VIASM
VIETNAM INSTITUTE FOR
ADVANCED STUDY IN MATHEMATICS

ĐẠI HỌC FPT
QUY NHƠN AI CAMPUS

CƠ SỞ TOÁN HỌC CHO HỌC MÁY

Mathematical Foundations for Machine Learning

(VIASM - Mathematical Summer School 2024)

Assoc. Prof. Trần Vũ Khanh, PhD

International University – VNU HCM

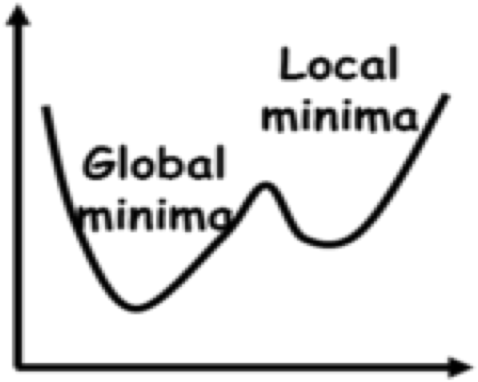
Linear Algebra

$$\vec{a} \longrightarrow \vec{b}$$

$$\begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix}$$

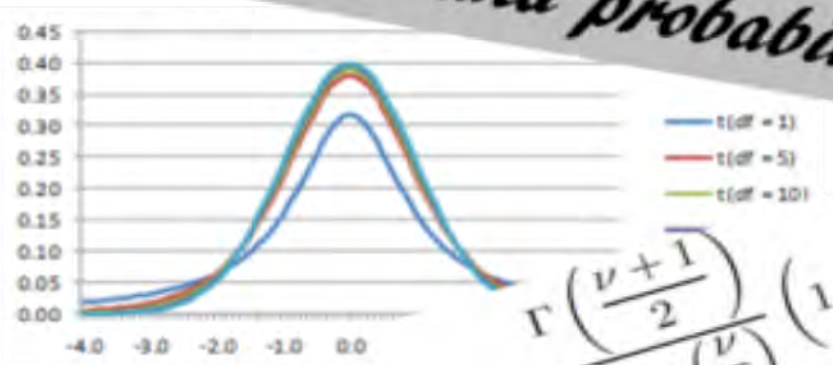
Multi-variable Calculus

$$\frac{dy}{dx} = 0$$



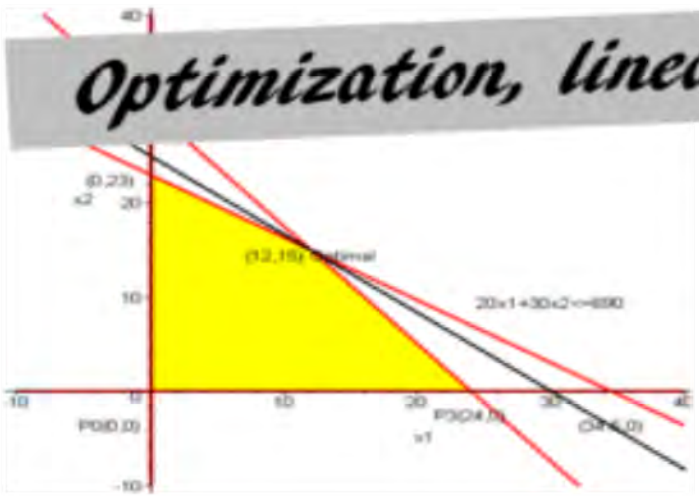
$$\frac{\partial}{\partial x}\left(f_{x,t}, g_{x,w}\right)$$

Statistics and probability



$$f(t) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{t^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

Optimization, linear programming



Outline

- Introduction to ML
- Mathematical foundations for Machine Learning
- Linear Algebra
- Vector Calculus
- Probability and distributions
- Math in ML examples: Linear regression, deep learning, PCA

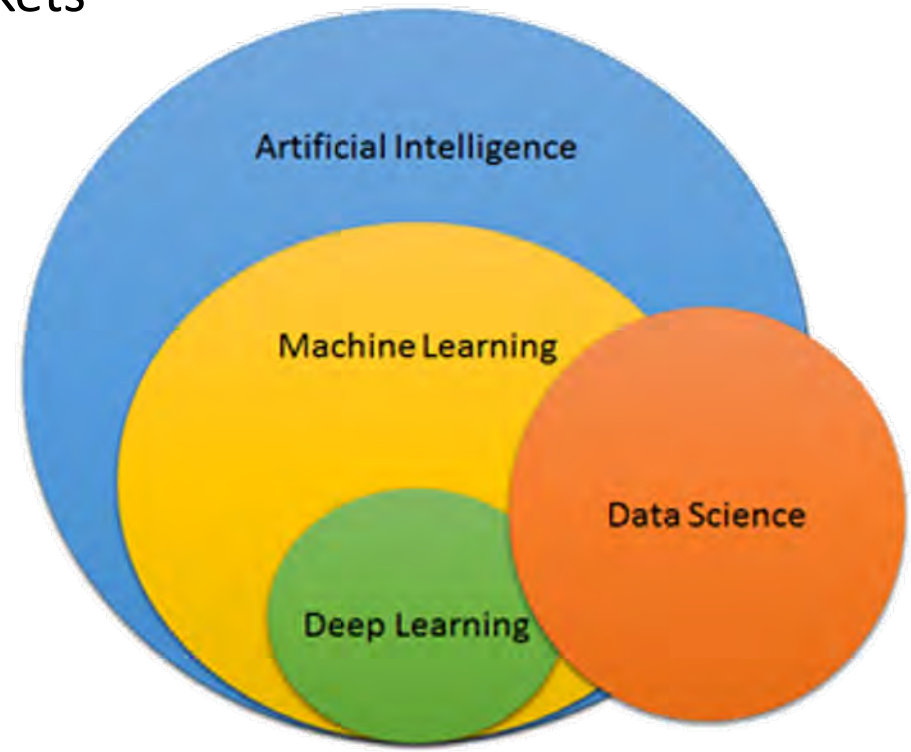
What is machine learning ?

Common definition:

Machine Learning is the study of computer algorithms that improve automatically through experience

Found its way in a lot of daily life applications, e.g.

- Self driving/flying cars/trains/planes/rockets
- Computer games
- Image & Video & Sound processing
- Speech recognition (Siri, Alexa,…)
- Translation (Google translate)
- Weather forecast
- Data analysis
- Email spam
- Robotics
- ChatGPT
- and much much more…



Not machine learning:
“Classical machines / programs”

Programmer understands the problem and can implement an explicit algorithm, which turns an input into an output.

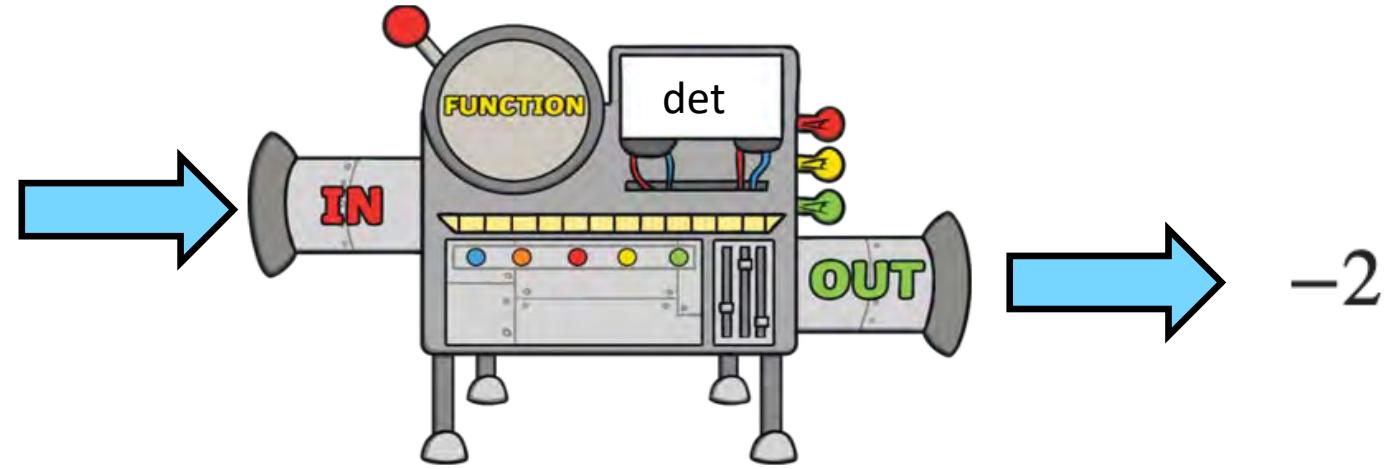
```
def determinant(A):
    total=0
    indices = list(range(len(A)))
    if len(A) == 2 and len(A[0]) == 2:
        val = A[0][0] * A[1][1] - A[1][0] * A[0][1]
        return val
    for fc in indices:
        As = A
        As = As[1:]
        height = len(As)
        for i in range(height):
            As[i] = As[i][0:fc] + As[i][fc+1:]
        sign = (-1) ** (fc % 2)
        sub_det = determinant(As)
        total += sign * A[0][fc] * sub_det
    return total
```

Example: Determinant of a matrix

```
determinant([[1,2],[3,4]])
```

```
-2
```

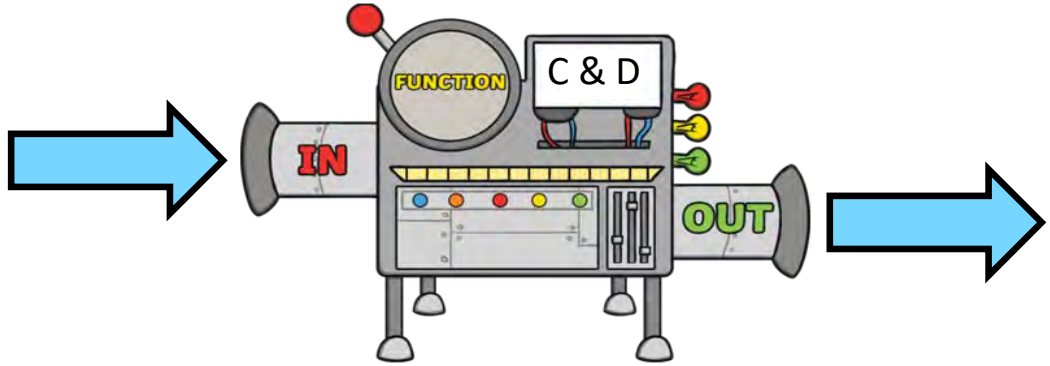
$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$



-2

What do you see?

Want: Cat & Dog detection machine



Cat & Dog detector

“This is a dog!”

Smart person

Of course, I know how cats and dogs look like. But I do not know how to describe it with a formula or algorithm!

How do we know?

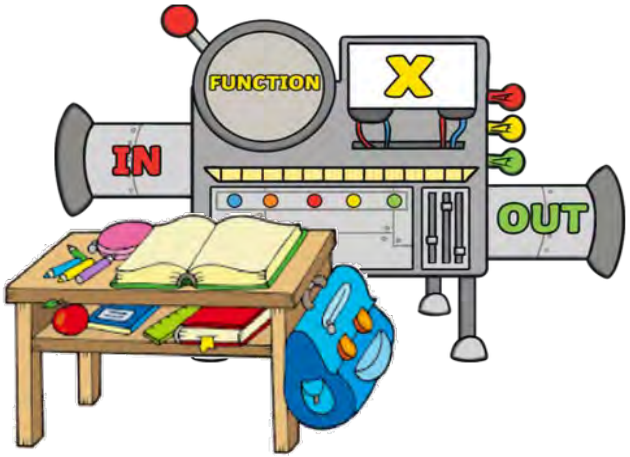
This is a dog because…

- I saw a lot of dogs & cats in my life.
- Whenever I saw a cat I was told by my parents „This is a cat!“. When I saw a dog they told me “This is a dog.”.
- I **learned** to distinguish between dogs and cats.
- But it is hard for me to explain the exact difference.

