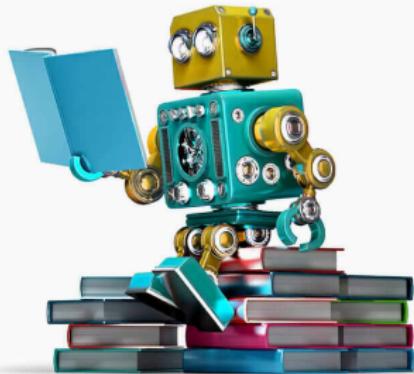


ECE 285
Machine Learning for Image Processing

Chapter I – Introduction

Charles Deledalle

October 2, 2019



(Source: Jeff Walsh)

Who?

Who am I?

- Visiting professor at UCSD since Jan 2017.
 - Computer Engineering degree (2008).
 - M.Sc. in Artificial Intelligence (2008).
 - Ph.D. in Digital Signal Processing (2011).
 - Research in Image Processing / Applied Maths (since 2012).
-

- Office: Jacobs Hall EBU1 4808
- Email: cdeledalle@ucsd.edu
- Webpage: www.charles-deledalle.fr

What?

What is it about?

Machine learning / Deep learning

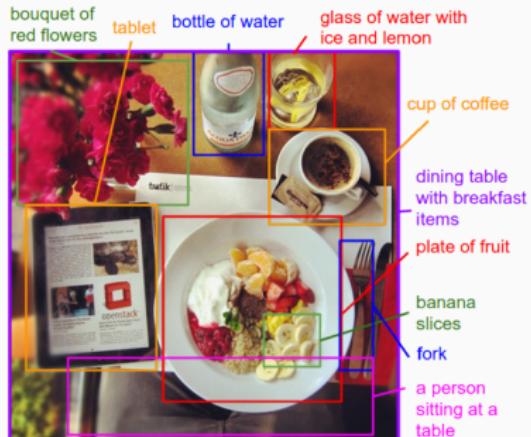
applied to

Image processing / Computer vision

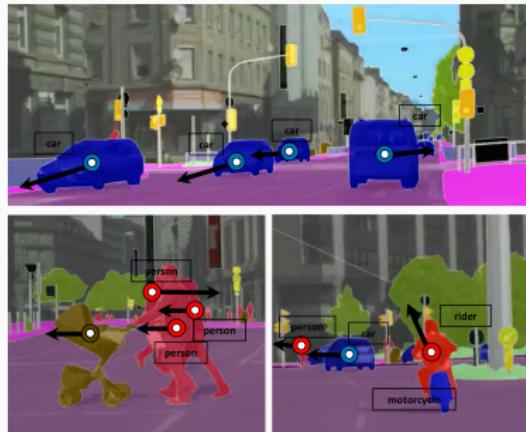
- A bit of theory (but not exhaustive), a bit of math (but not too much),
- Mainly: concepts, vocabulary, recent successful models and applications.

What?

What is it about? – Two examples



(Source: Luc et al., 2017)

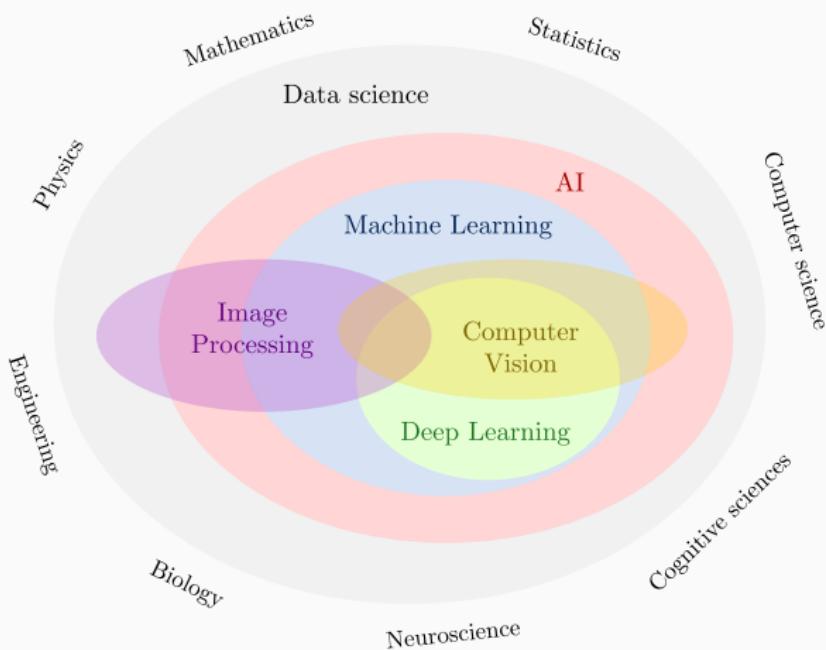


(Karpathy & Fei-Fei, 2015)

**(CV:) Automatic extraction of high level information from images/videos,
(ML:) by learning from tons of (annotated) examples.**

What?

What is it about? – A multidisciplinary field



What? Syllabus

- **Introduction to image sciences and machine learning**
 - Examples of image processing and computer vision tasks,
 - Overview of learning problems, approaches and workflow.
- **Preliminaries to deep learning**
 - Perceptron, Artificial Neural Networks (NNs),
 - Backpropagation, Support Vector Machines.
- **Basics of deep learning**
 - Representation learning, auto-encoders, algorithmic recipes.
- **Applications**
 - Image classification
 - Object detection
 - Image captioning
 - Image generation
 - Super resolution
 - Style transfer

⇒ Convolutional NNs, Recurrent NNs, Generative adversarial networks.
- **Assignments and project using Python & PyTorch.**

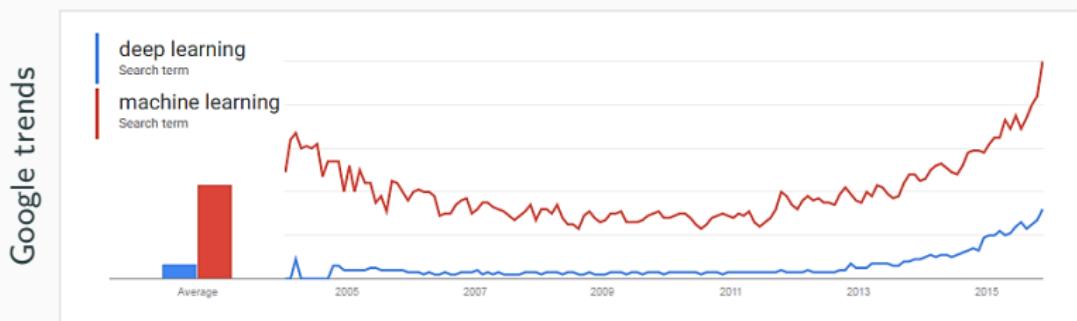
Why?

Why machine learning / deep learning?

- In the past 10 years, machine learning and artificial intelligence have shown **tremendous progress**.
- The recent success can be attributed to:
 - **Explosion of data,**
 - Cheap computing cost – CPUs and **GPUs**,
 - Improvements of machine learning models.
- Much of the current excitement concerns a subfield of it called "**deep learning**".



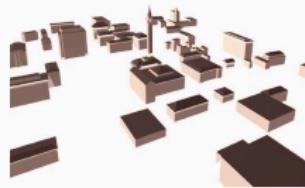
(Source: Poo Kuan Hoong)



Why?

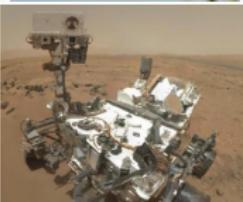
Why image processing / computer vision?

- Images become a **major communication media**.
- Images need to be analyzed **automatically**.
- Reduce the burden of human operators by teaching a computer to see.
- Many applications: robotic, medical, video games, sport, smart cars, ...



Why?

Why? More examples...



Top row, left to right:

Image by [Augustas Didžgalvis](#) is licensed under [CC BY-SA 3.0](#); changes made
Image by [Nestier](#) is licensed under [CC BY-SA 2.0](#)
Image is [CC0 1.0](#) public domain
Image is [CC0 1.0](#) public domain

Middle row, left to right

Image by [BGPHP Conference](#) is licensed under [CC BY 2.0](#); changes made
Image is [CC0 1.0](#) public domain
Image by [NASA](#) is licensed under [CC BY 2.0](#); changes made
Image is [CC0 1.0](#) public domain

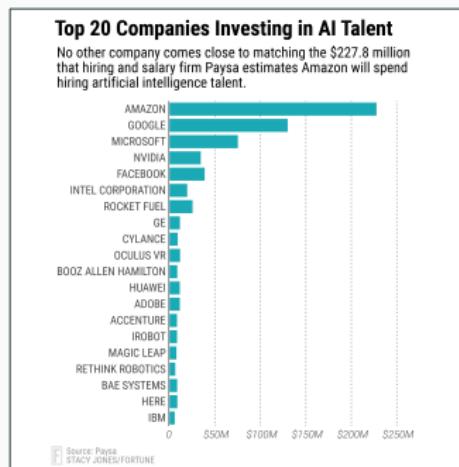
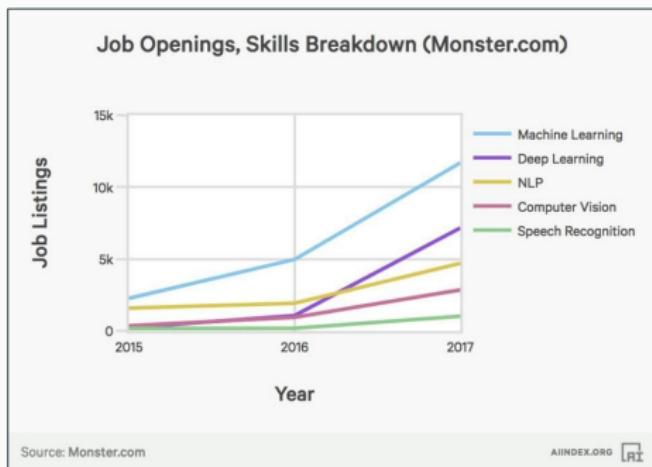
Bottom row, left to right

Image is [CC0 1.0](#) public domain
Image by [Derek Keats](#) is licensed under [CC BY 2.0](#); changes made
Image is [public domain](#)
Image by [Ted Eytan](#) is licensed under [CC BY-SA 2.0](#); changes made

What for?

What for?

- **Industry:** be able to use or implement latest machine learning techniques to solve image processing and computer vision tasks.

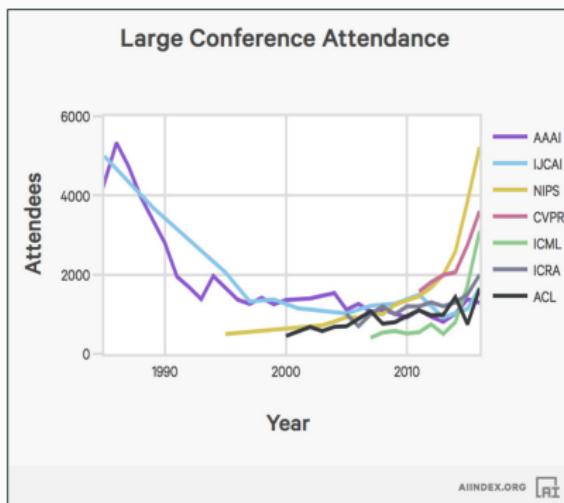
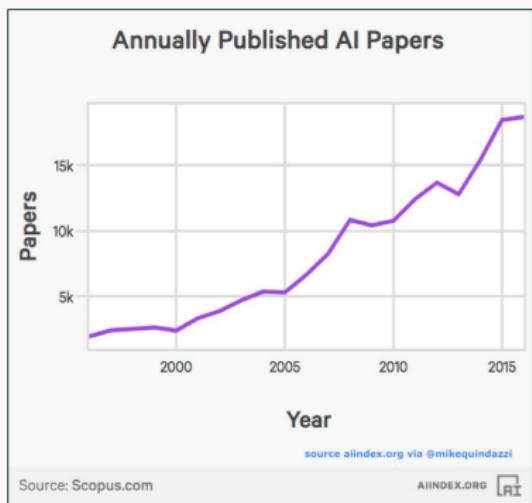


- **Big actors:** Amazon, Google, Microsoft, Facebook, ...

What for?

What for?

- **Academic:** be able to read and understand latest research papers, and possibly publish new ones.



- **Big actors:** Stanford, New York U., U. of Montreal, U. of Toronto, ...
- **Main conferences:** NIPS, CVPR, ICML, ...

How?

How? – Teaching staff

Instructor



Charles Deledalle

Teaching assistants



Anurag Paul



Inderjot
Singh Saggub



Raghav K
Subramanian

How?

How? – Schedule

- 30× **50 min lectures** (10 weeks)
 - Mon/Wed/Fri 3:00-3:50pm
 - Room CENTR 109
- **Office hours**
 - Charles Deledalle, weekly on Wed 10am-12pm, Jacobs Hall 4808.
 - TAs, weekly, (TBA, refer to Google's calendar)
- **Google calendar:** <https://tinyurl.com/y4q652qw>

How?

How? – Assignments / Project / Evaluation

- **4 assignments** in Python/Pytorch (individual) 40%
 - Don't wait for the lectures to start,
 - You can start doing them all now.
- **1 project** open-ended or to choose among 3 proposed subjects 30%
 - In groups of 3 or 4 (start looking for a group now),
 - Details to be announced in a couple of weeks.
- **3 quizzes** (~45 mins each) 30%
 - Multiple choice on the topics of **all** previous lectures,
 - Dates are: Oct 25, Nov 15, Dec 13,
 - No documents allowed.

How?

How? – What assignments?

Assignment 1 (Backpropagation): Create from scratch a simple machine learning technique to recognize hand-written digits from 0 to 9.



Assignment 2 (CNNs and PyTorch): Develop a deep learning technique and learn how to use GPUs with PyTorch.

Improve your results to 98%!

How?

How? – What assignments?

Assignment 3 (Transfer learning): Teach a program how to recognize bird species when only a small dataset is available.



→ **Mocking bird!**

How?

How? – What assignments?

Assignment 4 (Image Denoising): Teach a program how to remove noise.



How?

How? – Assignments and Project Deadlines

Calendar	Deadline
① Assignment 0 – Python/Numpy/Matplotlib (Prereq)	optional
② Assignment 1 – Backpropagation	Oct 16
③ Assignment 2 – CNNs and PyTorch	Oct 30
④ Assignment 3 – Transfer Learning	Nov 13
⑤ Assignment 4 – Image Denoising	Nov 27
⑥ Project	Dec 6

Refer to the Google calendar: <https://tinyurl.com/y4q652qw>

How?

How? – Prerequisites

- Linear algebra + Differential calculus + Basics of optimization + Statistics/Probabilities
 - Python programming (at least Assignment 0)

Optional: cookbook for data scientists

Cookbo	Cookbo	Cookbo	Cookbo	Cookbo	Cookbo
Convex optimization					
Convex function					
Let $f: \mathbb{R}^n \rightarrow \mathbb{R}$ be convex function. Partial derivative sequence as defined at $x = x_0 + \eta e_i$ and $x_{0+i} = x_0 + \eta e_i$ with $\eta \in \mathbb{R}$. Then $f(x) = f(x_0) + \nabla f(x_0)^T (x - x_0)$ $= f(x_0) + \eta \nabla f(x_0)^T e_i$ with $\eta \geq 0$. $\nabla f(x_0) = \lim_{\eta \rightarrow 0} \frac{f(x_0 + \eta e_i) - f(x_0)}{\eta} \geq 0$					
Lipschitz gradient					
$f: \mathbb{R}^n \rightarrow \mathbb{R}$ is Lipschitz gradient if $\ \nabla f(x) - \nabla f(y)\ \leq L \ x - y\ $ $\forall \nabla f(x), \nabla f(y) \in \mathbb{R}^n$. If f is twice differentiable, then $L = \ \nabla^2 f(x)\ $, i.e., the largest singular value of $\nabla^2 f(x)$ among all possible x .					
Convexity					
$f: \mathbb{R}^n \rightarrow \mathbb{R}$ is convex if for all x, y and λ : $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$					
f is strictly convex if the inequality is strict ($<$ instead of \leq). f is convex if it is not strictly negative definite. f is strictly concave if $-f$ is strictly convex. A function f is convex if and only if its domain is convex and the set of minima is convex.					
Convex function					
Let $f: \mathbb{R}^n \rightarrow \mathbb{R}$ be convex function. Then x is a local minimum of f if and only if $\nabla f(x) = 0$. If f is strictly convex, then there is exactly one local minimum.					
Convex optimization					
Let $f: \mathbb{R}^n \rightarrow \mathbb{R}$ be convex function with a Lipschitz gradient, then for all $x \in \mathbb{R}^n$, the unique $x_{\text{min}} = \arg \min_x f(x)$ is given by $\nabla f(x_{\text{min}}) = -\nabla^2 f(x_{\text{min}})^{-1} \nabla f(x_{\text{min}})$ $\nabla f(x_{\text{min}}) = -\nabla^2 f(x_{\text{min}})^{-1} \nabla f(x_{\text{min}})$ $\nabla f(x_{\text{min}})^T (\nabla f(x) - \nabla f(x_{\text{min}})) = 0$					
Multivariate differential calculus					
Partial derivatives					
Let $f: \mathbb{R}^n \rightarrow \mathbb{R}$ be function partial derivative as defined at $x = x_0 + \eta e_i$ and $x_{0+i} = x_0 + \eta e_i$ with $\eta \in \mathbb{R}$. Then $f(x) = f(x_0) + \nabla f(x_0)^T (x - x_0)$ $= f(x_0) + \eta \nabla f(x_0)^T e_i$ with $\eta \geq 0$. $\nabla f(x_0) = \lim_{\eta \rightarrow 0} \frac{f(x_0 + \eta e_i) - f(x_0)}{\eta} \geq 0$					
Chain rule					
Let $g: \mathbb{R}^m \rightarrow \mathbb{R}^n$ and $f: \mathbb{R}^n \rightarrow \mathbb{R}$ partial derivative of f at x with respect to y_j is $\frac{\partial f}{\partial y_j}(x) = \frac{\partial f}{\partial z_i}(x) \frac{\partial z_i}{\partial y_j}(x)$ where $i = 1, \dots, n$ and $j = 1, \dots, m$					
Jacobian matrix					
Let $g: \mathbb{R}^m \rightarrow \mathbb{R}^n$ and $f: \mathbb{R}^n \rightarrow \mathbb{R}$ then $\frac{\partial f}{\partial y_j}(x) = \frac{\partial f}{\partial z_i}(x) \frac{\partial z_i}{\partial y_j}(x)$ where $i = 1, \dots, n$ and $j = 1, \dots, m$					
Matrix multiplication					
Let $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$ then $AB \in \mathbb{R}^{m \times p}$ and $(AB)_{ij} = \sum_k A_{ik} B_{kj}$					
Inverse matrix					
Let $A \in \mathbb{R}^{n \times n}$ then $A^{-1} A = I_n$ and $A A^{-1} = I_n$					
Transpose					
Let $A \in \mathbb{R}^{m \times n}$ then $A^T \in \mathbb{R}^{n \times m}$ and $(A^T)_{ij} = A_{ji}$					
Trace					
Let $A \in \mathbb{R}^{n \times n}$ then $\text{Tr}(A) = \sum_{i=1}^n A_{ii}$					
Diagonal matrix					
Let $A \in \mathbb{R}^{n \times n}$ then $\text{diag}(A) \in \mathbb{R}^n$ and $\text{diag}(A) = [A_{11}, \dots, A_{nn}]^T$					
Vector products, angle and cosine					
$\langle u, v \rangle = u^T v = \sum_{i=1}^n u_i v_i$ (dot product) $\ u\ = \sqrt{\langle u, u \rangle} = \sqrt{\sum_{i=1}^n u_i^2}$ (l2 norm) $\langle u, v \rangle \leq \ u\ \ v\ $ (Cauchy-Schwarz inequality) $\ u + v\ ^2 = \ u\ ^2 + \ v\ ^2 + 2 \langle u, v \rangle$ (law of cosine) $\ u\ ^2 = \sum_{i=1}^n u_i^2, \ u\ \geq 0, \ u\ = 0 \iff u = 0$ (l1 norm) $\ u\ _p = \left(\sum_{i=1}^n u_i ^p \right)^{1/p}, p \geq 1$ (generalized l1 norm)					
Orthogonality, vector space, basis, dimension					
$u \perp v \iff \langle u, v \rangle = 0$ (Orthogonality) $u \perp v \iff \ u + v\ ^2 = \ u\ ^2 + \ v\ ^2$ (Pythagorean)					
Linear algebra I					
Notations					
A, B, C are $n \times n$ matrices of \mathbb{C} \mathbb{C}^n is the set of n -tuples of complex numbers $\mathbb{M}_n(\mathbb{C})$ is the set of $n \times n$ matrices of \mathbb{C} $\mathbb{M}_m \times n(\mathbb{C})$ is the set of $m \times n$ matrices of \mathbb{C}					
Elementary operations					
$(A \cdot B)_i = \sum_{j=1}^n A_{ij} B_{j1}$ $(AB)_{ij} = \sum_{k=1}^n A_{ik} B_{kj}$					
Matrix inverse					
$(A^{-1})_{ij} = \frac{1}{\det(A)} \text{adj}(A)_{ij}$					
Properties of determinants					
$\det(A) = \det(A^T)$ $\det(A) = \det(A^{-1})^{-1}$ $\det(A) = \det(A^T)$ $\det(A) = \det(A^{-1})^{-1}$					
Properties of eigenvalues					
λ is an eigenvalue of A if $\exists \mathbf{v} \neq 0$ such that $A \mathbf{v} = \lambda \mathbf{v}$					
λ is an eigenvector of A if $\exists \mathbf{v} \neq 0$ such that $A \mathbf{v} = \lambda \mathbf{v}$					
Properties of eigenvectors					
$\lambda_1, \dots, \lambda_n$ are eigenvalues of A , then A is invertible. $A^{-1} = \lambda_1^{-1} \mathbf{v}_1 \mathbf{v}_1^T + \dots + \lambda_n^{-1} \mathbf{v}_n \mathbf{v}_n^T$					
Properties with eigenvalues					
$(A + \beta I_n)^{-1} = \sum_{i=1}^n \frac{1}{\lambda_i + \beta} \mathbf{v}_i \mathbf{v}_i^T$ $\text{If } A \text{ is invertible, then } \det(A) = \prod_{i=1}^n \lambda_i$					
Properties with eigenvalues and eigenvectors					
$\text{rank}(A) = \text{rank}(A^T)$ $\text{rank}(A) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T)$ $\text{rank}(A) = \text{rank}(A^{-1})$					
Properties of independence and zeros					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of independence and zeros					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear functions					
$\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$ $\text{rank}(A) = \text{rank}(A^T) = \text{rank}(A^{-1})$					
Properties of linear					

