Deep Learning

Formalism, Data Visualization, from ML to DeepLearning

Alexandre Fournier Montgieux

M2 BDMA, CentraleSupélec, Université Paris Saclay

September 18, 2024

Plan

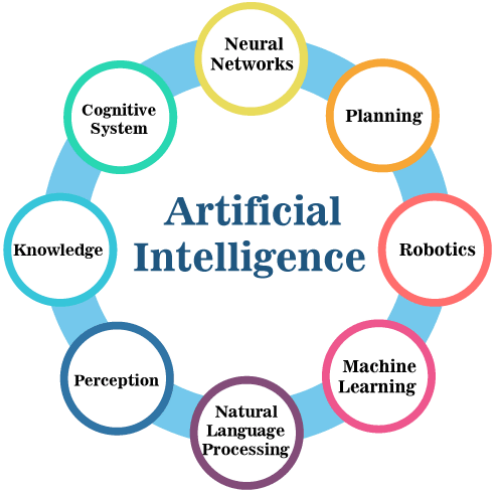
About Us

- Alexandre Fournier Montgieux
  - PhD student at CEA LIST LASTI laboratory and STIC DS (Univeristy Paris Saclay)
  - Generative model for bias mitigation applied to FR task
  - email: alexandre.fournier-montgieux@centralesupelec.fr
- Hugo Boulanger
- Akash Malhotra
- Guillaume Thomas
- Youssef Attia El Hili

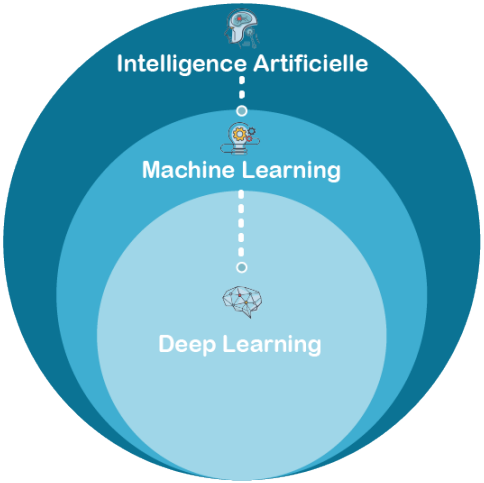
What is CEA LIST ?

- CEA: Alternative Energies and Atomic Energy Commission is a french public government funded research organisation in the areas of energy, defense and security, information technologies and health technologies
- Created in 1945 by Charles de Gaules after WW2
- LIST: Laboratory for Integration of Systems and Technology
  - Four main topics: Advanced manufacturing, embedded systems, **data intelligence**, health ionizing radiations
  - **Recruits interns for reasearch internships !**
  - Applications in October - November
  - `https://kalisteo.cea.fr/index.php/apply-for-a-job/`

What is Artificial Intelligence (AI) ?

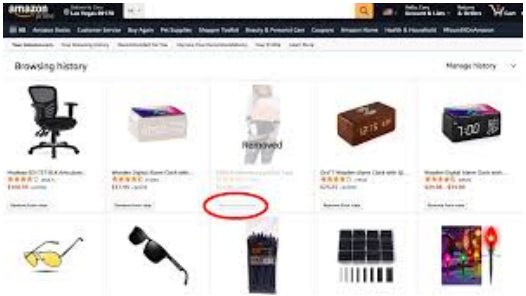


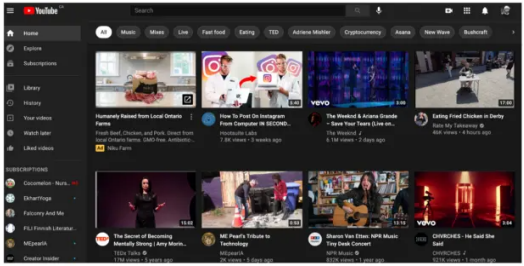
Deep Learning: One subcategory of AI



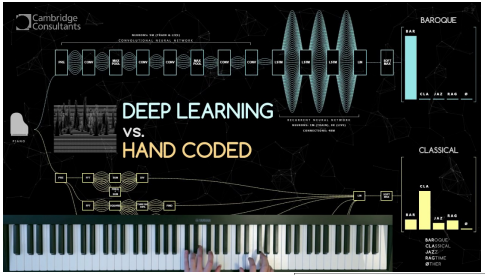
Deep Learning in our life: virtual assistants

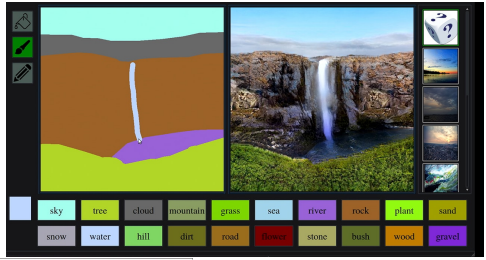
Deep Learning in our life: Recommendation systems



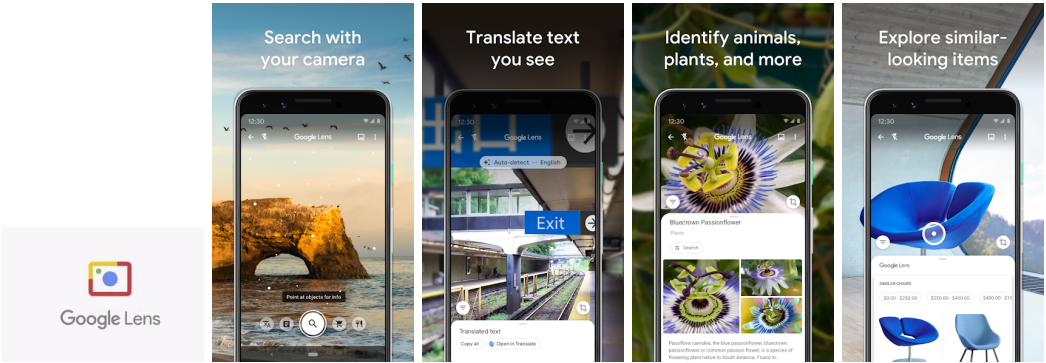


Deep Learning in our life: Making art





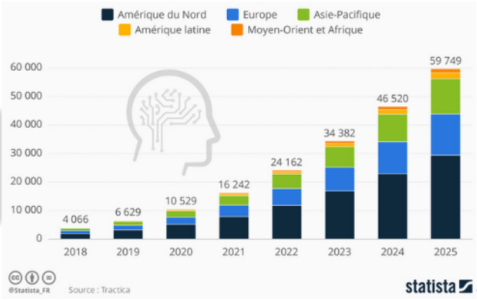
Deep Learning in our life: Identify everything in images



