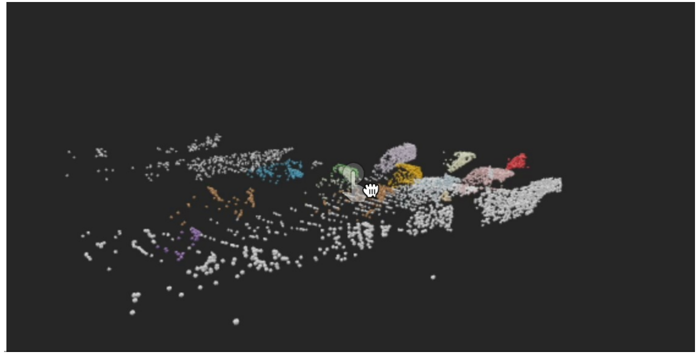
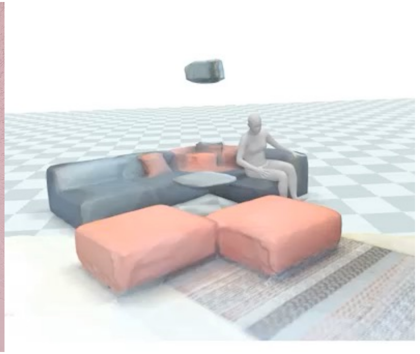
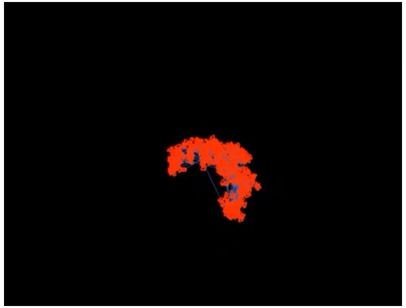
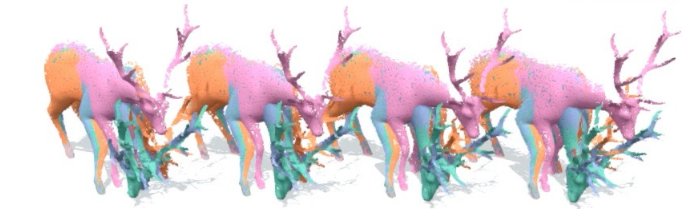
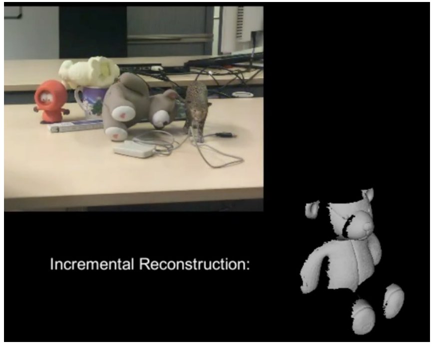
Mathematics for Machine Learning -- An Introduction

Tolga Birdal

Computer Vision & Geometric Machine Learning Group @ Imperial College London



TB

Bana yapay zekayi tarif eder misin?

Yapay Zeka, insan düşüncesi ve öğrenme gibi fonksiyonları taklit eden veya geçen makine öğrenmesi ve benzeri teknolojilerle programlanan bir sistemdir. Yapay Zeka, verileri analiz etme, karar verme, öğrenme ve problem çözme gibi görevleri yerine getirme kapasitesine sahiptir.

TB

Peki yapay zeka ve yapay öğrenmenin farkı nedir?

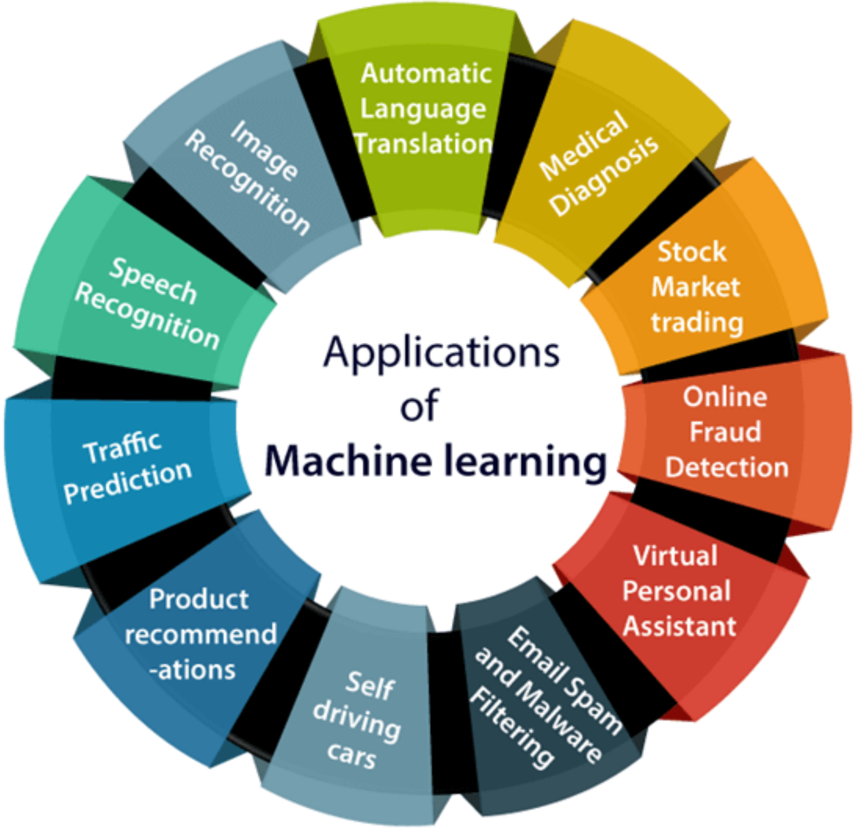
Yapay Zeka (AI) ve Yapay Öğrenme (ML), birbirine benzer ama aynı şey değildir. Yapay Zeka, insan düşüncesi ve öğrenme gibi fonksiyonları taklit eden bir sistemi ifade eder. Bununla birlikte, Yapay Öğrenme, verilerden model oluşturma ve bu modele göre tahmin yapma yeteneğini ifade eder. Yapay Zeka, birçok farklı teknolojiyi içerebilir ve Yapay Öğrenme, Yapay Zeka içindeki bir alt teknolojidir. Özet olarak, Yapay Zeka makinelere insana benzeyen davranışlar ve fonksiyonlar kazandırmayı amaçlar, Yapay Öğrenme ise verilerden model oluşturma ve tahmin yapma yeteneğini sağlar.

