_2, \ldots, X_n$ drawn IID from $f(x;\theta)$ for some "true" $\theta = \theta_*$

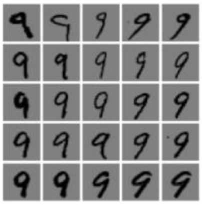
Customer segmentation / clustering
Can we identify distinct groups of customers by their behavior?

- θ_* describes "center" of distinct groups
- $X_1, X_2, \ldots$ are individual customers



Data exploration
What are the degrees of freedom of the dataset?

- θ_* describes the principle directions of variation
- $X_1, X_2, \ldots$ are the individual images



Predict

Observe $X_1, X_2, \ldots, X_n$ drawn IID from $f(x;\theta)$ for some "true" $\theta = \theta_*$

Content recommendation
Can we predict how much someone will like a movie based on past ratings?

- θ_* describes user's preferences
- $X_1, X_2, \ldots$ are (movie, rating) pairs

Object recognition / classification
Identify a flower given just its picture?

- θ_* describes the characteristics of each kind of flower
- $X_1, X_2, \ldots$ are the (image, label) pairs

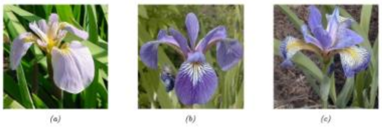


Figure 1.1: Three types of Iris flowers: Setosa, Versicolor and Virginica. Used with kind permission of Dennis Kramb and SIGNA.

index	sl	sw	pl	pw	label
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
	...				
50	7.0	3.2	4.7	1.4	Versicolor
	...				
149	5.9	3.0	5.1	1.8	Virginica

Generate

Observe $X_1, X_2, \ldots, X_n$ drawn IID from $f(x;\theta)$ for some "true" $\theta = \theta_*$

Text generation
Can AI generate text that could have been written like a human?

- θ_* describes language structure
- $X_1, X_2, \ldots$ are text snippets found online

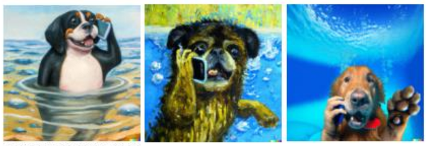
"Kaia the dog wasn't a natural pick to go to mars. No one could have predicted she would..."

https://chat.openai.com/chat

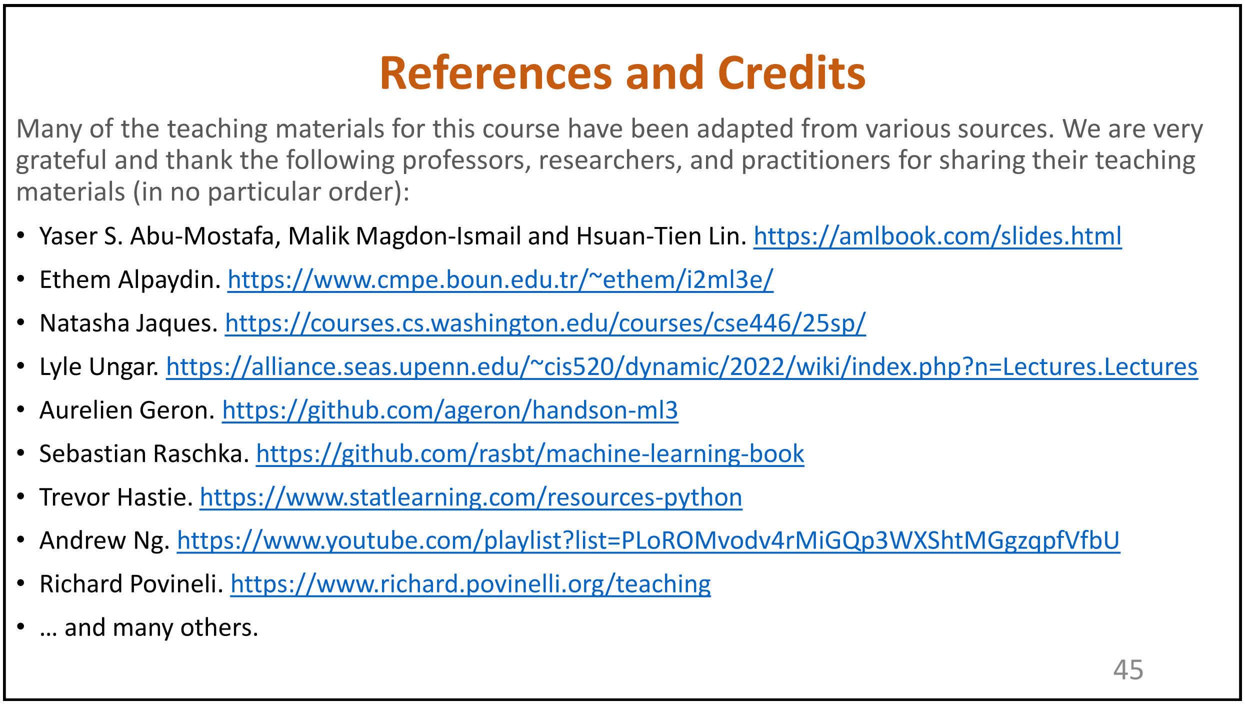
Image to text generation
Can AI generate an image from a prompt?

- θ_* describes the coupled structure of images and text
- $X_1, X_2, \ldots$ are the (image, caption) pairs found online

"dog talking on cell phone under water, oil painting"



https://labs.openai.com/



References and Credits

Many of the teaching materials for this course have been adapted from various sources. We are very grateful and thank the following professors, researchers, and practitioners for sharing their teaching materials (in no particular order):

- Yaser S. Abu-Mostafa, Malik Magdon-Ismail and Hsuan-Tien Lin. https://amlbook.com/slides.html
- Ethem Alpaydin. https://www.cmpe.boun.edu.tr/~ethem/i2ml3e/
- Natasha Jaques. https://courses.cs.washington.edu/courses/cse446/25sp/
- Lyle Ungar. https://alliance.seas.upenn.edu/~cis520/dynamic/2022/wiki/index.php?n=Lectures.Lectures
- Aurelien Geron. https://github.com/ageron/handson-ml3
- Sebastian Raschka. https://github.com/rasbt/machine-learning-book
- Trevor Hastie. https://www.statlearning.com/resources-python
- Andrew Ng. https://www.youtube.com/playlist?list=PLoROMvodv4rMiGQp3WXShtMGgzqpfVfbU
- Richard Povineli. https://www.richard.povinelli.org/teaching
- … and many others.