How?

How? – Piazza

<https://piazza.com/ucsd/fall2019/ece285mlip>

ECE 285 MLIP: Machine learning for image processing

Lecture Notes

Chapter 1 - Introduction

Chapter 2 - Probabilistic vs deep learning

Lecture Date

Sep 27, 2018

Oct 8, 2018

Homework

Assignment 1 - Python, Numpy and Matplotlib

Date Due

Oct 12, 2018

Copyright © 2017 Piazza Technologies, Inc. All Rights Reserved.

If you cannot access it contact me asap
at cdeledalle@ucsd.edu
(title: “[ECE285-MLIP] [Piazza] Access issues”).

Misc

Programming environment: Python/PyTorch/Jupyter

- We will use UCSD's DSMLP cluster with GPU/CUDA. Great but busy.
- We recommend you to install Conda/Python 3/Jupyter on your laptop. (please refer to additional documentations on Piazza), or to use any other platforms you may have access to.

Communication:

- All your emails **must have** a title starting with “[ECE285-MLIP]”
→ or it will end up in my spam/trash.

Note: “[ECE 285-MLIP]”, “[ece285 MLIP]”, “(ECE285MLIP)” are invalid!

- But avoid emails, use Piazza to communicate instead.
- For questions that may interest everyone else, post on Piazza forums.

Some references

Reference books



C. Bishop

Pattern recognition and Machine Learning

Springer, 2006



T. Hastie, R. Tibshirani, J. Friedman

The Elements of Statistical Learning: Data Mining, Inference, and Prediction

Springer, 2009

<http://web.stanford.edu/~hastie/ElemStatLearn/>



D. Barber

Bayesian Reasoning and Machine Learning

Cambridge University Press, 2012

<http://www.cs.ucl.ac.uk/staff/d.barber/brml/>



I. Goodfellow, Y. Bengio and A. Courville.

Deep Learning

MIT Press book, 2017

<http://www.deeplearningbook.org/>

Some references

Reference online classes



Fei-Fei Li, Justin Johnson and Serena Yeung, 2017 (Stanford)
CS231n: Convolutional Neural Networks for Visual Recognition
<http://cs231n.stanford.edu>



Giró et al, 2017 (Catalonia)
Deep Learning for Artificial Intelligence
<https://telecombcn-dl.github.io/2017-dlai/>



Leonardo Araujo dos Santos.
Artificial Intelligence
<https://www.gitbook.com/@leonardoaraujosantos>

Image sciences

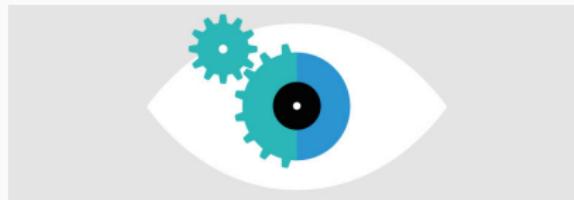
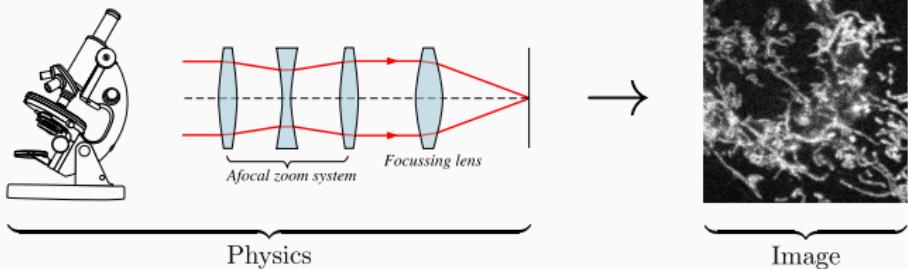


Image sciences

- Imaging:

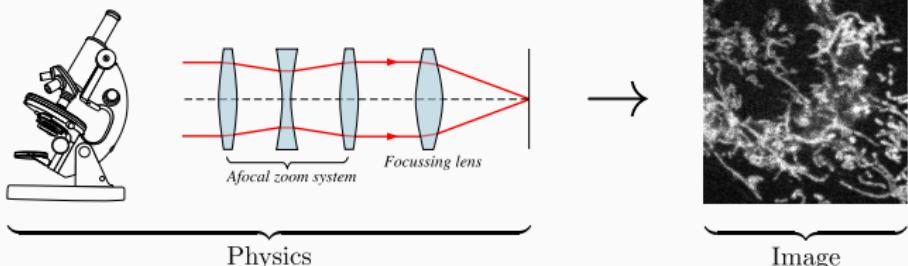


Modeling the image formation process

Imaging sciences – Overview

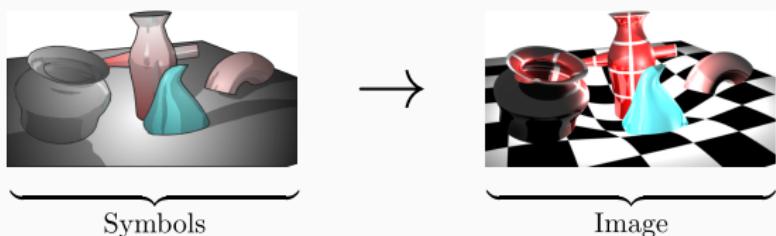
Image sciences

- Imaging:



Modeling the image formation process

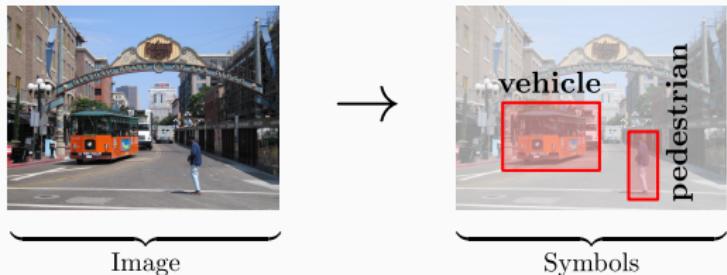
- Computer graphics:



Rendering images/videos from symbolic representation

Image sciences

- Computer vision:



Extracting information from images/videos

Imaging sciences – Overview

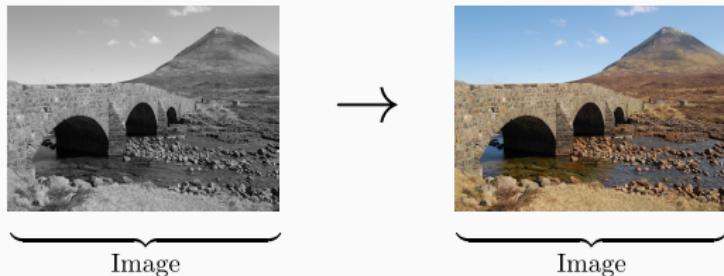
Image sciences

- Computer vision:



Extracting information from images/videos

- Image/Video processing:



Producing new images/videos from input images/videos

Spectrum from image processing to computer vision

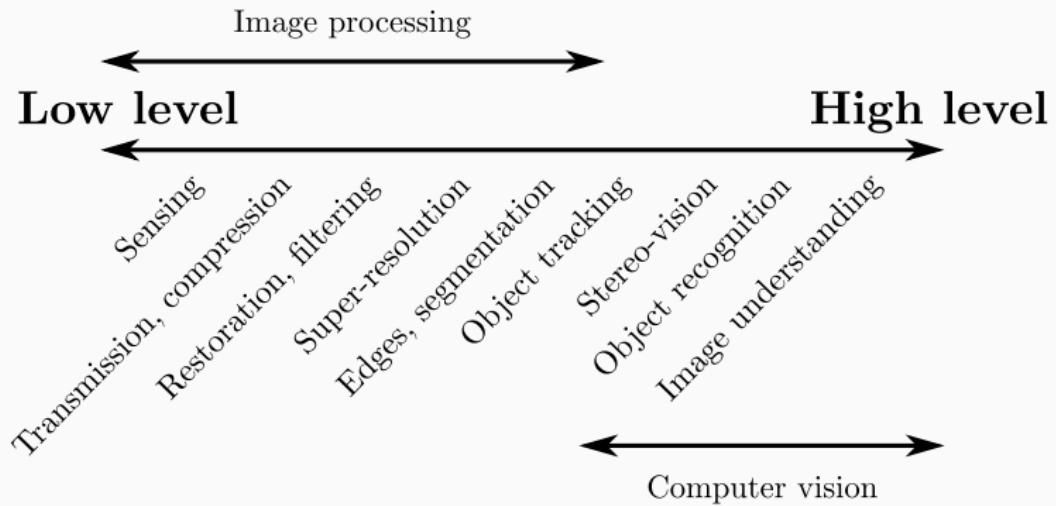


Image processing

Denoising



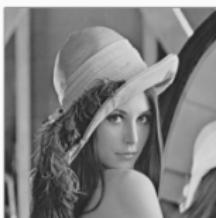
Enhancement



Compression

ctf_2	32 KB	JPEG Image
ctf_2	916 KB	PostScript

Feature detection



Inpainting



Super-resolution



(Source: Iasonas Kokkinos)

- Image processing: define a new image from an existing one
- Video processing: same problems + motion information

Image processing

Denoising



Enhancement



Compression

ctf_2	32 KB	JPEG Image
ctf_2	916 KB	PostScript

Feature detection



Inpainting



Super-resolution



(Source: Iasonas Kokkinos)

- Image processing: define a new image from an existing one
- Video processing: same problems + motion information

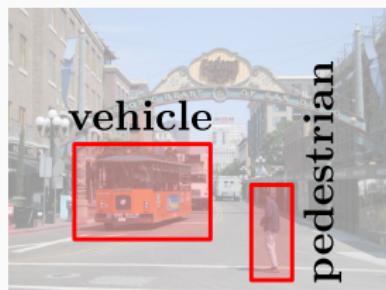
Computer vision

Definition (The British Machine Vision Association)

Computer vision (CV) is concerned with the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images.



Image



Symbols

CV is a subfield of Artificial Intelligence.

Computer vision – Artificial Intelligence (AI)

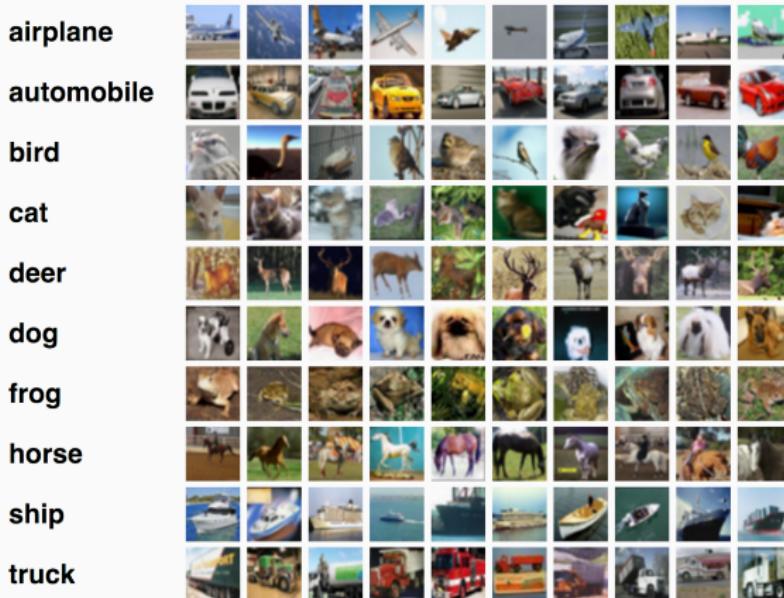
Definition (Oxford dictionary)

artificial intelligence, *noun*: the theory and development of computer systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation.

Remark:

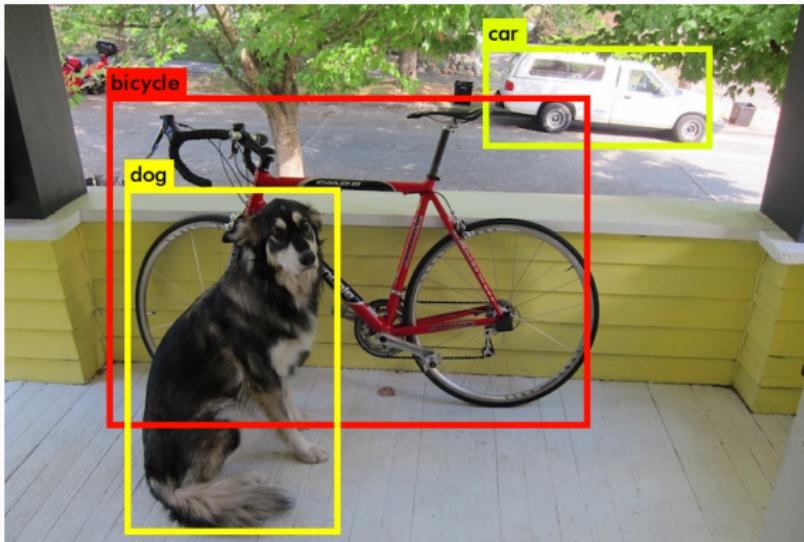
CV is a subfield of AI,
CV's new very best friend is machine learning (ML),
ML is also a subfield of AI,
but not all computer vision algorithms are ML.

Computer vision – Image classification



Goal: to assign a given image into one of the predefined classes.

Computer vision – Object detection



(Source: Joseph Redmon)

Goal: to detect instances of objects of a certain class (such as human).

Computer vision – Image segmentation



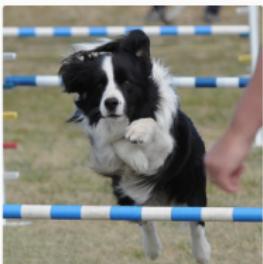
(Source: Abhijit Kundu)

Goal: to partition an image into multiple segments such that pixels in a same segment share certain characteristics (color, texture or semantic).

Computer vision – Image captioning



"girl in pink dress is jumping in air."



"black and white dog jumps over bar."



"young girl in pink shirt is swinging on swing."



"man in blue wetsuit is surfing on wave."



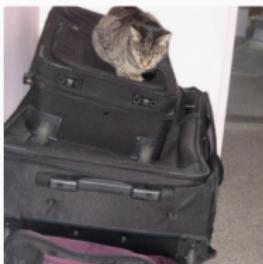
"little girl is eating piece of cake."



"baseball player is throwing ball in game."



"woman is holding bunch of bananas."