On the possibility of AI

- **P1**: Any function that can be computed by a mechanical procedure can be computed by a Turing machine. (Church-Turing thesis)
- **P2**: Thinking is nothing more than the computing of functions by mechanical procedures (i.e., thinking is symbol manipulation). (Functionalist-Computationalist thesis)
- **C1**: Therefore, thinking can be performed by a Turing machine.
- **P3**: Turing machines are multiply realizable. In particular, they can be realized by computers, robots, etc.
- $\therefore$

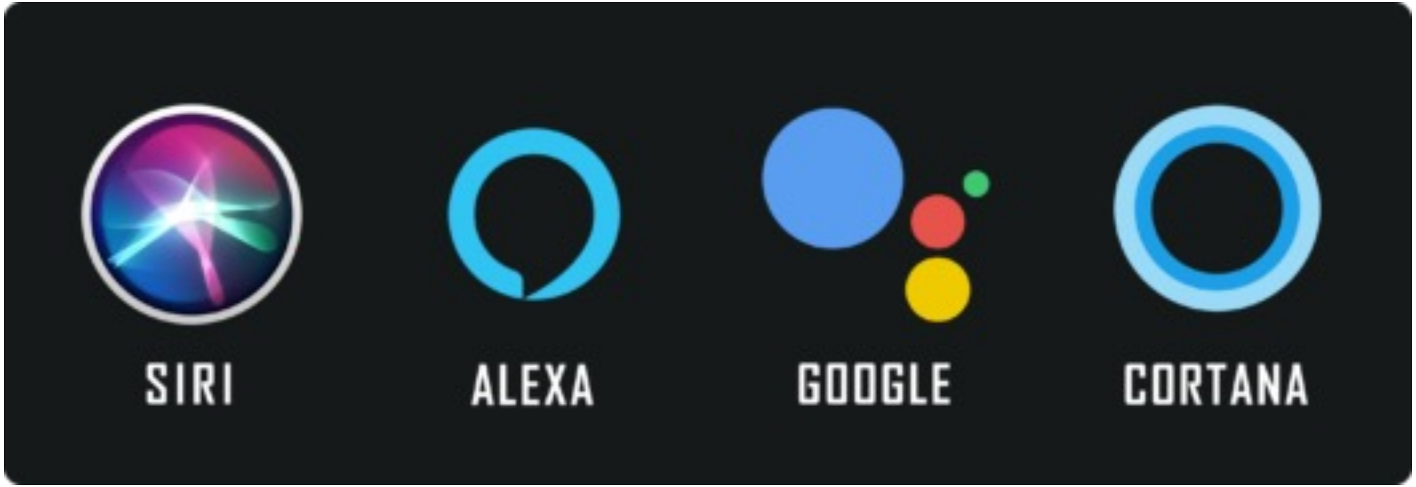
  It is possible to build a computer, robot, etc. that can think.
  → That is, AI is possible.