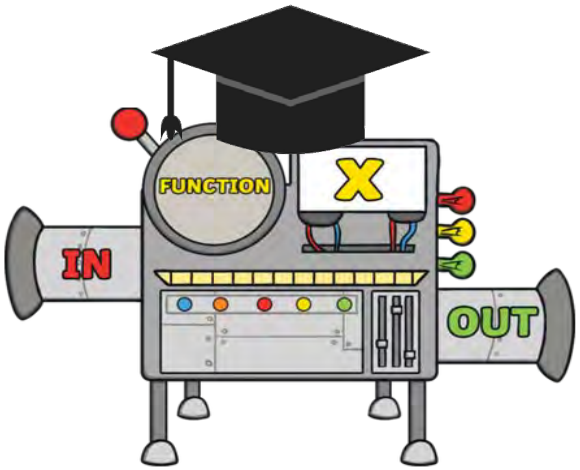
Machine learning

We can create machines/programs which we can teach

Training



After Training

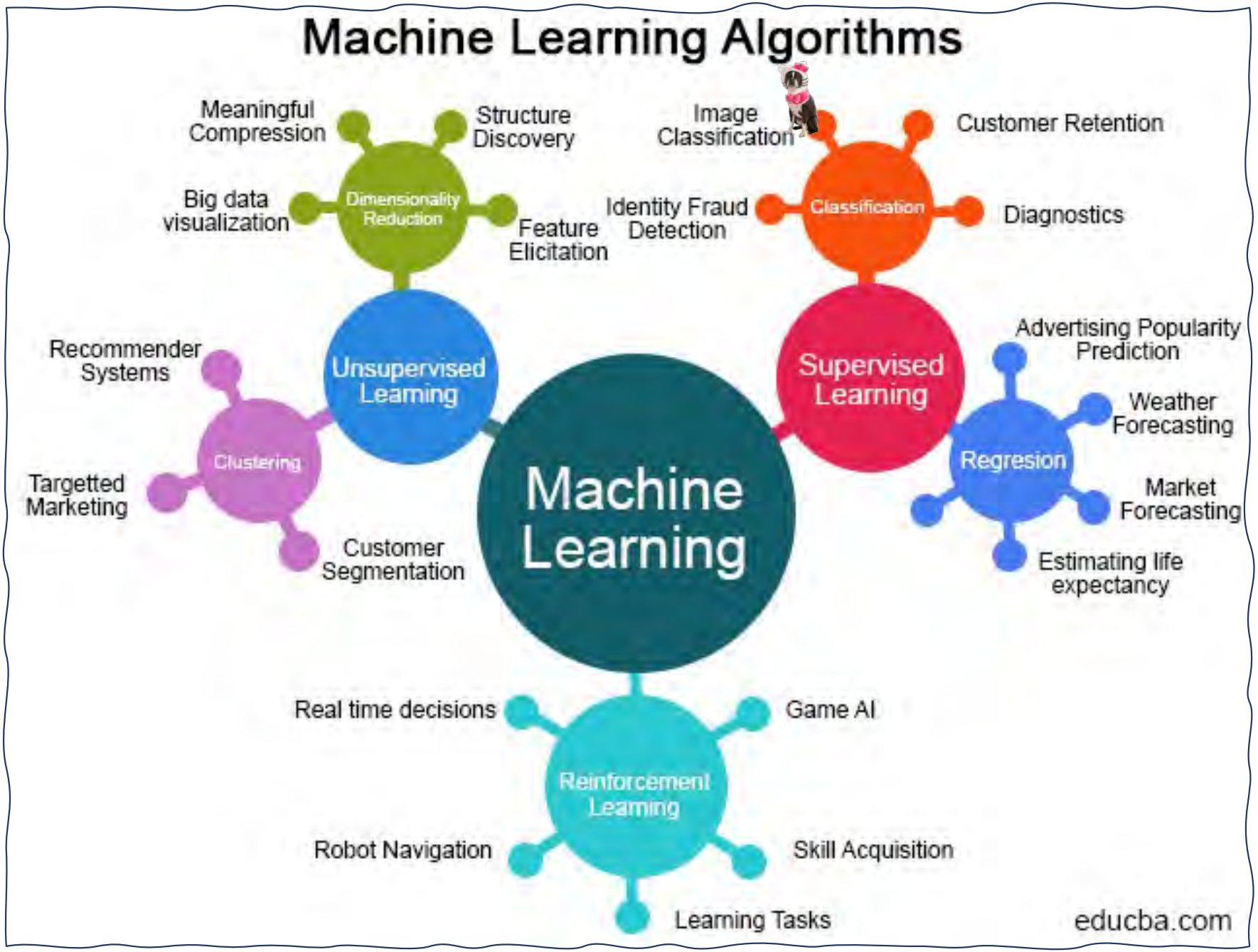


Cat & Dog detector

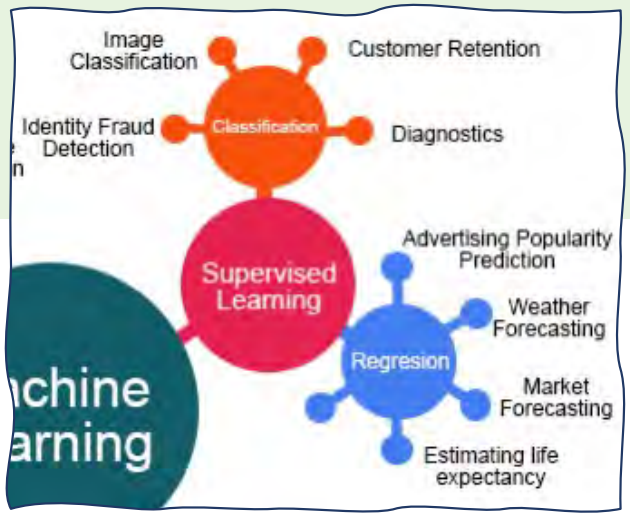
Dog!

… with 95% certainty!

Machine learning overview

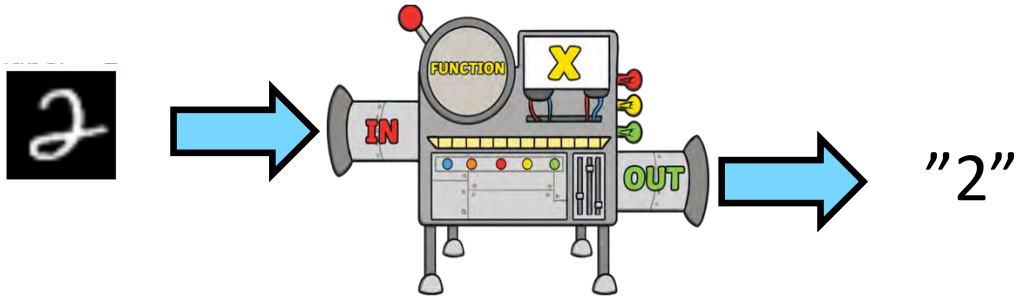


Supervised learning



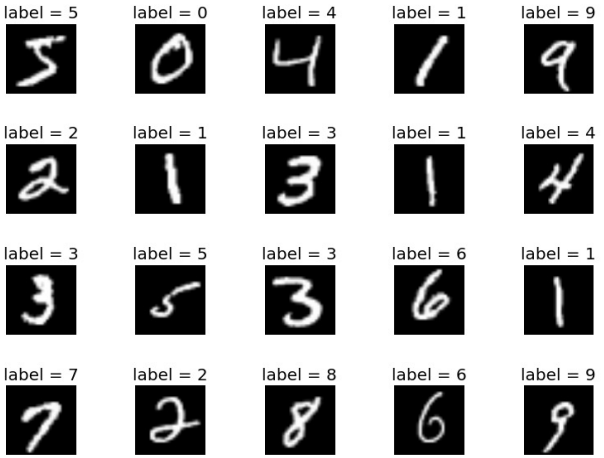
Classification (discrete output)

Output = category (e.g. “dog”, “cat” or “1”,”2”,…,”9”)



”2”

Handwriting recognition

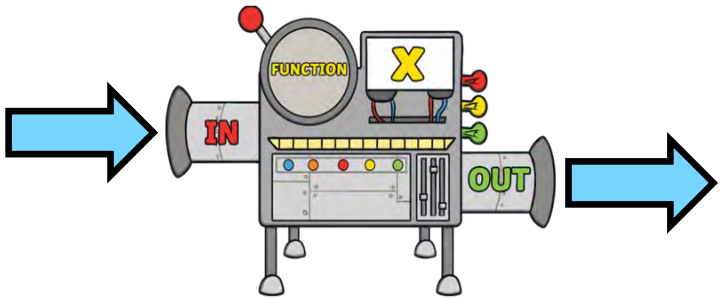


Learning by using some trainingsdata

Regression (continuous output)

Output = real number (like “price”, “weight”,…)