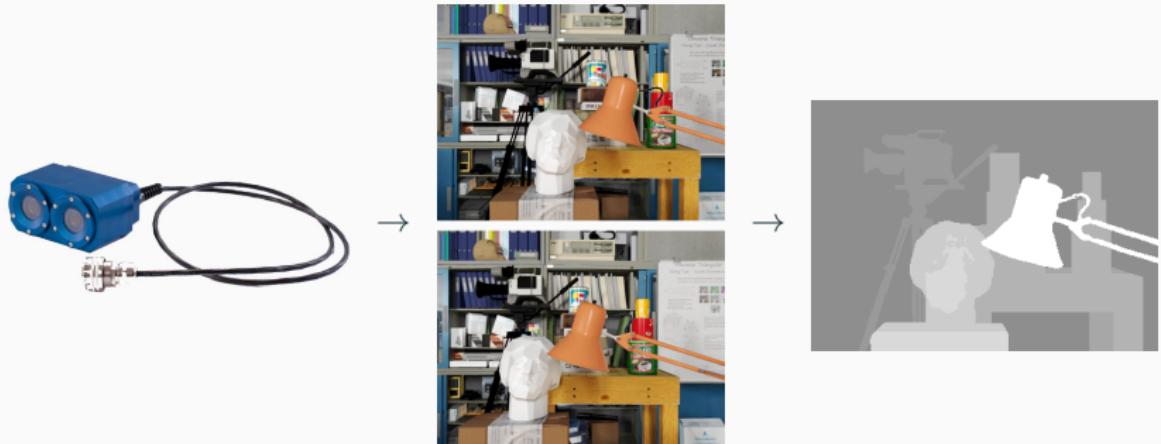


"black cat is sitting on top of suitcase."

(Karpathy, Fei-Fei, CVPR, 2015)

Goal: to write a sentence that describes what is happening.

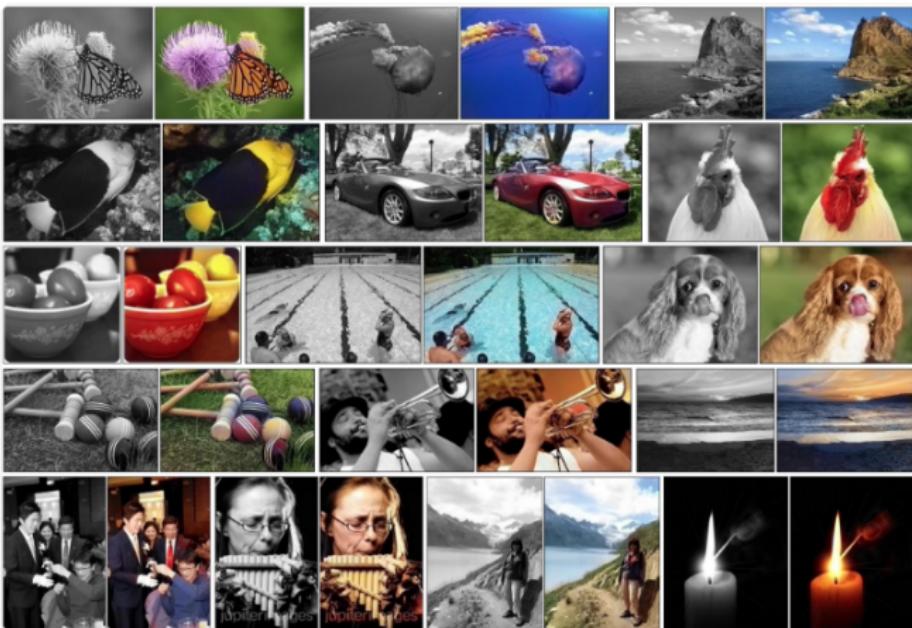
Computer vision – Depth estimation



(*Stereo-vision: from two images acquired with different views.*)

Goal: to estimate a depth map from one, two or several frames.

Image colorization



(Source: Richard Zhang, Phillip Isola and Alexei A. Efros, 2016)

Goal: to add color to grayscale photographs.

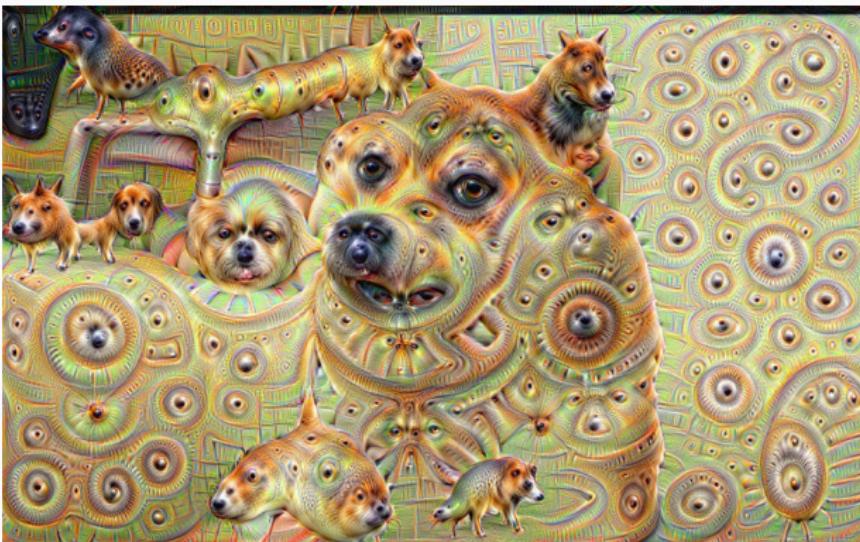
Image generation



Generated images of bedrooms (Source: Alec Radford, Luke Metz, Soumith Chintala, 2015)

Goal: to automatically create realistic pictures of a given category.

Image generation – DeepDream



(Source: Google Deep Dream, Mordvintsev et al., 2016)

Goal: to generate arbitrary photo-realistic artistic images,
and understand/visualizing deep networks.

Image stylization



Goal: to create stylized images from rough sketches.

Style transfer



(Source: Gatys, Ecker and Bethge, 2015)

Goal: transfer the style of an image into another one.

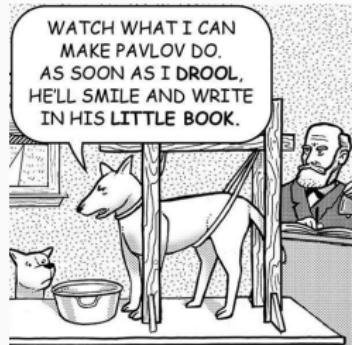
Machine learning



What is learning?

Herbert Simon (Psychologist, 1916–2001):

Learning is any process by which a system improves performance from experience.



Pavlov's dog (Mark Stivers, 2003)

Tom Mitchell (Computer Scientist):

*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**.*

Machine learning (ML)

Definition

machine learning, *noun*: type of Artificial Intelligence that provides computers with the ability to **learn without being explicitly programmed**.

Traditional Programming



Machine Learning

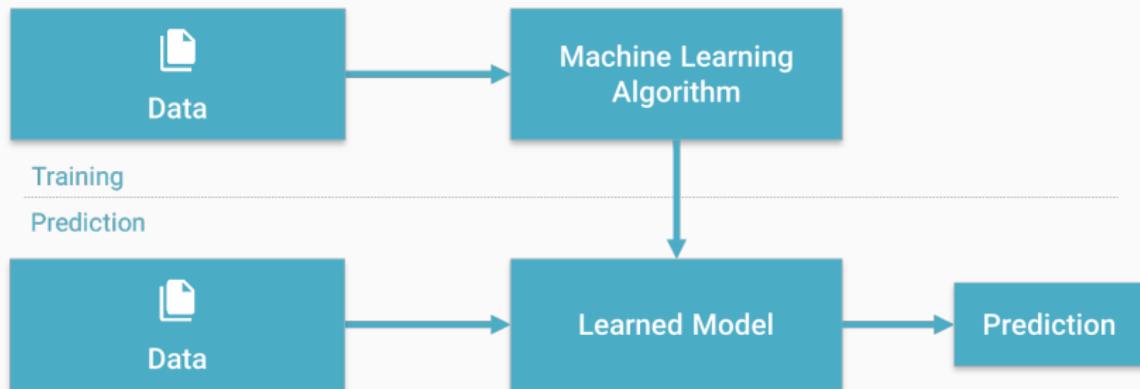


(Source: Pedro Domingos)

Machine learning (ML)

Provides **various techniques** that can learn from and make predictions on data.

Most of them follow the same general structure:



(Source: Lucas Masuch)

Learning from examples

3 main ingredients

- ① Training set / examples:

$$\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$$

- ② Machine or model:

$$\mathbf{x} \rightarrow \underbrace{f(\mathbf{x}; \theta)}_{\text{function / algorithm}} \rightarrow \underbrace{\mathbf{y}}_{\text{prediction}}$$

θ : parameters of the model

- ③ Loss, cost, objective function / energy:

$$\operatorname{argmin}_{\theta} E(\theta; \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$$

Learning from examples

Tools:

$$\left\{ \begin{array}{l} \text{Data} \leftrightarrow \text{Statistics} \\ \text{Loss} \leftrightarrow \text{Optimization} \end{array} \right.$$

Goal: to extract information from the training set

- relevant for the given task,
- relevant for other data of the same kind.

Terminology

Sample (Observation or Data): item to process (e.g., classify). *Example: an individual, a document, a picture, a sound, a video...*

Features (Input): set of distinct traits that can be used to describe each sample in a quantitative manner. Represented as a multi-dimensional vector usually denoted by x . *Example: size, weight, citizenship, ...*

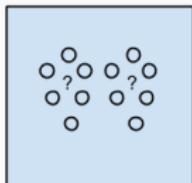
Training set: Set of data used to discover potentially predictive relationships.

Testing set: Set used to assess the performance of a model.

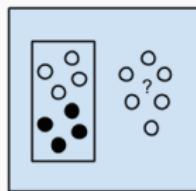
Label (Output): The class or outcome assigned to a sample. The actual prediction is often denoted by y and the desired/targeted class by d or t . *Example: man/woman, wealth, education level, ...*

Machine learning – Learning approaches

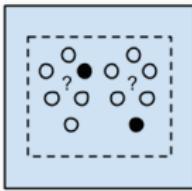
Learning approaches



Unsupervised Learning
Algorithms



Supervised Learning
Algorithms



Semi-supervised
Learning Algorithms

Unsupervised learning: Discovering patterns in unlabeled data. *Example: cluster similar documents based on the text content.*

Supervised learning: Learning with a labeled training set. *Example: email spam detector with training set of already labeled emails.*

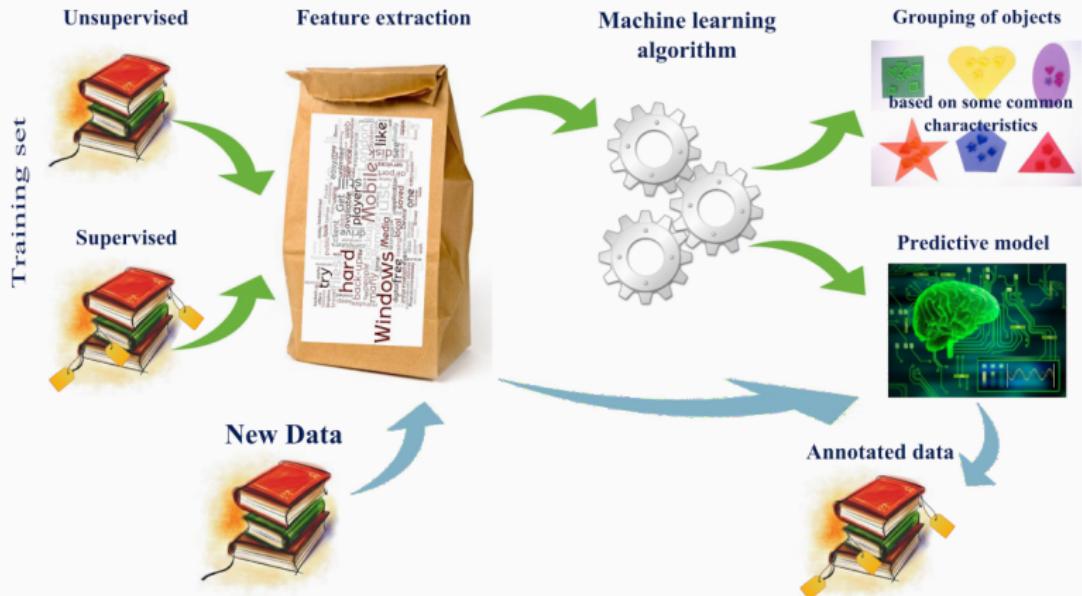
Semisupervised learning: Learning with a small amount of labeled data and a large amount of unlabeled data. *Example: web content and protein sequence classifications.*

Reinforcement learning: Learning based on feedback or reward. *Example: learn to play chess by winning or losing.*

(Source: Jason Brownlee and Lucas Masuch)

Machine learning – Workflow

Machine learning workflow



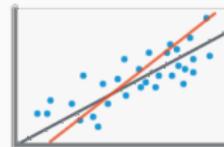
(Source: Michael Walker)

Machine learning – Problem types

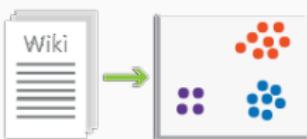
Problem types



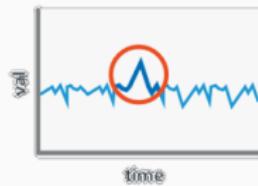
Classification
(supervised – predictive)



Regression
(supervised – predictive)



Clustering
(unsupervised – descriptive)



Anomaly Detection
(unsupervised – descriptive)

(Source: Lucas Masuch)

Unsupervised learning

Unsupervised learning

- **Training set:** $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$ where $\mathbf{x}_i \in \mathbb{R}^d$.
- **Goal:** to find **interesting structures** in the data \mathbf{X} .

Examples: $\left\{ \begin{array}{l} \bullet \text{ clustering,} \\ \bullet \text{ quantile estimation,} \\ \bullet \text{ outlier detection,} \\ \bullet \text{ dimensionality reduction.} \end{array} \right.$

Statistical point of view

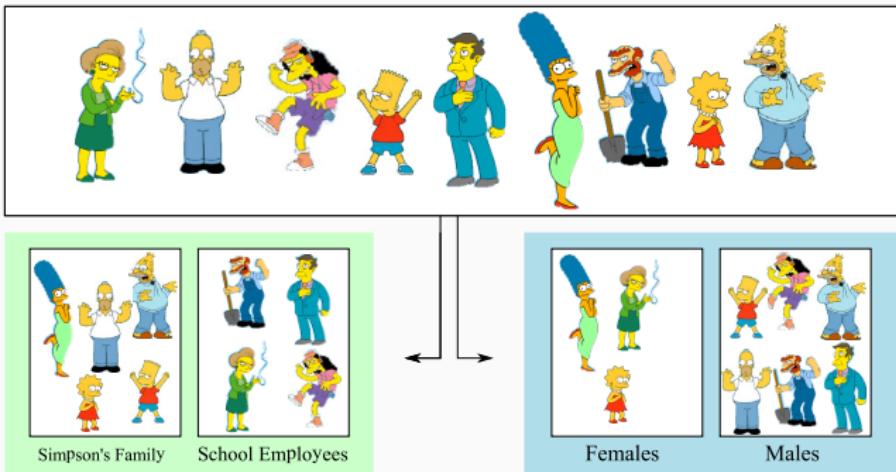
To estimate a density p which is likely to have generated \mathbf{X} , i.e., such that

$$\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N \stackrel{\text{i.i.d}}{\sim} p$$

(i.i.d = identically and independently distributed).

Clustering

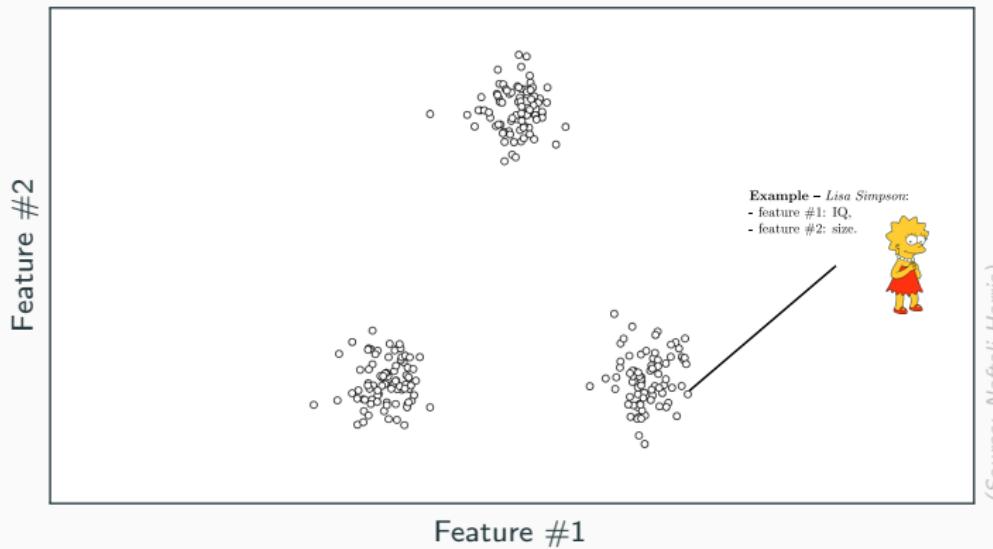
Clustering: group observations into “meaningful” groups.



(Source: Kasun Ranga Wijeweera)

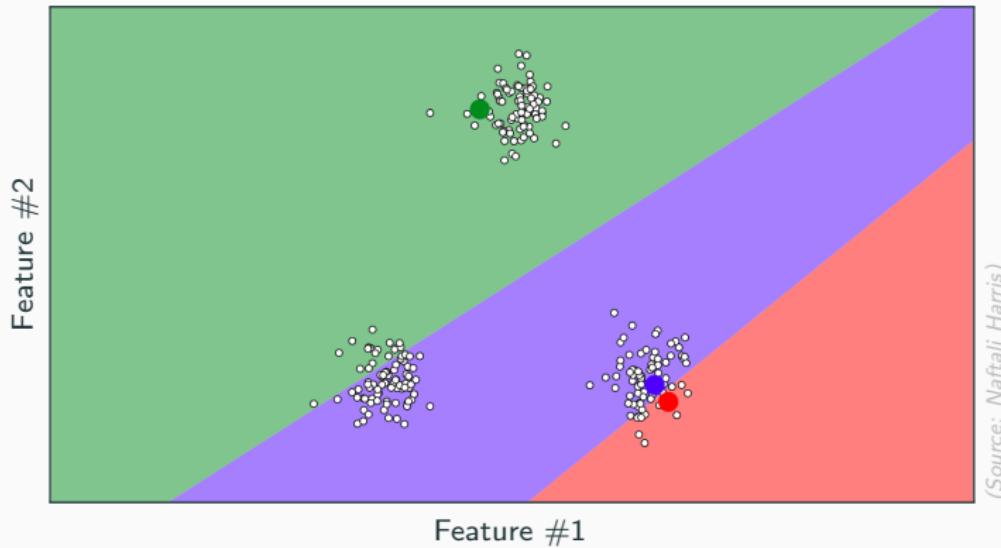
- Task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other.
- Popular ones are K-means clustering and Hierarchical clustering.