Applications of Machine Learning

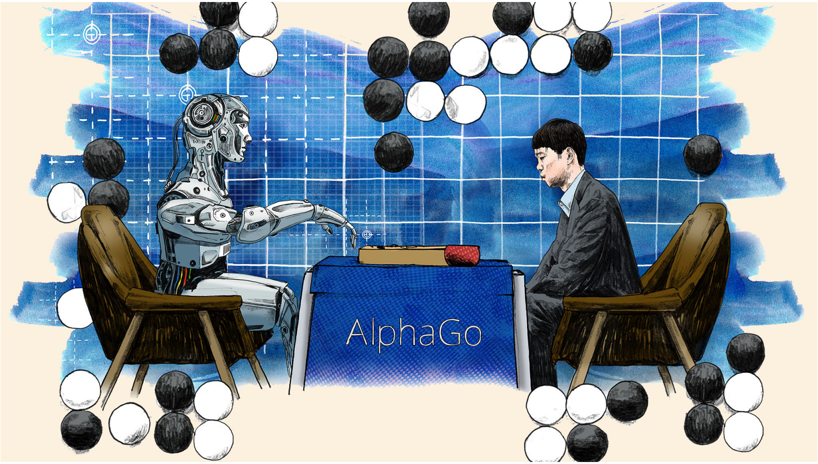


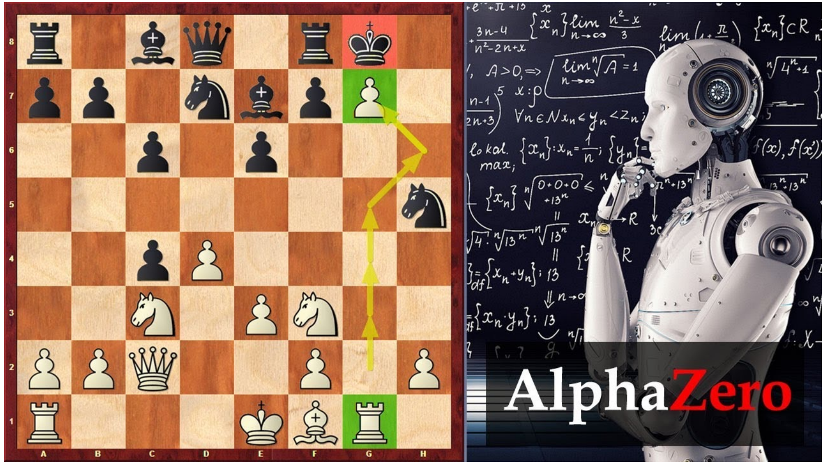
Recommendation Systems

Virtual Personal Assistants

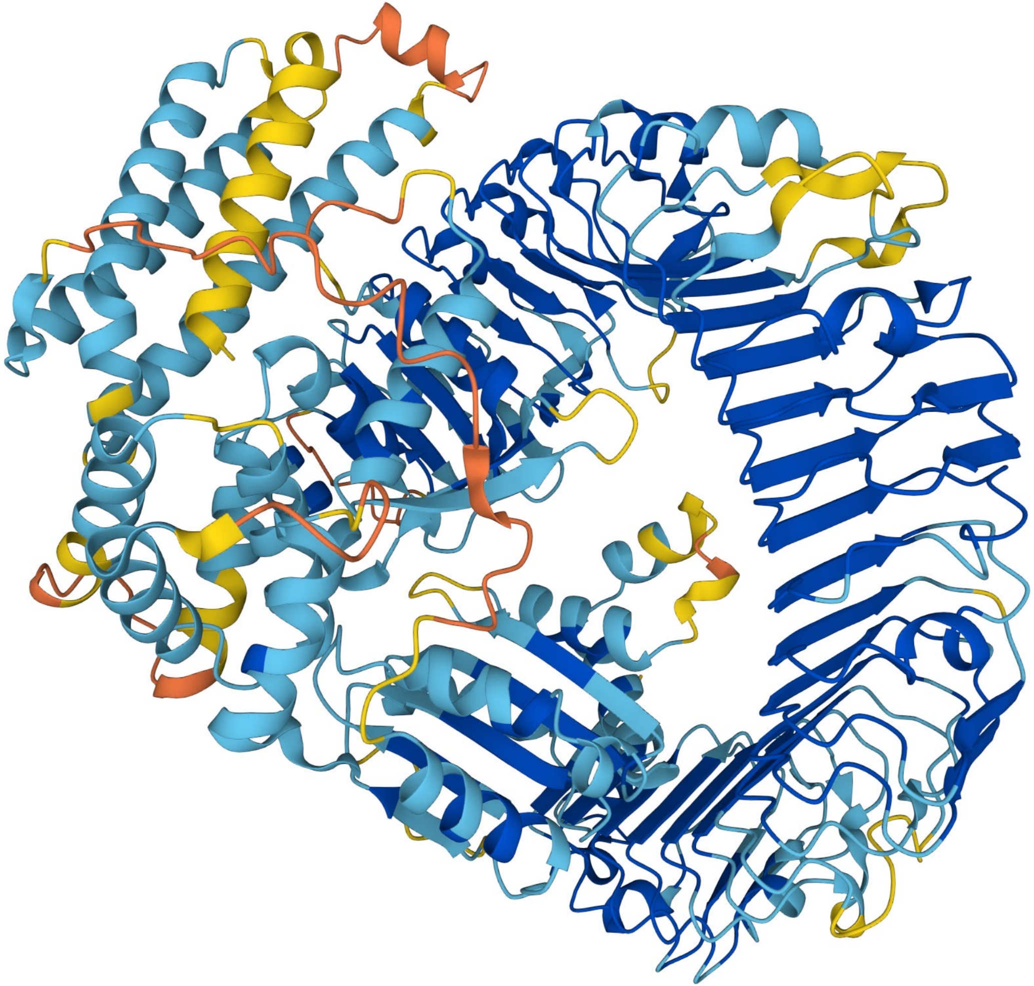


Playing Games





Protein Folding (Alpha Fold)



Self Driving Cars

Generative Art (NFTs)

Generative Art

Applications of Machine Learning

Harmonai

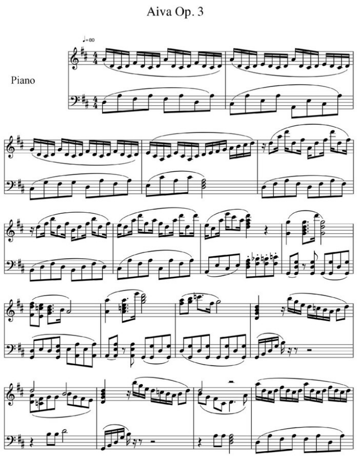
Infinite
Bass
Solo

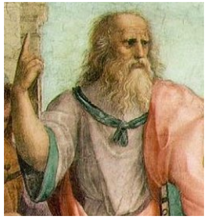
Applications of Machine Learning

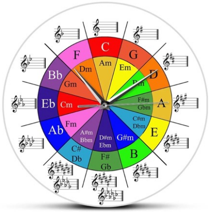
Video
Clip
Synthesis

Applications of Machine Learning

By **AIVA** in 2016

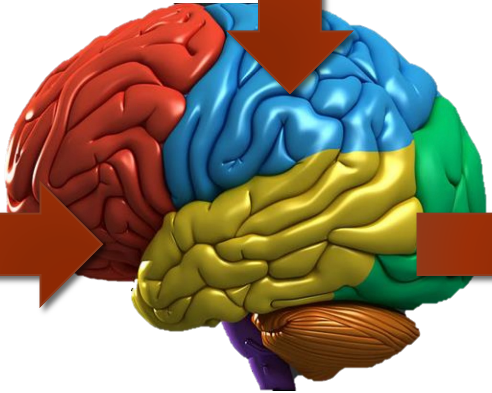
Recognized by music society!