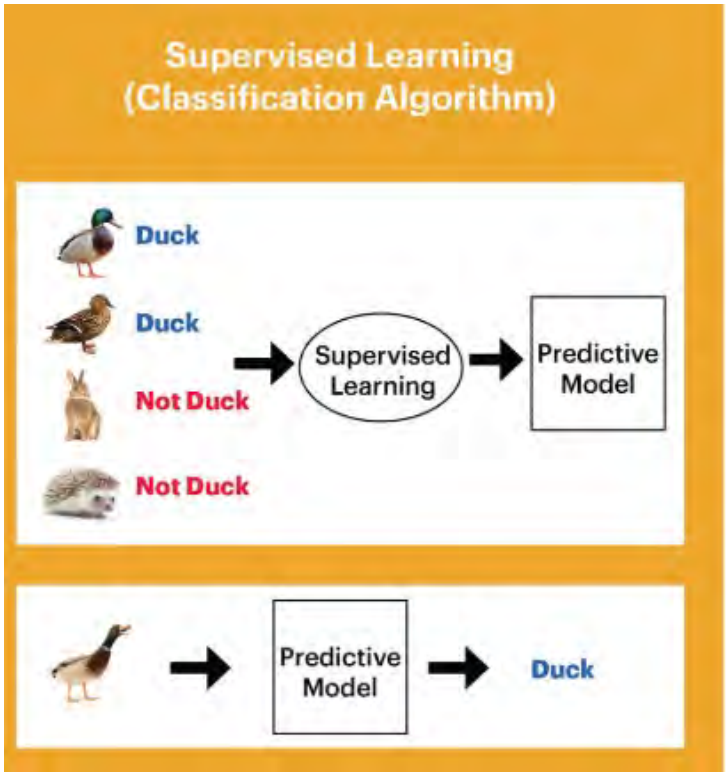
Size: $70m^2$
Rooms: 3

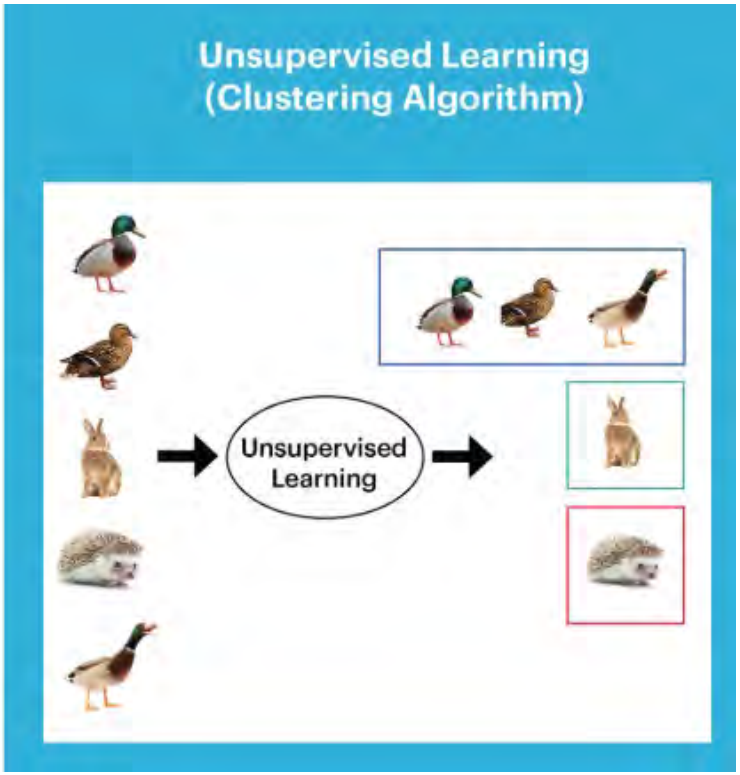


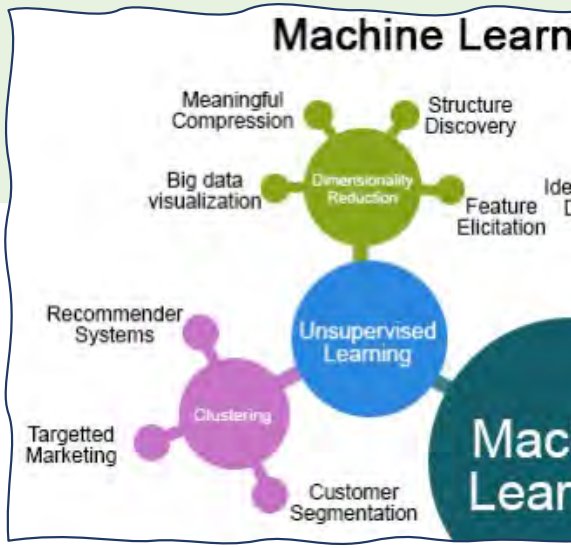
90.000 Yen

Apartment price prediction

Unsupervised learning

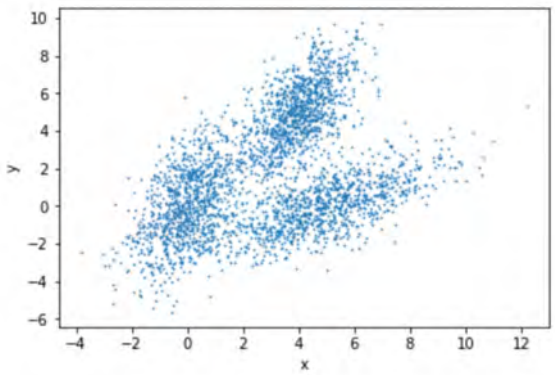




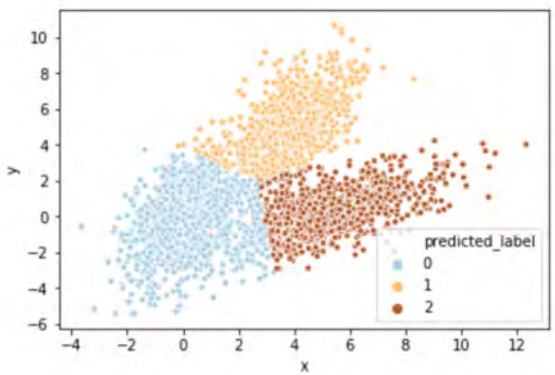


Clustering

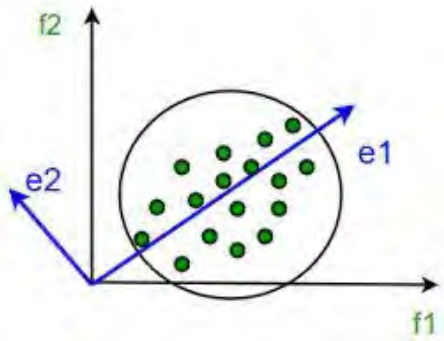
Before



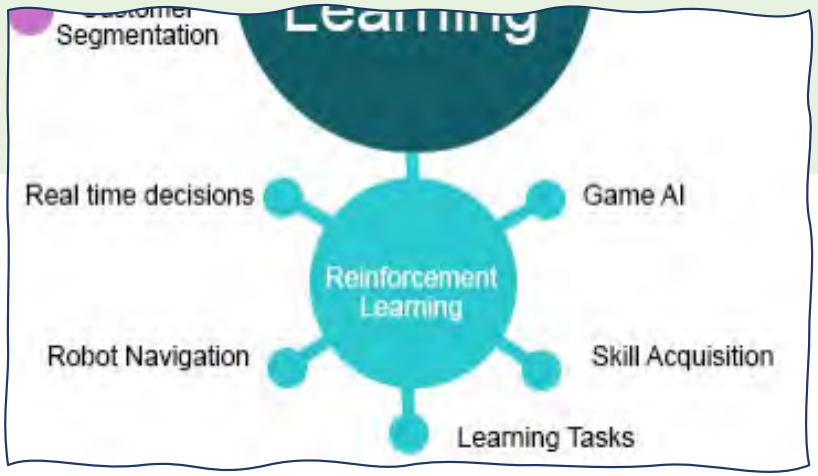
After

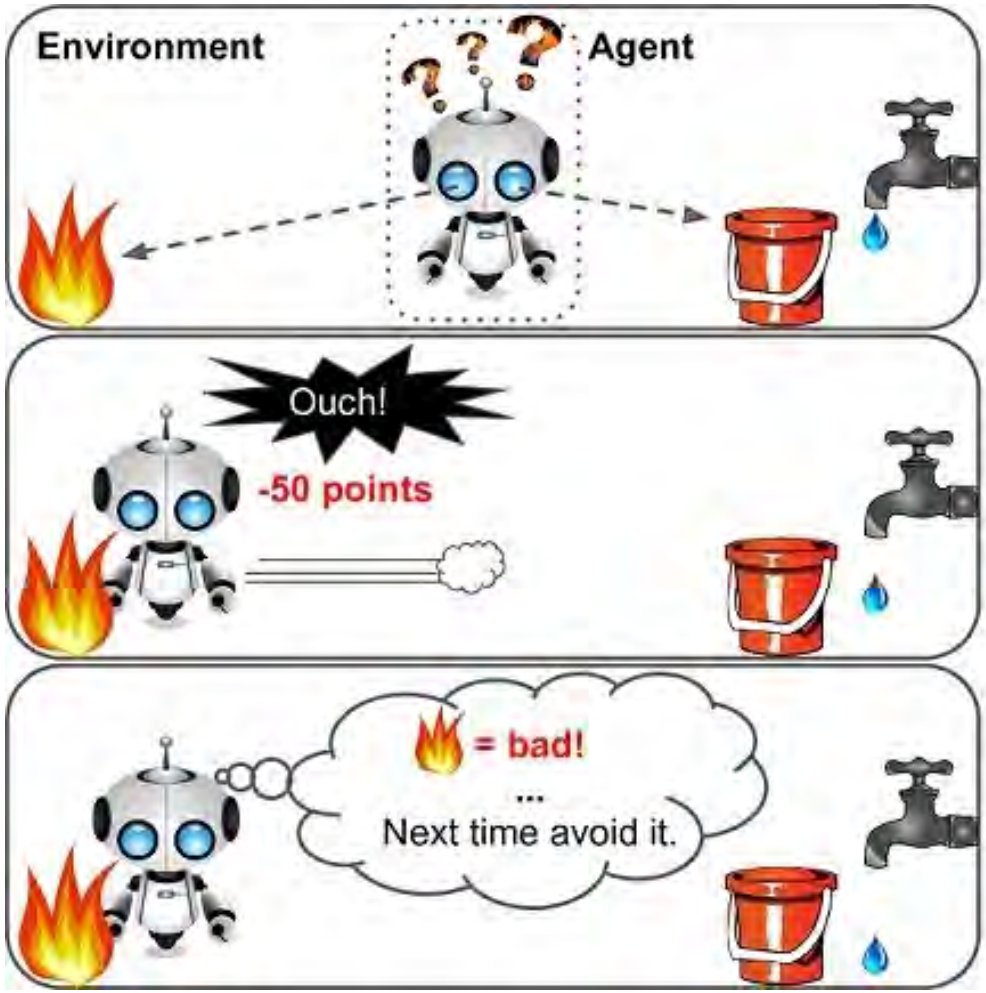


Dimension reduction



Reinforcement learning





1. Observe
2. Select action using policy
3. Action!
4. Get reward or penalty
5. Update policy (learning step)
6. Iterate until an optimal policy is found

Machine Learning Limitations

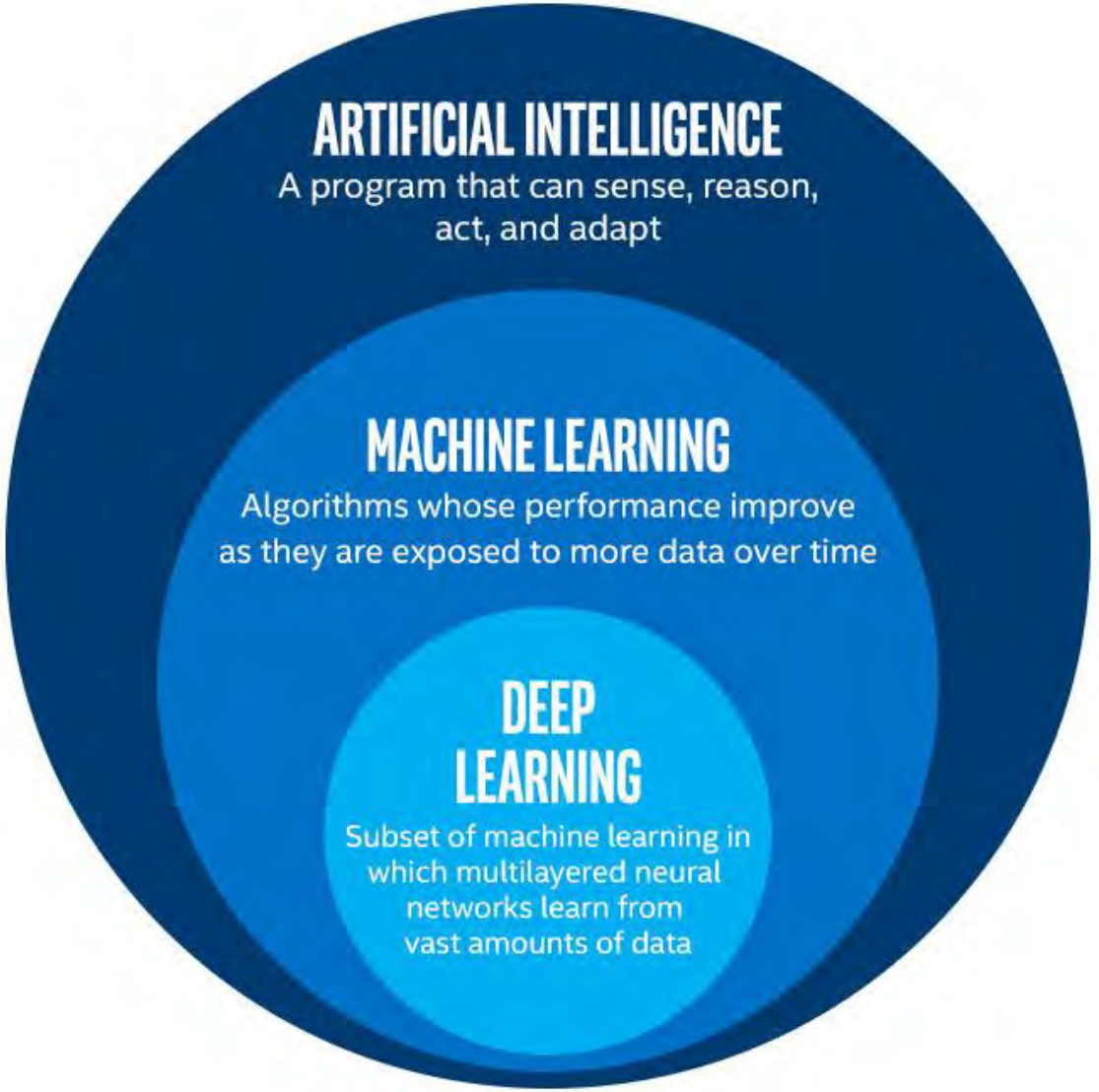
- Suppose you wanted to determine if an image is of a cat or a dog.
- What features would you use?
- *This is where **Deep Learning** can come in.*

Dog and cat recognition

Deep Learning

"Machine learning that involves using very complicated models called "deep neural networks"." (Intel)

Models determine best representation of original data; in classic machine learning, humans must do this.



Deep Learning Example

Classic Machine Learning

Step 1: Determine features.

Step 2: Feed them through model.

Feature Detection

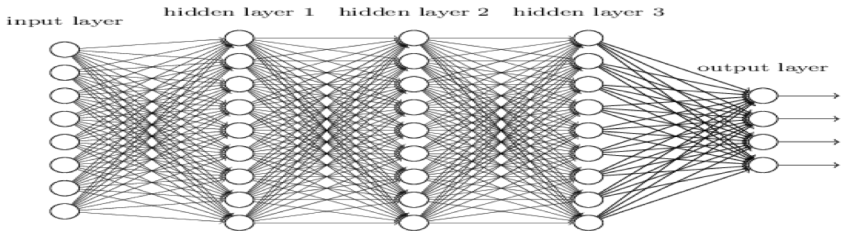
Machine Learning Classifier Algorithm

"Arjun"

Deep Learning

Steps 1 and 2 are combined into 1 step.

Neural Network



"Arjun"

Examples of AI in Chess, Go, Computer games, biology, math

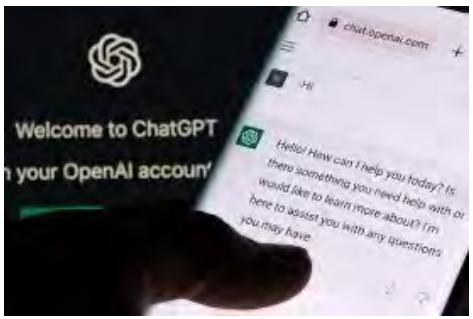
- 1996: Deep blue (not machine learning)
- 2016: Alpha Go
- 2017: Alpha Zero (plays chess, shōgi & go)
- 2019: AlphaStar (Starcraft 2)
- 2020: AlphaFold
- 2022: ChatGPT, AlphaTensor

Reinforcement learning

1996: Garry Kasparov (Chess world champion) loses against Deep blue.

2016: Lee Sedol loses against Alpha Go.

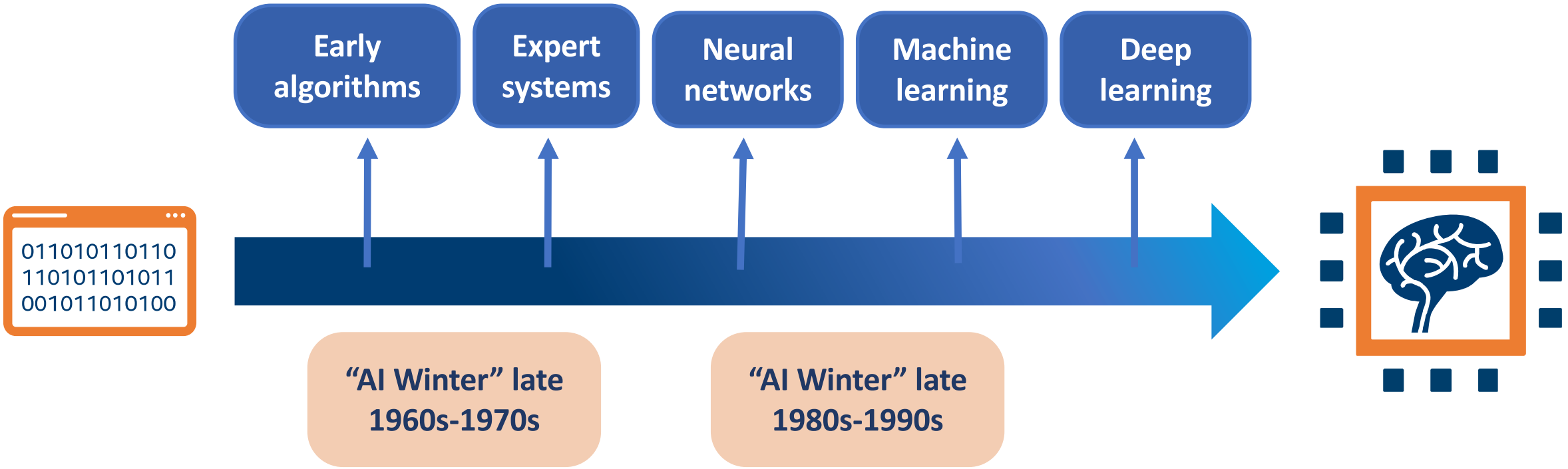
2019: AlphaStar beats Starcraft 2 pro gamers



2022: ChatGPT… for lab reports and more

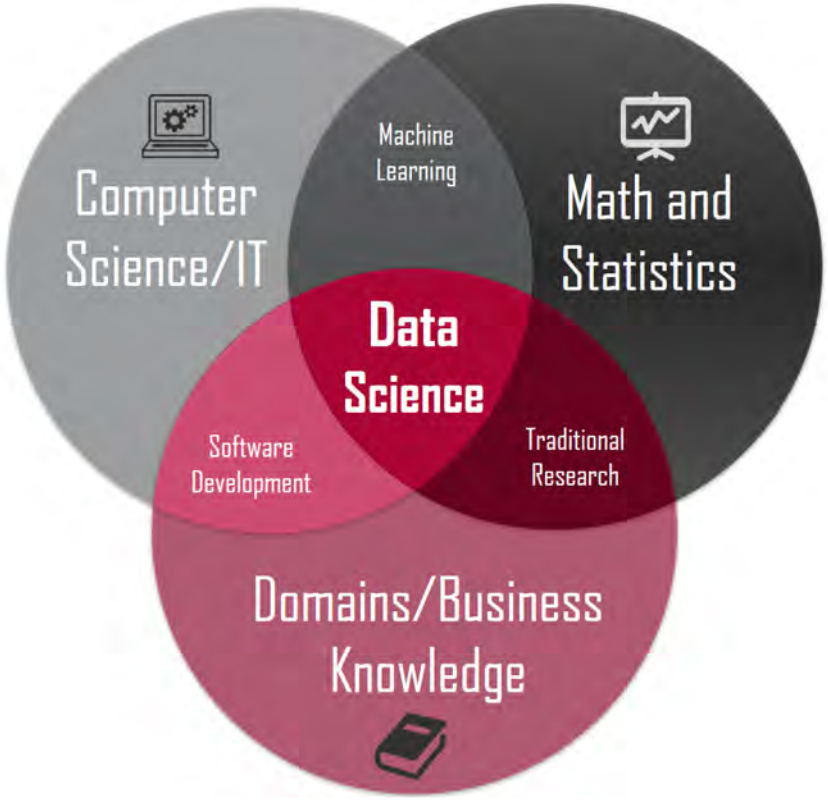
History of AI

AI has experienced several hype cycles, where it has oscillated between periods of excitement and disappointment.



