Pattern Recognition
and Applications Lab

Machine Learning
Introduction to the Course

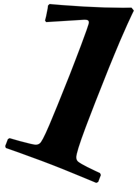
Battista Biggio

battista.biggio@unica.it

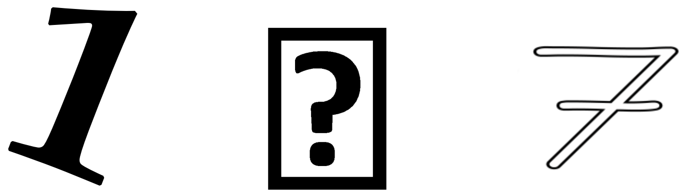
What Number Is This?

7

What Number Is This?



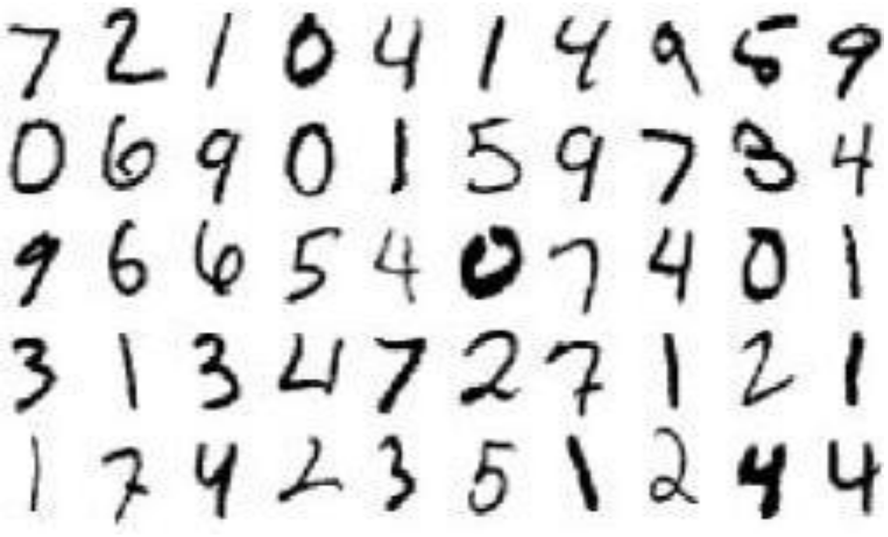
What Number Is This?



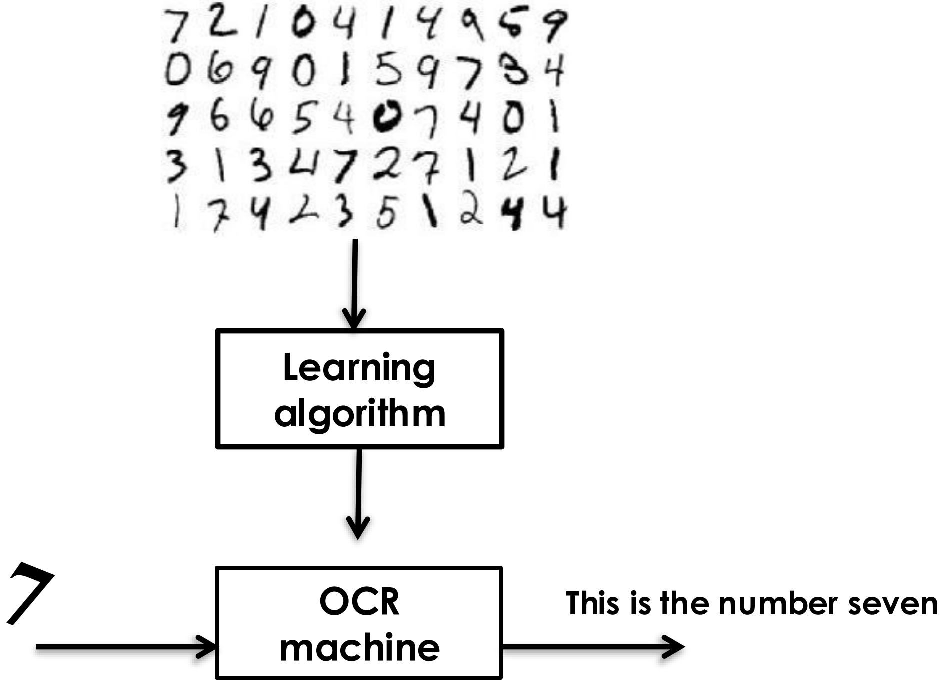
- Are you able to write in Python (or any other language) the **exact algorithm** (step after step) that you use to **recognize** the above numbers?

Writing a **deterministic** algorithm to recognize numbers from images is very difficult…

But we can collect easily many example images…



If We Could Design a Machine that Learns from Examples…



So, What Is Machine Learning?

*Machine learning is the technology that we use to solve a problem by **learning** the solution **by examples***

“The goal of machine learning is to build computer systems that automatically improve with experience”

Tom M. Mitchell, The discipline of Machine Learning, 2006

Take-Home Messages

1. Machine learning is very useful when **no algorithmic solution** is known. It also avoids a detailed algorithm to overfit known cases, reducing errors

2. When you are able to devise algorithmic solutions (*step after step through every possible corner case*) that work 100% of the time, **you should not use machine learning**!

When Did It Start?