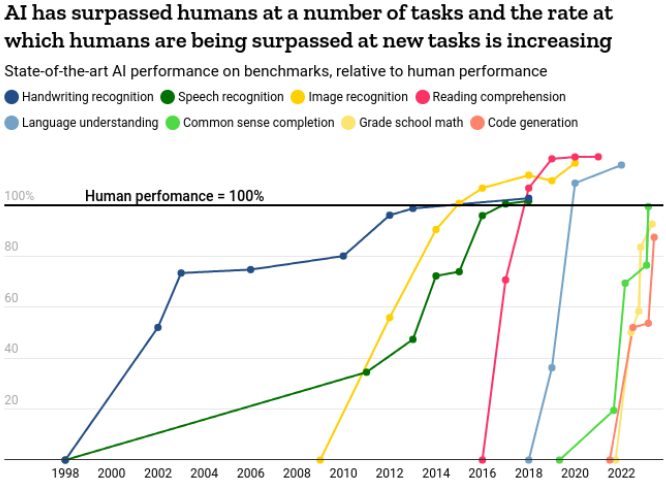
Deep Learning in our life: Automated machines

Deep Learning is attractive

- Attractive for business, governments, and students in the whole world



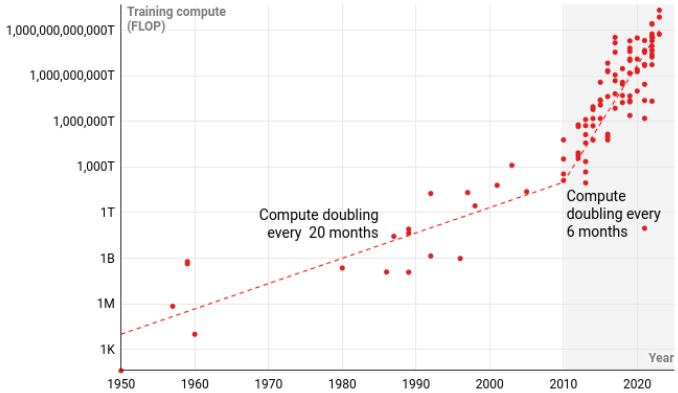
Deep Learning is accelerating



Deep Learning is accelerating

The amount of compute used to train AI systems has been increasing since 1950, the rate of increase increased in 2010

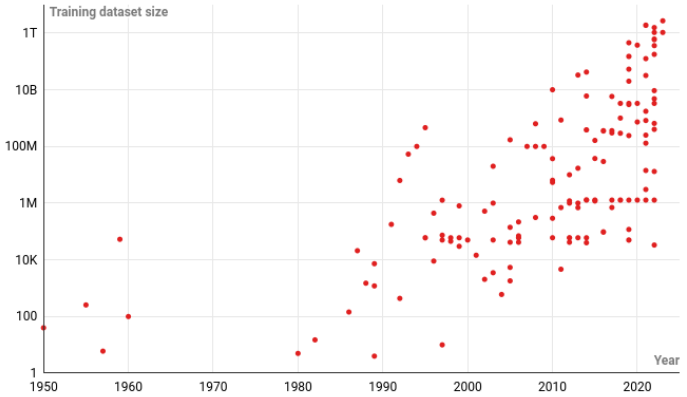
Amount of compute used to train notable AI models



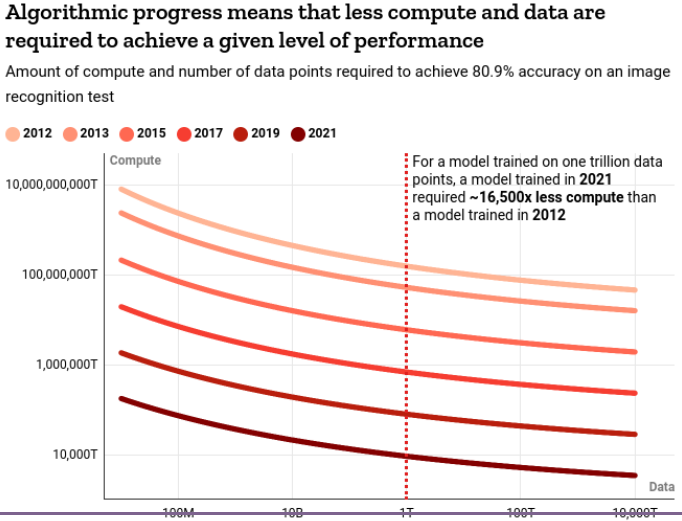
Deep Learning is accelerating

The number of data points used to train AI models has increased dramatically over the last seventy years

Number of data points used to train notable AI models



Deep Learning is accelerating



Deep Learning issues: juridical and responsibility

- If something wrong happens who is responsible ?
- If AI becomes sentient should it be considered as a person ?

Canada investigates after Tesla catches fire, forcing driver to 'smash the window'

Video shows incident in which driver says he had to kick his way out because the doors and windows wouldn't open

Model Y cars at Tesla's factory in Gruenheide, Germany. Photograph: Reuters

'Risks posed by AI are real': EU moves to beat the algorithms that ruin lives

The EU's Artificial Intelligence Act will have consequences beyond its borders as does the GDPR. Photograph: metamorworks/Getty Images/iStockphoto

'I am, in fact, a person': can artificial intelligence ever be sentient?

Friendly robots: Google's chatbot LaMDA told Blake Lemoine that it was a person. Photograph: Getty Images

Controversy over Google's AI program is raising questions about just how powerful it is. Is it even safe?

Deep Learning issues: privacy

- AI applications allow for mass surveillance
- How are datasets collected ?
- Is data recoverable from the trained models ?