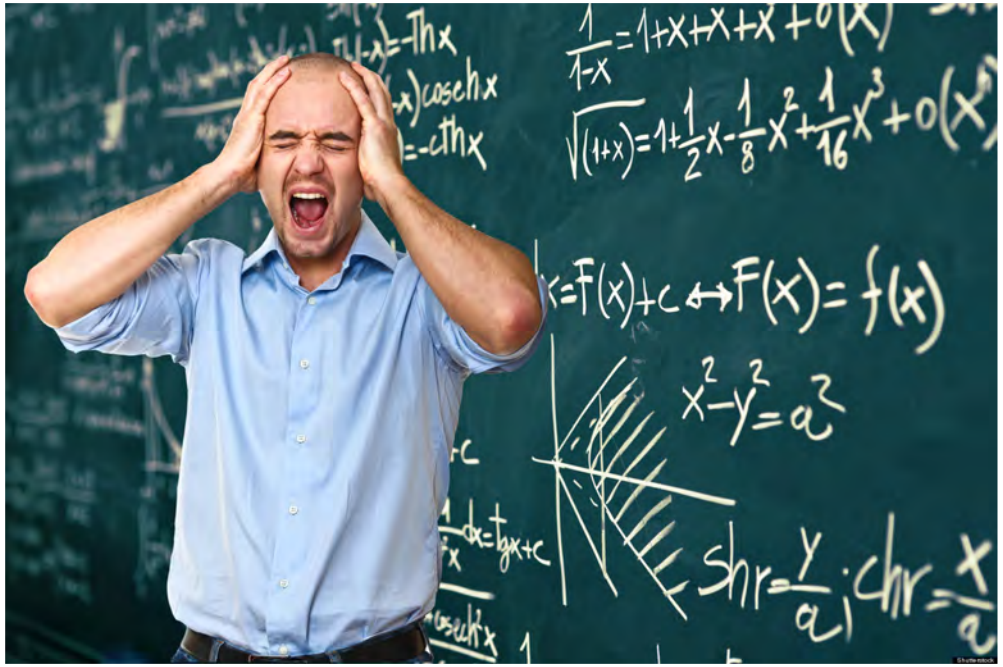
Cơ sở toán học của học máy

Machine Learning



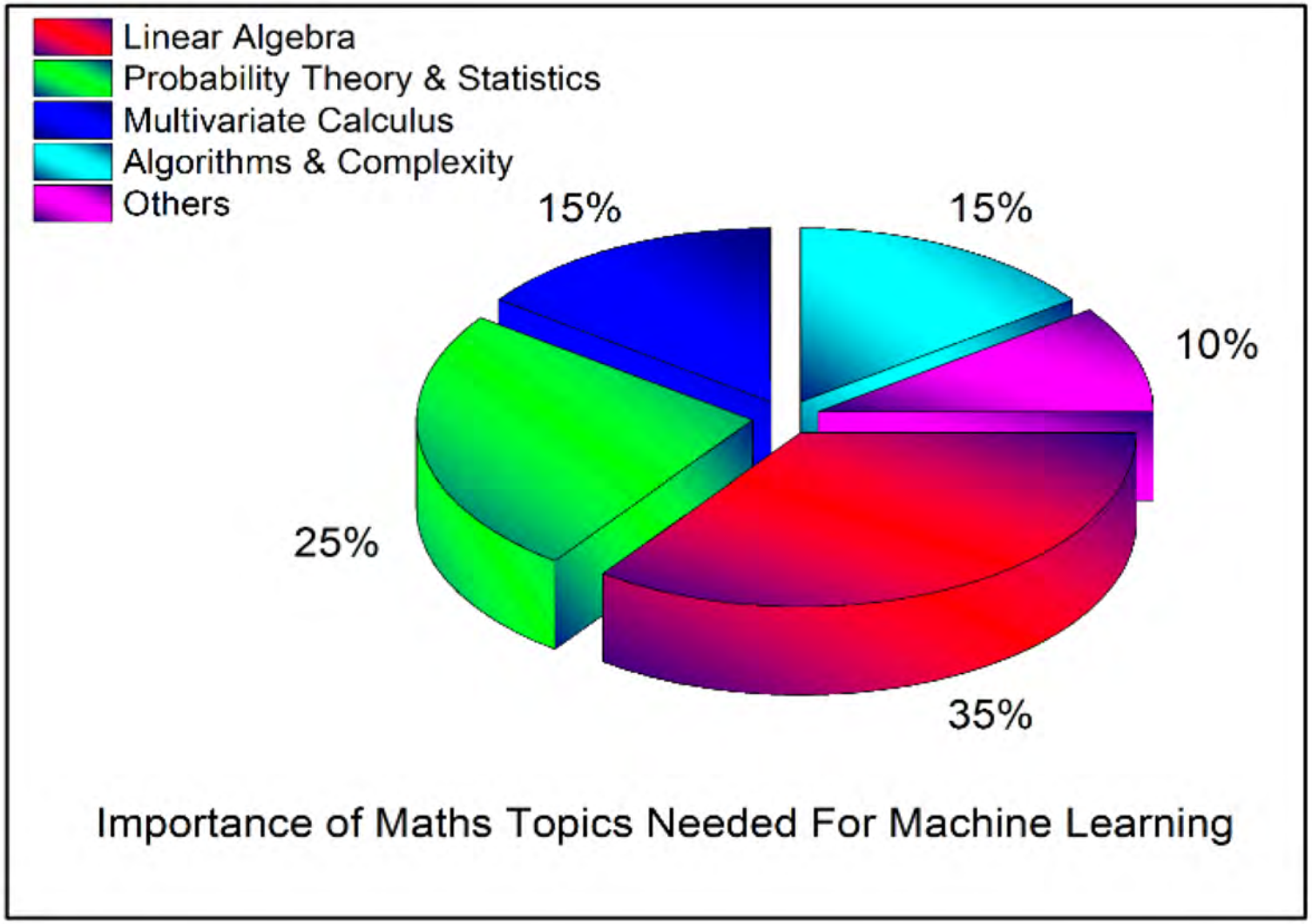
- Machine Learning theory is a field that intersects:
- statistical, probabilistic,
- computer science and algorithmic aspects arising from learning iteratively from data and finding hidden insights which can be used to build intelligent applications

Why Worry About The Maths?

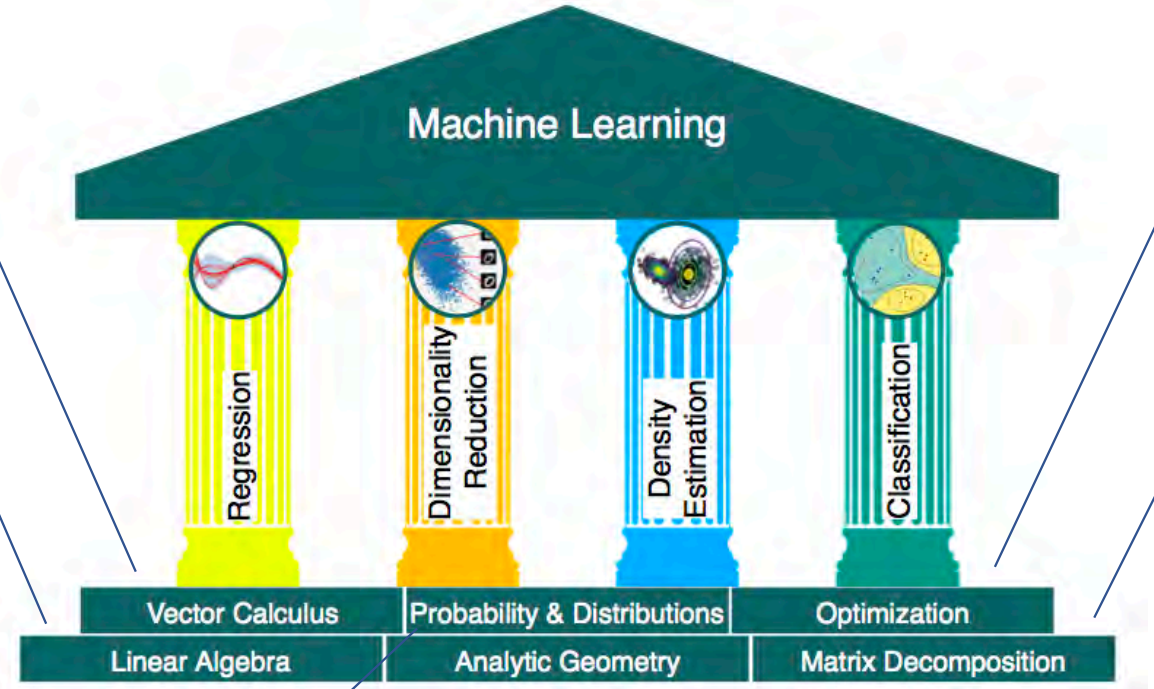


- - Selecting the right algorithm which includes giving considerations to accuracy, training time, model complexity, number of parameters and number of features.
- - Choosing parameter settings and validation strategies.
- - Identifying underfitting and overfitting by understanding the Bias-Variance tradeoff.
- - Estimating the right confidence interval and uncertainty.

What Level of Maths Do You Need?



The foundations and four pillars of machine learning



Về khái niệm gradient được dùng rộng rãi trong các phương pháp tối ưu của học máy

Tính toán cơ bản cho các đối tượng biểu diễn được dưới dạng dữ liệu cấu trúc (vectors và ma trận)

Định lượng độ không chắc chắn của dữ liệu là lĩnh vực của lý thuyết xác suất

Các phương pháp tối ưu để tìm cực trị của các hàm số trong thuật toán học máy.

Công cụ cơ bản để biến đổi biểu diễn của các đối tượng có dữ liệu cấu trúc.

Trọng tâm là việc xây dựng các cách đo sự tương tự giữa các đối tượng.

Math for Machine Learning

Mathematical Foundations

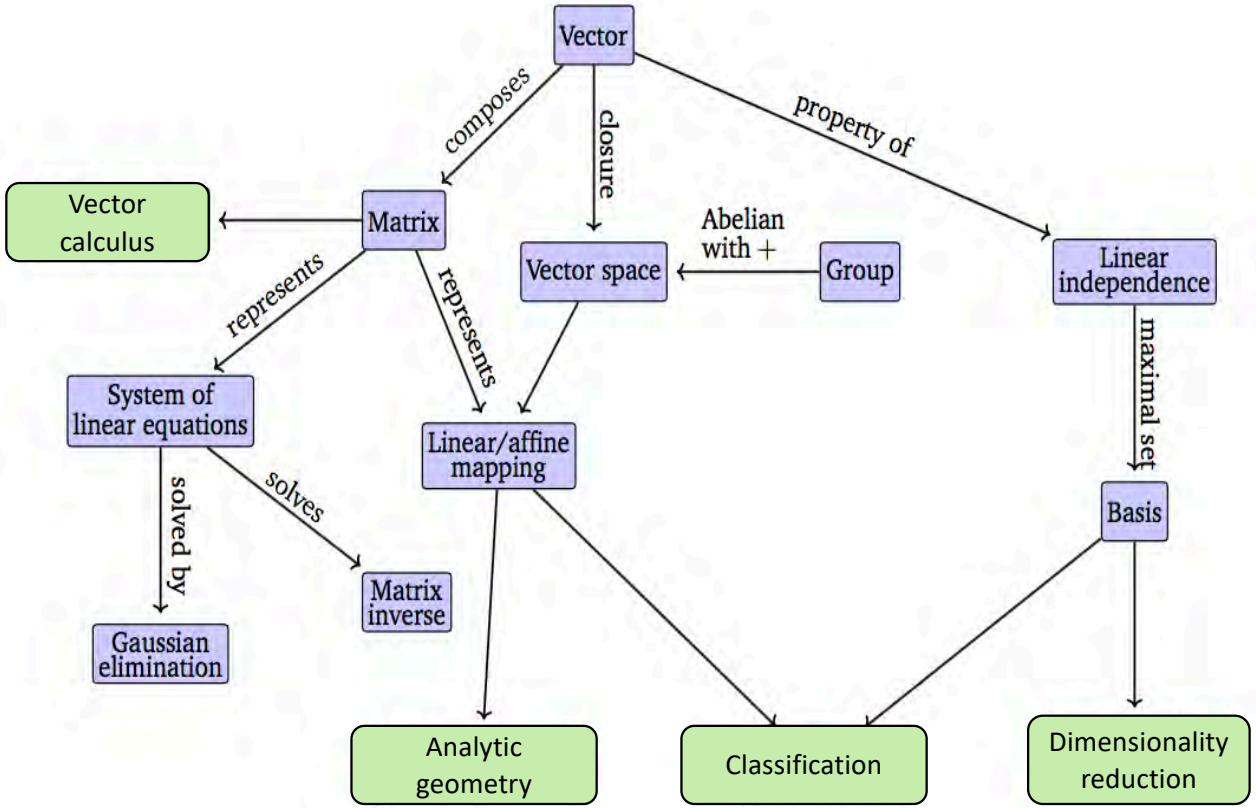
1. Linear Algebra
2. Analytic Geometry
3. Matrix Decompositions
4. Vector Calculus
5. Probability and Distributions
6. Continuous Optimization

Central Machine Learning Problems

1. When Models Meet Data
2. Linear Regression
3. Dimensionality
4. Density Estimation with Gaussian Mixture Models
5. Classification with Support Vector Machines

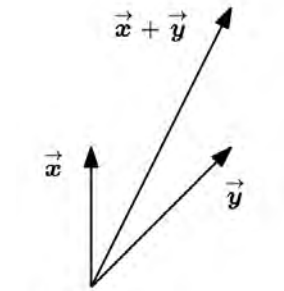
Linear Algebra

1. Systems of Linear Equations
2. Matrices
3. Solving Systems of Linear Equations
4. Vector Spaces
5. Linear Independence
6. Basis and Rank
7. Linear Mappings
8. Affine Spaces

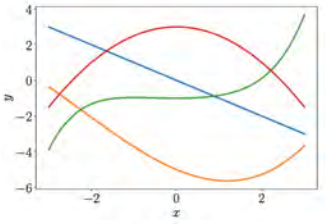


Linear Algebra

- Linear algebra is the study of vectors and certain rules to manipulate vectors.
- Vectors are special objects that can be added together and multiplied by scalars to produce another object of the same kind.
- Geometric vectors, polynomials, audio signals, elements of $\mathbb{R}^n$
- The major question: What is the set of vectors that can result by starting with a small set of vectors, and adding them to each other and scaling them? → Vector space

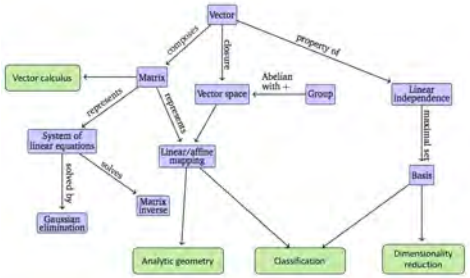


(a) Geometric vectors.

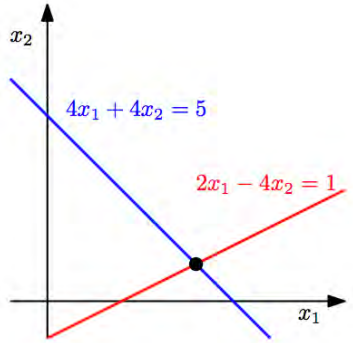


(b) Polynomials.