Ethics
Copyrights
Credits / Licensing
and all that fuss…



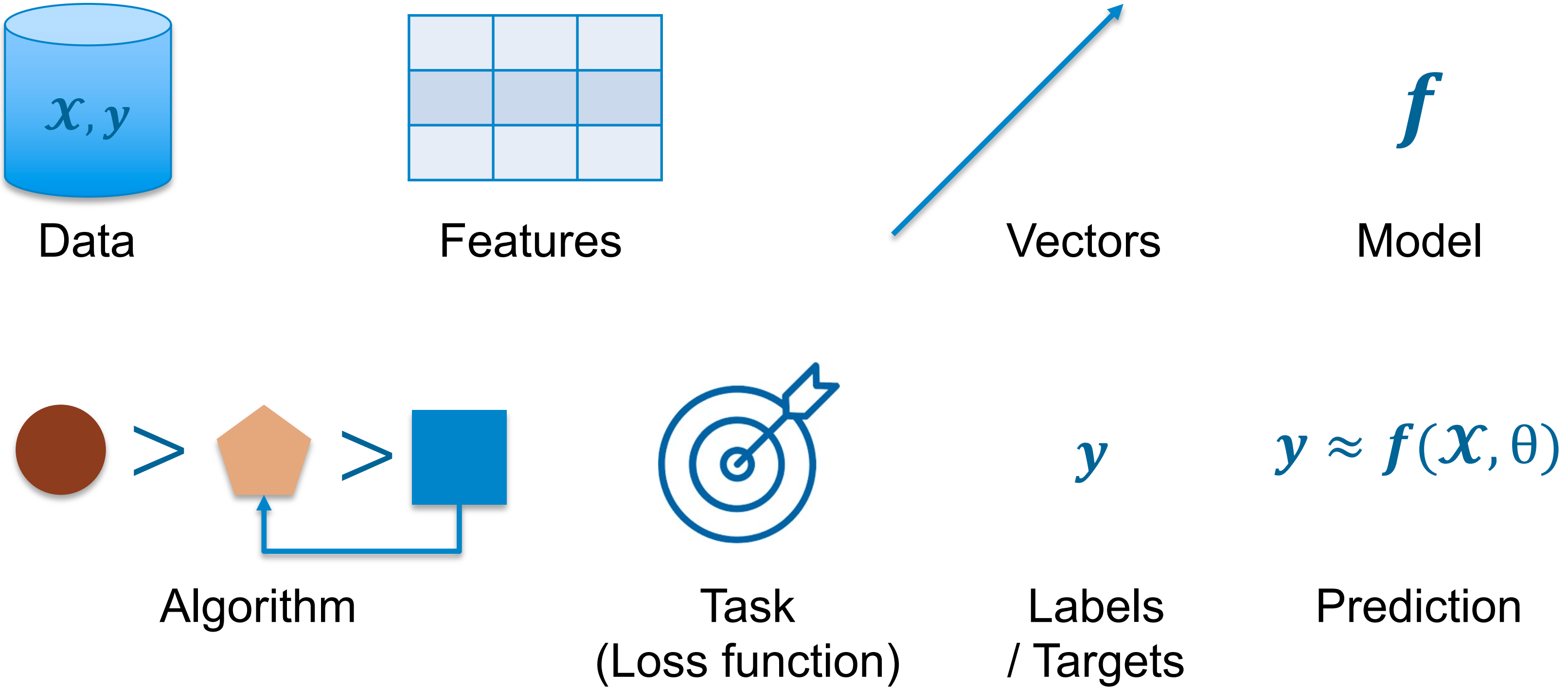


Large Corpus of Recorded Music

Meta Data

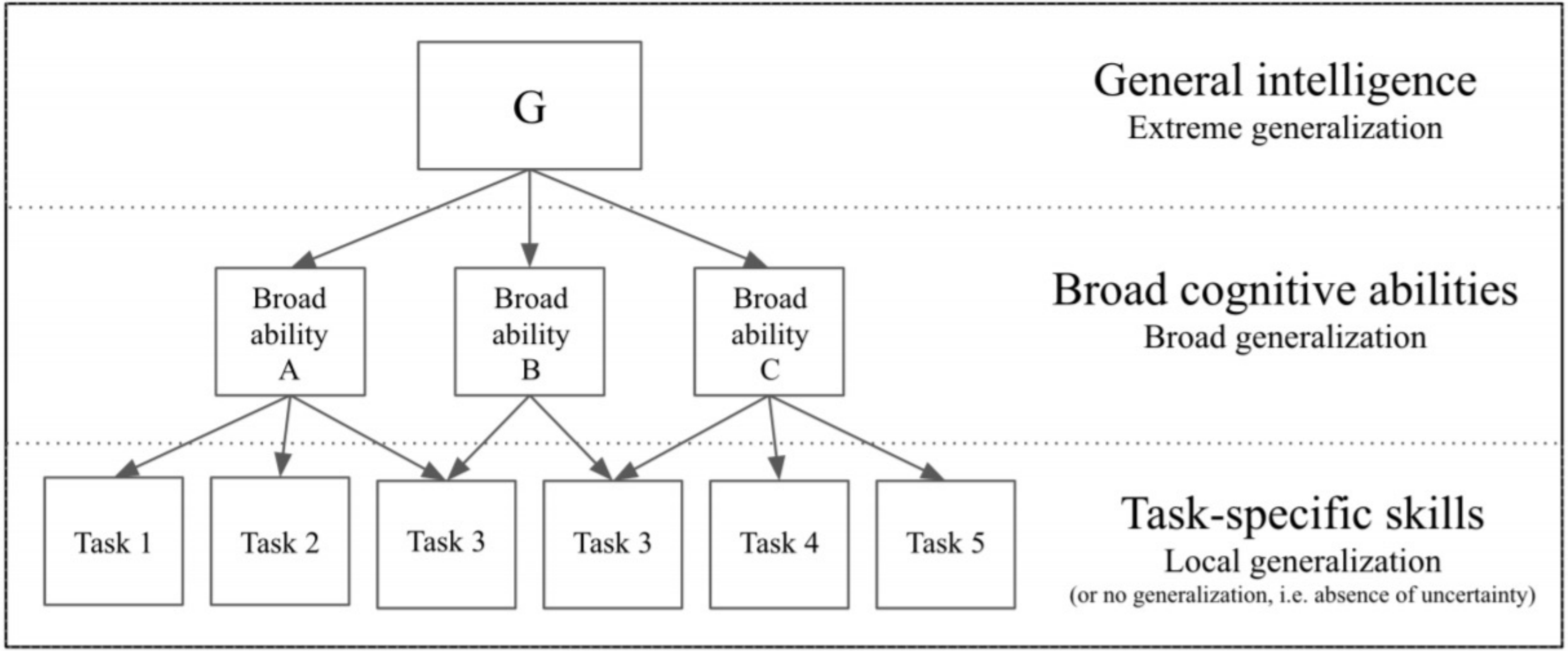


Machine Learning: A Glossary



"... You insist that there is something a machine cannot do. If you will tell me precisely what it is that a machine cannot do, then I can always make a machine which will do just that."

-- von Neumann



Chollet, François. "On the measure of intelligence." *arXiv preprint arXiv:1911.01547* (2019).

Machine Learning: Prediction vs Inference

Philosophically:

a prediction is an **educated guess** that can be **confirmed or denied**,
an inference is more concerned with the **implicit**.

In Statistics:

a prediction is an **educated guess** that can be **confirmed or denied**,
an inference is more concerned with the **implicit**.

In Twisted Machine Learning Terms:

prediction is a specific type of inference that forecasts future outcomes,