A Dad Took Photos of His Naked Toddler for the Doctor. Google Flagged Him as a Criminal.

Google has an automated tool to detect abusive images of children. But the system can get it wrong, and the consequences are serious.

China, Russia, India: facial recognition surveillance projects far from the capitals

Jul 18, 2022, 10:10 am EDT | Frank Hersev

CATEGORIES Biometrics News | Facial Recognition | Surveillance

Unmanned police surveillance vehicles patrol a city in Xinjiang, a Russian city in the Arctic Circle deploys a powerful facial recognition network and the southern Indian state of Telangana continues to saturate itself with CCTV.

Deep Learning issues: data generation

- AI can create mistrust, violence, paranoïa

Are these guys for real? How to keep your business safe from deepfakes

Scammers are using manipulated video and audio to dupe employees into handing over money. But protection is possible

A journalist views a video manipulated with artificial intelligence to potentially deceive viewers, or 'deepfake'. Photograph: Alexandra Robinson/AFP/Getty Images

Countering disinformation: Three tools for detecting bots on Twitter

September 02 12:11 2022

by Wasilat Azeez
0 Comments

Print This Article

Facebook Twitter Telegram Copy Link WhatsApp More 4

If you follow Elon Musk, the world's richest person, you should be aware that he is trying to terminate his $44 billion deal to buy the microblogging platform.

Why study Deep Learning ?

- Deep Learning is not only bad stuff and can helps on various "good" applications: health, agriculture, defense, ...
- Deep Learning is a key component to deal with data
  - higher performance than traditional Machine Learning for huge amount of data
- AI is a formidable tool to invent new services and make decisions
- It receives backslash from grand public (legitimate issues)
  - job suppression, individual liberties, environmental, bias, ...
  - several area of research to work on this issues
  - need for more education to better understand its limits

For who is this course ?

- People desiring to learn about Deep Learning and various applications (for text, image, generation, robotics, ...)
- People knowing how to code (basic knowledge required for practical sessions)
- People desiring to work in the industry or in research in Deep Learning applications
- This course is not a comprehensive course about Deep Learning, most interested students should dive deeper in the various domains introduced

Ressources

- Two main ressources have been used for this course
- *Deep Learning Book*, Goodfellow et al. 2016 (Free)
- *Dive into Deep Learning*, `https://d2l.ai/index.html` (Free)
- Other greats resources: *Deep Learning Specialization* on Coursera, ...

General information

The course has 2 main parts: **deep learning** and *reinforcement learning*.

- 3h of class every week
- Either full course or half course + practical session
- Alternating **deep learning** and *reinforcement learning* every week
- Equal evaluation on both subjects through paper presentations, projects and MCQ evaluations

Plan for the course

- 20/09: **Course 3h: Introduction to Deep Learning, ML reminders and Deep Learning fundamentals**
- 27/09: *Course 3h: Introduction to Reinforcement Learning - Markov Decision Processes*
- 4/10: **Course 1h30: (Multi Layer) Perceptron, optimization, regularization**
  TP 1h30: Manual implementation of Neural Networks and basics of Pytorch
- 11/10: *Course 1h30: Model-Free Prediction*
  TD 1h30: Model-Free Prediction
- 18/10: **Course 1h30: Convolutional Neural Networks and Image Processing**
  TP 1h30: Convolutional Neural Networks and Finetuning
- 25/10: *Course 1h30: Model-Free Control*
  TD 1h30: Model-Free Control
- 8/11: **Course 1h30: Sequential Models: RNN and Transformers**
- **TP 1h30: Action classification for videos**

Plan for the course

- 15/11: **Course 1h30: Generative Models part 1**
  TP 1h30: Generative Models part 1
- 22/11: *Course 1h30: Deep Reinforcement Learning*
  TP 1h30: Deep Reinforcement Learning
- 29/11:**Course 1h30: Generative Models part 2**
  TP 1h30: Generative Models part 2
- 6/12: Paper presentations by students 3h
- 13/12: *Policy Gradient Methods, Continuous Control and Robotics*
  TP 1h30: Policy Gradient Methods, Continuous Control and Robotics
- 20/12: Project Session 3h
- 10/01: **Deep Learning MCQ + Project presentations 3h**
- 17/01: Course 3h: Recent advances in Deep Learning and Reinforcement Learning
- 24/01:*Reinforcement Learning MCQ + Project presentations 3h*

Evaluation

- Deep Learning Written Exam 20%
- Reinforcement Learning Written Exam 20%
  - (Multiple Choice Questions and open questions)
- Paper presentation 10%
- Project 50%
  - Kaggle project (or else) working in collaboration in pairs to solve a task
  - Choice of a project among a pool of propositions (object detection, text analysis, ...)
  - Submit a report for last session of 2-4 pages
  - 15 minutes presentation at last session (same as writtenexam)
  - Grades take into account: technical difficulty, understanding of the problem, understanding of the solution, quality of the report, quality of the presentation

Plan

1950: Turing Test

Turing test, called the **Imitation game**, Turing 1950, aims to answer the question "Can machines think?" through a test

- 3 players: A a man, B a female and C an interrogator
- C asks question to determine which one is the man or female
- A tries to trick C and B assist C
- What happens if A is replaced by a computer ?

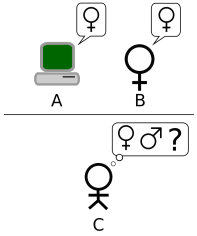


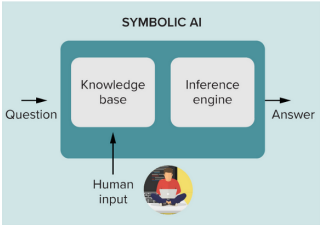
Figure: Imitation Game

1956: Birth of AI

Artificial Intelligence term invented at the Dartmouth Workshop in 1956 organized by IBM to describe

> *every aspect of learning or any other feature of intelligence can be so precisely described that a machine can be made to simulate it*

1956-1974: Symbolic AI



- Several area of research
  - **Reasoning as search**: a different sets of action leads to a certain goal
  - **Natural Language** tools have been developed following grammar and language rules
  - **Micro world**: few blocks to identify and move
- Birth of connectionism: Percreptron in 1958
- large funding was made to support AI research
- A lack of perspective by scientists
  - 1965, H. A. Simon: "machines will be capable, within twenty years, of doing any work a man can do."