Clustering – K-means



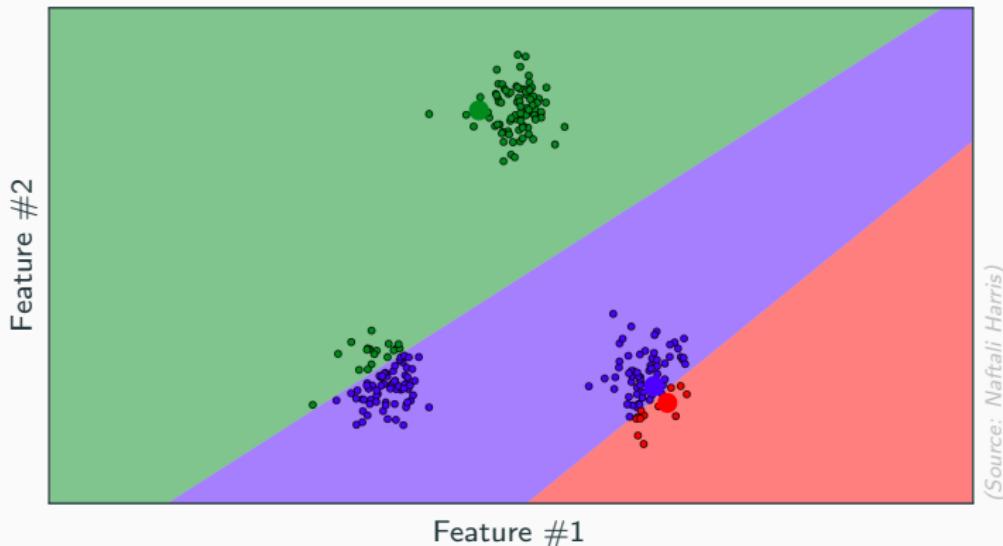
- ① Consider data in \mathbb{R}^2 spread on three different clusters,

Clustering – K-means



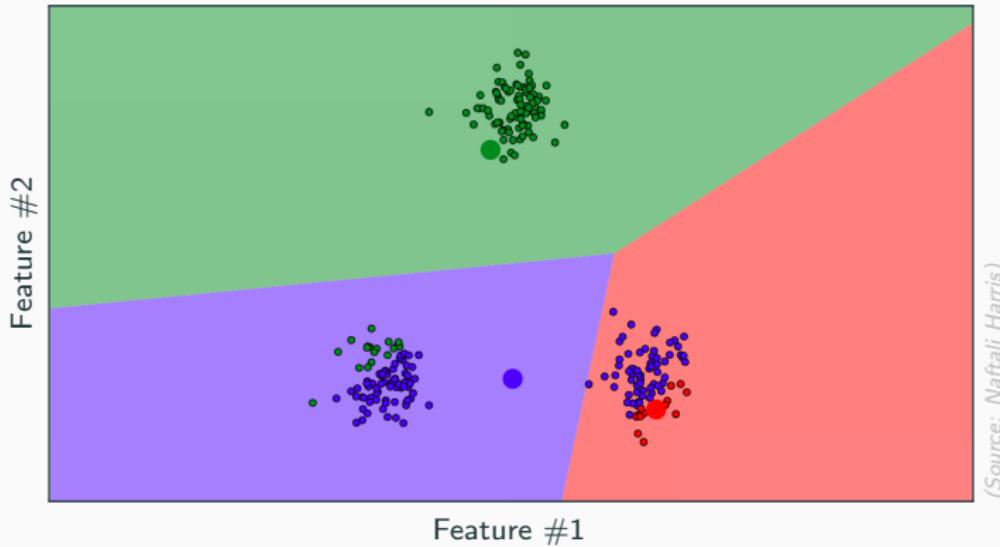
- ① Consider data in \mathbb{R}^2 spread on three different clusters,
- ② Pick randomly $K = 3$ data points as cluster centroids,

Clustering – K-means



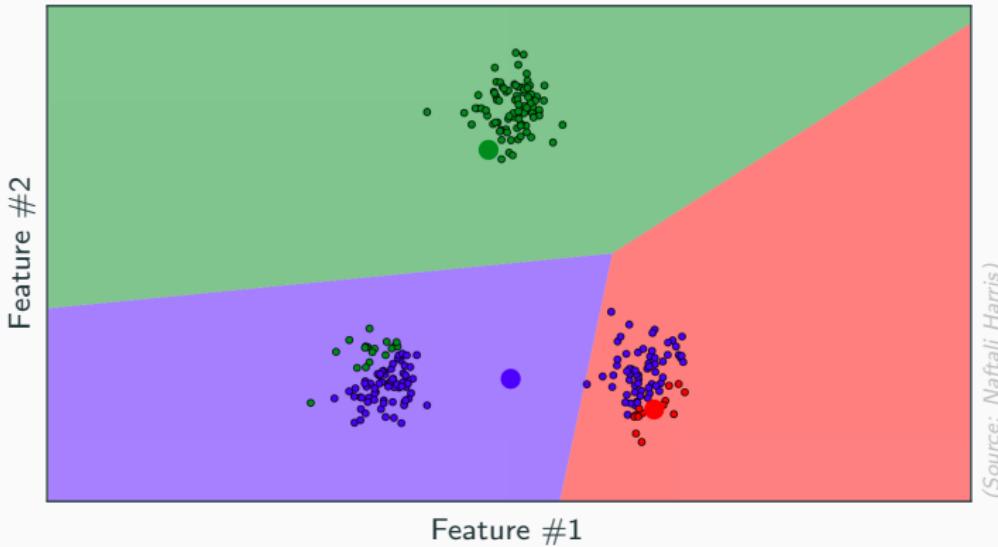
- ① Consider data in \mathbb{R}^2 spread on three different clusters,
- ② Pick randomly $K = 3$ data points as cluster centroids,
- ③ Assign each data point to the class with closest centroid,

Clustering – K-means



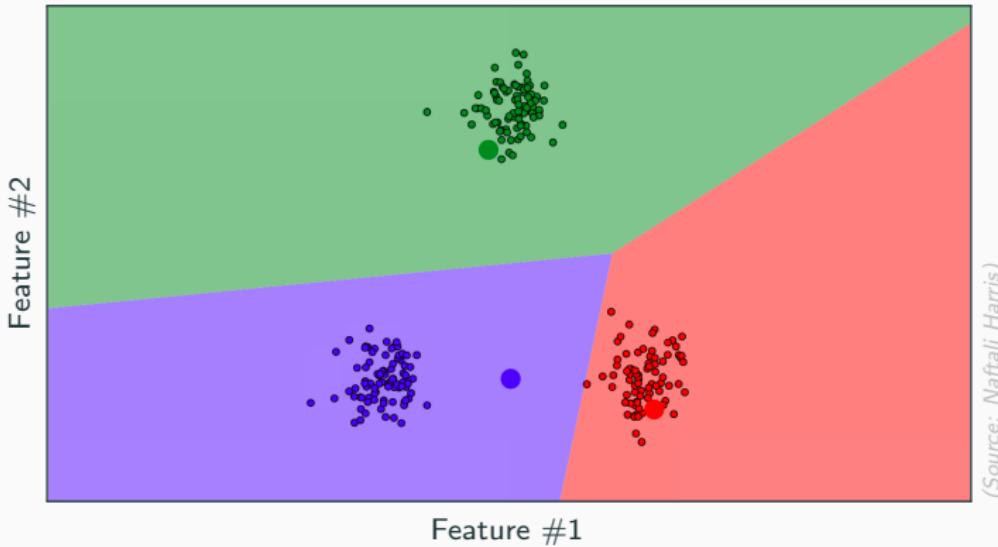
- ① Consider data in \mathbb{R}^2 spread on three different clusters,
- ② Pick randomly $K = 3$ data points as cluster centroids,
- ③ Assign each data point to the class with closest centroid,
- ④ Update the centroids by taking the means within the clusters,

Clustering – K-means



- ① Consider data in \mathbb{R}^2 spread on three different clusters,
- ② Pick randomly $K = 3$ data points as cluster centroids,
- ③ Assign each data point to the class with closest centroid,
- ④ Update the centroids by taking the means within the clusters,
- ⑤ Go back to 3 until no more changes.

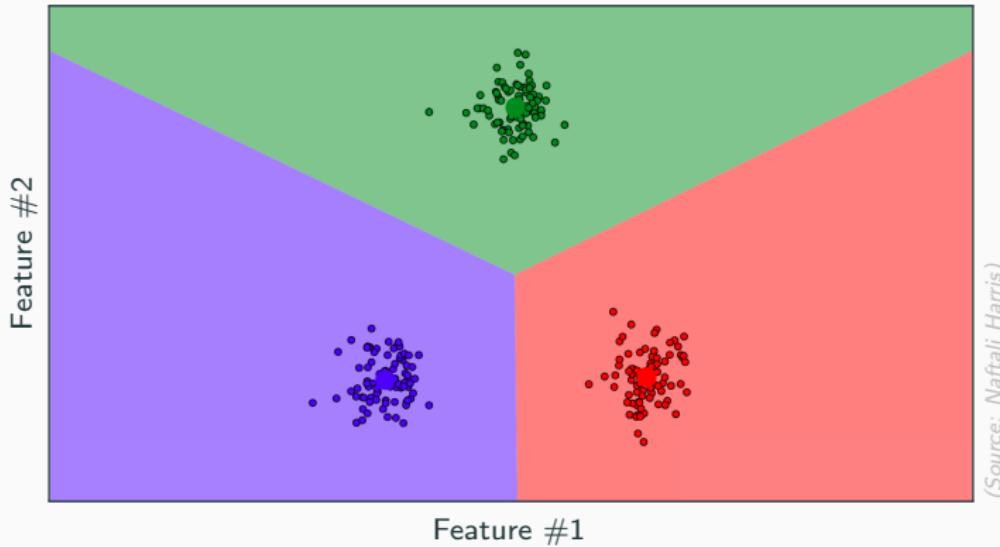
Clustering – K-means



(Source: Naftali Harris)

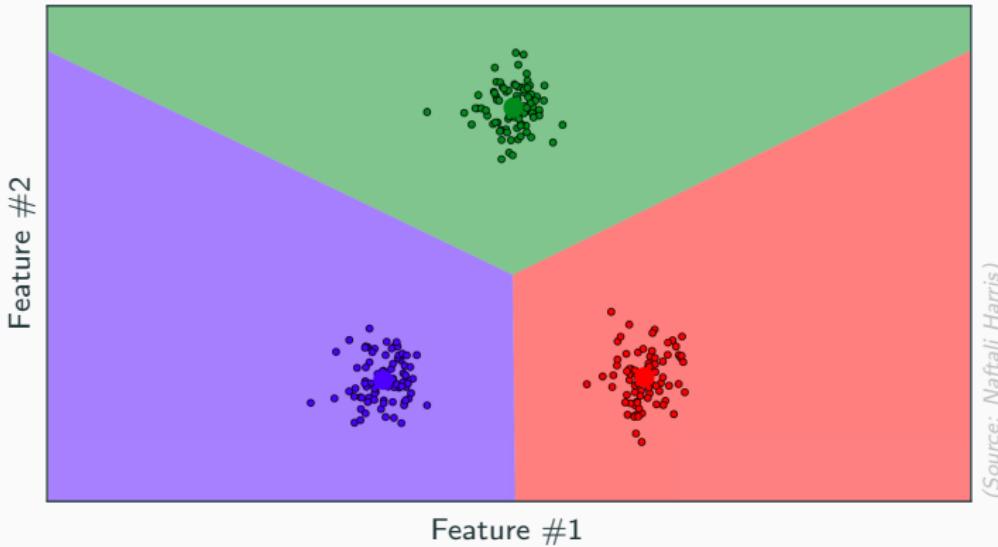
- ① Consider data in \mathbb{R}^2 spread on three different clusters,
- ② Pick randomly $K = 3$ data points as cluster centroids,
- ③ Assign each data point to the class with closest centroid,
- ④ Update the centroids by taking the means within the clusters,
- ⑤ Go back to 3 until no more changes.

Clustering – K-means



- ① Consider data in \mathbb{R}^2 spread on three different clusters,
- ② Pick randomly $K = 3$ data points as cluster centroids,
- ③ Assign each data point to the class with closest centroid,
- ④ Update the centroids by taking the means within the clusters,
- ⑤ Go back to 3 until no more changes.

Clustering – K-means



(Source: Naftali Harris)

- ① Consider data in \mathbb{R}^2 spread on three different clusters,
- ② Pick randomly $K = 3$ data points as cluster centroids,
- ③ Assign each data point to the class with closest centroid,
- ④ Update the centroids by taking the means within the clusters,
- ⑤ Go back to 3 until no more changes.

Clustering – K-means

- Parameter: a partition \mathcal{P} of $[1, N]$ in K subsets \mathcal{P}_k ,
- Try to be optimal in terms of inter-class variability (loss):

$$E(\mathcal{P}; \mathbf{x}_1, \dots, \mathbf{x}_N) = \sum_{k=1}^K \sum_{i \in \mathcal{P}_k} \|\mathbf{x}_i - \mathbf{c}_k\|_2^2 \quad \text{with} \quad \mathbf{c}_k = \underbrace{\frac{1}{|\mathcal{P}_k|} \sum_{i \in \mathcal{P}_k} \mathbf{x}_i}_{\text{centroid of cluster } k}$$

- Solutions **strongly depend on the initialization** (local minima),
 - Good initializations can be obtained by K-means++ strategy.
- The number of class **K is often unknown**
 - not a parameter but a *hyper-parameter* (not optimized),
 - usually found by trial and error on a validation set (\neq training set).
- The data dimension d is often much larger than 2,
 - subject to the curse of dimensionality. *(we will come back to this)*

Supervised learning

Supervised learning

- A training labeled set: $(\mathbf{x}_1, d_1), (\mathbf{x}_2, d_2), \dots, (\mathbf{x}_N, d_N)$.
- Goal: to learn a relevant mapping f st

$$y_i = f(\mathbf{x}_i; \theta) \approx d_i$$

- Examples:
- | | | | |
|---|---|---|---|
| • classification (d is a categorical variable ^a), | { | } | • regression (d is a real variable), |
| a. can take one of a limited, and usually fixed, number of possible values. | | | |

Statistical point of view

- Discriminative models: to estimate the posterior distribution $p(d|\mathbf{x})$.
- Generative models: to estimate the likelihood $p(\mathbf{x}|d)$,
or the joint distribution $p(\mathbf{x}, d)$.

Supervised learning – Bayesian inference

Bayes rule

In the case of a categorical variable d and a real vector \mathbf{x}

$$p(d|\mathbf{x}) = \frac{p(\mathbf{x}, d)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|d)p(d)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|d)p(d)}{\sum_d p(\mathbf{x}|d)p(d)}$$

- $p(d|\mathbf{x})$: probability that \mathbf{x} is of class d ,
- $p(\mathbf{x}|d)$: distribution of \mathbf{x} within class d ,
- $p(d)$: frequency of class d .

Example of final classifier: $f(\mathbf{x}; \theta) = \operatorname{argmax}_d p(d|\mathbf{x})$

Generative models carry more information:

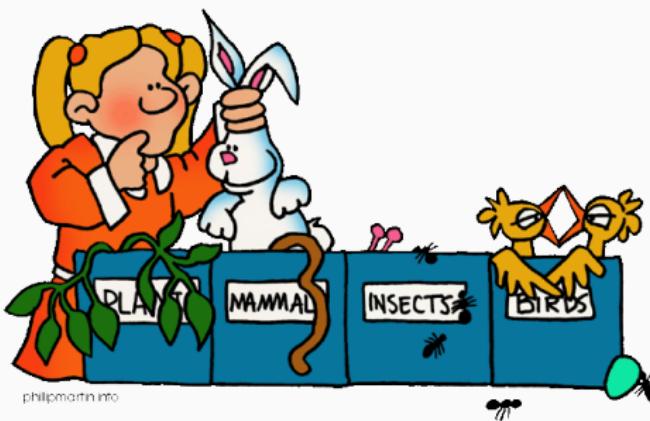
Learning $p(\mathbf{x}|d)$ and $p(d)$ allows to deduce $p(d|\mathbf{x})$.

But they often require many more parameters and more training data.

Discriminative models are usually easier to learn and thus more accurate.

Classification

Classification: predict class d from observation x .



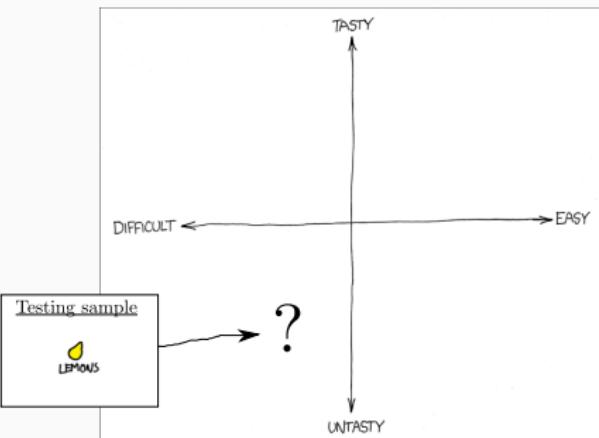
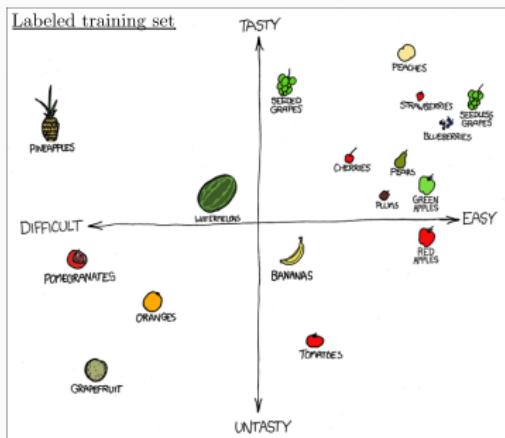
(Source: Philip Martin)

- Classify a document into a predefined category.
- Documents can be text, images, videos. . .
- Popular ones are Support Vector Machines and Artificial Neural Networks.

Machine learning – Regression

Regression

Regression (prediction): predict value(s) from observation.



- Statistical process for estimating the relationships among variables.
- Regression means to predict the output value using training data.
→ related to interpolation and extrapolation.
- Popular ones are linear least square and Artificial Neural Networks.

Classification vs Regression

Classification

- Assign to a class
- Ex: a type of tumor is harmful or not
- Output is discrete/categorical

v.s

Regression

- Predict one or several output values
- Ex: what will be the house price?
- Output is a real number/continuous



Classification

Will it be Cold or Hot tomorrow?



Regression

What is the temperature going to be tomorrow?



(Source: Ali Reza Kohani)

Quiz, which one is which?

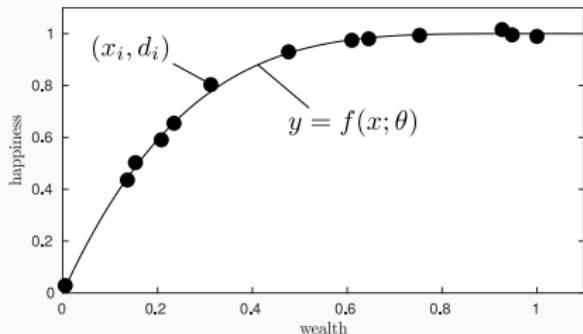
denoising, identification, verification, approximation.