inference involves conclusion-drawing tasks based on data and evidence.

“Learning”

Herbert Simon:
“Learning denotes changes in a system that ... enable a system to do the same task … more efficiently the next time.”

Ryszard Michalski:
“Learning is constructing or modifying representations of what is being experienced.”

Marvin Minsky:
“Learning is making useful changes in our minds.”

Machine Learning

Wikipedia: The study of computer algorithms that improve automatically through ***experience***.

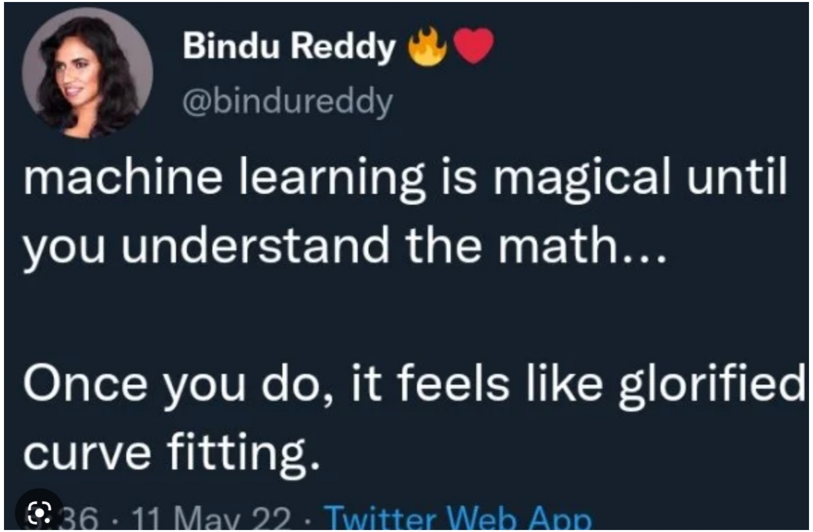
Cambridge: The process of computers changing the way they carry out tasks by ***learning*** from new data, without a human being needing to give instructions in the form of a program.

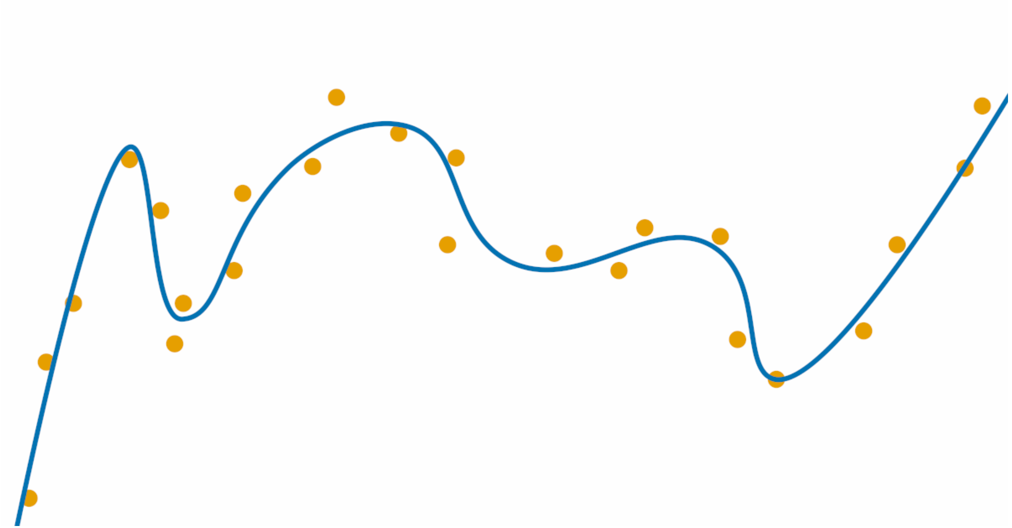
Others:

- Automating automation.
- Getting computers to program themselves.
- Writing software is the bottleneck, let the data do the work instead.