1974-1980: First Winter of AI

- Several problems found
  - few computational resources
  - unscalability of models
  - **Moravec's paradox**: "*it is comparatively easy to make computers exhibit adult level performance on intelligence tests or playing checkers, and difficult or impossible to give them the skills of a one-year-old when it comes to perception and mobility*"
  - Devastating critics of connectionism by Marvin Minsky
- Stop of funding because of the lack of progress

1980-1987: Boom of Expert Knowledge Systems

- An **expert system** solves specifc tasks following an ensemble of rules based on expert knowledge
- In 1980, XCON sorting system was developed for the Digitial Equipment Corporation that helps saving 40M$ per year
- Connectionism is also back thanks to Backpropagation applied to Neurons by Geoffrey Hinton
- Fundings were back

1987-1994: Second Winter of AI

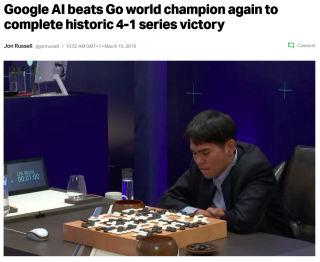
- Several companies were disappointed and AI was seen as a technology that couldn't solve wide varieties of tasks
- Funding were withdrawn
- A lot of AI companies went bankrupt and AI economy was shutdown

1994-2011: AI returns in the industry

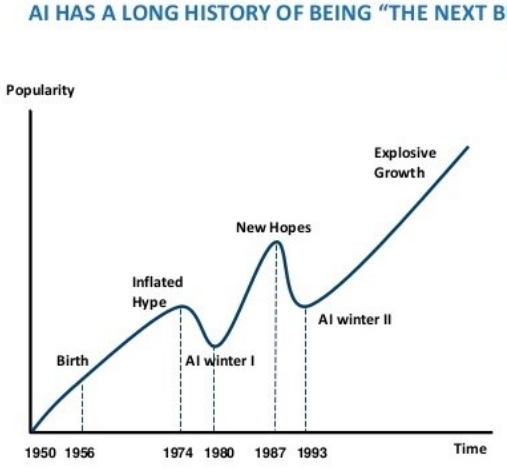
- 1997: Deep Blue defeats Garry Kasparov
- **Moore's Law** states that speed and memory of computers will double every two years
- Definition of **intelligent agents**: a system that perceives its environment and takes actions which maximize its chances of success. Human intelligence is not anymore the only form of intelligence studied or mimicked
- Use of **probabilistic reasoning** tools such as Bayesian networks, hidden Markov models, information theory, SVM, ...
- To attract fundings AI researches renamed their work in mathematics, computer science, physics, ...

2011-today: Deep Learning

- Deep Learning are deep graph of processing layers mimicking human neurons interactions.
- Possible thanks to the advances of hardware technologies
- Show spectacular results on various tasks such as Computer Vision, Natural Language Processing, Anomaly Detection



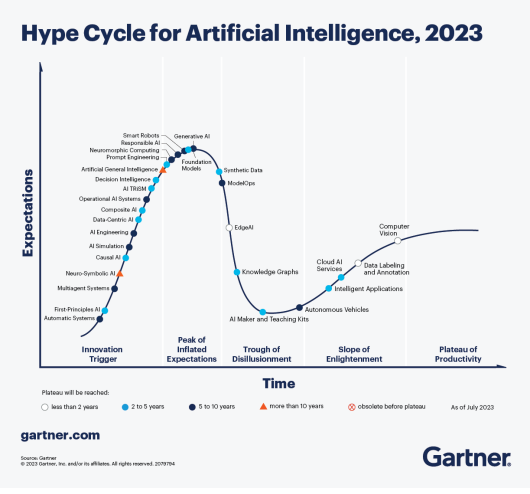
AI: a succession of hype cycles



Timeline of AI Development

- **1950s-1960s**: First AI boom - the age of reasoning, prototype AI developed
- **1970s**: AI winter I
- **1980s-1990s**: Second AI boom: the age of Knowledge representation (appearance of expert systems capable of reproducing human decision-making)
- **1990s**: AI winter II
- **1997**: Deep Blue beats Gary Kasparov
- **2006**: University of Toronto develops Deep Learning
- **2011**: IBM's Watson won Jeopardy
- **2016**: Go software based on Deep Learning beats world's champions

AI: a succession of hype cycles



Plan

Notations

- **Scalars**: a single number, example n the number of hidden neurons.
- **Vectors**: an array of number, example of a $\mathbb{R}^n$ vector elements $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$
- **Matrices**: a 2 dimensional array of numbers, example of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n} = \begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{m1} & \cdots & A_{mn} \end{bmatrix}$
- **Tensors**: a n dimensional array of numbers, example $\mathbf{A} \in \mathbb{R}^{m \times k \times p}$ a 3 dimensional tensor

Vectors and matrices operations

- Matrix transposition: $\mathbf{A}^T$ transposed of $\mathbf{A}$ defined as $(\mathbf{A}^T)_{i,j} = A_{i,j}$
- Matrix multiplication: Let $\mathbf{A} \in \mathbb{R}^{m \times k}$, $\mathbf{B} \in \mathbb{R}^{k \times n}$, $\mathbf{C} = \mathbf{AB}$ is defined for each $\mathbf{C}$ element as $C_{i,j} = \sum_k A_{i,k} B_{k,j}$ with $\mathbf{C} \in \mathbb{R}^{m \times n}$
- Transposed matrix multiplication: $(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$
- Matrix-vector multiplication: $\mathbf{Ax} = \sum_i x_i \mathbf{A}_{:,i}$
- Identity matrix: A square matrix that preserve any vectors it is multiplied with. For vectors of size n, the identity matrix $\mathbf{I}_n$ is defined by $\mathbf{I}_n \mathbf{x} = \mathbf{x}$
- Matrix inverse: $\mathbf{A}^{-1}$ inverse of $\mathbf{A}$ defined by $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}_n$