Polynomial curve fitting

- Consider N individuals answering a survey asking for
 - their wealth: x_i
 - level of happiness: d_i
- We want to learn how to predict d_i (the desired output) from x_i as

$$d_i \approx y_i = f(x_i; \theta)$$

where f is the predictor and y_i denotes the predicted output.



Quiz

Supervised or unsupervised?

Classification or regression?

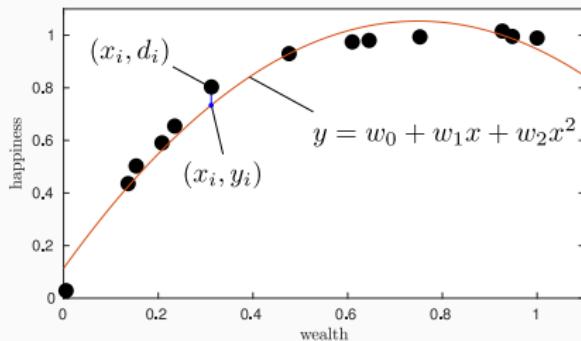
Polynomial curve fitting

- We assume that the relation is M -order polynomial

$$y_i = f(x_i; \mathbf{w}) = w_0 + w_1 x_i + w_2 x_i^2 + \dots + w_M x_i^M = \sum_{j=0}^M w_j x_i^j$$

where $\mathbf{w} = (w_0, w_1, \dots, w_M)^T$ are the polynomial coefficients.

- The (multi-dimensional) parameter θ is the vector \mathbf{w} .



Polynomial curve fitting

- Let $\mathbf{y} = (y_1, y_2, \dots, y_N)^T$ and $\mathbf{X} = \begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^M \\ 1 & x_2 & x_2^2 & \dots & x_2^M \\ \vdots & & & \ddots & \vdots \\ 1 & x_N & x_N^2 & \dots & x_N^M \end{pmatrix}$, then

$$\mathbf{y} = \mathbf{X}\mathbf{w} \quad \text{with} \quad \mathbf{w} = (w_0, w_1, \dots, w_M)^T$$

- Polynomial curve fitting is linear regression.
linear regression = linear relation between \mathbf{y} and θ , even though f is non-linear.
- Standard procedures involve minimizing the sum of square errors (SSE)

$$E(\mathbf{w}) = \sum_{i=1}^N (y_i - d_i)^2 = \|\mathbf{y} - \mathbf{d}\|_2^2 = \|\mathbf{X}\mathbf{w} - \mathbf{d}\|_2^2$$

also called sum of square differences (SSD), or
mean square error (MSE, when divided by N).

Linear regression + SSE → Linear least square regression

Polynomial curve fitting

Recall: $E(\mathbf{w}) = \|\mathbf{X}\mathbf{w} - \mathbf{d}\|_2^2 = (\mathbf{X}\mathbf{w} - \mathbf{d})^T(\mathbf{X}\mathbf{w} - \mathbf{d})$

Note that: $\nabla_{\mathbf{w}}^T \mathbf{A}\mathbf{w} = (\mathbf{A} + \mathbf{A}^T)\mathbf{w}$ and $\nabla_{\mathbf{b}}^T \mathbf{w} = \mathbf{b}$

- The solution is obtained by canceling the gradient

$$\nabla E(\mathbf{w}) = 0 \Rightarrow \underbrace{\mathbf{X}^T(\mathbf{X}\mathbf{w} - \mathbf{d})}_{\text{normal equation}} = 0$$

- As soon as we have $N \geq M + 1$ distinct x_i , the solution is unique

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{d}$$

- Otherwise, there is an infinite number of solutions.

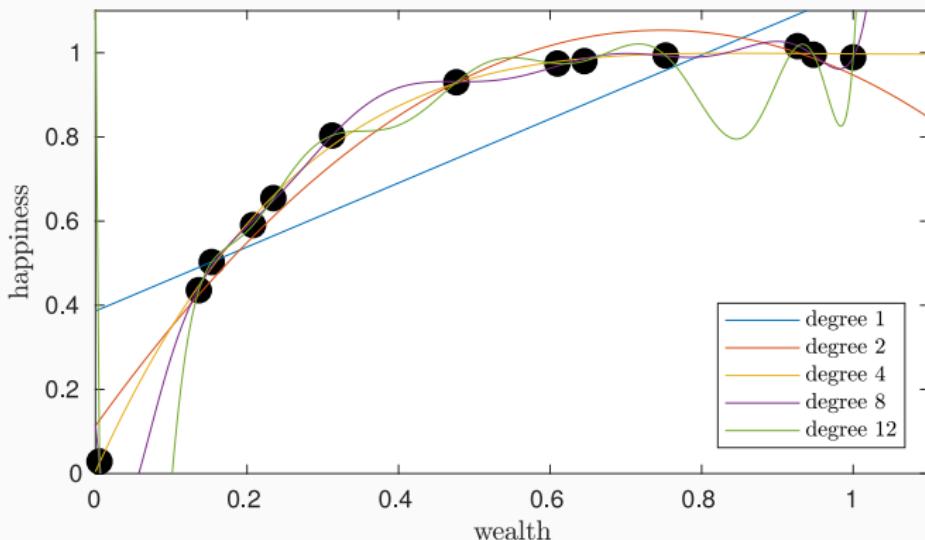
Polynomial curve fitting

- **Training data:** answers to the survey
- **Model:** polynomial function of degree M
- **Loss:** sum of square errors
- **Machine learning algorithm:** linear least square regression

The methodology for **Deep Learning** will be the exact same one.

The only difference is that the relation between
 y and θ will be (extremely) non-linear.

Polynomial curve fitting



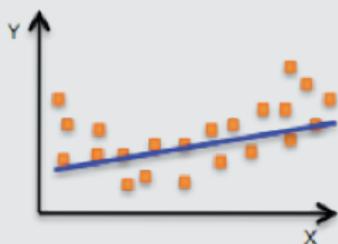
As M increases, unwanted oscillations appear (Runge's phenomenon), even though $N \geq M + 1$.

How to choose the hyper-parameter M ?

Difficulty of learning

- **Fit:** to explain the training samples,
→ requires some flexibility of the model.
- **Generalization:** to be accurate for samples outside the training dataset.
→ requires some rigidity of the model.

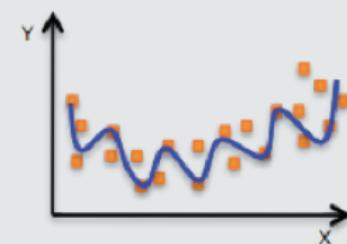
Example (Regression)



← Underfitting



Balanced

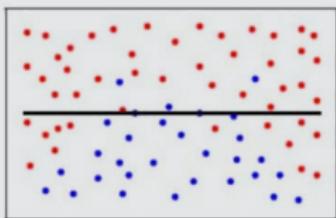


Overfitting →

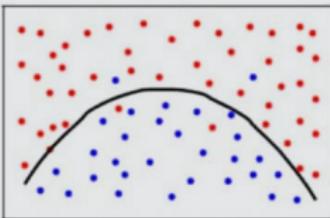
Difficulty of learning

- **Fit:** to explain the training samples,
→ requires some flexibility of the model.
- **Generalization:** to be accurate for samples outside the training dataset.
→ requires some rigidity of the model.

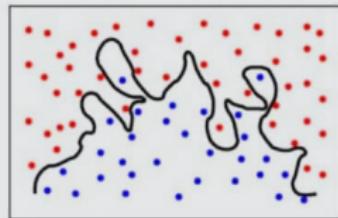
Example (Binary classification)



← Underfitting



Balanced

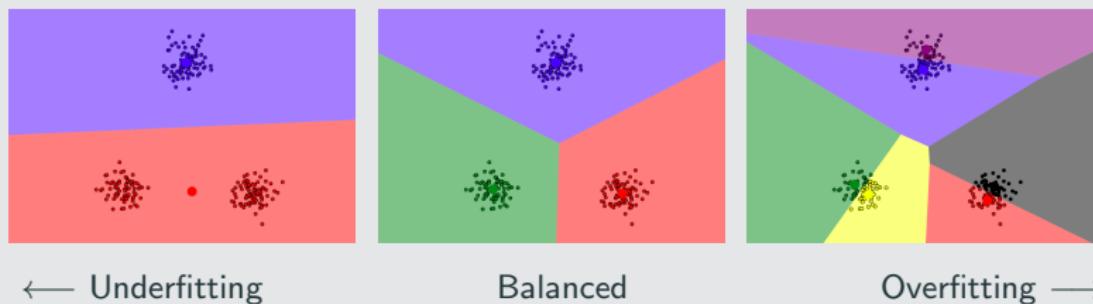


Overfitting →

Difficulty of learning

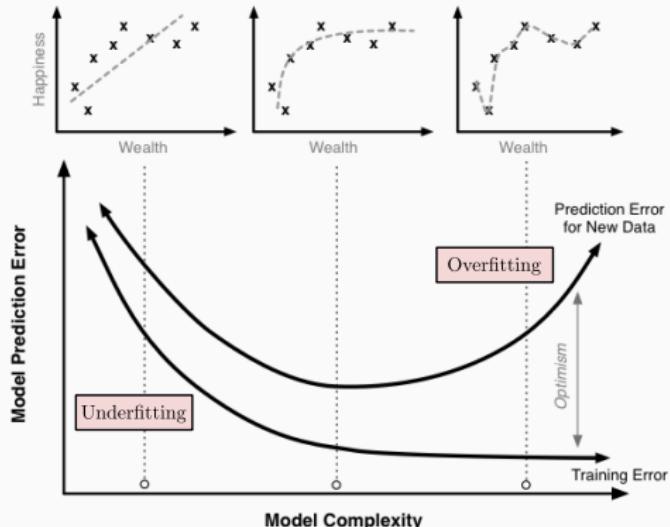
- **Fit:** to explain the training samples,
→ requires some flexibility of the model.
- **Generalization:** to be accurate for samples outside the training dataset.
→ requires some rigidity of the model.

Example (Clustering)



Machine learning – Overfitting and Generalization

Difficulty of learning



- Training error is optimistic towards overfitting.
- Prediction error must be evaluated on a validation set, not the training set.
- For the same reason, the performance of the best selected model must be evaluated on yet another set.

Complexity: number of parameters, degrees of freedom, capacity, richness, flexibility, see also Vapnik–Chervonenkis (VC) dimension.

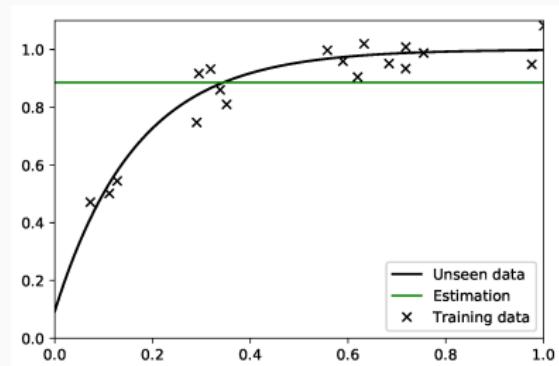
→ Often controlled by hyper-parameters (polynomial order, #clusters, etc).

Difficulty of learning

How to measure under and overfitting?

Variance: how much the **predictions** of my model on unseen data fluctuate if trained over different but similar training sets (high \Rightarrow overfitting).

Bias: how off is the **average** of these predictions (high \Rightarrow underfitting).



(a) Training set #1

Figure 1 – Example with polynomial curve fitting of order $M = 1$

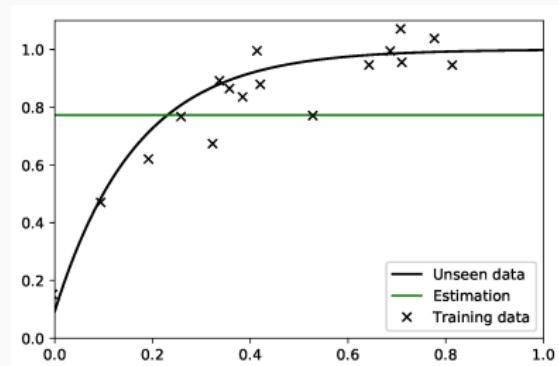
Machine learning – Overfitting and Generalization

Difficulty of learning

How to measure under and overfitting?

Variance: how much the **predictions** of my model on unseen data fluctuate if trained over different but similar training sets (high \Rightarrow overfitting).

Bias: how off is the **average** of these predictions (high \Rightarrow underfitting).



(a) Training set #2

Figure 1 – Example with polynomial curve fitting of order $M = 1$

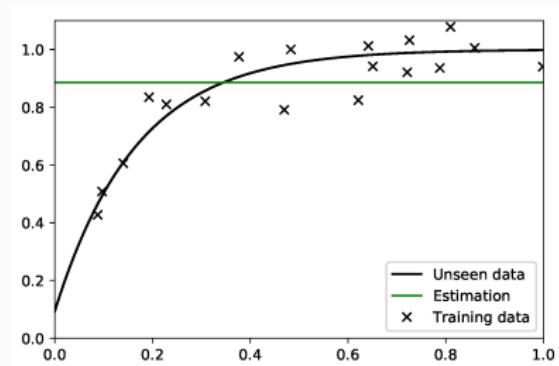
Machine learning – Overfitting and Generalization

Difficulty of learning

How to measure under and overfitting?

Variance: how much the **predictions** of my model on unseen data fluctuate if trained over different but similar training sets (high \Rightarrow overfitting).

Bias: how off is the **average** of these predictions (high \Rightarrow underfitting).



(a) Training set #3

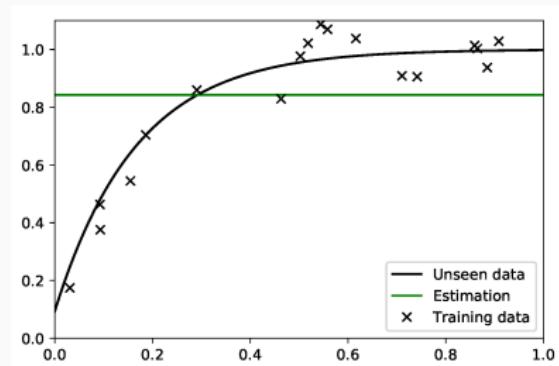
Figure 1 – Example with polynomial curve fitting of order $M = 1$

Difficulty of learning

How to measure under and overfitting?

Variance: how much the **predictions** of my model on unseen data fluctuate if trained over different but similar training sets (high \Rightarrow overfitting).

Bias: how off is the **average** of these predictions (high \Rightarrow underfitting).



(a) Training set #4

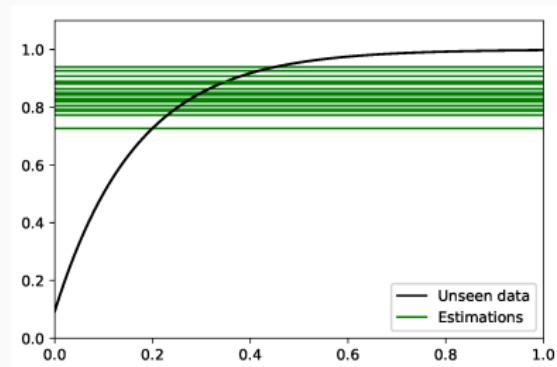
Figure 1 – Example with polynomial curve fitting of order $M = 1$

Difficulty of learning

How to measure under and overfitting?

Variance: how much the **predictions** of my model on unseen data fluctuate if trained over different but similar training sets (high \Rightarrow overfitting).

Bias: how off is the **average** of these predictions (high \Rightarrow underfitting).



(a) Training sets

Figure 1 – Example with polynomial curve fitting of order $M = 1$

Machine learning – Overfitting and Generalization

Difficulty of learning

How to measure under and overfitting?

Variance: how much the **predictions** of my model on unseen data fluctuate if trained over different but similar training sets (high \Rightarrow overfitting).

Bias: how off is the **average** of these predictions (high \Rightarrow underfitting).

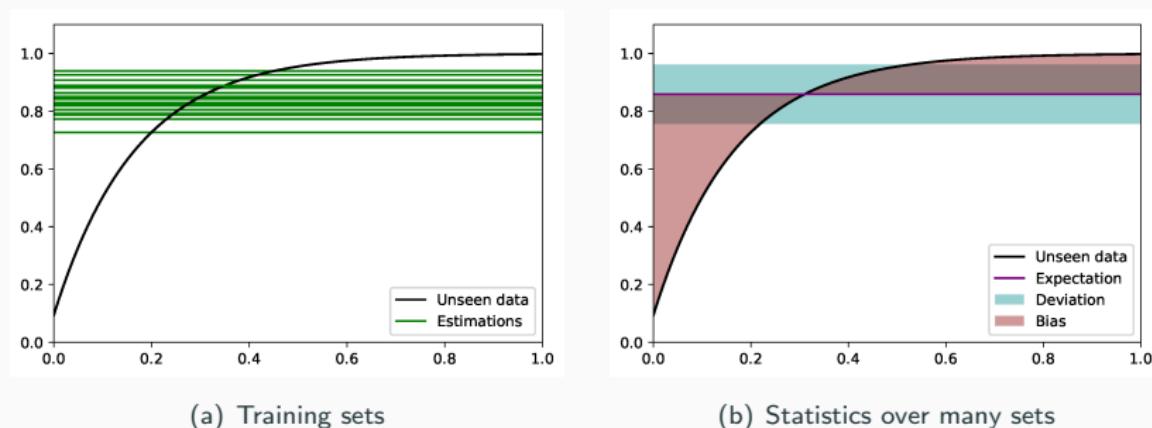


Figure 1 – Example with polynomial curve fitting of order $M = 1$

Difficulty of learning

How to measure under and overfitting?

Variance: how much the **predictions** of my model on unseen data fluctuate if trained over different but similar training sets (high \Rightarrow overfitting).

Bias: how off is the **average** of these predictions (high \Rightarrow underfitting).