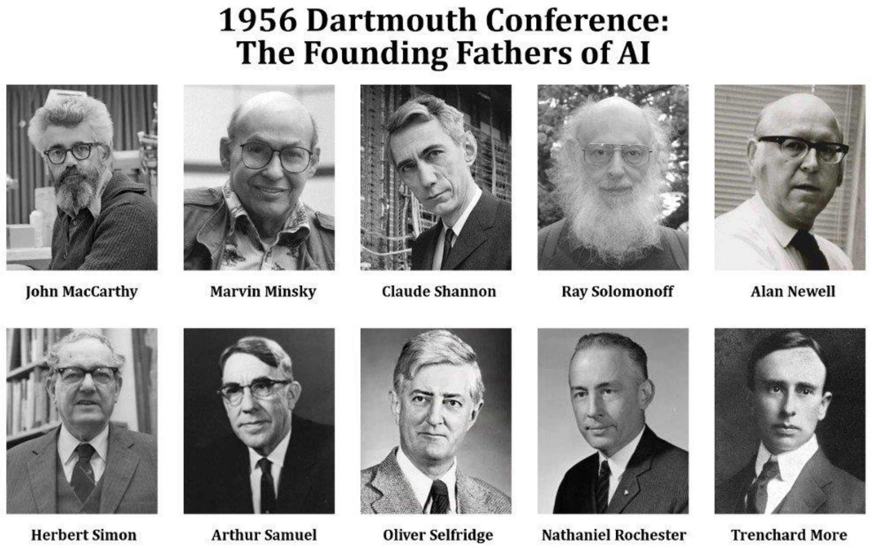
It All Started In 1955

http://www.aaai.org/ojs/index.php/aimagazine/article/view/1904

A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence

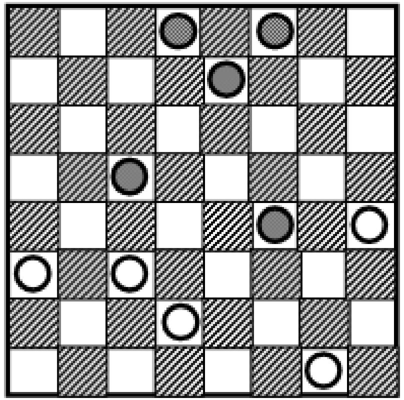
August 31, 1955

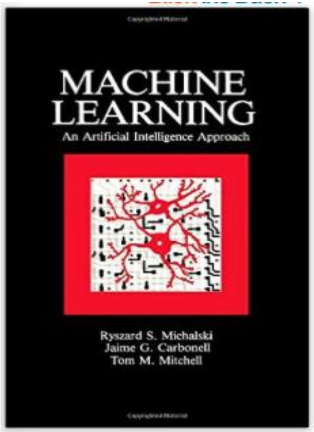
John McCarthy, Marvin L. Minsky, Nathaniel Rochester, and Claude E. Shannon



Machine Learning at the Beginning…

- Arthur Samuel (1959) wrote a program that **learned** to play *checkers*
  - (“draughts” if you’re British)



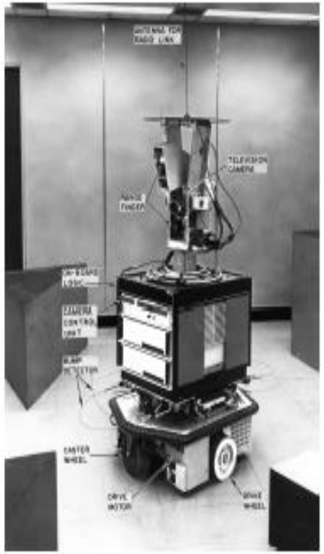


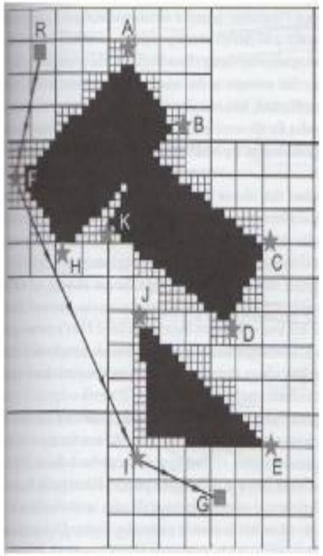
R.S. Michalski, J.G. Carbonell, T.M. Mitchell, *Machine Learning: An Artificial Intelligence Approach*, 1985

Earlier Approaches: 1960-1980

- Algorithms based on the reasoning-as-research "paradigm" worked well for well-defined and circumscribed problems, with strong domain knowledge
  - Game of chess, robot navigation in defined environments, etc.

- But they showed significant limits when:
  - tackling the problem required enormous "knowledge" of the real world to avoid a "combinatorial explosion" of the research space
  - no explicit formulation of the problem was available





Artificial Intelligence and Machine Learning Today

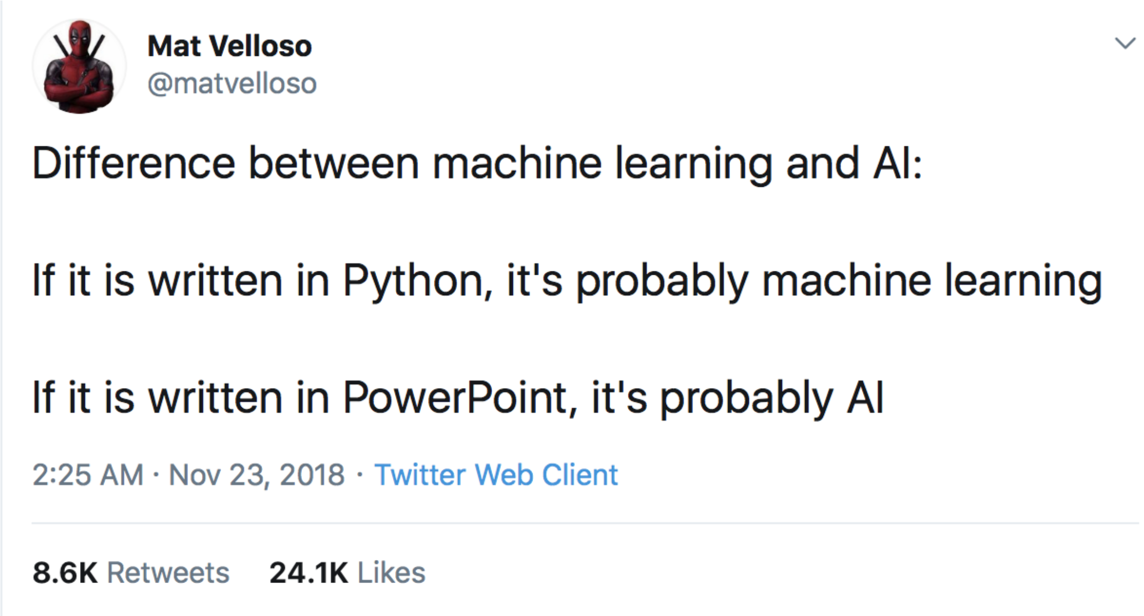
AI is going to transform industry and business as electricity did about a century ago

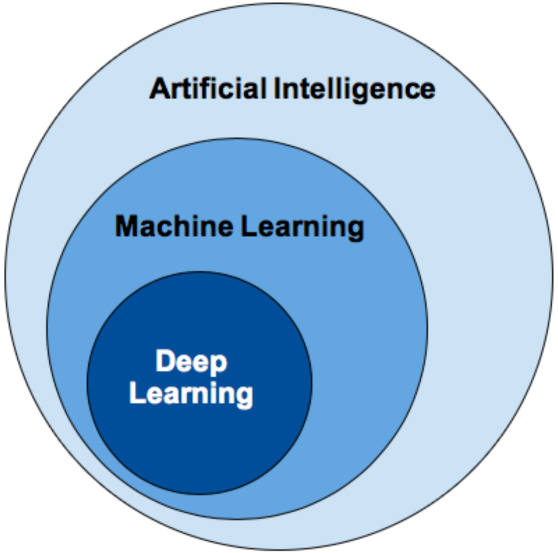
(*Andrew Ng, Jan. 2017*)

Applications:

- Cybersecurity
- Robotics
- Healthcare
- Speech recognition
- Virtual assistants
- ...