Systems of Linear Equations

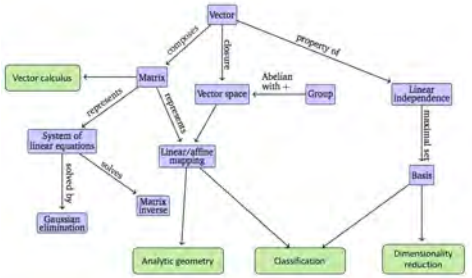


- Many problems can be formulated as systems of linear equations
- The solution a system of linear equations with n variables can be geometrically interpreted as the intersection of n hyperplanes.



$$x_1 \begin{bmatrix} a_{11} \\ \vdots \\ a_{m1} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ \vdots \\ a_{m2} \end{bmatrix} + \cdots + x_n \begin{bmatrix} a_{1n} \\ \vdots \\ a_{mn} \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} \iff \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

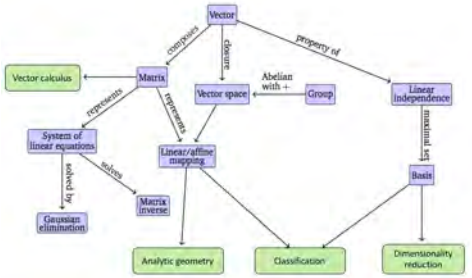
Matrices & Solving Systems of Linear Equations



- Matrix operations: addition, multiplication, inverse, transpose, multiplication by a scalar.
- Particular and general solution
- Elementary transformations
- Algorithms for solving $\boldsymbol{Ax} = \boldsymbol{b} \rightarrow \boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$ (if A is square and invertible) In general case, assuming the solution exists, i.e., $\boldsymbol{A}$ needs to have linearly independent columns (if not find approximate solution, e.g. by regression)

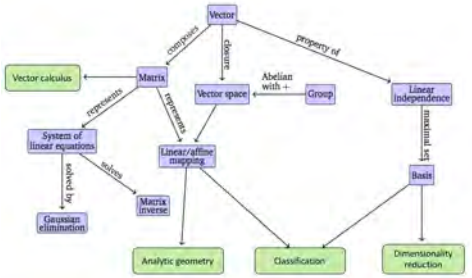
$$\boldsymbol{Ax} = \boldsymbol{b} \iff \boldsymbol{A}^\top\boldsymbol{Ax} = \boldsymbol{A}^\top\boldsymbol{b} \iff \boldsymbol{x} = (\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top\boldsymbol{b}$$

Vector spaces



- Groups: a set of elements and an operation defined on these elements that keeps some structure of the set intact, $(G,\otimes)$ $\otimes$: $G\times G \rightarrow G$
- Group properties: closure, associativity, neutral element, inverse element
- Groups: $(\mathbb{Z},+)$, $(\mathbb{R}\setminus\{0\},.)$, $(\mathbb{R}^n,+)$, $(\mathbb{Z}^n,+)$, $(\mathbb{R}^{m\times n},+)$ …
- A real-valued *vector space* $V=(\mathcal{V},+,.)$, $+:\mathcal{V}\times\mathcal{V}\rightarrow\mathcal{V}$, $.:\mathbb{R}\times\mathcal{V}\rightarrow\mathcal{V}$ that satisfy conditions …
- $V=(\mathcal{V},+,.)$ is a vector spaces, $\mathcal{U}\subseteq\mathcal{V}$, $\mathcal{U}\neq\emptyset$ then $U=(\mathcal{U},+,.)$ is *subspace* of *V* if $U=(\mathcal{U},+,.)$ is a vector space, and + and . restricted on $\mathcal{U}\times\mathcal{U}$ and $\mathbb{R}\times\mathcal{V}$.

Linear Independence

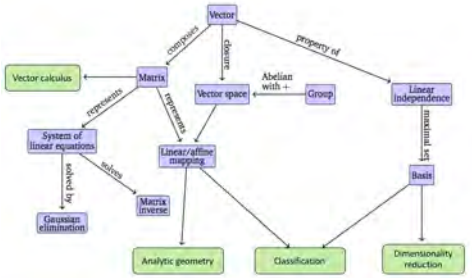


- $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k \in V$ and $\lambda_1, \lambda_2, \ldots, \lambda_k \in \mathbb{R}$, v is a *linear combination* of $\boldsymbol{x}_\mathrm{i}$

$$\boldsymbol{v} = \lambda_1 \boldsymbol{x}_1 + \cdots + \lambda_k \boldsymbol{x}_k = \sum_{i=1}^{k} \lambda_i \boldsymbol{x}_i \in V$$

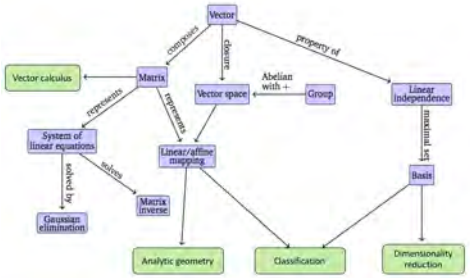
- Let us consider a vector space V with $k \in \mathbb{N}$ $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k \in V$. If there is a non-trivial linear combination, such that $\sum_{i=1}^{k} \lambda_i \boldsymbol{x}_i = 0$ with at least one $\lambda_i \neq 0$, the vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$ are ***linearly dependent***.
- If only the trivial solution exists, i.e., $\lambda_1, = \lambda_2 = \cdots = \lambda_k = 0$ the vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$ are ***linearly independent***.

Basis and Rank



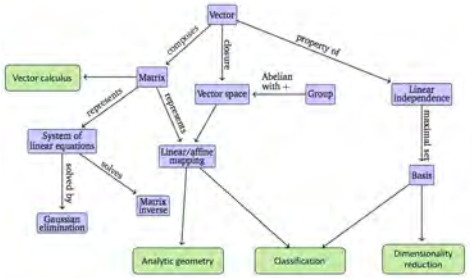
- Consider a vector space $V = (\mathcal{V}, +, .)$ and set of vectors $\mathcal{A} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k\} \subseteq \mathcal{V}$. If every vector $v \in V$ can be expressed as a linear combination of $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k, \mathcal{A}$ is called a ***generating set*** of V. The set of all linear combinations of vectors in $\mathcal{A}$ is called the *span* of $\mathcal{A}$.
- A vector space $V = (\mathcal{V}, +, .)$ and $\mathcal{A} \subseteq \mathcal{V}$. A generating set $\mathcal{A}$ of $\mathcal{V}$ is called *minimal* if there exists no smaller set $\tilde{\mathcal{A}} \subsetneq \mathcal{A} \subseteq \mathcal{V}$ that spans V. Every linearly independent generating set of V is minimal d is called a ***basis*** of V.
- Each vector of V is a unique linear combination of elements from a basis $\mathcal{B}$.

Basis and Rank



- Every vector space V possesses a basis B.
- There is no unique basis, however, all bases possess the same number of elements, the ***basis vectors***.
- The ***dimension*** of V is the number of basis vectors of V, denoted by $\dim(V)$. If $U \subseteq V$ is a subspace of V, then $\dim(U) \leq \dim(V)$.
- The number of linearly independent columns of a matrix $A \in \mathbb{R}^{m \times n}$ equals the number of linearly independent rows and is called the ***rank*** of A and is denoted by $rk(A)$.

Linear mappings

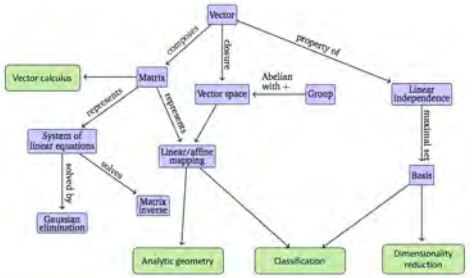


- Mappings on vector spaces that preserve their structure, which allow us to define the concept of a coordinate.
- For vector spaces V, W, a mapping $\Phi: V \to W$ is called a ***linear mapping*** (or *vector space homomorphism* or *linear transformation*) if

$$\forall x, y \in V, \forall \lambda, \psi \in \mathbb{R}: \Phi(\lambda x + \psi y) = \lambda\Phi(x) + \psi\Phi(y)$$

- $\Phi: \mathcal{V} \to \mathcal{W}$ is called
  - *Injective* if $\forall x, y \in V, \Phi(x) = \Phi(y) \Longrightarrow x = y$.
  - *Surjective* if $\Phi(\mathcal{V}) = \mathcal{W}$.
  - *Bijective* if it is injective and surjective.

Matrix Representation of Linear Mappings

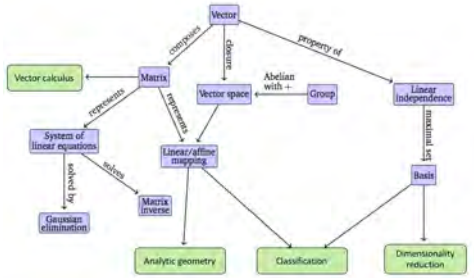


- Consider a n-dimensional vector space V and an ordered basis $B = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n\}$ of V. For any $x \in V$ we obtain a unique representation (linear combination) of x w.r.t B, $x = \alpha_1\boldsymbol{b}_1 + \cdots + \alpha_n\boldsymbol{b}_n$.
- We call $\alpha_1, \alpha_2, \ldots, \alpha_n$ are the *coordinates* of x w.r.t. B, and the vector

$$\alpha = \begin{bmatrix} \alpha_1 \\ \cdots \\ \alpha_n \end{bmatrix} \in \mathbb{R}^n$$

is the *coordinate vector/coordinate representation* of x with respect to the ordered basis B.

Transformation Matrix



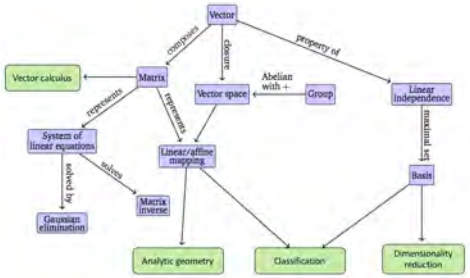
- Consider vector spaces V, W with corresponding (ordered) bases $B = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n\}$ and $C = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_m\}$) and a linear mapping $\Phi : V \rightarrow W$. For $j \in \{1, \ldots, n\}$,

$$\Phi(\boldsymbol{b}_j) = \alpha_{1j}\boldsymbol{c}_1 \ldots + \alpha_{mj}\boldsymbol{c}_m = \sum_{i=1}^{m} \alpha_{ij}\boldsymbol{c}_i$$

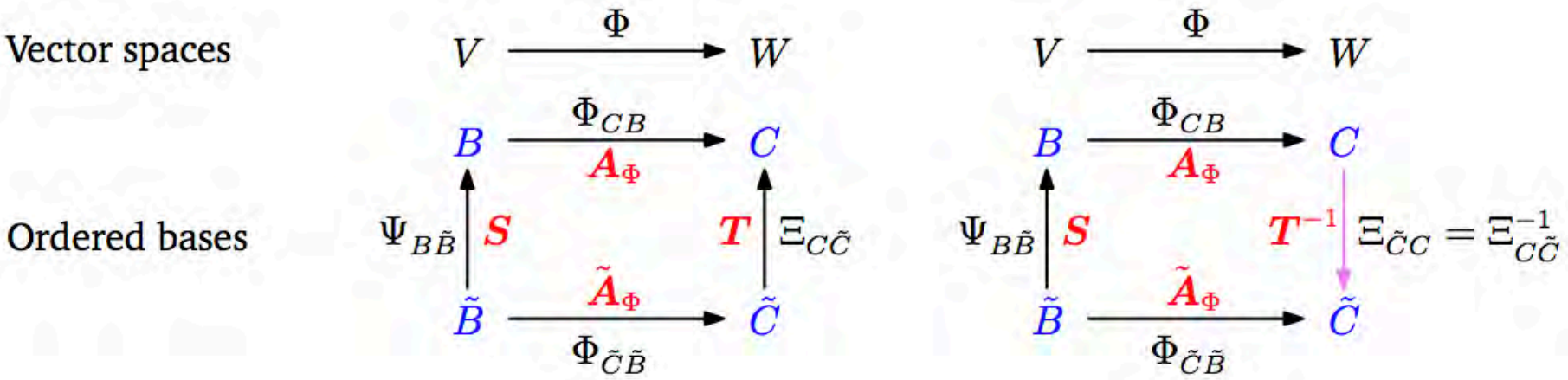
  is the unique representation of $\Phi(\boldsymbol{b}_j)$ w.r.t. C. We call the $m \times n$ -matrix A_Φ, whose elements are given by $A_\Phi(i,j) = \alpha_{ij}$, the *transformation matrix* of Φ (with respect to the ordered bases B of V and C of W).

- We have $\hat{\boldsymbol{y}} = A_\Phi \hat{\boldsymbol{x}}$

Basis change

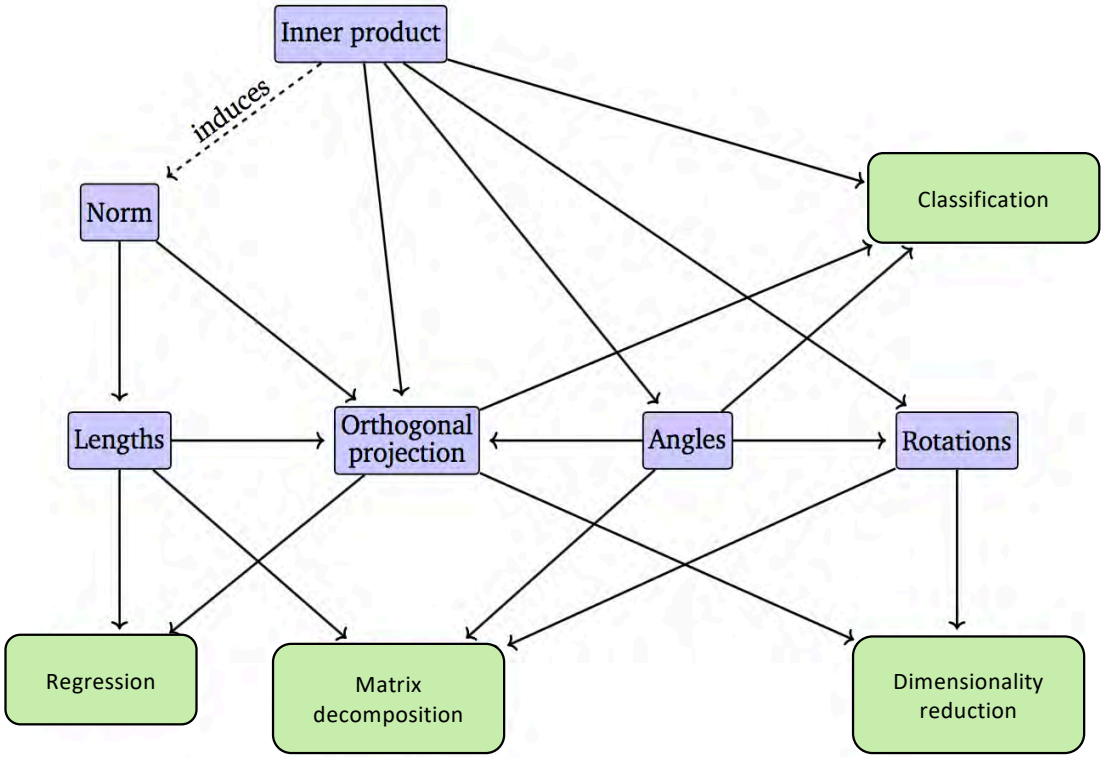


- How transformation matrices of a linear mapping $\Phi: V \to W$ change if we change the bases in V and W?