Norms

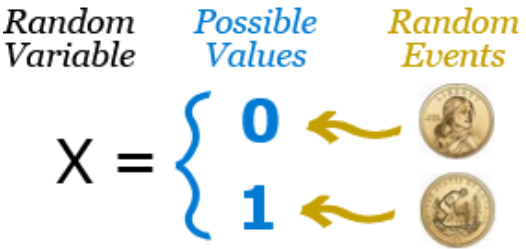
- A **norm** is a function f that measures the size of vectors with the following properties:
  - **Positive definiteness:** $f(x) = 0 \implies \mathbf{x} = 0$
  - **Triangle inequality:** $f(\mathbf{x} + \mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$
  - **Absolute homogeneity** $\forall \alpha \in \mathbb{R}, f(\alpha \mathbf{x}) = |\alpha| f(\mathbf{x})$
- The L^p norm is defined for a vector $\mathbf{x}$ as

$$||\mathbf{x}||_p = \left(\sum_i |x_i|^p \right)^{\frac{1}{p}}$$

- **Euclidian norm** is the L^2 norm, noted $||\mathbf{x}||$ and equal to $\sqrt{\mathbf{x}^T \mathbf{x}} = \sqrt{\sum_i x_i^2}$. Generally, the squared Euclidian norm is used
- **Manhattan norm** is the L^1 norm and is used when the difference between zero and nonzero elements is important
- **Max norm** is defined as $||\mathbf{x}||_\infty = \max_i |x_i|$

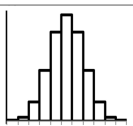
Random Variables

- A **random variable** is a variable that can take different values randomly
- Notation: a random variable $\mathbf{x}$ can take different x values
- Random variables can be **discrete**, like the number of a dice, or **continuous**, like the humidity in the air.



Probability Distribution

- A p probability distribution is a **Probability Mass Function (PMF)** for discrete random variables and a **Probability Density Function (PDF)** for continuous random variables
- p has to satisfy the following properties:
  - p domain describe all possible states of **x**
  - $\forall x \in \mathbf{x}, p(x) \geq 0$
  - $\int p(x)\, dx = 1$

Discrete	Continuous
	
Probability Mass Function	Probability Density Function
Count,Sum,Proportion	Integration
P(X = x) = f(x)	$P(X=x) = \int f(x).dx$
CMF,PMF = Sum, Difference	CDF,PDF = Integrate, Differentiate

Marginal and Conditional Probabilities

We have two random variables $\mathbf{x}$ and $\mathbf{y}$ and we know the probability distribution $p(x, y)$

- **Marginal probability**:

$$\forall x \in \mathbf{x}, p(\mathbf{x} = x) = \int p(x, y)\, dy$$

- **Conditional probability**:

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

- **Chain rule of Conditional Probabilities**:

$$p(x^{(1)}, \ldots, x^{(n)} = p(x^{(1)} \prod_{i=2}^{n} p(x^{(i)}|x^{(1)}, \ldots, x^{(i-1)})$$

Independance and Conditional Independance

- $\mathbf{x}$ and $\mathbf{y}$ are **independant**, $\mathbf{x} \perp \mathbf{y}$, if

$$\forall x \in \mathbf{x}, y \in \mathbf{y}, p(\mathbf{x} = x, \mathbf{y} = y) = p(\mathbf{x} = x)p(\mathbf{y} = y)$$

- $\mathbf{x}$ and $\mathbf{y}$ are **conditionally independe**

$$\forall x \in \mathbf{x},$$

$$p(\mathbf{x} = x, \mathbf{y} = y | \mathbf{z} = z) =$$

We have two random variables $\mathbf{x}$ and $\mathbf{y}$ and we know the probability distribution $p(x, y)$

- **Marginal probability**:

$$\forall x \in \mathbf{x}, p(\mathbf{x} = x) = \int p(x, y)\, dy$$

- **Conditional probability**:

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

- **Chain rule of Conditional Probabilities**:

$$p(x^{(1)}, \ldots, x^{(n)} = p(x^{(1)} \prod_{i=2}^{n} p(x^{(i)} | x^{(1)}, \ldots, x^{(i-1)})$$

Expectation, Variance and Covariance

- **Expectation** of a function $f(x)$ given $p(x)$ is the average value of f on x

$$\mathbb{E}_{x\sim p}[f(x)] = \int p(x)f(x)\,dx$$

- **Variance** of $f(x)$ measure how the values of f varies from its average

$$\textbf{Var}(f(x)) = \mathbb{E}\left[(f(x) - \mathbb{E}[f(x)])^2\right]$$

  and the **standard deviation** is the square root of the variance
- **Covariance** of two random variables provides information of how much two values are linearly related

$$\textbf{Cov}(f(x), g(y)) = \mathbb{E}\left[(f(x) - \mathbb{E}[f(x)])(g(y) - \mathbb{E}[g(y)])\right]$$

Objective

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.

- **Task T**: classification, regression, translation, generation, anomaly detection, ...
- **Performance measure P**: specific to the tasks such as accuracy for classification. It is measured on a **test set**.
- **Experience E**: two main categories supervised and unsupervised
  - Supervised learning: a dataset of points associated with a label or a target
  - Unsupervised learning: a dataset of points without labels or targets

Mathematically

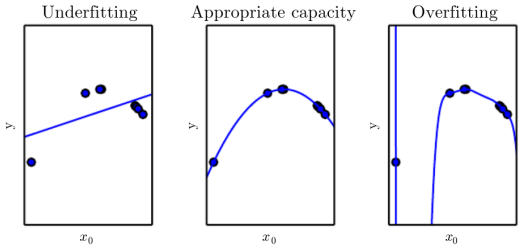
- A dataset of m points and k features can be represented as a matrix $\mathbf{X} \in \mathbb{R}^{m \times k}$.
- In case of supervised learning $\mathbf{X}$ is associated with a vector of labels $\mathbf{y}$ and we aim to learn a joint distribution $p(\mathbf{x}, y)$ to infer

$$p(x|\mathbf{y}) = \frac{p(\mathbf{x}, y)}{\sum_{y'} p(\mathbf{x}, y')}$$

- The goal of machine learning is to find a function $\tilde{f}$ such as $\tilde{f}$ associate each $\mathbf{x}$ to the best approximation of $\mathbf{y}$ and that it is capable to generalize to unseen data.
- $\tilde{f}$ can be parameterized by a set of parameters θ

The capacity of a model

- The main challenge of a machine learning model is **generalization** to unseen data estimated on *test* data after training on *training* data
- **Overfitting** occurs when the gap between training error and test error is too large and **underfitting** when the training error is too low
- The **capacity** of a model is the range of functions it is able to learn and control how likely the model can overfit or underfit



Curse of Dimensionality

- Supposing we have a 1D line, sampled with 10 points.
- How many points do we need for an equivalent sampling in 2D? in 3D?
- Curse of dimensionality: There is an exponential increase in the number of points w.r. the space dimension in order to keep the same sampling quality.