But… What’s the Difference between AI/ML?





Data-Driven AI/ML (1990-now)

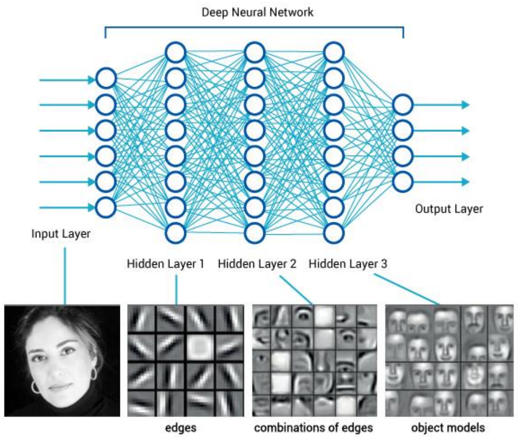
Big Data

Facebook 350 millions of images per day

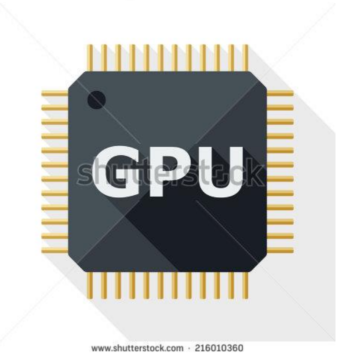
Walmart 2.5 Petabytes customer data hourly

YouTube 300 hours of videos per minute

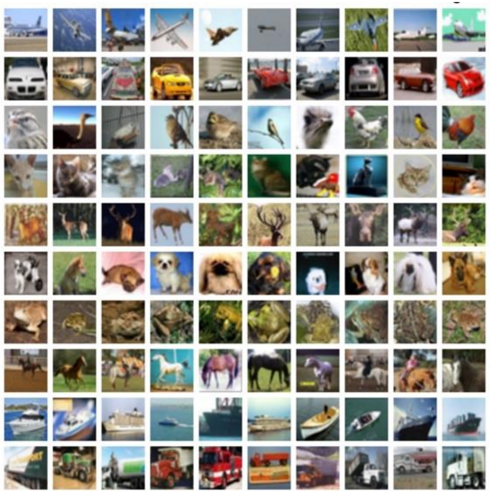
Deep Learning

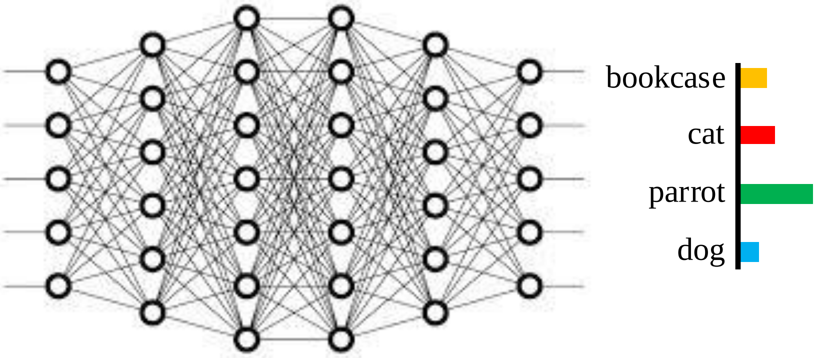


GPU



Modern AI is Numerical Optimization + Big Data





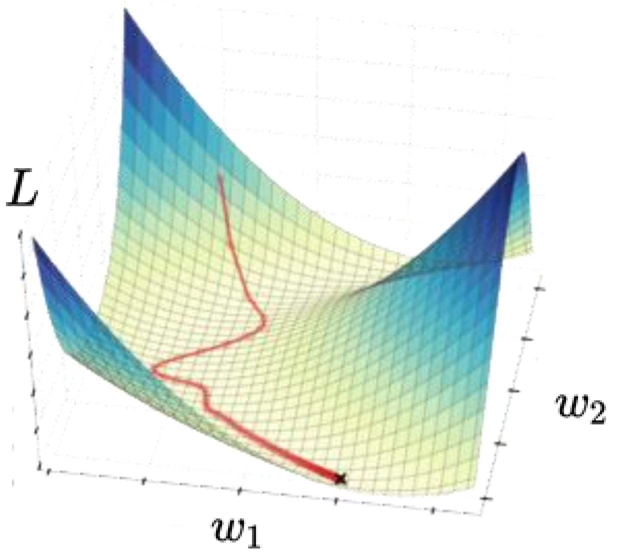
$$\min_{\boldsymbol{w}} L(D; \boldsymbol{w})$$