Analytic Geometry

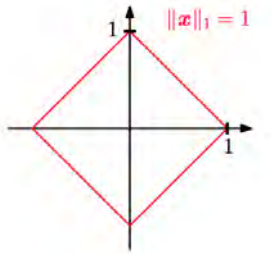
1. Norms
2. Inner Products
3. Lengths and Distances
4. Angles and Orthogonality
5. Orthonormal Basis
6. Orthogonal Complement
7. Inner Product of Functions
8. Orthogonal Projections
9. Rotations



Geometric interpretation and intuition to the concepts
of vectors, vector spaces, and linear mappings

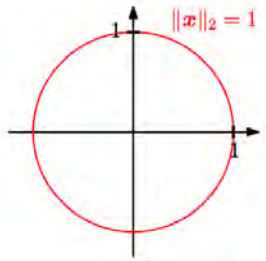
Norms and Inner product

- A ***norm*** on a vector space V is a function $\|.\| : V \to \mathbb{R}, x \mapsto \|x\|$ which assigns each vector x its *length* $\|x\| \in \mathbb{R}$
- ***Dot product*** (scalar product) $x^T y = \sum_{i=1}^{n} x_i y_i$
- Let V be a vector space and $\Omega: V \times V \to \mathbb{R}$ be a bilinear mapping that takes two vectors and maps them onto a real number. Then
  - A positive definite, symmetric bilinear mapping $\Omega: V \times V \to \mathbb{R}$ is called an ***inner product*** on V. We typically write $\langle x, y \rangle$.
  - The pair $(V, \langle \cdot, \cdot \rangle)$ is called an ***inner product space*** or (real) *vector space with inner product*. If we use the dot product, we call $(V, \langle \cdot, \cdot \rangle)$ a *Euclidean vector space*.



Manhattan norm l_1

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$



Euclidean norm l_2

$$\|x\|_2 = \sqrt[2]{\sum_{i=1}^{n} x_i^2}$$

Symmetric, Positive Definite Matrices

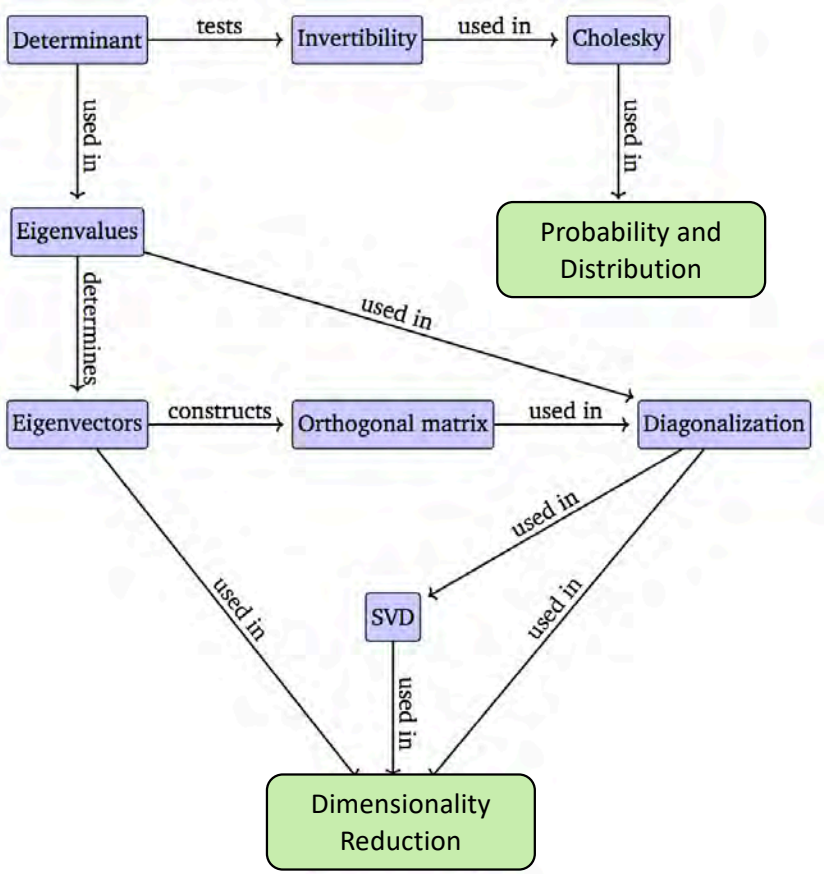
- A symmetric matrix $A \in \mathbb{R}^{n \times n}$ that satisfies $\forall x \in V \setminus \{0\}: x^T A x > 0$ $(*)$ is called ***symmetric, positive definite***, or just *positive definite*. If only (*) holds, then A is called *symmetric, positive semidefinite*.
- Consider an inner product space $(V, \langle \cdot, \cdot \rangle)$. Then the ***distance*** between $\boldsymbol{x}$ and $\boldsymbol{y}$ for $\boldsymbol{x}, \boldsymbol{y} \in V$ is defined as

$$d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\| = \sqrt{\langle \boldsymbol{x} - \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{y} \rangle}$$

- $\cos\omega = \frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{\|\boldsymbol{x}\| \, \|\boldsymbol{y}\|}$, ω is the ***angle*** between $\boldsymbol{x}$ and $\boldsymbol{y}$.
- Vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are ***orthogonal*** if and only if $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$, we write $\boldsymbol{x} \perp \boldsymbol{y}$.

Matrix Decompositions

1. Determinant and Trace
2. Eigenvalues and Eigenvectors
3. Cholesky Decomposition
4. Eigendecomposition and Diagonalization
5. Singular Value Decomposition
6. Matrix Approximation
7. Matrix Phylogeny



Also called Matrix Factorizations

Determinant and Trace

- The *determinant* of a square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is a function that maps $\boldsymbol{A}$ onto a real number.
- *Laplace Expansion*: $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, *for all* $j = 1, \ldots, n$

  1. Expansion along column j

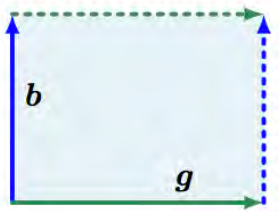
$$Det(\boldsymbol{A}) = \sum_{k=1}^{n} (-1)^{k+j} a_{kj} \det(\boldsymbol{A}_{k,j})$$

  2. Expansion along row j

$$Det(\boldsymbol{A}) = \sum_{k=1}^{n} (-1)^{k+j} a_{jk} \det(\boldsymbol{A}_{j,k})$$

Here $\boldsymbol{A}_{k,j} \in \mathbb{R}^{(n-1)\times(n-1)}$ is the submatrix of $\boldsymbol{A}$ that we obtain when deleting row k and column j.

$$\det(\boldsymbol{A}) = \begin{vmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{vmatrix}$$

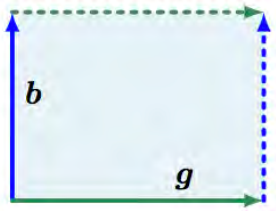


The area of the parallelogram (shaded region) spanned by the vectors **b** and **g** is |det([**b, g**])|.

Determinant and Trace

- Trace: $tr(A) = \sum_{i=1}^{n} a_{ii}$
- A square matrix $\boldsymbol{A} \in \mathbb{R}^{(n-1)\times(n-1)}$ has $\det(A) \neq 0$ if and only if $rk(A) = n$. In other words, $\boldsymbol{A}$ is invertible if and only if it is full rank.

$$\det(\boldsymbol{A}) = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix}$$



The area of the parallelogram (shaded region) spanned by the vectors ***b*** and ***g*** is |det([**b, g**])|.

Eigenvalues and Eigenvectors

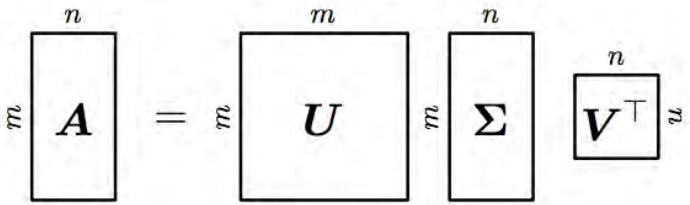
- Let $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ be a square matrix. Then $\lambda \in \mathbb{R}$ is an ***eigenvalue*** of A and $x \in \mathbb{R}^n \setminus \{0\}$ is the corresponding ***eigenvector*** of A if $\boldsymbol{Ax} = \lambda \boldsymbol{x}$.
- For $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, the set of all eigenvectors of $\boldsymbol{A}$ associated with an eigenvalue λ spans a subspace of $\mathbb{R}^n$, which is called the ***eigenspace*** of $\boldsymbol{A}$ with respect to λ and is denoted by E_λ.
- The eigenvectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ of a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ with n distinct eigenvalues $\lambda_1, \ldots, \lambda_n$ are linearly independent.
- **Spectral Theorem**. If $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is symmetric, there exists an orthonormal basis of the corresponding vector space V consisting of eigenvectors of A, and each eigenvalue is real.

Eigendecomposition and Diagonalization

- $\det(\boldsymbol{A}) = \prod_{i=1}^{n} \lambda_i$ and $tr(\boldsymbol{A}) = \sum_{i=1}^{n} \lambda_i$
- *Theorem* (Eigendecomposition). A square matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ can be factored into $\boldsymbol{A} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^{-1}$ where $\boldsymbol{P} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{D}$ is a diagonal matrix whose diagonal entries are the eigenvalues of $\boldsymbol{A}$, if and only if the eigenvectors of A form a basis of $\mathbb{R}^n$.

Eigendecomposition and Diagonalization

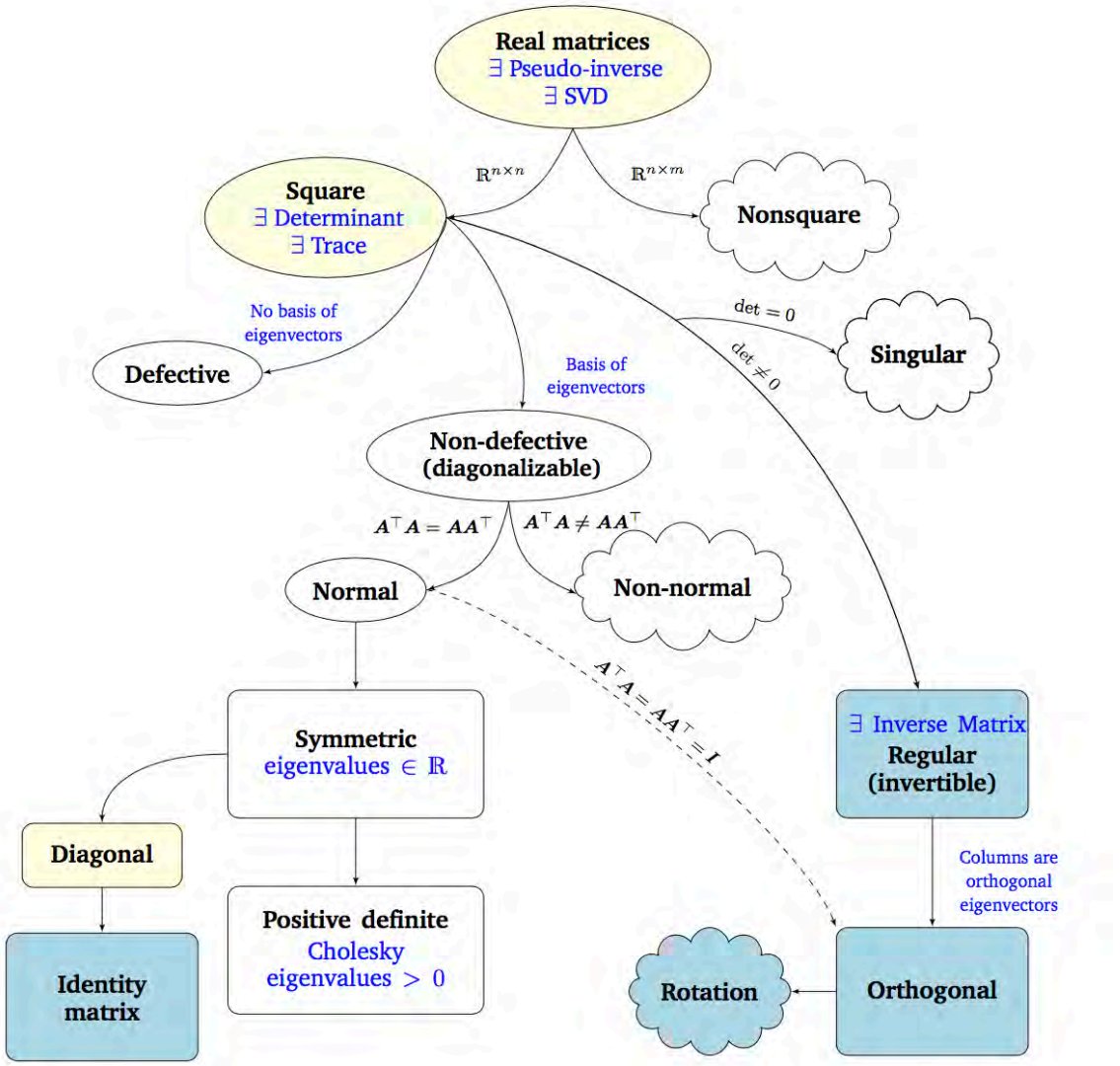
- *SVD Theorem*. Let $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ be a rectangular matrix of rank $r \in [0, \min(m, n)]$. The SVD of A is a decomposition of the form



with an orthogonal matrix $\boldsymbol{U} \in \mathbb{R}^{m \times m}$ with column vectors $u_i, i = 1, \ldots, m$, and an orthogonal matrix $\boldsymbol{V} \in \mathbb{R}^{n \times n}$ with column vectors $v_i, j = 1, \ldots, n$. Moreover, Σ is an $m \times n$ matrix with $\Sigma_{ii} = \sigma_i \geq 0, \Sigma_{ij} = 0, i \neq j$