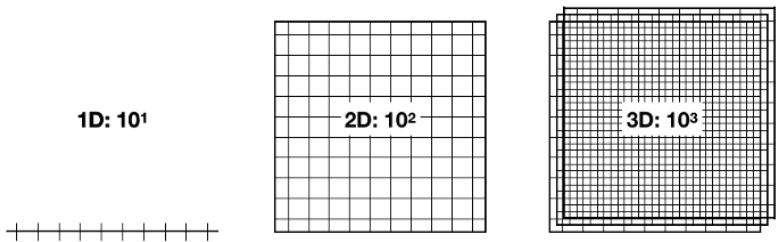


Figure: source:https://builtin.com/data-science/curse-dimensionality

Estimators

- The best parameter θ is unknown and we seek to estimate it through an **estimator** $\hat{\theta}$
- A **point estimator** is any function of the data **X**

$$\hat{\theta} = g(\mathbf{X})$$

- The **bias** of an estimator is

$$\text{bias}(\hat{\theta}) = \mathbb{E}(\hat{\theta}) - \theta$$

  An estimator is unbiased if $\text{bias}(\hat{\theta}) = 0$
- The **variance** of an estimator is

$$\text{Var}(\hat{\theta})$$

Maximum Likelihood Estimation

- To make a good estimator one of the most common principle is **maximum likelihood**.
- Consider $\mathbf{X} = \{\mathbf{x^{(1)}}, \ldots, \mathbf{x^{(m)}}\}$ and $p(\mathbf{x}; \theta)$ a parametric family of probability distributions that maps for each x a real number estimation the probability $p_{data}(x)$
- The **maximum likelihood estimator** is

$$\begin{aligned}\theta_{\mathbf{ML}} &= \arg\max_{\theta} p_{model}(\mathbf{X}; \theta) \\ &= \arg\max_{\theta} \prod_{i=1}^{m} p_{model}(\mathbf{x^{(i)}}; \theta)\end{aligned}$$

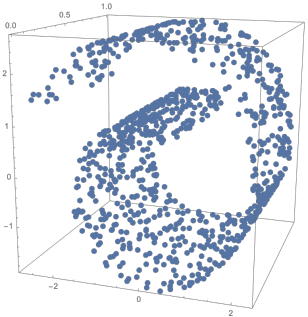
- The maximum log-likelihood estimator is often used to remove the product:

$$\theta_{\mathbf{ML}} = \arg\max_{\theta} \sum_{i=1}^{m} log(p_{model}(\mathbf{x^{(i)}}; \theta))$$

Plan

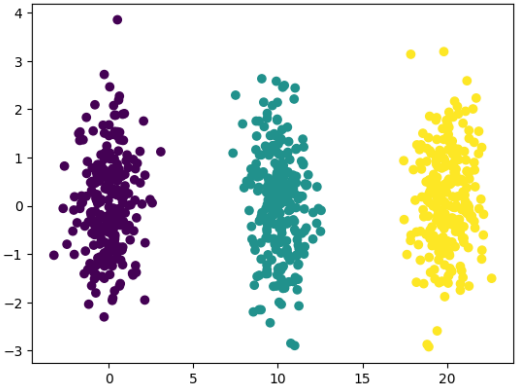
Common Characteristics for Data

- **Redundancy:** Original dimensions of the data are often higher than what is needed
- **Clusterable data**: Groups can be defined to describe the input data
- **Structured data**: Data can follow a given structure (Shape, Manifold etc...)



Occam's razor (Law of parsimony), *William of Ockham*

pluralitas non est ponenda sine necessitate: Plurality should not be posited without necessity.



Projection through PCA(Principal Component Analysis

Definition

Supposing we have a dataset $X \in \mathbf{R}^{n \times m}$, PCA projection consists in finding a new orthonormal coordinate system, such as the first coordinates retain most of the variance. I.e., with Y the projected data, the first j component $Y_{1...j}$ maximizes the variance.

PCA: notations and objective

Definition

- The orthogonal projection of a vector x on a normalized vector u is $P_u(x) = (x^T u)u$
- We thus look for directions u that maximize the variance of the projection $Var(||P_u(x_i)||) = \frac{1}{n-1}\sum_i (x_i^T u - \bar{x}^T u)^2 = u^T \Sigma u$
- I.e. we want to find $argmax_u u^T \Sigma u$, s.t. $||u|| = 1$

PCA as a constraint problem

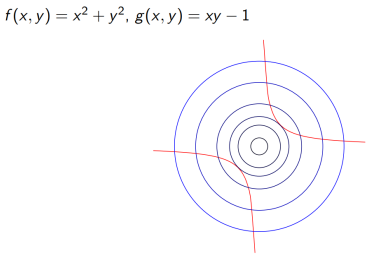


Figure: Optimisation sous contraintes avec le lagrangien, *Agathe Herrou, CNRS*

- **Objective:** While staying on the line $g = 0$, we want to maximize f. We thus look for stationary point (local maxima) of f when walking on g.
- It happens on the contour line of f (i.e., $f(x, y) = c$) which occurs formally when the lines defined by $g = 0$ and $f = c$ are parrallel.

PCA as a constraint problem

- f and g are parallel when ∇f and ∇g are parrallel.
- i.e. when $\exists \lambda$, s.t. $\nabla f = \lambda \nabla g$
- Lagrangian function : $L(x, \lambda) = f(x) + \lambda g(x)$
- Under the condition of $-\nabla_x^2 L$ being positive semi-definite, finding x s.t. $\nabla_\lambda L = \nabla_x L = 0$ thus fulfills the constraint objective.