The goal is to minimize the fraction of *classification errors*

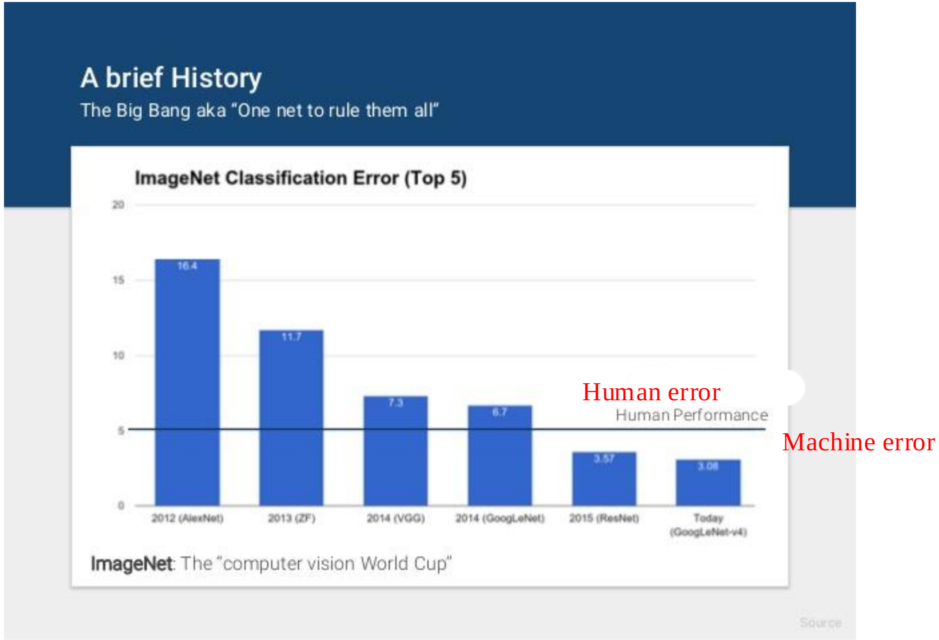
... by iteratively updating the classifier parameters $\boldsymbol{w}$ along the gradient direction $\nabla_{\boldsymbol{w}} L(D; \boldsymbol{w})$

The Workhorse of Machine Learning: *Gradient Descent*

w $\leftarrow \mathbf{w}_0$
$i \leftarrow 0$
while $i < maxiter$ **do**
    $\mathbf{w} \leftarrow \mathbf{w} - \eta \, \nabla_{\mathbf{w}} L(\mathbf{X}, \mathbf{y})$
    $i \leftarrow i + 1$
end while
return w



The Bright Side of AI: Super-Human Performance



What does it mean to build an “intelligent” machine?

Definition of AI and Turing test

- What does it mean for a machine to be intelligent?
  - Asking whether a machine can think can be ambiguous
  - What is the meaning attributed to the word *think*?

- Alan Turing proposed a different view in 1950, suggesting a behavioral definition of intelligence

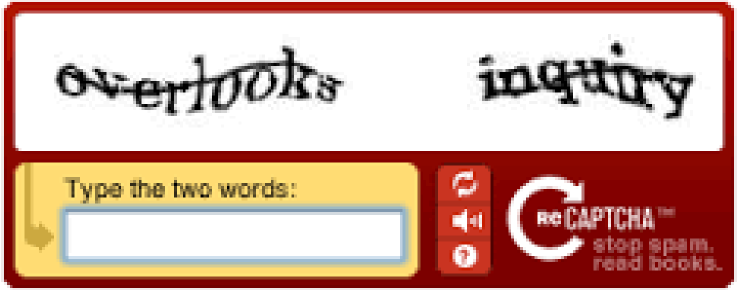
- This definition states that a computer can be defined as intelligent
  - if it is able to pass a test (Turing test)
  - which demonstrates its ability to achieve performance comparable to that of humans in all cognitive tasks

- It is a good definition from an engineering point of view

A. Turing, "Computing Machinery & Intelligence," *Mind*, Vol. 59(236), 1950.

Turing Test: A Popular Example

Completely Automated Public Turing test to tell Computers and Humans Apart (**CAPTCHA**)



Pattern Recognition as a Classification Problem

- This ML course focuses on pattern **classification**. We use the term **recognition** instead of classification if the context makes the meaning clear and there is no ambiguity

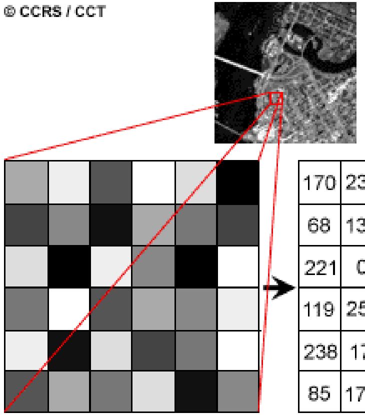
- **Pattern Classification:** assigning a “pattern” (input data) to a category/class

- In this picture, the pattern is the specific grouping of pixels that represent the number 7

Pattern Recognition as a Classification Problem

- Pattern classification is about assigning class labels to patterns

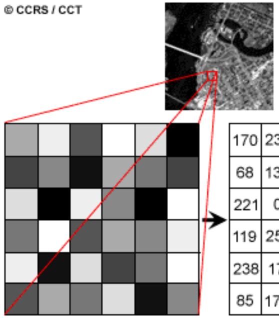
- Patterns are described by a set of measurements called *features* (or attributes)
  - For images, feature/input values could correspond to the brightness of each pixel



170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

Basic Concepts: Class and Features

- In this course, we assume that each pattern is described by a feature vector with "d" elements: $\mathbf{x} = (x_1, x_2, \ldots, x_d)$.