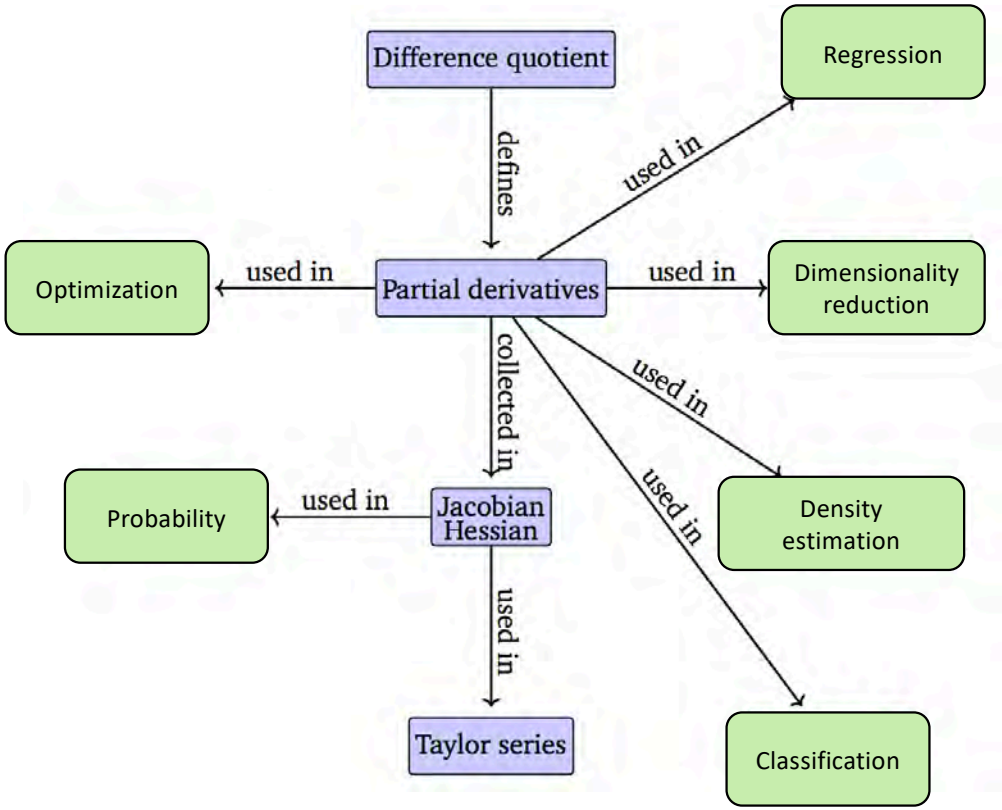
A functional phylogeny of matrices encountered in machine learning.

Phát sinh chức năng của ma trận gặp trong học máy.



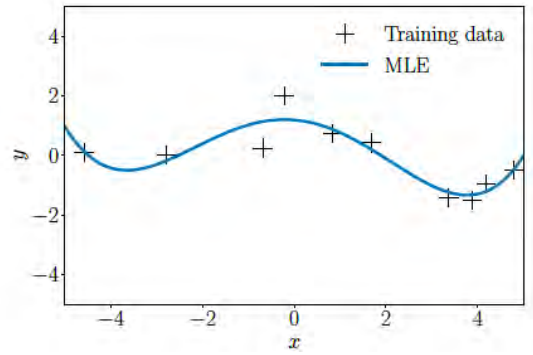
Vector calculus

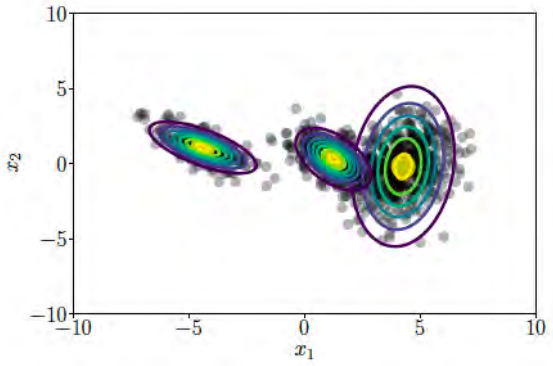
1. Differentiation of Univariate Functions
2. Partial Differentiation and Gradients
3. Gradients of Vector-Valued Functions
4. Gradients of Matrices
5. Useful Identities for Computing Gradients
6. Backpropagation and Automatic Differentiation
7. Higher-Order Derivatives
8. Linearization and Multivariate Taylor Series



Vector calculus

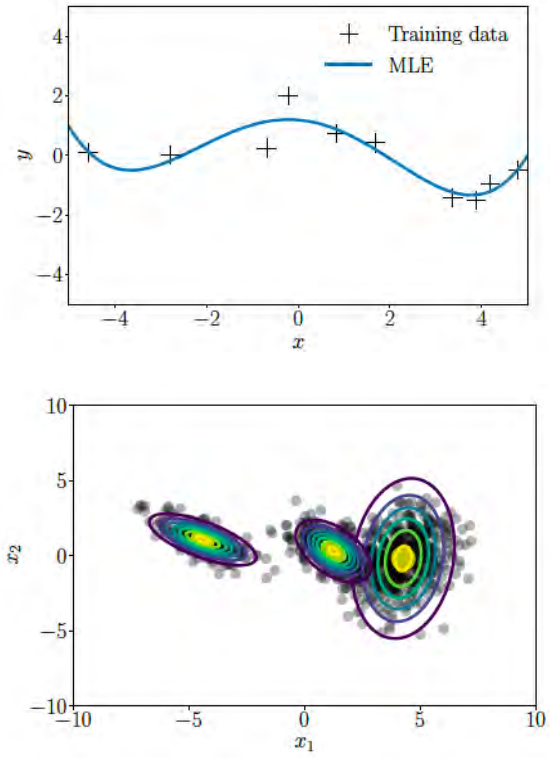
- Many algorithms in machine learning optimize an objective function w. r. t. a set of desired model parameters that control how well a model explains the data: Finding good parameters can be phrased as an optimization problem.
- Linear regression: we look at curve-fitting problems and optimize linear weight parameters to maximize the likelihood;
- Auto-encoders for dimensionality reduction: parameters are the weights and biases of each layer, and where we minimize a reconstruction error by repeated application of the chain rule
- Gaussian mixture models for modeling data distributions, where we optimize the location and shape parameters of each mixture component to maximize the likelihood of the model.





Vector calculus

- Nhiều thuật toán trong học máy tối ưu hóa một hàm mục tiêu gắn với các tham số mô hình để kiểm soát mức độ giải thích dữ liệu của mô hình: Việc tìm các tham số tốt là một bài toán tối ưu hóa.
  - Trong hồi quy tuyến tính: ta tối ưu hóa các tham số trọng số tuyến tính để tối đa hóa khả năng xảy ra;
  - Bộ mã hóa tự động để giảm kích thước: các tham số là trọng lượng và độ lệch của mỗi lớp và nơi chúng tôi giảm thiểu lỗi xây dựng lại bằng cách áp dụng lặp lại quy tắc chuỗi
  - Mô hình hỗn hợp Gaussian để lập mô hình phân phối dữ liệu, trong đó chúng tôi tối ưu hóa các thông số về vị trí và hình dạng của từng thành phần hỗn hợp để tối đa hóa khả năng của mô hình.



Differentiation of Univariate Functions

- **Difference quotient** computes the slope of the secant line through two points on the graph of f

$$\frac{\delta y}{\delta x} = \frac{f(x + \delta x) - f(x)}{\delta x}$$

- **Derivative** of f at x is define as the limit, for $h > 0$

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Partial Differentiation and Gradients

- **Partial Derivative**: For a function $f: \mathbb{R}^n \to \mathbb{R}, x \mapsto f(x), x \in \mathbb{R}^n$ of n variables $x_1, \ldots, x_n$

$$\frac{\partial f}{\partial x_1} = \lim_{h \to 0} \frac{f(x_1 + h, x_2, \ldots, x_n) - f(\boldsymbol{x})}{h}$$

$$\vdots$$

$$\frac{\partial f}{\partial x_n} = \lim_{h \to 0} \frac{f(x_1, \ldots, x_{n-1}, x_n + h) - f(\boldsymbol{x})}{h}$$

- The **gradient** of f (Jacobian)

$$\nabla_{\boldsymbol{x}} f = \operatorname{grad} f = \frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{x}} = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1} & \frac{\partial f(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial f(\boldsymbol{x})}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{1 \times n}$$

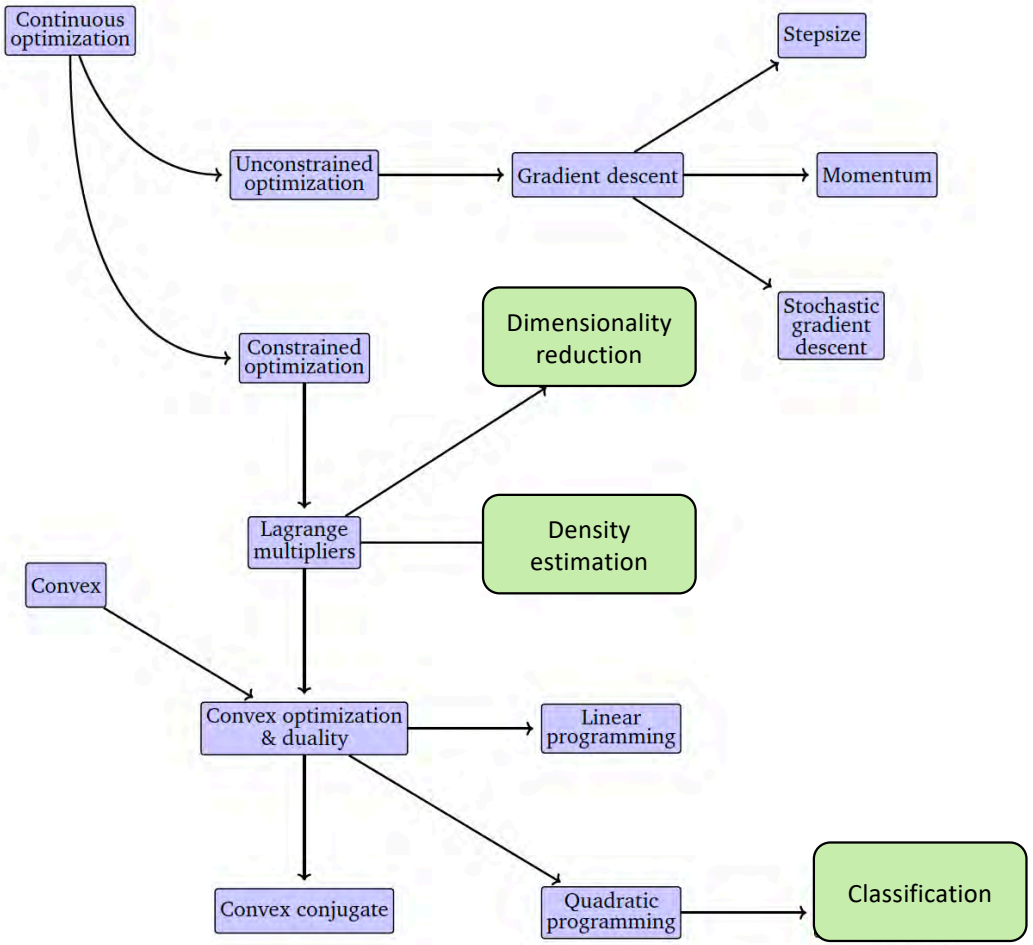
Gradients of Vector-Valued Functions

- Consider function $f: \mathbb{R}^n \to \mathbb{R}^m$, vector $\boldsymbol{x} = [x_1, \ldots, x_n]^T \in \mathbb{R}^n$ we can write $\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_m(\boldsymbol{x}) \end{bmatrix} \in \mathbb{R}^m$ and f as $[f_1, \ldots, f_m]^T$, $f_i: \mathbb{R}^n \to \mathbb{R}$
- the partial derivative of a vector-valued function $f: \mathbb{R}^n \to \mathbb{R}^m$ is

$$\frac{\partial \boldsymbol{f}}{\partial x_i} = \begin{bmatrix} \frac{\partial f_1}{\partial x_i} \\ \vdots \\ \frac{\partial f_m}{\partial x_i} \end{bmatrix} = \begin{bmatrix} \lim_{h \to 0} \frac{f_1(x_1, \ldots, x_{i-1}, x_i + h, x_{i+1}, \ldots x_n) - f_1(\boldsymbol{x})}{h} \\ \vdots \\ \lim_{h \to 0} \frac{f_m(x_1, \ldots, x_{i-1}, x_i + h, x_{i+1}, \ldots x_n) - f_m(\boldsymbol{x})}{h} \end{bmatrix} \in \mathbb{R}^m .$$

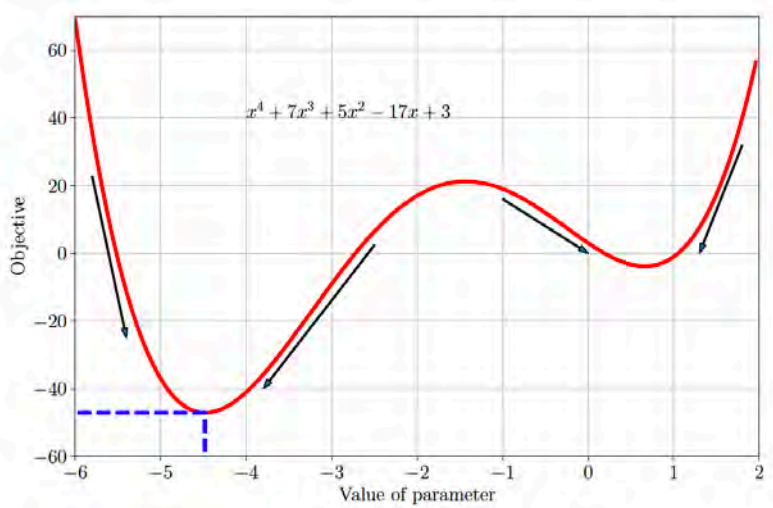
Continous optimization

- Optimization Using Gradient Descent
- Constrained Optimization and Lagrange Multipliers
- Convex Optimization



Optimization Using Gradient Descent

- Solving for the minimum of a real-valued function $\min_x f(x), f\colon \mathbb{R}^d \to \mathbb{R}$ and assume that f is differentiable.
- **Gradient descent** is a first-order optimization algorithm to find a local minimum of a function: takes steps proportional to the negative of the gradient of the function at the current point.
- Gradient descent exploits the fact that $f(x_0)$ decreases fastest if one moves from x_0 in the direction of the negative gradient $-((\nabla f)(x_0))^T$



Example of objective function. Negative gradients are indicated by arrows, and the global minimum is indicated by the dashed blue line.