PCA as a constraint problem

- In our case, the Lagrangian is :

$$L(u, \lambda) = u^T \Sigma u + \lambda(1 - u^T u)$$

-

$$\nabla_u L = 0 \Leftrightarrow \Sigma u = \lambda u$$
$$\nabla_\lambda L = 0 \Leftrightarrow u^T u = 1$$

- $\Sigma u = \lambda u \Rightarrow u^T \Sigma u = \lambda$, so to maximize $u^T \Sigma u$ we need the find the eigenvector associated to the highest eigen value

PCA as a constraint problem

- We can repeat this process, adding the orthogonality constraint d time
- Or, using the spectral theorem, we know that σ, being a positive-semidefinite matrix, has d orthonormal eigenvectors, with d the number of nonnull eigenvalues.

PCA computation: summary

- Center data (Why?)
- Use singular value decomposition on $X^T X$, which will give you eigenvalues and associated eigenvectors.
- Ordere the eigen vector w.r. the eigenvalues (from highest to lowest), and project the centered data on the d first eigenvectors

PCA limitations

- Performs linear transformations on the data (linear projection)
- Mean of the data and covariance matrix are supposed to be enough to reduce dimensionality
- Assume that large variance is an interest criterion

PCA limitations

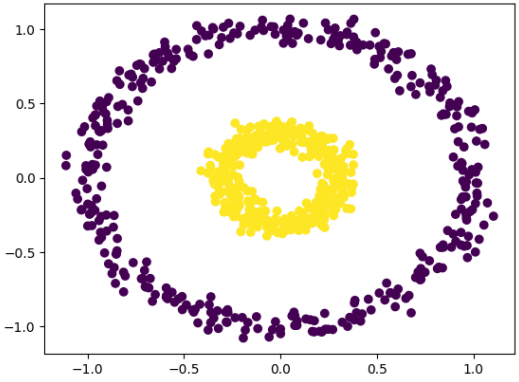


Figure: Is PCA useful here ?

Kernel PCA

- Idea: data are separable in higher dimension
- **Theoritically** relies on a non-linear transformation

$$\phi : \mathbf{R}^m \to \mathbf{k}, \quad x \to \phi(x)$$

with $k > m$ .This transformation (also called mapping) is complicated to find, instead it is easier to define a kernel function $k(x, y) = \phi(x)^T \phi(y)$

Kernel PCA: quick theoretical summary

- We apply the same logic as with PCA but on the $\phi(x_i)$ (feature space) instead of just x_i
- we obtain $\Sigma = \frac{1}{n-1} \sum_i^n \phi(x_i)\phi(x_i)^T$
- Solving $\Sigma u = \lambda u$ we obtain

$$u = \sum_i \frac{1}{\lambda(n-1)} (\phi(x_i)^T u)\phi(x_i) = \sum_i \alpha_i \phi(x_i)$$

- Sparing you the details we obtain a the end :

$$\forall j \in 1...n : \frac{1}{n-1} \sum_{s=1}^{n} \alpha_s \sum_i k(x_j, k_i)k(x_i, x_s) = \lambda \sum_i \alpha_i k(x_i, x_j)$$

Kernel PCA: quick theoretical summary

- Noting $\mathbb{K} = \{k(x_i, x_j)\}_{i,j \in [|1,n|]^2}$ and $\boldsymbol{\alpha} = \{\alpha_i\}_{i \in [|1,n|]}$, we have :

$$\mathbf{K}\boldsymbol{\alpha} = \lambda(n-1)\boldsymbol{\alpha}$$

- The constraint $u^T u = 1$ gives, by replacing u with its expression

$$\boldsymbol{\alpha}^T \boldsymbol{\alpha} = \frac{1}{\lambda(n-1)} = \frac{1}{c}$$

- Solving these equations is thus equivalent to find the eigen vectors of $\mathbf{K}$ and divide them by $c_i = \lambda_i(n-1)$
- We can then find for each component j of a sample x $y_j = \sum_i \alpha_i^j k(x, x_i)$

Kernel PCA: which kernel function to use?

- k must satisfy the **Mercier's condition**
- In finite dimension, it is simplified by having $\mathbf{K}$ semi-definite positive
- Commonly used kernels are the gaussian kernel ($k(x, y) = e^{\frac{||x-y||^2}{\sigma^2}}$ or the polynomial one $k(x, y) = (x^T y)^a$

Other Projections approaches: t-SNE and UMAP

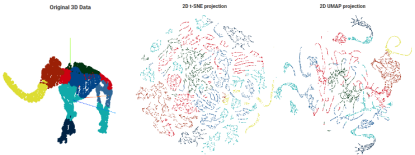


Figure: https://pair-code.github.io/understanding-umap/

- t-SNE apply for each data point a Gaussian kernel on this point and its neighbors and normalize the values to obtain probabilities values. Modelizing the probabilities of the lower-dimensioned space with a student kernel, t-SNE minimizes the KL divergence between the probabilities in the lower dimension and probabilities in the higher one.
- UMAP relies on topological analysis of the high-dimension data to define a structure of the data. Optimize the projected space to reproduce the structure of the original space.

Clustering Algorithms : K-means

Algorithm 1: K-means Clustering

Input: Data points $X = \{x_1, x_2, \ldots, x_n\}$, number of clusters k
Output: Cluster centroids $C = \{c_1, c_2, \ldots, c_k\}$, cluster assignments for each point

Initialize C with k random points from X;
repeat
  foreach $x_i \in X$ **do**
    Assign x_i to the nearest centroid c_j;
  foreach $c_j \in C$ **do**
    Update c_j to be the mean of all points assigned to it;
until *convergence*;
return C, cluster assignments;

Clustering Algorithms and their behviour

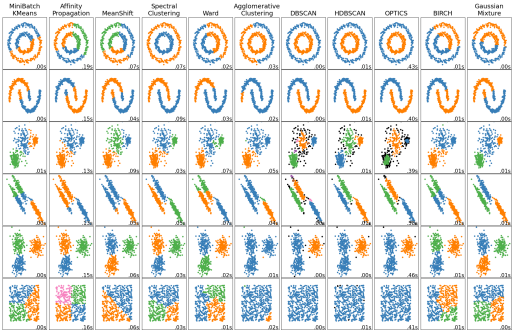


Figure: https://scikit-learn.org/stable/auto$_e xamples/cluster/plot_c luster_c omparison.html$