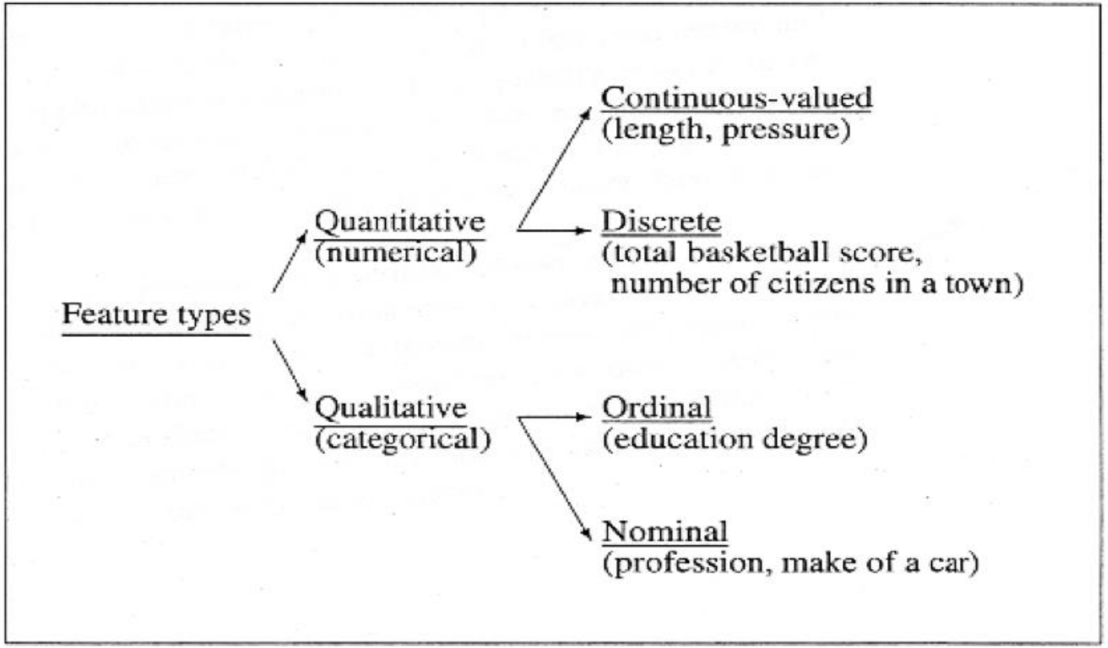


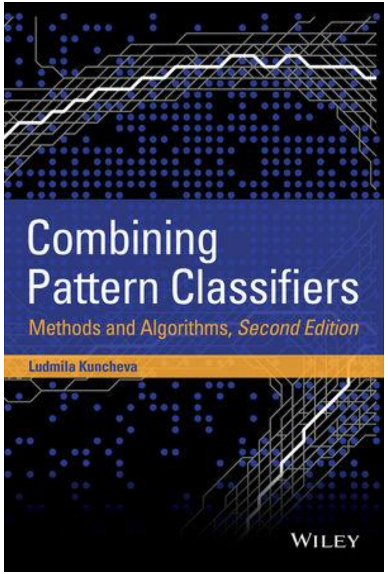
170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

$\mathbf{x} = (x_1, x_2, \ldots, x_d) = (170, 238, 85 \ldots 136)$

- **Class**: intuitively, a class contains similar patterns, whereas patterns from different classes are dissimilar (e.g., dogs and cars)
  - In this course, we assume that there are c possible classes, denoted with: $\Omega = \{\omega_1, \omega_2, \ldots, \omega_c\}$. Each pattern belongs to one of the "c" classes of the set Ω
  - We say that each pattern has a class **label**

Different Feature Types





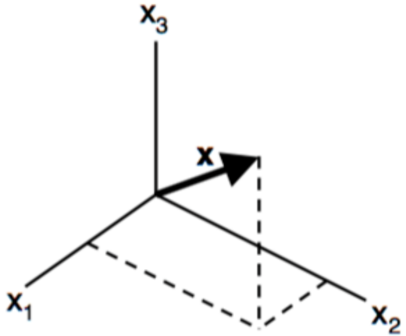
Ludmila Kuncheva, *Combining pattern classifiers*, Wiley, 2004

- **Statistical pattern classification** uses (mostly) *numerical* features

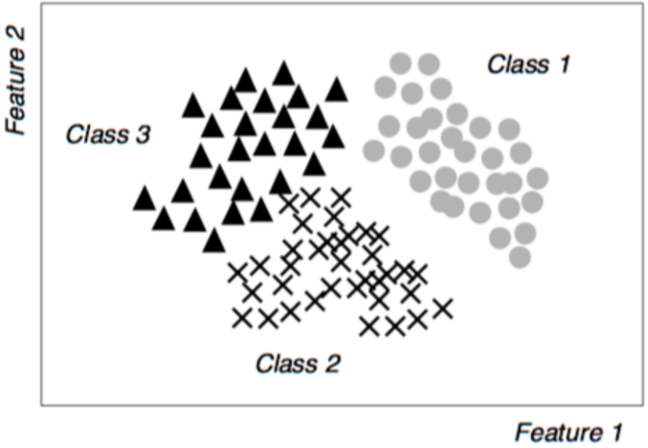
Basic Concepts: Feature Vector and Feature Space

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

Feature vector



Feature space (3D)



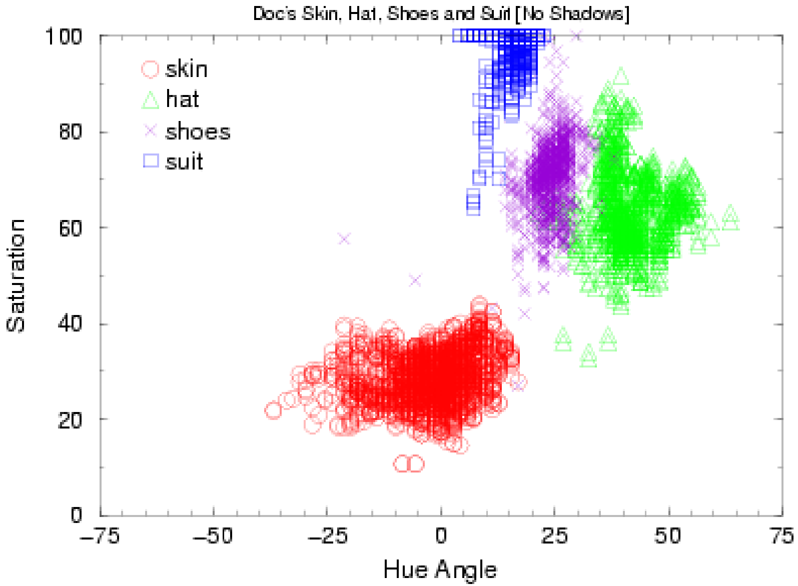
Scatter plot (2D)

Basic Concepts: Feature Space

- The feature values are arranged as a *d*-dimensional vector

- The real space is called the feature space

- Each axis/dimension corresponds to a feature

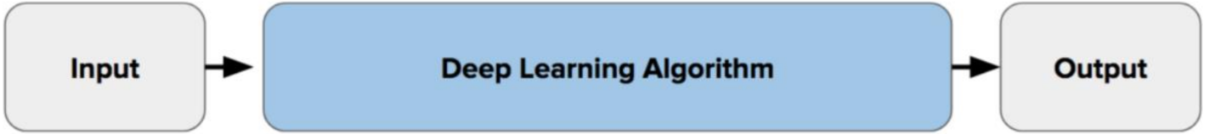


Hand-crafted vs. Non-handcrafted (Learned) Features

- In the previous example, we have seen what is named **handcrafted** features that are manually engineered by the human designer