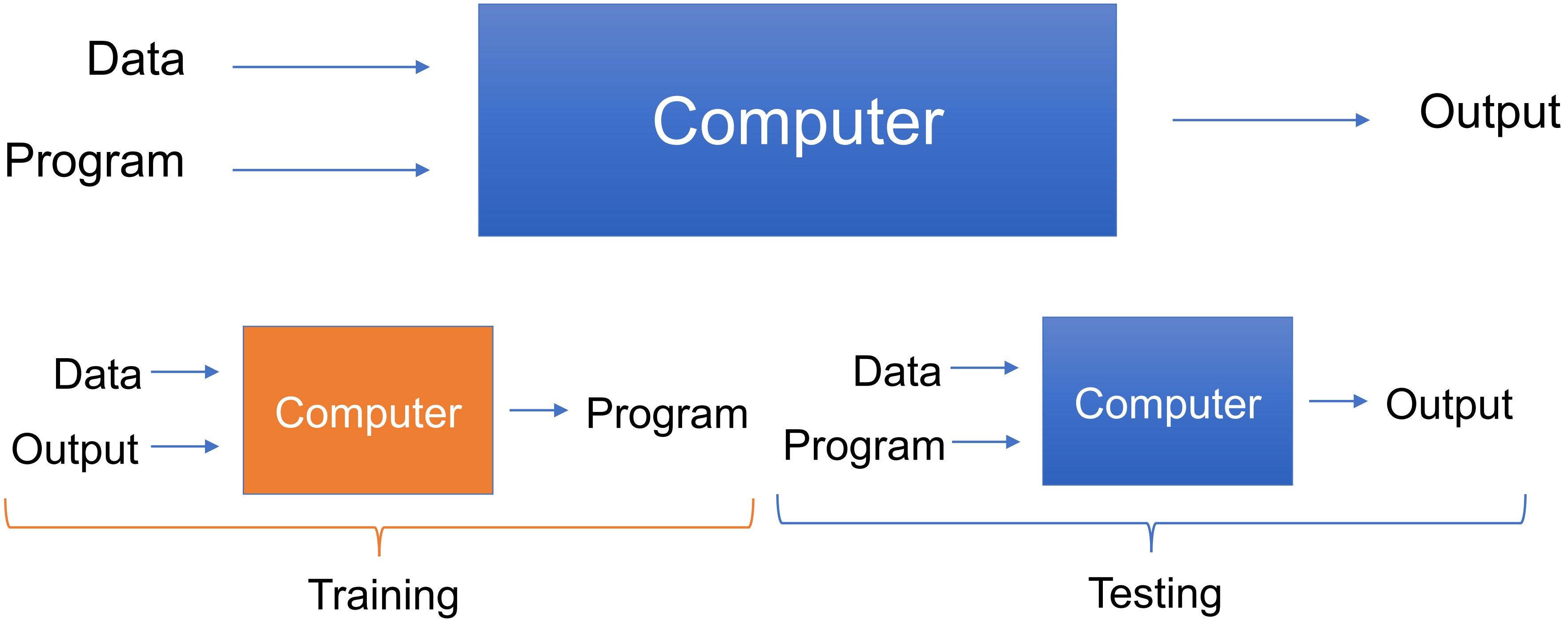
After this course:

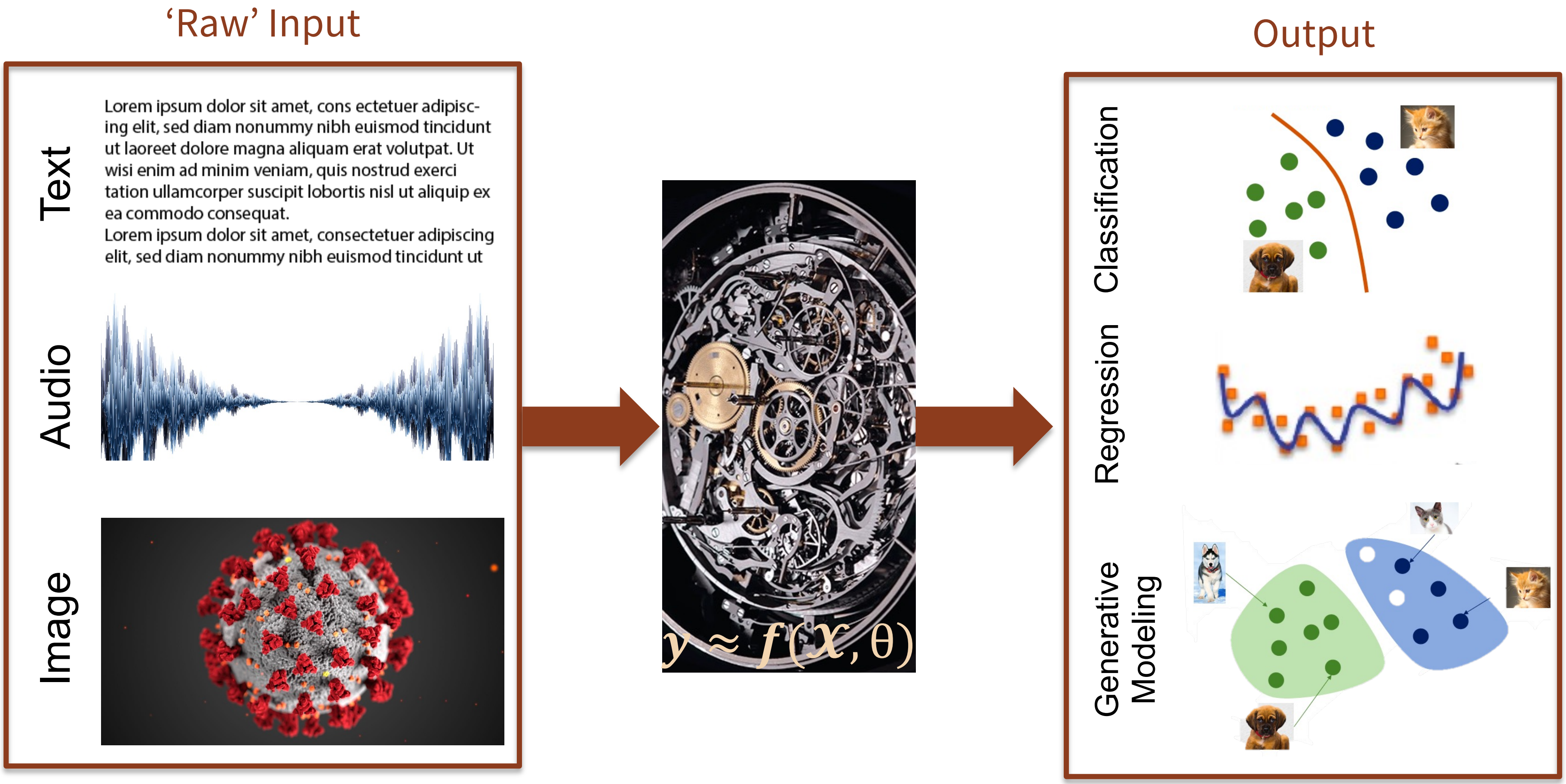
General purpose methods for extracting **patterns from data**.