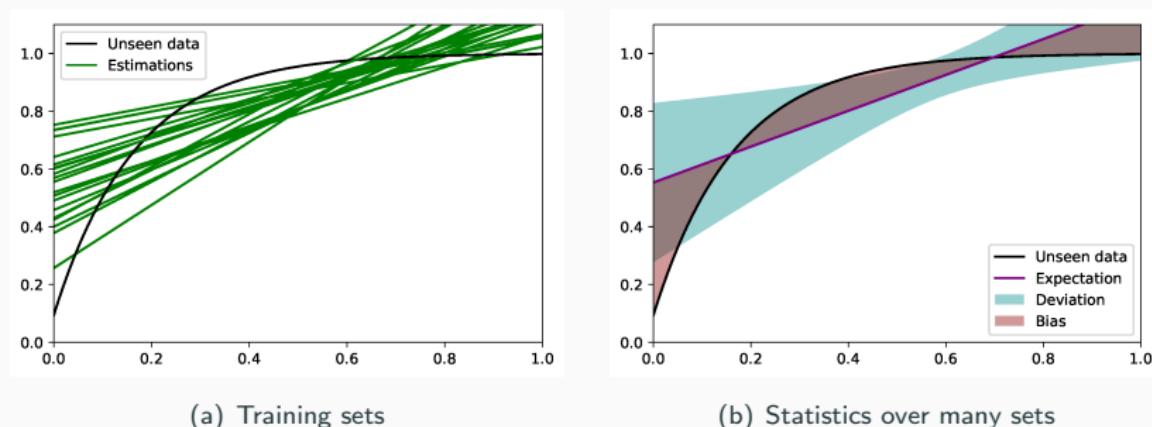


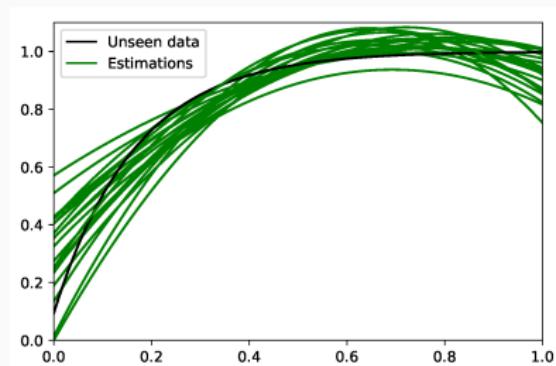
Figure 1 – Example with polynomial curve fitting of order $M = 2$

Difficulty of learning

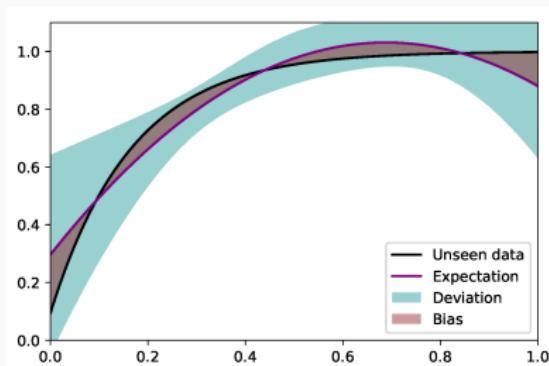
How to measure under and overfitting?

Variance: how much the **predictions** of my model on unseen data fluctuate if trained over different but similar training sets (high \Rightarrow overfitting).

Bias: how off is the **average** of these predictions (high \Rightarrow underfitting).



(a) Training sets



(b) Statistics over many sets

Figure 1 – Example with polynomial curve fitting of order $M = 3$

Difficulty of learning

How to measure under and overfitting?

Variance: how much the **predictions** of my model on unseen data fluctuate if trained over different but similar training sets (high \Rightarrow overfitting).

Bias: how off is the **average** of these predictions (high \Rightarrow underfitting).

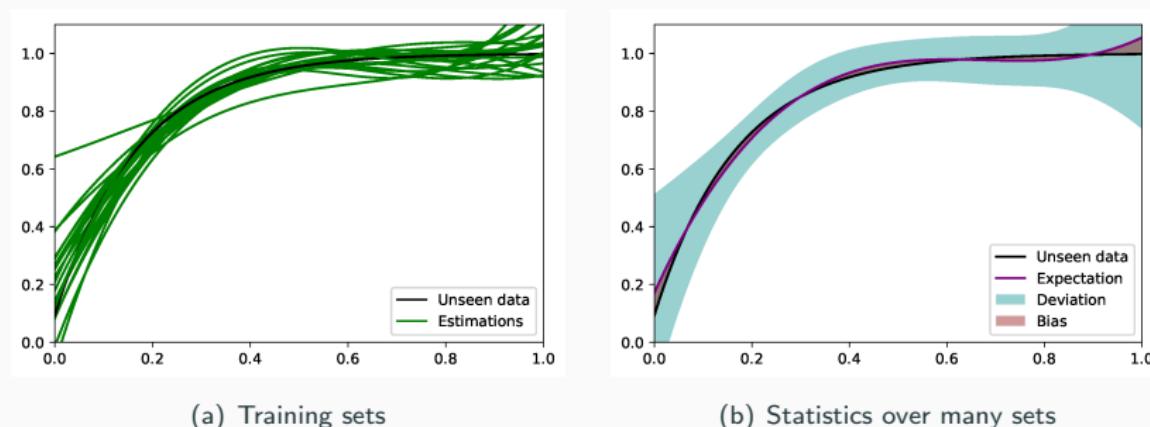


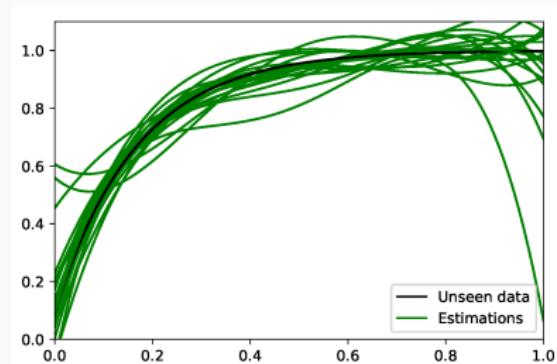
Figure 1 – Example with polynomial curve fitting of order $M = 4$

Difficulty of learning

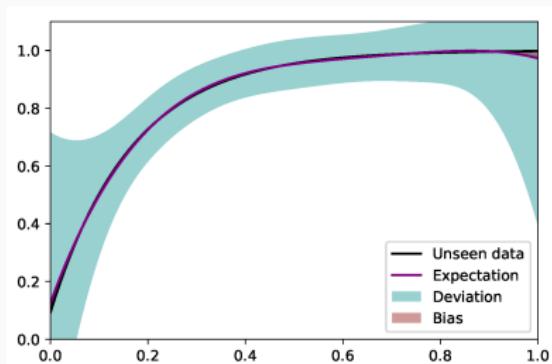
How to measure under and overfitting?

Variance: how much the **predictions** of my model on unseen data fluctuate if trained over different but similar training sets (high \Rightarrow overfitting).

Bias: how off is the **average** of these predictions (high \Rightarrow underfitting).



(a) Training sets



(b) Statistics over many sets

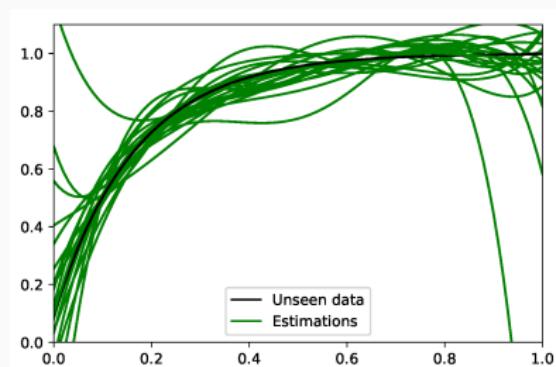
Figure 1 – Example with polynomial curve fitting of order $M = 5$

Difficulty of learning

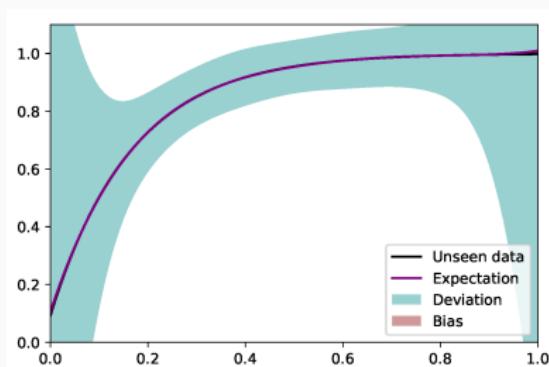
How to measure under and overfitting?

Variance: how much the **predictions** of my model on unseen data fluctuate if trained over different but similar training sets (high \Rightarrow overfitting).

Bias: how off is the **average** of these predictions (high \Rightarrow underfitting).



(a) Training sets



(b) Statistics over many sets

Figure 1 – Example with polynomial curve fitting of order $M = 6$

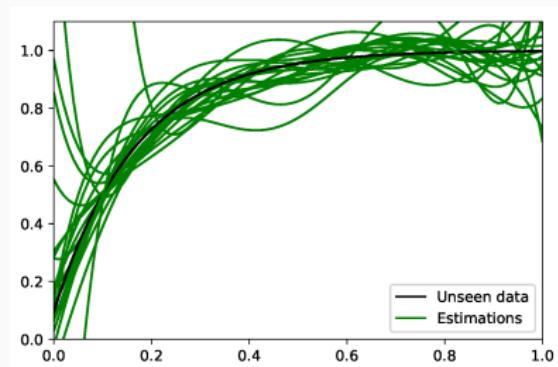
Machine learning – Overfitting and Generalization

Difficulty of learning

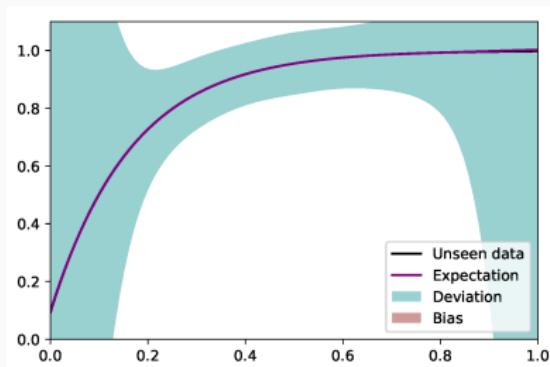
How to measure under and overfitting?

Variance: how much the **predictions** of my model on unseen data fluctuate if trained over different but similar training sets (high \Rightarrow overfitting).

Bias: how off is the **average** of these predictions (high \Rightarrow underfitting).



(a) Training sets

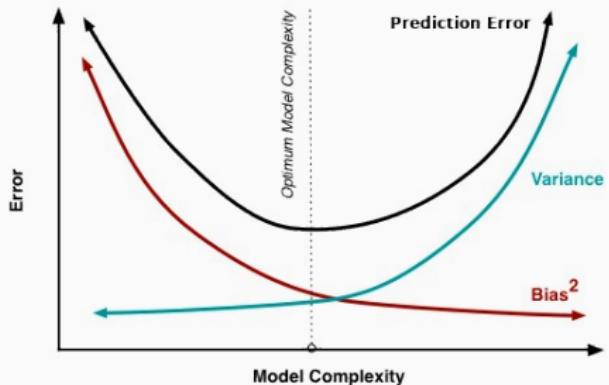


(b) Statistics over many sets

Figure 1 – Example with polynomial curve fitting of order $M = 7$

Difficulty of learning

Tradeoff Underfitting/Overfitting \equiv Tradeoff Bias/Variance



Bias decreases with model complexity.

Variance increases with model complexity.

$$\underbrace{\text{Prediction Error}}_{\text{MSE on unseen data}} = \text{Bias}^2 + \text{Variance}$$

The quality of the optimal model (at the best trade-off) depends on:

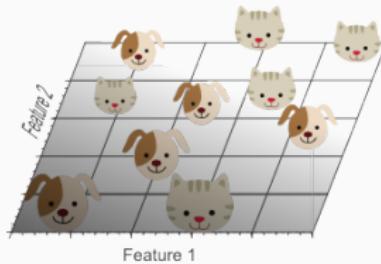
- Intrinsic complexity of the phenomenon to be predicted,
- Size of the training set: the larger the better,
- Size of the feature vectors: larger or smaller?

Curse of dimensionality

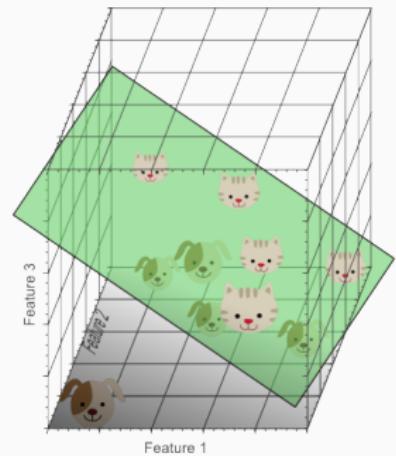
Is there a (hyper)plane that perfectly separates dogs from cats?



No perfect separation



No perfect separation



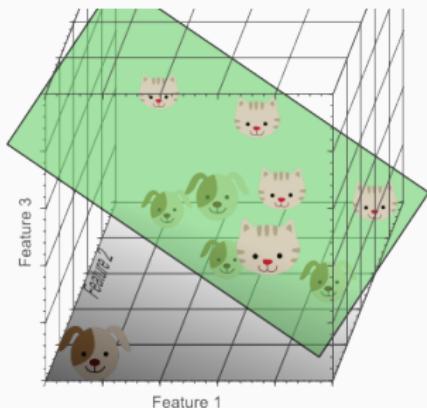
Linearly separable case

Looks like the more features we have, the better it is. But...

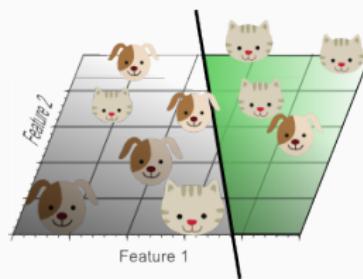
(Source: Vincent Spruyt)

Curse of dimensionality

Is there a (hyper)plane that perfectly separates dogs from cats?



Yes, but overfitting

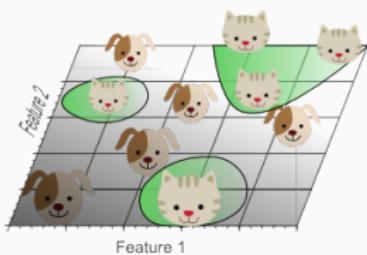


No, but better on unseen data

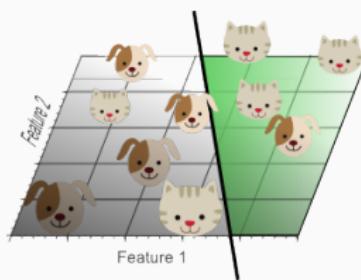
(Source: Vincent Spruyt)

Curse of dimensionality

Is there a (hyper)plane that perfectly separates dogs from cats?



Yes, but overfitting

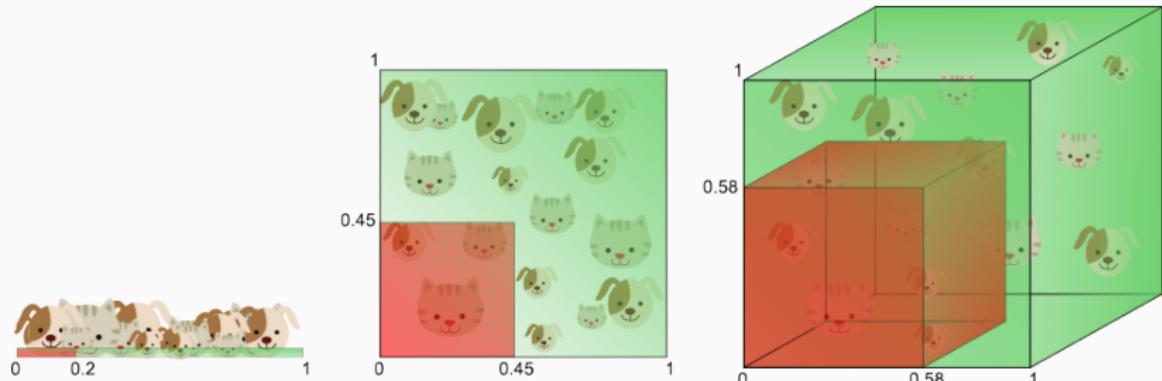


No, but better on unseen data

Why is that?

(Source: Vincent Spruyt)

Curse of dimensionality



The red area (corresponding to 20% of the feature range) become very sparse as the number of dimensions increases.

The amount of training data needed to cover this area must grow exponentially with the number of dimensions (+1 feature $\Rightarrow 10\times$ more data).

\Rightarrow Reducing the feature dimension is thus often favorable.

"Many algorithms that work fine in low dimensions become intractable when the input is high-dimensional." Bellman, 1961.

Feature engineering

- **Feature selection:** choice of distinct traits used to describe each sample in a quantitative manner.

Ex: fruit → acidity, bitterness, size, weight, number of seeds, ...

Correlations between features: weight vs size, seeds vs bitterness,

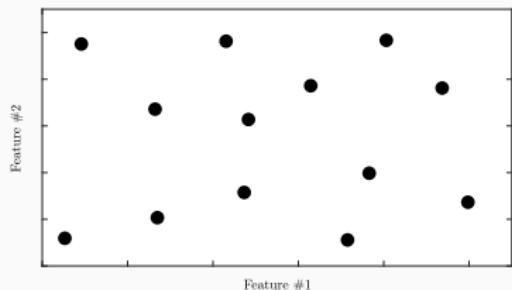
⇒ **Information is redundant and can be summarized with less but more relevant features.**

- **Feature extraction:** extract/generate new features from the initial set of features intended to be informative, non-redundant and facilitating the subsequent task.

⇒ **Common procedure: Principal Component Analysis (PCA)**

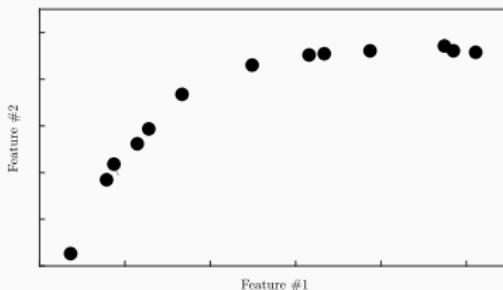
Principal Component Analysis (PCA)

In most applications examples are not spread uniformly throughout the example space, but are concentrated on or near a low-dimensional subspace/manifold.



No correlations

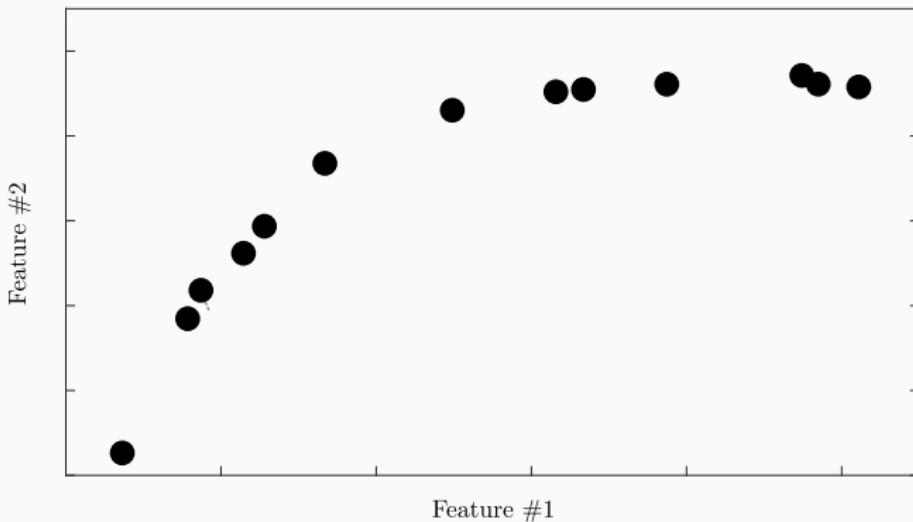
- ⇒ Both features are informative,
- ⇒ No dimensionality reductions.