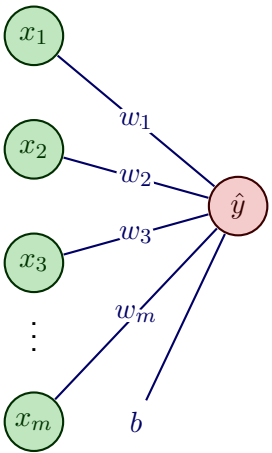
Plan

Perceptron

- A perceptron is an algorithm for supervised learning of binary classifiers (two classes).
- Suppose we have a dataset $\mathbf{X} \in \mathbb{R}^{n \times m}$ associated with a vector of labels $\mathbf{y} \in \{0, 1\}^n$
- It learns a function $\tilde{f}$ parametrized by a vector of weights $\mathbf{w} \in \mathbb{R}^m$ and a bias b such as:

$$\tilde{f}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases}$$

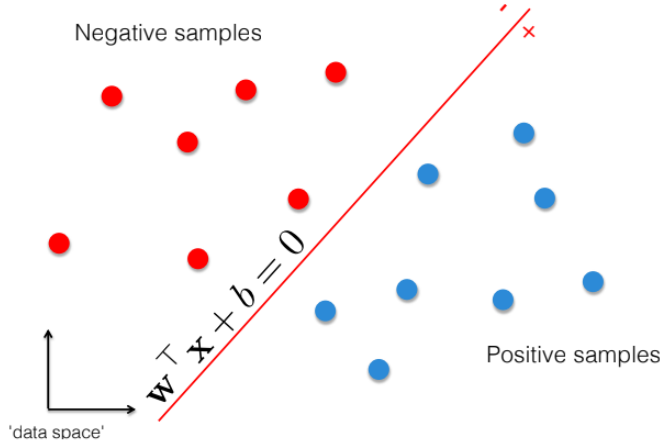
Perceptron visualization



$$\hat{y} = w_1 x_1 + w_2 x_2 + ... + w_m x_m + b$$

Perceptron Decision Boundary

A perceptron can linearly separate data



Gradient Descent for Perceptron

Algorithm 2: Gradient Descent for Perceptron

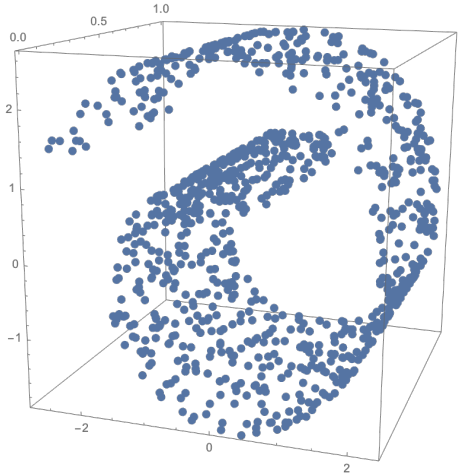
Input: Data points $X = \{x_1, x_2, \ldots, x_n\}$, labels $Y = \{y_1, y_2, \ldots, y_n\}$, learning rate η, number of epochs T

Output: Weights w, bias b

Initialize w and b randomly;
for $t = 1$ **to** T **do**
  foreach $(x_i, y_i) \in (X, Y)$ **do**
    Compute $\hat{y}_i = wx_i + b$;
    Compute loss $L = \frac{1}{2}(\hat{y}_i - y_i)^2$;
    Compute gradients $\frac{\partial L}{\partial w} = (\hat{y}_i - y_i)x_i$;
    Compute gradients $\frac{\partial L}{\partial b} = (\hat{y}_i - y_i)$;
    Update $w := w - \eta \frac{\partial L}{\partial w}$;
    Update $b := b - \eta \frac{\partial L}{\partial b}$;

Perceptron limitation

A perceptron cannot separate non linear data



Percepton limitations mitigation : Adaboost

Let's $\{h\}$ be a set of perceptions (weeak classifiers), $H = sign(\sum(h(x))$, we want to find the right h to get the best H possible.

Adaboost Algorithm

Input: $\mathbf{H}$: a chosen class of "weak" binary classifiers
Output: $F_t = \text{sign}(H_t)$

Initialize: $w_1(i) = \frac{1}{n}$, $H_0 = 0$;
for $t = 1$ **to** T **do**
  $h_t = \arg\min_{h \in \mathbf{H}} \epsilon_t(h)$;
  where $\epsilon_t(h) = \sum_{i \sim w_t} [h(x_i) \neq y_i]$;
  Choose α_t;
  Update w_{t+1};
  $H_t = H_{t-1} + \alpha_t h_t$;
end

Output: $F_T = \text{sign}(H_T)$;

Updating parameters

- $\alpha_t = \frac{1}{2} log(\frac{1-\epsilon_t}{\epsilon_t}$
- $w_{t+1} = \frac{w_t(I) e^{-\alpha_t y_i h_t(x_i)}}{Z_{t+1}}$, Z is chosen so that the sum of w is equal to 1

- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.
- Karras, Tero, Samuli Laine, and Timo Aila (2019). "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, pp. 4401–4410. DOI: 10.1109/CVPR.2019.00453. URL: http://openaccess.thecvf.com/content_CVPR_2019/html/Karras_A_Style-Based_Generator_Architecture_for_Generative_Adversarial_Networks_CVPR_2019_paper.html.

Park, Taesung et al. (2019). "Semantic Image Synthesis With Spatially-Adaptive Normalization". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, pp. 2337–2346. DOI: 10.1109/CVPR.2019.00244. URL: http://openaccess.thecvf.com/content_CVPR_2019/html/Park_Semantic_Image_Synthesis_With_Spatially-Adaptive_Normalization_CVPR_2019_paper.html.

Turing, A. M. (Oct. 1950). "Computing Machinery and Intelligence". In: *Mind* LIX.236, pp. 433–460.