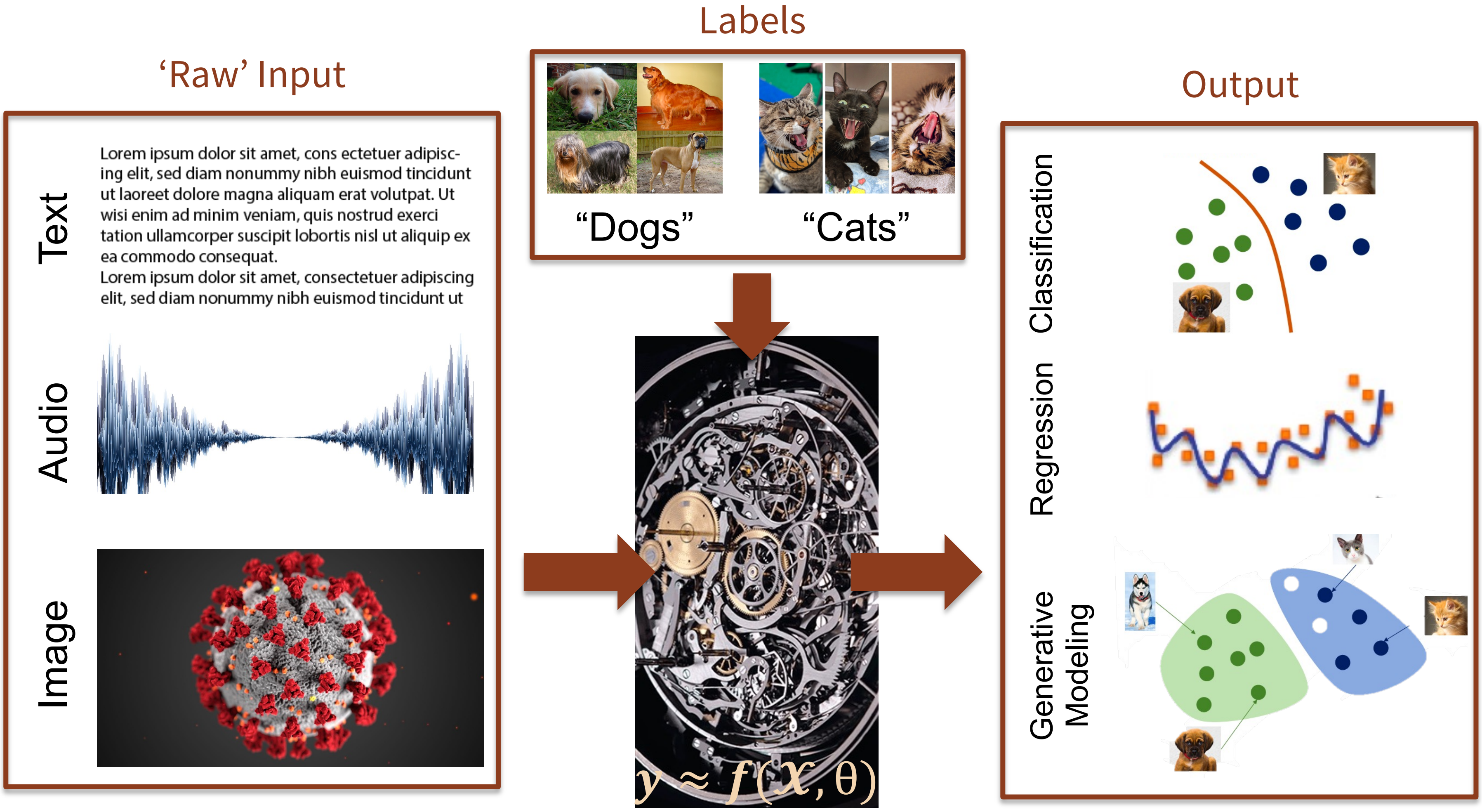




Machine Learning



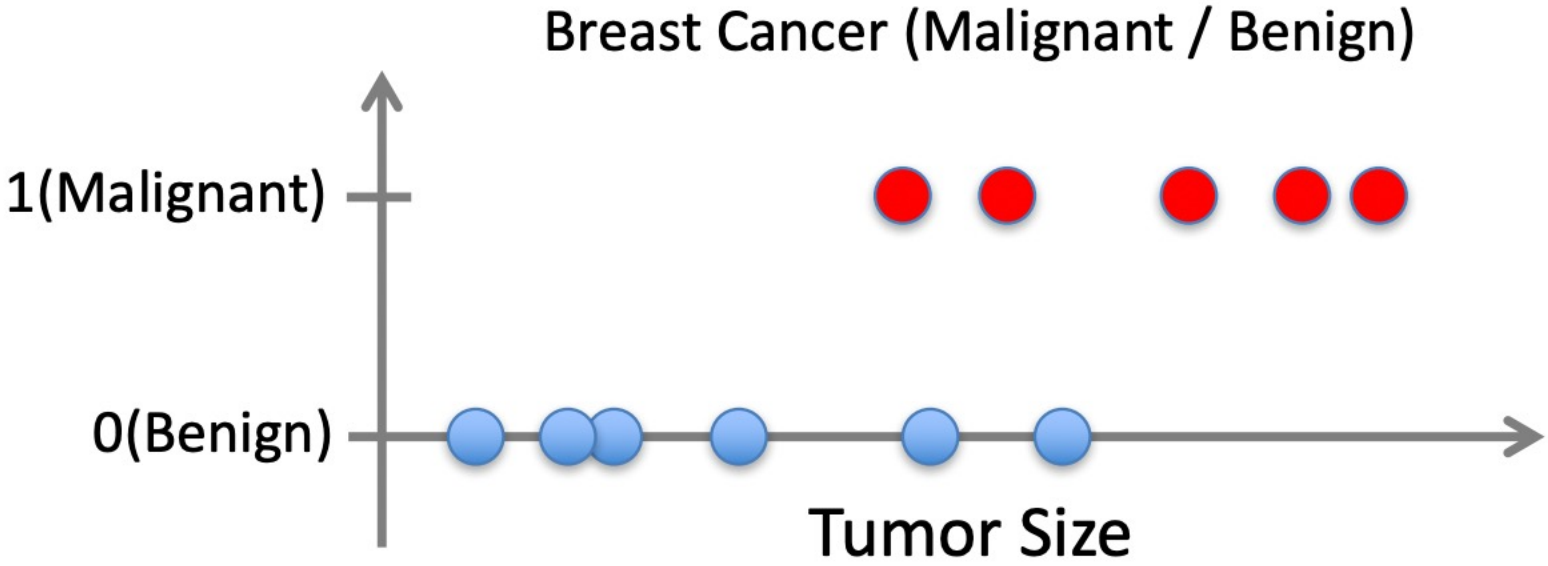

‘Raw’ Input
Text
Lorem ipsum dolor sit amet, cons ectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut
Audio
Image
$y \approx f(x, \theta)$
Output
Classification
Regression
Generative Modeling


Labels
'Raw' Input
Output
Text
Lorem ipsum dolor sit amet, cons ectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut
Audio
Image