Processing flow for extraction of **handcrafted** features

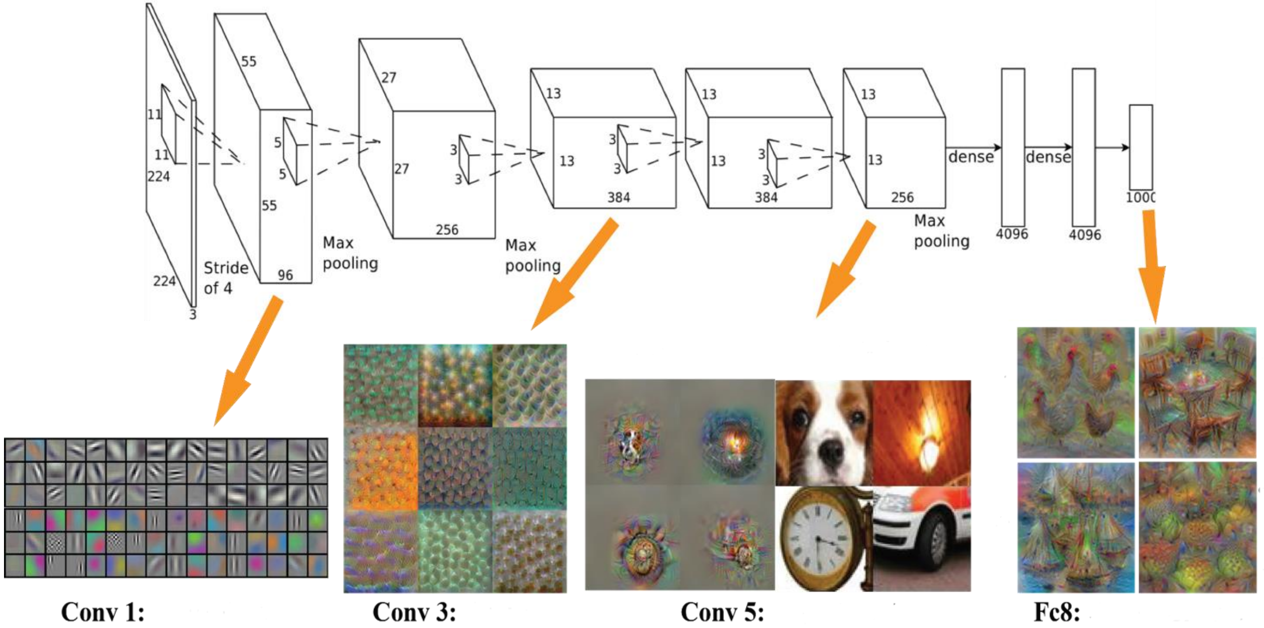
- Today, we can extract **non-handcrafted** features that are automatically learned from a machine learning algorithm



Processing flow for learning **non-handcrafted** features («learned» features)

Learning Non-handcrafted Features

- **Non-handcrafted** features can be automatically learned with **deep neural networks** (we will see them later).



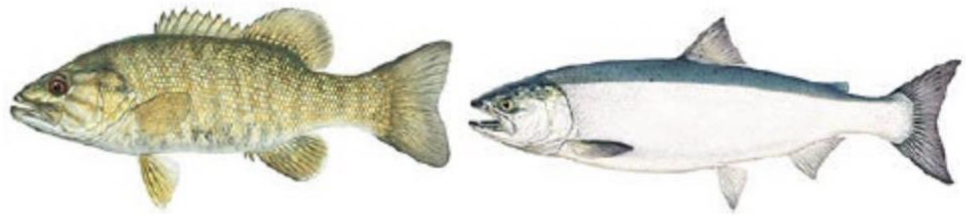
Classification Model

Pattern Classification, R. O. Duda, P. E. Hart, D. G. Stork, John Wiley & Sons, 2000

- **Classification**: after *feature extraction*, we should select a classification model using such feature vectors as inputs to classify patterns

- Let us assume that want to recognize 2 classes of fish: *salmon* and *sea bass*

- We use only one feature: length value (random variable *L*).



Sea bass **Salmon**

Classification Model

- A very simple classification model based on a simple heuristic rule could be:
  - *A sea bass is generally longer than a salmon*

- We can rewrite more formally this heuristic rule as follows:
  - if $L > L^*$ then fish=sea bass , else fish=salmon

- The threshold value L^* can be an heuristic value that we know, otherwise we should estimate it

- How can we estimate L^* ? We need a set of samples/examples of the two fish types (called “design o **training set**”)

Basic Concept: Design or Training Dataset

- The information to design a pattern classifier is usually in the form of a labeled data set ***D*** (called design or training set):

$$\boldsymbol{D} = [\mathbf{x}_1, \mathbf{x}_2, \ldots., \mathbf{x}_n]$$

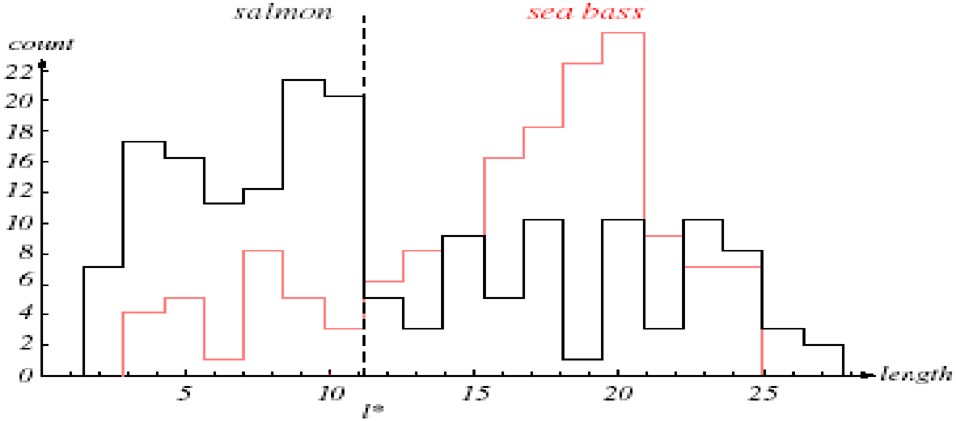
$$\mathbf{x}_i = (\mathbf{x}_{i1}, \mathbf{x}_{i2}, \ldots., \mathbf{x}_{id}) \text{ i=1,..,n}$$

$$\mathbf{x}_i \text{ belongs to one of the “c” classes } (\mathbf{x}_i \; \varepsilon \; \omega_j \;\; j=1,\ldots,c)$$

- In the previous example, ***D*** is the data set used to compute the empirical distributions of the length of the two fish types
- This allows us to estimate the threshold value L* that discriminates between salmon and sea bass