Stochastic Gradient Descent

- In machine learning, given $n = 1, \ldots, N$ data points, we often consider objective functions that are the sum of the losses L_n incurred by each. example n. In mathematical notation, we have the form

$$L(\theta) = \sum_{i=1}^{N} L_n(\theta)$$

where θ is the vector of parameters of interest, i.e., we want to find that minimizes L. For example, in regression it is the sum over log-likelihoods of individual

$$\mathrm{L}(\theta) = \sum_{i=1}^{\mathrm{N}} \log p\,(y_n | x_n, \theta)|$$

Constrained Optimization and Lagrange Multipliers

- $f: \mathbb{R}^D \to \mathbb{R}, g_i: \mathbb{R}^D \to \mathbb{R}, i = 1, \ldots, m$. Consider the constrained optimization problem (primal problem)

$$\min_x f(x)$$
$$\text{subject to} \quad g_i(x) \leq 0, \quad \text{for all } i = 1, \ldots, m$$

- We associate to problem the Lagrangian by introducing the Lagrange multipliers $\lambda_i > 0$ corresponding to each inequality constraint respectively

$$\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(\boldsymbol{x})$$
$$= f(\boldsymbol{x}) + \boldsymbol{\lambda}^T g(\boldsymbol{x})$$

Constrained Optimization and Lagrange Multipliers

- The associated **Lagrangian dual problem** is given by

The associated Lagrangian dual problem is given by

$$\text{subject to } \lambda \geq 0$$

where are the dual variables and $\mathfrak{D}(\lambda) = \min_{x \in \not\approx \mathbb{R}^d} \mathfrak{L}(x, \lambda)$

Convex Optimization

- A set $\mathcal{C}$ is a **convex set** if for any $x, y \in \mathcal{C}$ and for any scalar θ with $0 \leq \theta \leq 1$ we have

$$\theta x + (1 - \theta) y \in \mathcal{C}$$

- Let function $f: \mathbb{R}^D \rightarrow \mathbb{R}$ be a function whose domain is a convex set. The function f is a **convex function** if for all x, y in the domain convex function of f, and for any scalar θ with $0 \leq \theta \leq 1$, we have

$$f(\theta x + (1 - \theta) y) \leq \theta f(x) + (1 - \theta) f(y)$$

Convex Optimization

- A constrained optimization problem is called a convex optimization problem if

$$\min_x f(x)$$

$$\text{subject to } g_i(x) \leq 0 \text{ for all } i = 1, \ldots, m$$

$$h_j(x) = 0 \text{ for all } j = 1, \ldots, n$$

  where all functions $f(x)$ and $g_i(x)$ are convex functions, and all $h_j(x) = 0$ are convex sets.

- Two classes of convex optimization problems: Linear programming and quadratic programming.

Math for Machine Learning

Mathematical Foundations

1. Linear Algebra
2. Analytic Geometry
3. Matrix Decompositions
4. Vector Calculus
5. Probability and Distributions
6. Continuous Optimization

Central Machine Learning Problems

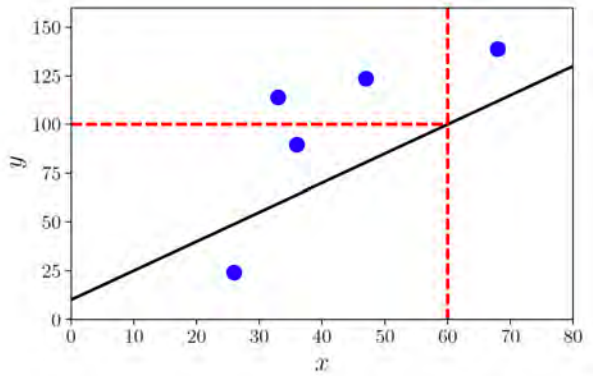
1. When Models Meet Data
2. Linear Regression
3. Dimensionality
4. Density Estimation with Gaussian Mixture Models
5. Classification with Support Vector Machines

When Models Meet Data: Data, Models, and Learning

- Major components of a machine learning system: data, models, and learning.
- The main question of machine learning is "What do we mean by good models?"
- One of the guiding principles of machine learning is that good models should perform well on unseen data.
- Structured data (tabular, vector representation) vs Unstructured data
- *Models as functions* ($f(\boldsymbol{x}) = \theta^T \boldsymbol{x} + \theta_0$) vs. *Model as probability distributions* (i.e., models describing the distribution of possible functions)

Models as Functions vs. Models a Probability Distribution

- Two major approaches in this book: a model as a function, and a model as a probabili distribution.
- Assume data are in vector representation and a predictor $f: \mathbb{R}^D \rightarrow \mathbb{R}$
- In case of linear function $f(x) = \theta^T x + \theta_0$ for unknown θ^T and θ_0.
- We could consider predictors to be probabilistic models, i.e., models describing the distribution of possible functions.



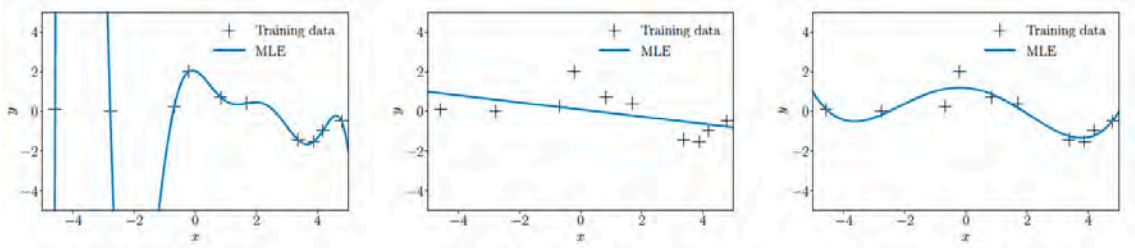
Example function (black solid diagonal line) and its prediction at. x = 60, i.e., f(60) = 100.

Learning is Finding Parameters

- The goal of learning is to *find a model* and its corresponding *parameters* such that the resulting predictor will perform well on unseen data. There are three distinct algorithmic phases when discussing machine learning algorithms:
  1. Prediction or inference
  2. Training or parameter estimation
  3. Hyperparameter tuning or model selection
- For the non-probabilistic model, we follow the principle of empirical risk minimization.
- With a statistical model, the principle of maximum likelihood is used maximum likelihood to find a good set of parameters

Parameter Estimation

- Maximum Likelihood Estimation (MLE)
  - MLE is to define a function of the parameters that enables us to find a model that fits the data well, $\mathcal{L}_x(\theta) = -\log p\,(x|\theta)$
- Maximum A Posteriori Estimation
  - $p(\theta|x) = \dfrac{p(x|\theta)p(\theta)}{p(x)}$
- Model Fitting



(a) Overfitting (b) Underfitting. (c) Fitting well.

Machine learning tribes

Tribes	Origins	Master Algorithm
Symbolists	Logic, philosophy	*Inverse deduction*
Evolutionary	Evolutionary biology	*Genetic programming*
Connectionists	Neuroscience	*Backpropagation*
Probabilistic	Statistics	*Bayes' theorem*
Analogizers	Psychology	*Support Vector Machines*

27 March 2019: Turing award 2019

Tom Mitchell | Steve Muggleton | Ross Quinlan

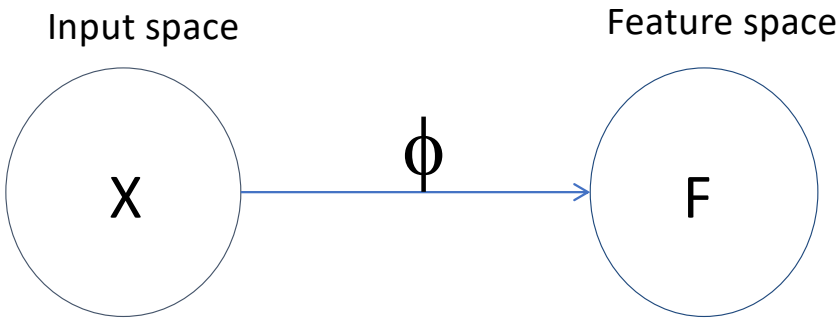
Geoff Hinton | Yann LeCun | Yoshua Bengio

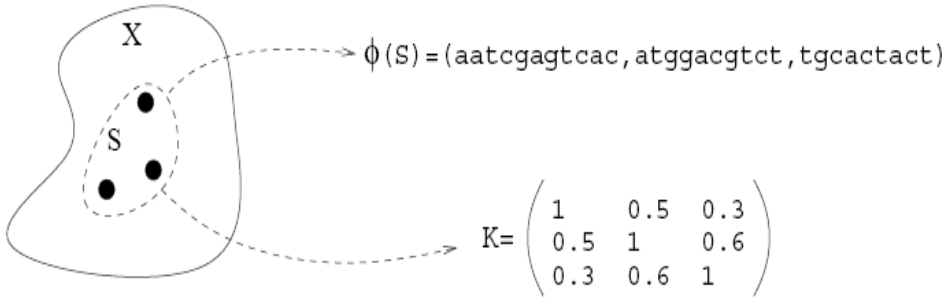
Vladimir Vapnik | Peter Hart | Douglas Hofstadter

John Koza | John Holland | Hod Lipson

David Heckerman | Judea Pearl | Michael Jordan

Data transformation





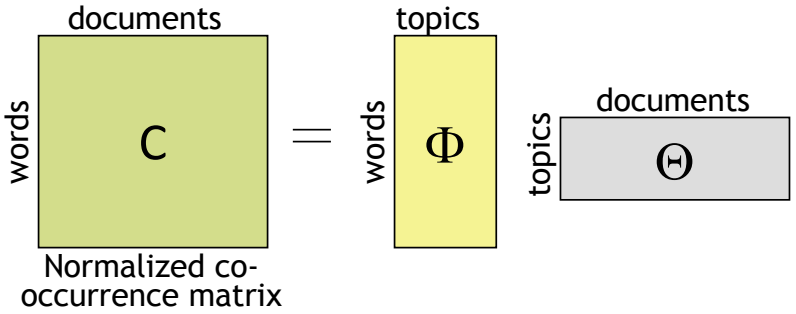
ϕ: X → F where the problem can be solved in F

X is the set of all oligonucleotides, S consists of three oligonucleotides, and S is represented in F as a matrix of pairwise similarity between its elements.

Topic models

The key ideas

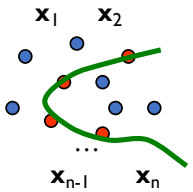
Topic models



- **Key idea**: documents are mixtures of latent topics, where a topic is a probability distribution over words.
- Hidden variables, generative processes, and statistical inference are the foundation of probabilistic modeling of topics.

Kernel methods

Input space X



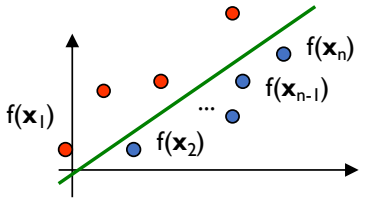
inverse map f^{-1}

$f(\mathbf{x})$

$k(\mathbf{x}_i,\mathbf{x}_j) = f(\mathbf{x}_i) \cdot f(\mathbf{x}_j)$

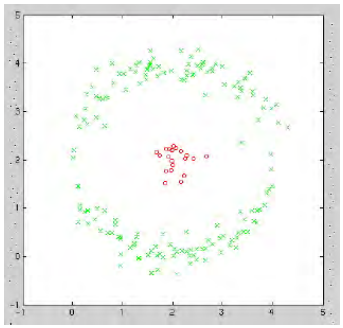
Kernel matrix K_{nxn}

Feature space F

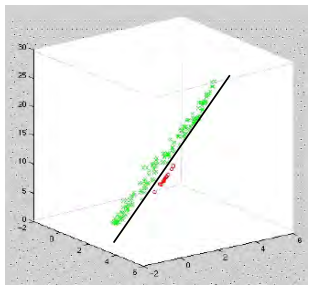


kernel function k: $X \times X \rightarrow R$

kernel-based algorithm on K
(computation done on kernel matrix)



$$\phi : \mathcal{X} = \mathcal{R}^2 \rightarrow \mathcal{H} = \mathcal{R}^3$$
$$(x_1, x_2) \mapsto (x_1, x_2, x_1^2 + x_2^2)$$



Regularized regression

- least squares estimator minimizes the training error

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - \beta^T X_i)^2$$

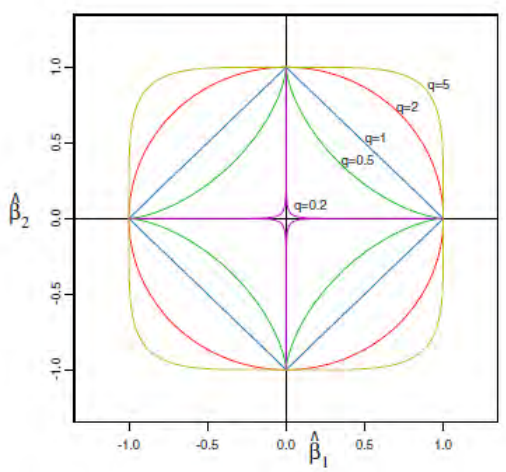
- Instead, we can minimize the penalized training error:

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - \beta^T X_i)^{\mathrm{q}} + \lambda\|\beta\|_{\mathrm{q}}$$

where $\|\beta\|_{\mathrm{q}} = \left(\sum_{\mathrm{j}} \beta_{\mathrm{j}}^{\mathrm{q}}\right)^{1/\mathrm{q}}$

- The solution is:

$$\hat{\beta} = (\mathbb{X}^T\mathbb{X} + \lambda I)^{-1}\mathbb{X}^T\mathbb{Y}$$



- ***q* = 2**: *ridge regression*.
- ***q* ≈ 0** penalty function places all its mass along the coordinate axes
- ***q* = 1** produces the lasso method having a diamond-shaped penalty function