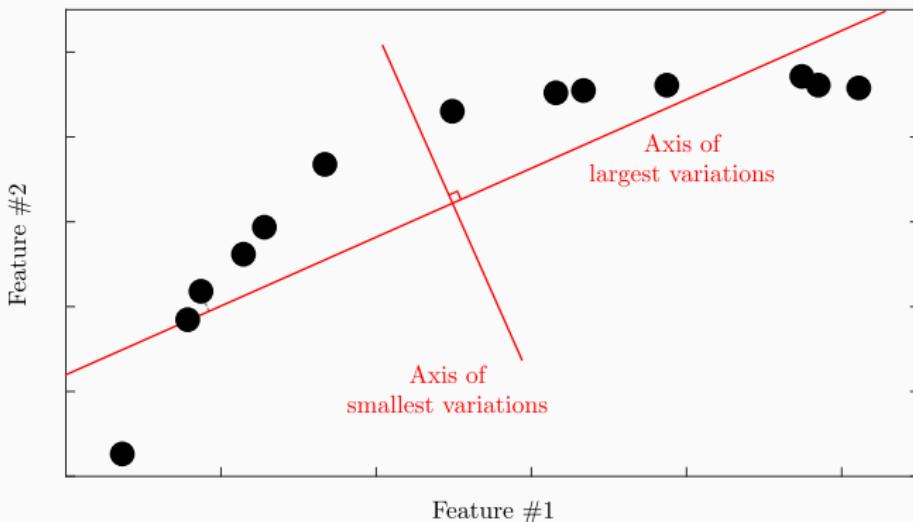
Strong correlation

- ⇒ Features “influence” each other,
- ⇒ Dimensionality reductions possible.

Principal Component Analysis (PCA)

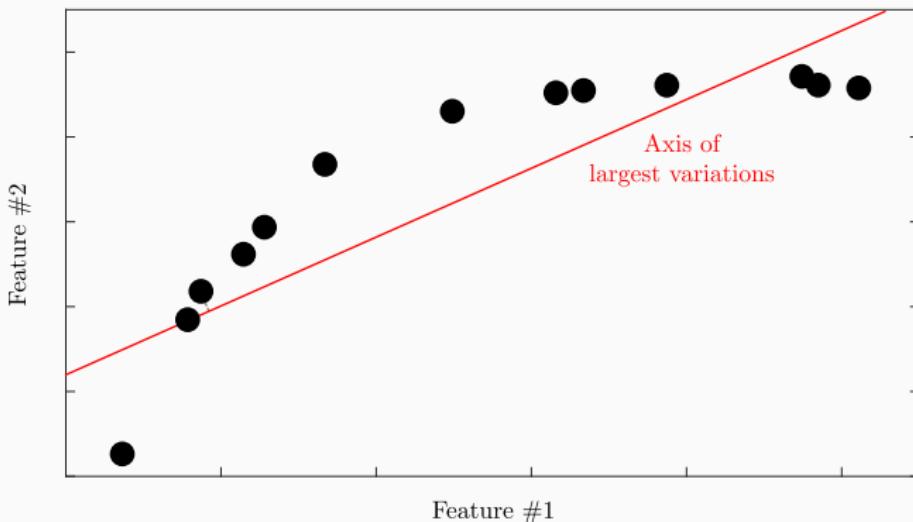


Principal Component Analysis (PCA)



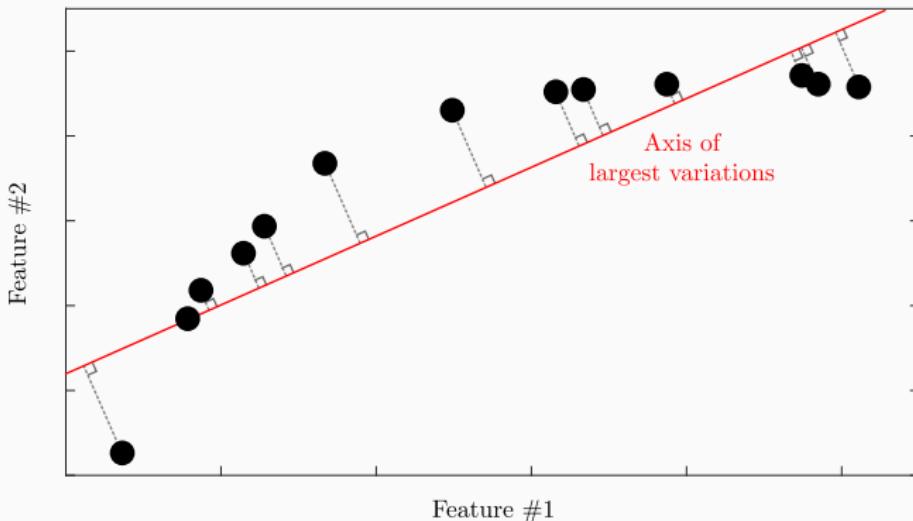
- Find the principal axes (eigenvectors of the covariance matrix),

Principal Component Analysis (PCA)



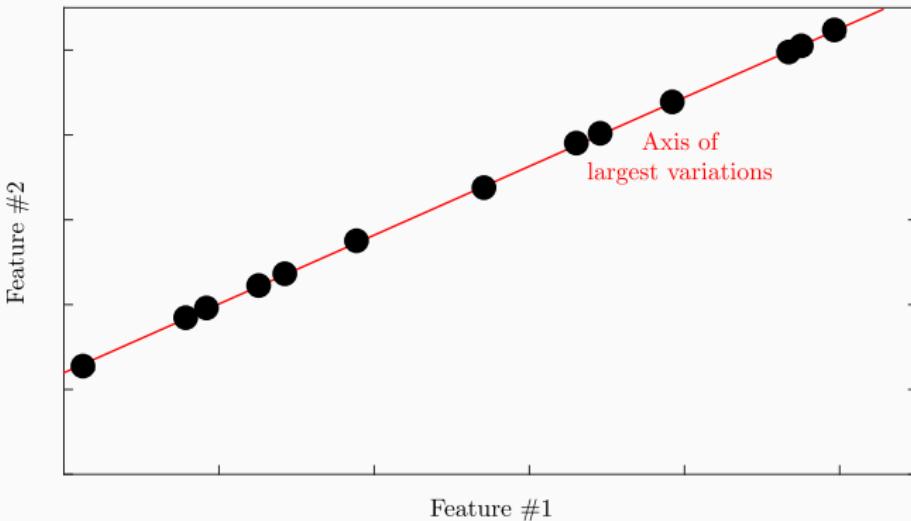
- Find the principal axes (eigenvectors of the covariance matrix),
- Keep the ones with largest variations (largest eigenvalues),

Principal Component Analysis (PCA)



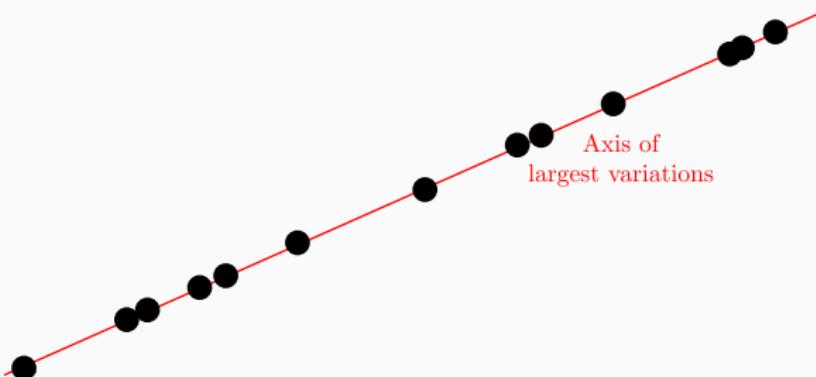
- Find the principal axes (eigenvectors of the covariance matrix),
- Keep the ones with largest variations (largest eigenvalues),
- Project the data on this low-dimensional space,

Principal Component Analysis (PCA)



- Find the principal axes (eigenvectors of the covariance matrix),
- Keep the ones with largest variations (largest eigenvalues),
- Project the data on this low-dimensional space,

Principal Component Analysis (PCA)



- Find the principal axes (eigenvectors of the covariance matrix),
- Keep the ones with largest variations (largest eigenvalues),
- Project the data on this low-dimensional space,
- Change system of coordinate to reduce data dimension.

Principal Component Analysis (PCA)



- Find the principal axes (eigenvectors of the covariance matrix),
- Keep the ones with largest variations (largest eigenvalues),
- Project the data on this low-dimensional space,
- Change system of coordinate to reduce data dimension.

Machine learning – Principal Component Analysis (PCA)

Principal Component Analysis (PCA)

- Find the principal axes of variations of $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathbb{R}^d$:

$$\boldsymbol{\mu} = \underbrace{\frac{1}{N} \sum_{i=1}^N \mathbf{x}_i}_{\text{mean (vector)}}, \quad \boldsymbol{\Sigma} = \underbrace{\frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T}_{\text{covariance (matrix)}}, \quad \underbrace{\boldsymbol{\Sigma} = \mathbf{V}^T \boldsymbol{\Lambda} \mathbf{V}}_{\substack{\text{eigen decomposition} \\ (\mathbf{V}\mathbf{V}^T = \mathbf{V}^T\mathbf{V} = \mathbf{I}\mathbf{d}_d)}}$$

$$\mathbf{V} = (\underbrace{\mathbf{v}_1, \dots, \mathbf{v}_d}_{\text{eigenvectors}}), \quad \boldsymbol{\Lambda} = \text{diag}(\underbrace{\lambda_1, \dots, \lambda_d}_{\text{eigenvalues}}) \quad \text{and} \quad \lambda_1 \geq \dots \geq \lambda_d$$

- Keep the $K < d$ first dimensions: $\mathbf{V}_K = (\mathbf{v}_1, \dots, \mathbf{v}_K) \in \mathbb{R}^{d \times K}$
- Project the data on this low-dimensional space:

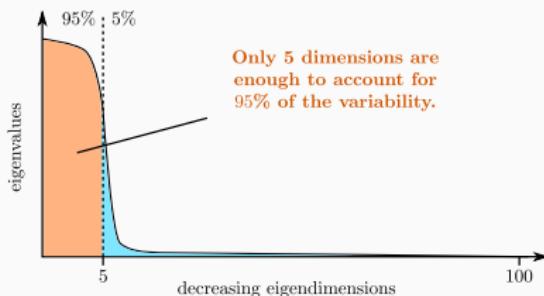
$$\tilde{\mathbf{x}}_i = \boldsymbol{\mu} + \sum_{k=1}^K \langle \mathbf{v}_k, \mathbf{x}_i - \boldsymbol{\mu} \rangle \mathbf{v}_k = \boldsymbol{\mu} + \mathbf{V}_K \mathbf{V}_K^T (\mathbf{x}_i - \boldsymbol{\mu}) \in \mathbb{R}^d$$

- Change system of coordinate to reduce data dimension:

$$\mathbf{h}_i = \mathbf{V}_K^T (\tilde{\mathbf{x}}_i - \boldsymbol{\mu}) = \mathbf{V}_K^T (\mathbf{x}_i - \boldsymbol{\mu}) \in \mathbb{R}^K$$

Principal Component Analysis (PCA)

- Typically: from hundreds to a few (one to ten) dimensions,
- Number K of dimensions often chosen to cover 95% of the variability:



$$K = \min \left\{ K \mid \frac{\sum_{k=1}^K \lambda_k}{\sum_{k=1}^d \lambda_k} > .95 \right\}$$

- **PCA is done on training data, not on testing data!:**
 - First, learn the low-dimensional subspace on training data only,
 - Then, project both the training and testing samples on this subspace,
 - It's an affine transform (translation, rotation, projection, rescaling):

$$\mathbf{h} = \mathbf{W}\mathbf{x} + \mathbf{b} \quad (\text{with } \mathbf{W} = \mathbf{V}_K^T \text{ and } \mathbf{b} = -\mathbf{V}_K^T \boldsymbol{\mu})$$

Deep learning does something similar but in an (extremely) non-linear way.

Machine learning – Feature extraction

What features for an image?



(Source: Michael Walker)

Image representation

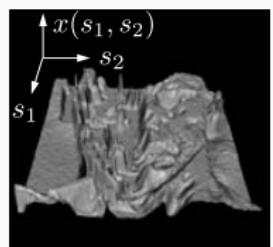
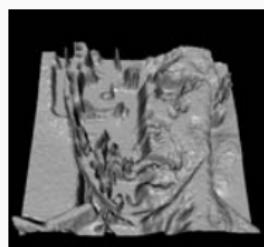
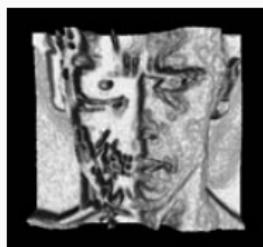
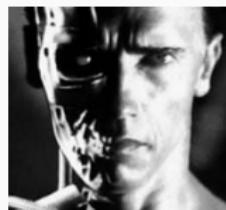


*La Trahison des images, René Magritte, 1928
(Los Angeles County Museum of Art)*

How do we represent images?

A two dimensional function

- Think of an image as a two dimensional function x .
- $x(s_1, s_2)$ gives the intensity at location (s_1, s_2) .



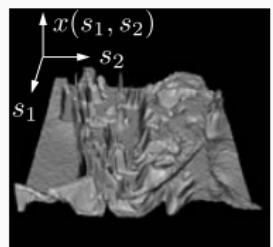
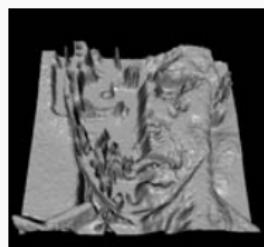
(Source: Steven Seitz)

Convention: larger values correspond to brighter content.

How do we represent images?

A two dimensional function

- Think of an image as a two dimensional function x .
- $x(s_1, s_2)$ gives the intensity at location (s_1, s_2) .



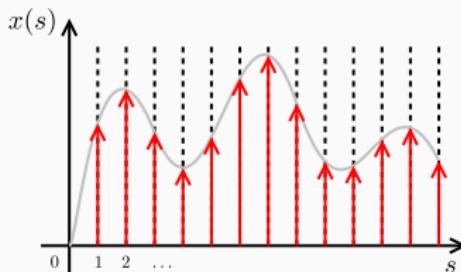
(Source: Steven Seitz)

Convention: larger values correspond to brighter content.

A color image is defined similarly as a 3 component vector-valued function:

$$x(s_1, s_2) = \begin{pmatrix} r(s_1, s_2) \\ g(s_1, s_2) \\ b(s_1, s_2) \end{pmatrix} .$$

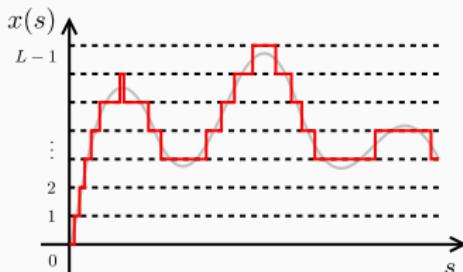
Digital imagery



Raster images

- Sampling: reduce the 2d continuous space to a discrete grid $\Omega \subseteq \mathbb{Z}^2$
- Gray level image: $\Omega \rightarrow \mathbb{R}$ (discrete position to gray level)
- Color image: $\Omega \rightarrow \mathbb{R}^3$ (discrete position to RGB)

Image representation – Types of images – Digital imagery

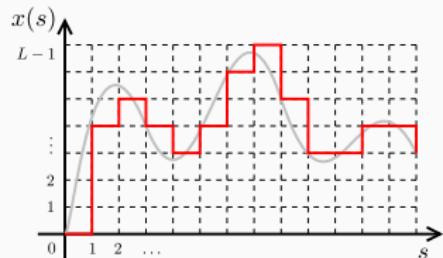


Bitmap image

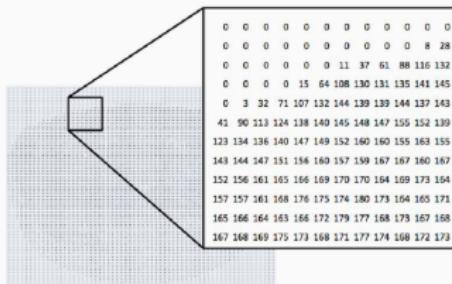
- Quantization: map each value to a discrete set $[0, L - 1]$ of L values
(e.g., round to nearest integer)
- Often $L = 2^8 = 256$ (8bit images \equiv unsigned char)
 - Gray level image: $\Omega \rightarrow [0, 255]$ $(255 = 2^8 - 1)$
 - Color image: $\Omega \rightarrow [0, 255]^3$
- Optional: assign instead an index to each pixel pointing to a color palette
(format: .png, .bmp)

Digital imagery

- Digital images: sampling + quantization:



→ 8bit images can be seen as a matrix of integer values



Each element of this matrix is referred to as a pixel ("picture element").

Image representation – Types of images – Digital imagery



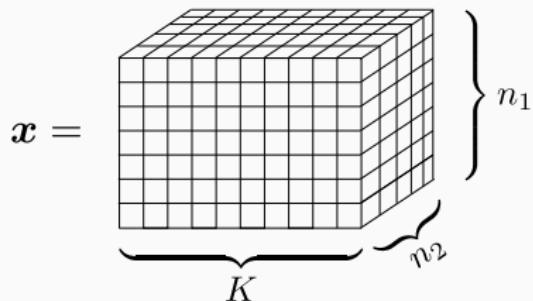
139	162	119	98	127	202
66	88	44	27	85	160
95	121	83	71	106	184
133	124	110	105	159	218
54	42	32	38	107	185
86	80	74	82	143	204
127	107	116	145	200	226
47	28	40	89	160	198
86	69	86	128	187	210
112	122	137	186	220	229
59	53	79	145	189	199
92	98	120	175	207	207
128	162	186	208	220	222
69	107	144	179	194	190
107	149	180	201	207	195
169	192	206	220	219	224
117	148	170	189	187	187
156	171	182	195	192	194

For color images each pixel has 3 values

- Color channels: Red, Green, Blue (RGB)
- RGB: Usual colorspace for acquisition and display
- There exist other colorspaces for different purposes:
HSV (Hue, Saturation, Value), YUV, YCbCr...

Image representation – Types of images – Digital imagery

- A $n_1 \times n_2$ image with K channels is a **multidimensional array**:



- In the deep learning community: they are referred to as **tensors**
(not to be confused with tensor fields or tensor imagery).