Classification Models

- This simple example suggests us a more general classification model. We could estimate the two probability functions:
  - P(length / salmon) and P(length / sea bass)
  - and then make a probabilistic decision…



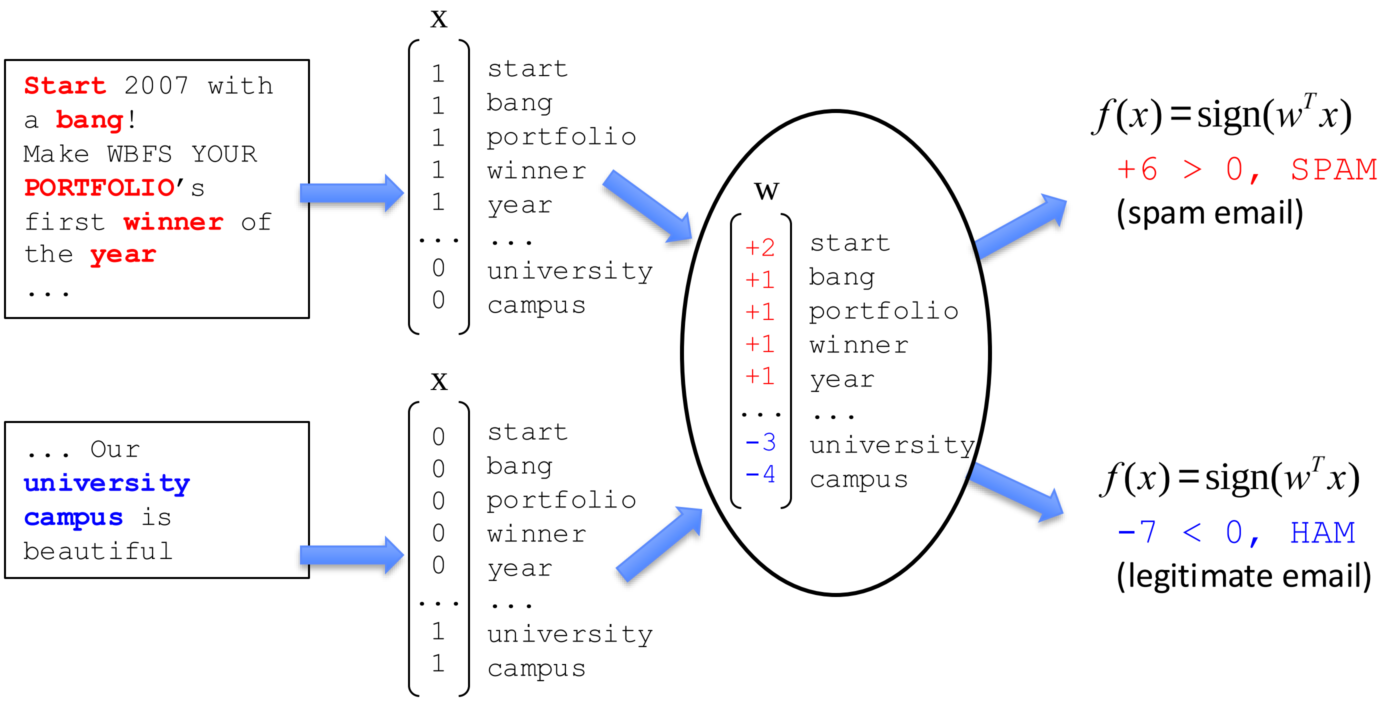
Classification Models

- In general, a classification model can be regarded as a **function** $f(\boldsymbol{x})$, that takes as input the vector **x** (representing the pattern) and provides as output the classification (class label)

- For example, the classification model could be a linear function:

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b = \sum_{j=1}^{d} w_j x_j + b$$

Example: Spam Filtering



Learning as Optimization

- We said that machine learning is "learning from experience"
  - i.e., improving classification performances over time
- How do we evaluate if we are improving?
- In order to develop a formal mathematical system of learning machines, we need to have formal measures of how good (or bad) our models are
- To this end, we use **loss functions** (or *cost functions*) to evaluate how good (or bad) our classification models are

Example of Loss Function

$$L(D, \boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(\boldsymbol{x}_i; \boldsymbol{\theta}))$$

- D: training set containing «n» examples
  - y_i: is the class label for training example $\boldsymbol{x}_i$

- $f(\boldsymbol{x}_i; \boldsymbol{\theta})$ is the classification model

- $\ell(y_i, f(\boldsymbol{x}_i; \boldsymbol{\theta}))$ could be the zero-one loss function
  - equal to 0 for correct predictions and 1 otherwise

$$\ell\big(y_i, f(\boldsymbol{x}_i; \boldsymbol{\theta})\big) = \begin{cases} 0, \text{classification is correct} \\ 1, \text{classification is incorrect} \end{cases}$$

Learning as an Optimization Problem

- Given a linear function $f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b = \Sigma_{j=1}^{\mathrm{d}} w_j x_j + b$
  - How do we estimate the classifier parameters $\boldsymbol{w}$ and b?

- Modern approaches formulate the learning problem as an **optimization problem**
  - This is generally true also for nonlinear classification functions $f(\boldsymbol{x};\boldsymbol{\theta})$, including modern deep-learning approaches and neural networks

$$\boldsymbol{w}^\star, b^\star = \underset{\boldsymbol{w},b}{\operatorname{argmin}} \underbrace{\frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(\boldsymbol{x}_i))}_{\substack{\text{loss term} \\ L(D,\boldsymbol{\theta})}} + \lambda \underbrace{\Omega(\boldsymbol{w})}_{\substack{\text{regularization term} \\ \Omega(\boldsymbol{w})}}$$

λ: regularization hyperparameter

Learning as an Optimization Problem

- The loss function $\ell(y_i, f(\boldsymbol{x}_i))$ measures how much a prediction is wrong
  - e.g., the zero-one loss is 0 if points are correctly predicted, and 1 if they are not

- The regularization term $\Omega(\boldsymbol{\theta})$ imposes a penalty on the magnitude of the classifier parameters to avoid overfitting and promote smoother functions, i.e., functions that change more gradually as we move across the feature space

- The hyperparameter λ tunes the trade-off between the training loss and regularization
  - Larger values tend to promote more regularized functions but with a larger training error
  - Smaller values tend to reduce the training error but learn more complex functions

Optimization Algorithms

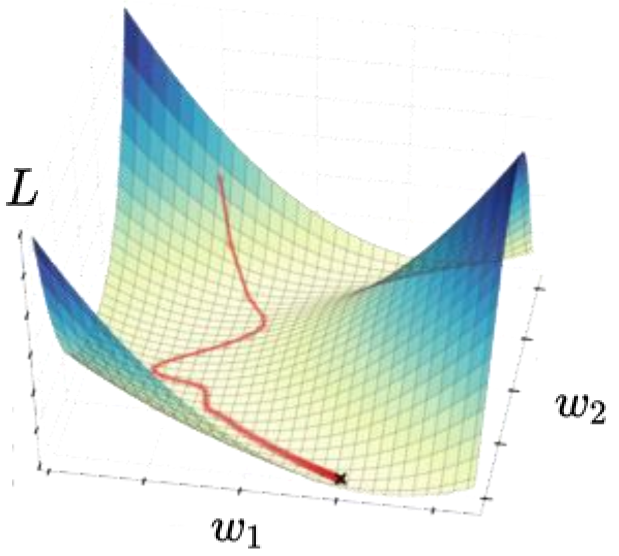
- In machine learning, we need an **optimization algorithm** (also called *solver*) capable of finding the best possible parameters that minimize the loss function

- The most popular optimization algorithms follow an approach called **gradient descent**

$$\boldsymbol{w}^\star, b^\star = \underset{\boldsymbol{w},b}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(\boldsymbol{x}_i)) + \lambda\, \Omega(\boldsymbol{w})$$

The Workhorse of ML: *Gradient Descent* (again!)

1: $\mathbf{w} \leftarrow \mathbf{w}_0$
2: $i \leftarrow 0$
3: **while** $i < maxiter$ **do**
4: $\quad \mathbf{w} \leftarrow \mathbf{w} - \eta \, \nabla_{\mathbf{w}} L(\mathbf{X}, \mathbf{y})$
5: $\quad i \leftarrow i + 1$
6: **end while**
7: **return** $\mathbf{w}$



Generalization Error - Overfitting

- The best values of the model's parameters are learned by minimizing the loss incurred on a ***training set*** consisting of some number of *examples* collected for training

- However, doing well on the training data does not guarantee that we will do well on (**unseen**) **test** data

- So we split the available data into two partitions: the training data (for fitting model parameters) and the test data (which is held out for evaluation), and then measure:
  - **Training Error -** The error on that data on which the model was trained
  - **Test Error -** This is the error incurred on an **unseen** test set (**generalization error**). This can deviate significantly from the training error. When a model performs well on the training data but fails to generalize to unseen data, we say that it is ***overfitting***

https://d2l.ai, Chapter 1

Two Main Kinds of Machine Learning

- **Supervised learning**
  - **in this course, we mainly focus on this case**

- **Unsupervised learning**
  - Learning from a set of unlabeled samples. The goal of **unsupervised learning** (also called "clustering") is basically to find groupings in the data ("clusters") which actually reflect the ground truth and the "natural properties" of the domain the data comes from

https://d2l.ai, Chapter 1

Other Kinds of Machine Learning Problems

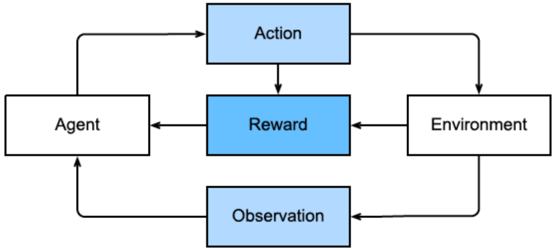
- **Regression**
  - for example, predicting the rating that a user will assign to a movie can be thought of as a regression problem
- **Tagging / Multi-label classification**
  - for example, assigning muliple lables to one image can be thought of as a tagging problem
- **Search and ranking**
  - for example, determining whether a particular web page is relevant for a user's query can be thought of as a search and ranking problem
- **Recommendation**
  - for example, providing movie recommedations to web users can thought of as a recommendation problem

https://d2l.ai, Chapter 1

Other Kinds of Machine Learning Problems

- **Sequence learning**
  - When you have a sequence of inputs and you have to provide a sequence of outputs; for example, speech recognition, text-to-speech, language translation, can be thought of as a sequence learning problems

- **Reinforcement learning (learning by interacting with an environment)**
  - Game of chess, driving a car, can be thought of as reinforcement learning problems



https://d2l.ai, Chapter 1

Course Objectives and Outcome

- **Objectives:** to provide students with the fundamental elements of **machine learning** and its applications to **pattern recognition**. The main concepts and methods of machine learning and statistical pattern recognition are presented, as well as basic methods to design and evaluate the performance of a pattern recognition system.

- **Outcome:** An understanding of fundamental concepts and methods of machine learning, statistical pattern recognition and its applications
  - *An ability to analyse and evaluate simple algorithms for pattern classification*
  - *An ability to design simple algorithms for pattern classification, code them with Python programming language and test them with benchmark data sets*

Machine Learning (7 CFU) - Tentative Course Outline

1. Introduction (2 hours)
2. Bayesian Decision Theory and Gaussian Pattern Classifiers (10 hours)
3. Non parametric methods and k-NN classifier (4 hours)
4. Linear discriminant functions and support vector machines (6 hours)
5. Artificial neural networks (4 hours)
6. Performance evaluation (2 hours)
7. Clustering Methods (2 hours)
8. Adversarial machine learning (2 hours)
9. Exercises (12 hours)
10. Python Programming language and computer exercises (16 hours)
11. **Deep Learning + PyTorch (10 hours)**

Course Grading and Material

Course Grading

- *Home computer-exercise assignment + Oral examination*
  - You can do the written assessment at the end of the course instead of the oral examination
- Teams of 3 students maximum can do the home computer exercise
- **Grading policy = Computer exercise (10/30) + Oral examination (20/30)**

Reference Books:

- Pattern Recognition and Machine Learning, C. Bishop, Springer, 2007
- Dive into Deep Learning, A. Zhang, Z. C. Lipton, M. Li, A. J. Smola, 2020: https://d2l.ai
- Pattern Classification (2nd ed.), R. O. Duda, P. E. Hart, e D. G. Stork, John Wiley & Sons, 2000

Website / Repository (The course materials are all made available there)

- https://github.com/unica-ml/ml

Teams Channel

- Please subscribe to the course Teams channel. The link can be found on the course website