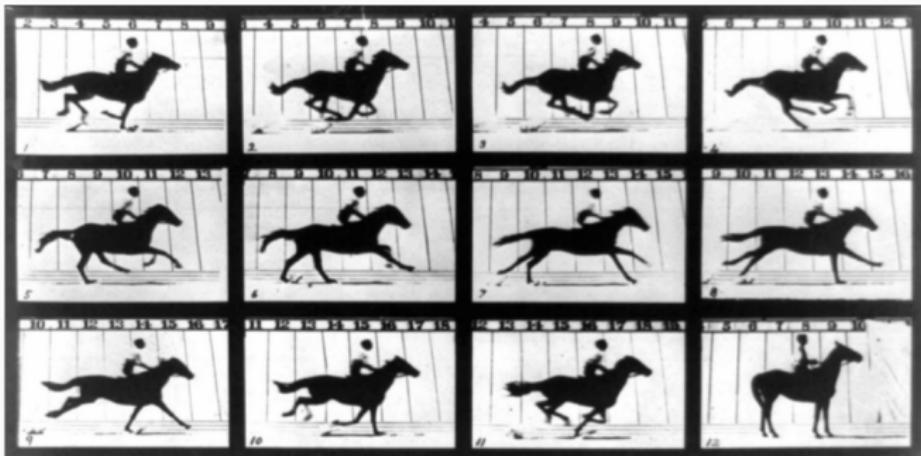
Image representation – Types of images – Digital imagery



Spectral image:

- Each of the K channels is a wavelength band
- For $K \approx 10$: multi-spectral imagery
- For $K \approx 200$: hyper-spectral imagery
- Used in astronomy, surveillance, mineralogy, agriculture, chemistry

Image representation – Types of images – Digital imagery

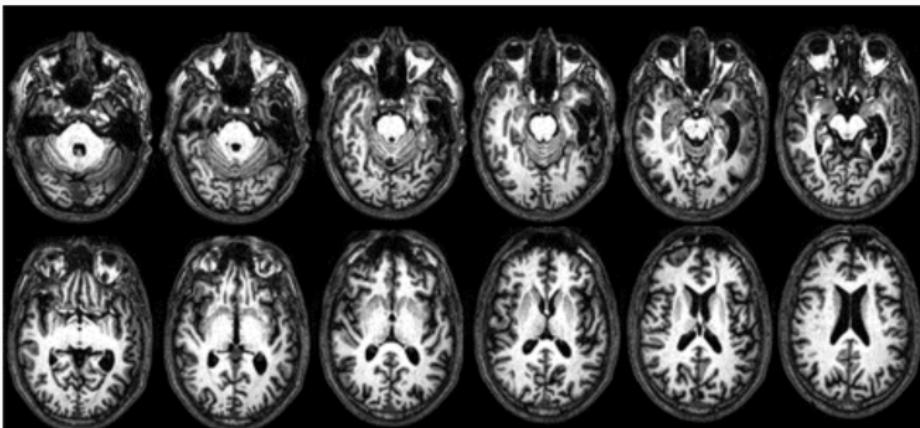


The Horse in Motion (1878, Eadweard Muybridge)

Videos:

- 2 dimensions for space
- 1 dimension for time

Image representation – Types of images – Digital imagery



MRI slices at different depths

3d brain scan:

- 3 dimensions for space
- 3d pixels are called voxels ("volume elements")

Image representation – Semantic gap

Semantic gap in CV tasks



08	02	22	97	35	15	00	40	00	75	04	05	07	78	52	12	50	31
49	49	99	40	17	81	18	57	60	87	17	40	98	43	69	44	05	56
81	49	31	73	55	79	14	29	93	71	40	62	54	15	30	03	49	13
52	70	95	23	04	60	11	62	63	05	54	01	32	54	71	37	02	36
22	31	16	71	51	63	03	89	41	92	36	54	22	40	40	28	66	33
24	47	11	60	39	05	45	02	44	75	33	53	78	36	04	20	35	17
32	98	81	28	64	23	67	10	26	38	40	67	59	54	70	66	18	38
70	47	26	20	48	02	62	12	20	95	63	94	39	63	08	40	91	66
21	36	55	58	05	66	72	99	26	97	17	78	78	96	83	14	88	34
72	21	36	23	09	75	09	76	44	20	45	35	14	00	41	33	97	34
95	71	17	53	28	22	73	31	67	15	94	03	80	04	62	14	14	09
59	16	39	05	42	96	33	31	47	55	58	88	24	00	17	54	24	36
57	86	56	00	48	35	73	89	07	05	44	44	37	44	60	21	58	51
58	19	80	81	68	05	94	47	69	28	73	92	13	86	52	17	77	04
66	49	52	08	83	97	35	99	16	07	97	57	32	16	26	79	33	27
69	66	16	68	87	57	62	20	72	03	46	33	67	44	55	12	32	63
69	93	04	42	16	73	29	84	38	11	24	94	72	18	08	46	29	32
36	14	20	69	36	41	72	30	23	88	39	89	69	82	67	59	85	74
16	14	20	73	35	29	78	31	90	01	74	31	49	73	51	02	16	23
54	54	70	54	71	83	51	54	69	16	92	33	48	65	43	52	01	39
48	48	48	48	48	48	48	48	48	48	48	48	48	48	48	48	48	48

What the computer sees

→ 82% cat
15% dog
2% hat
1% mug

Gap between tensor representation and its semantic content.

Old school computer vision

Semantic gap: initial representation of the data is too low-level,

Curse of dimensionality: reducing dimension is necessary for limited datasets,

Instead of considering images as a collection of pixel values (tensor),
we may consider other features/descriptors:

Designed from prior knowledge

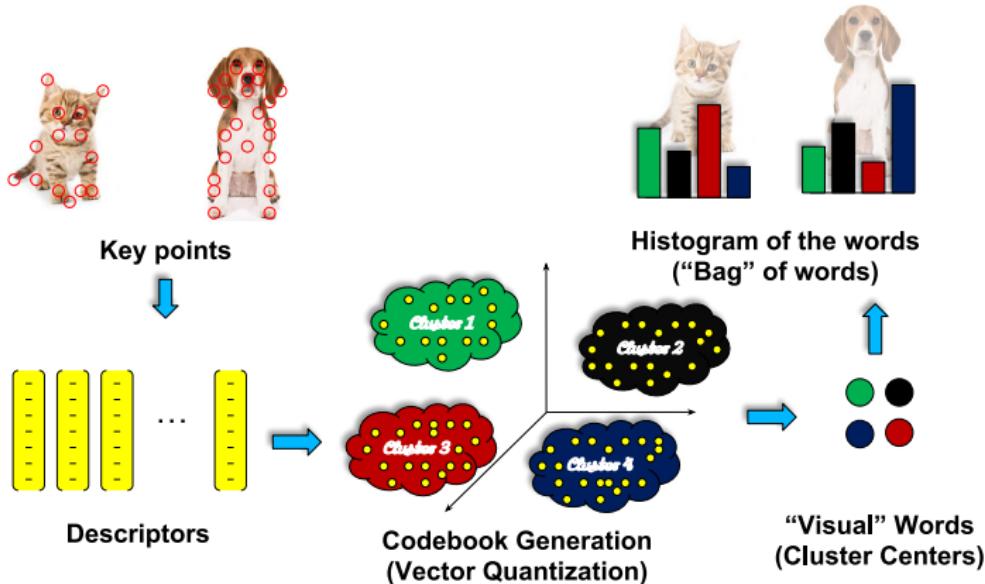
- Image edges,
- Color histogram,
- Local frequencies,
- High-level descriptor (SIFT).

Or learned by unsupervised learning

- Dimensionality reduction (PCA),
- Parameters of density distributions,
- Clustering of image regions,
- Membership to classes (GMM-EM).

Goal: Extract informative features, remove redundancy, reduce dimensionality, facilitating the subsequent learning task.

Example of a classical CV pipeline



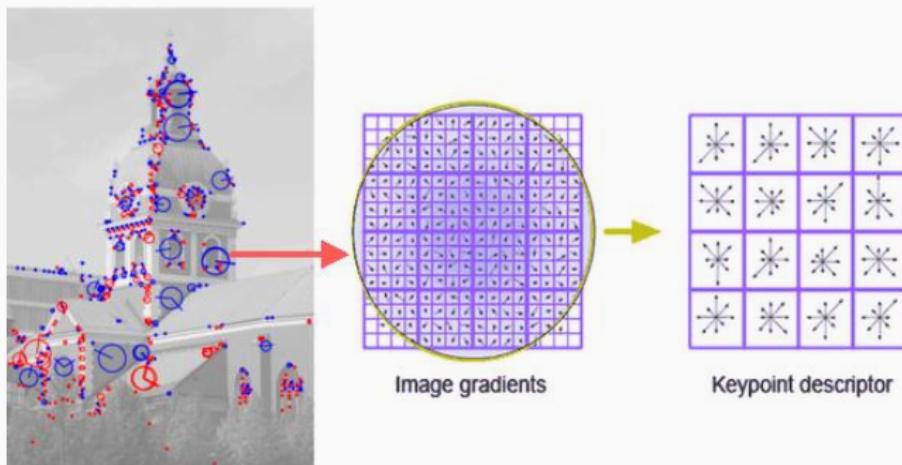
- ① Identify “interesting” key points,
- ② Extract “descriptors” from the interesting points,
- ③ Collect the descriptors to “describe” an image.

Key point detector



- Goal: to detect interesting points (without describing them).
- Method: to measure intensity changes in local sliding windows.
- Constraint: to be invariant to illumination, rotation, scale, viewpoint.
- Famous ones: Harris, Canny, DoG, LoG, DoH, ...

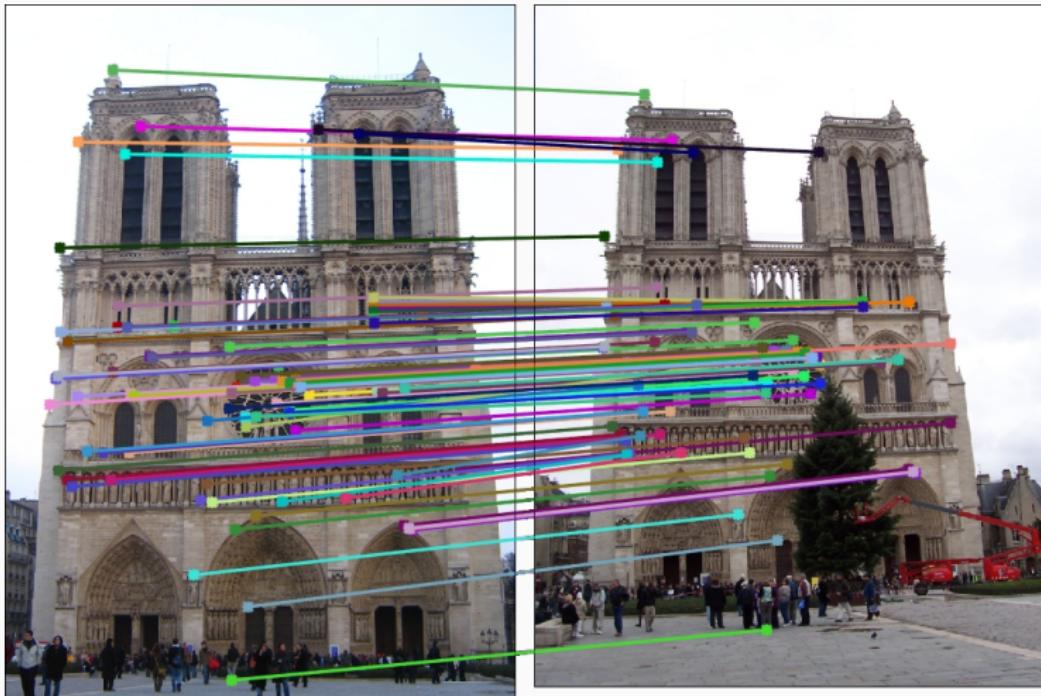
Scale-invariant feature transform (SIFT) (Lowe, 1999)



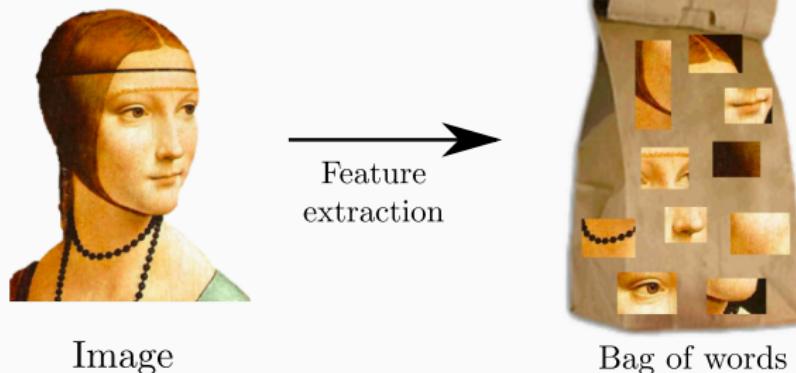
(Source: Ravimal Bandara)

- Goal: to provide a quantitative description at a given image location.
- Based on multi-scale analysis and histograms of local gradients.
- Robust to changes of scales, rotations, viewpoints, illuminations.
- Fast, efficient, very popular in the 2000s.
- Other famous descriptors: HoG, SURF, LBP, ORB, BRIEF, ...

SIFT – Example: Object matching



Bags of words



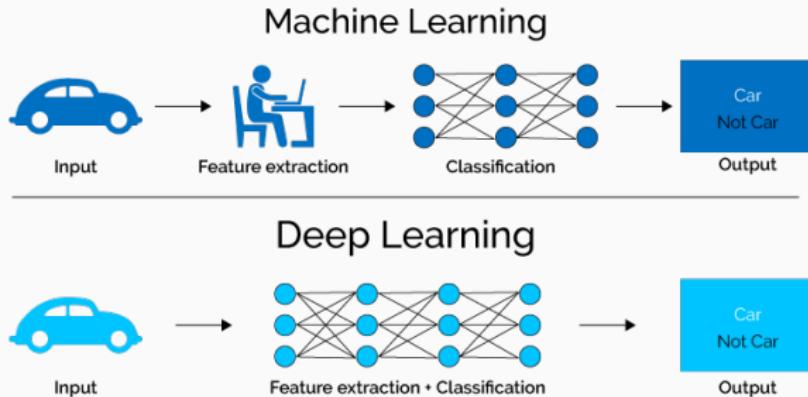
(Source: Rob Fergus & Svetlana Lazebnik)

Bag of words: vector of occurrence count of visual descriptors
(often obtained after vector quantization).

Before deep learning: most computer vision tasks were relying on feature engineering and bags of words.

Image representation – Deep learning

Modern computer vision – Deep learning



Deep learning is about **learning the feature extraction**, instead of designing it yourself.

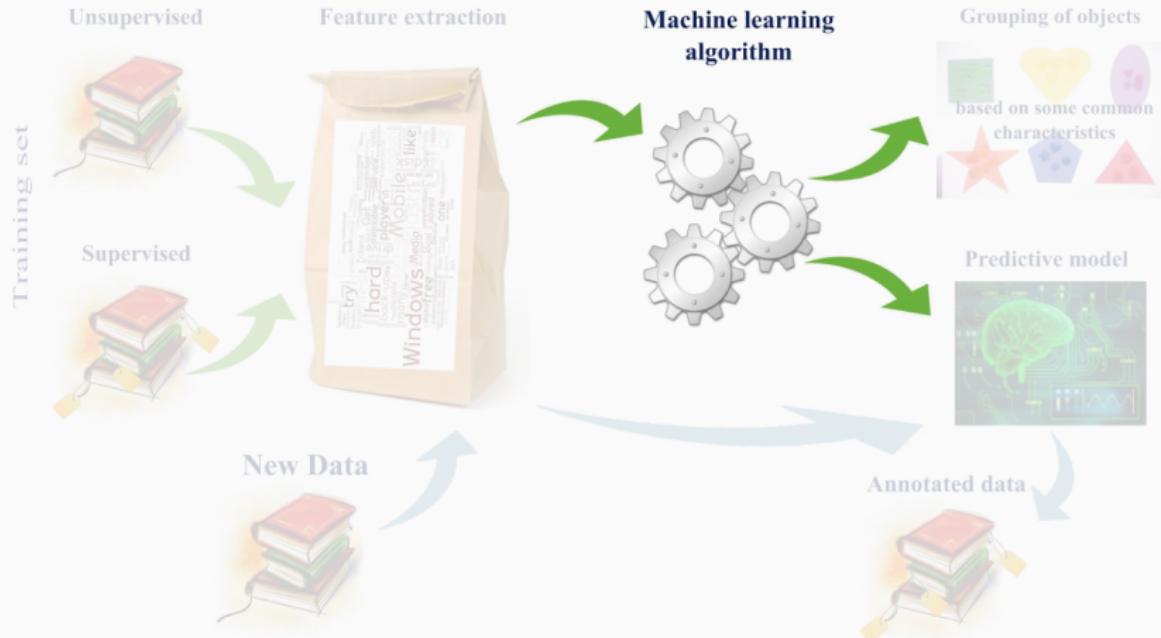
Deep learning requires a lot of data and hacks to fight the curse of dimensionality (i.e., reduce complexity and overfitting).

Quick overview of ML algorithms



Quick overview of ML algorithms

What about algorithms?



(Source: Michael Walker)

Quick overview of ML algorithms

In fact, most of statistical tools are machine learning algorithms.

Dimensionality reduction / Manifold learning

- Principal Component Analysis (PCA) / Factor analysis
- Dictionary learning / Matrix factorization
- Kernel-PCA / Self organizing map / Auto-encoders

Linear regression / Variable selection

- Least square regression / Ridge regression / Least absolute deviations
- LASSO / Sparse regression / Matching pursuit / Compressive sensing

Classification and non-linear regression

- K-nearest neighbors
- Naive Bayes / Decision tree / Random forest
- Artificial neural networks / Support vector machines

Quiz: Supervised or unsupervised?

Quick overview of ML algorithms

Clustering

- K-Means / Mixture models
- Hidden Markov Model
- Non-negative matrix factorization

Density estimation

- Maximum likelihood / a posteriori
- Parzen windows / Mean shift
- Expectation-Maximization

Recommendation

- Association rules
- Low-rank approximation
- Metric learning

Simulation / Sampling / Generation

- Variational auto-encoders
- Deep Belief Network
- Generative adversarial network

Often based on tools from **optimization, sampling or operations research**:

- Gradient descent / Quasi-Newton / Proximal methods / Duality
- Simulated annealing / Genetic algorithms
- Gibbs sampling / Metropolis-hastings / MCMC

Questions?

Next class: Preliminaries to deep learning

Sources, images courtesy and acknowledgment

L. Condat

R. Fergus

P. Gallinari

N. Harris

A. Horodniceanu

J. Johnson

A. Karpathy

S. Lazebnik

F.-F. Li

A. Newson

S. Parameswaran

D. C. Pearson

S. Seitz

V. Spruyt

V. Tong Ta

R. Wendell

Wikipedia