"Dogs"
"Cats"
$y \approx f(X, \theta)$
Classification
Regression
Generative Modeling

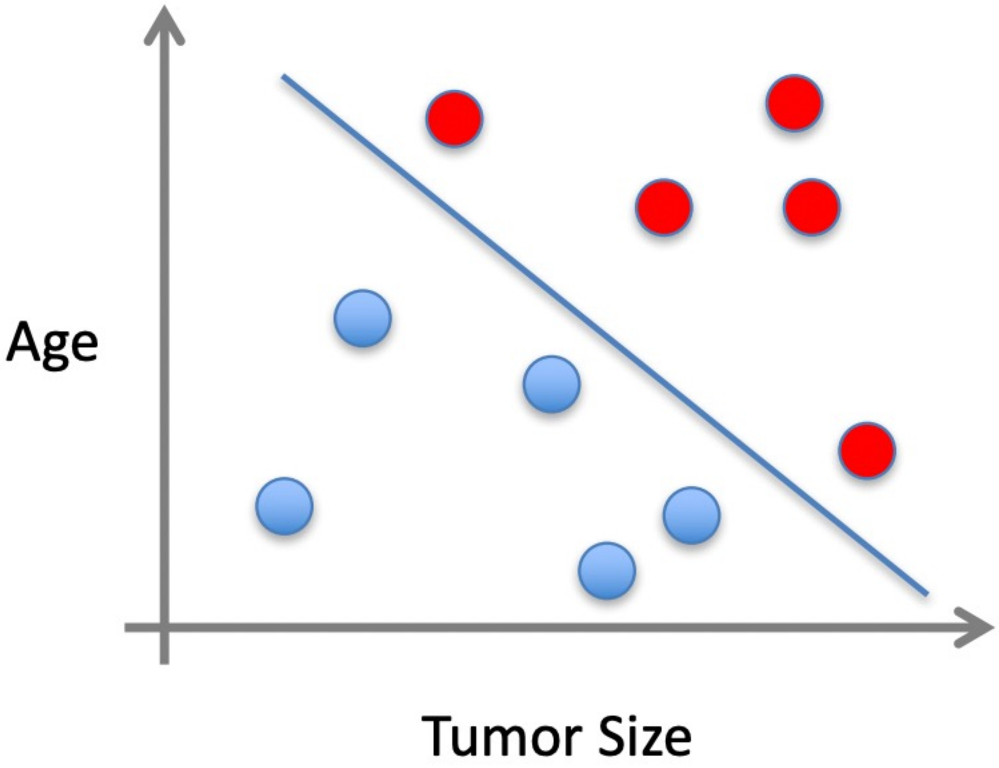
Types of Learning

- Supervised (inductive) learning
  - Given: training data + desired outputs (labels)
- Unsupervised learning
  - Given: training data (without desired outputs)
- Semi-supervised learning
  - Given: training data + a few desired outputs
- Reinforcement learning
  - Rewards from sequence of actions

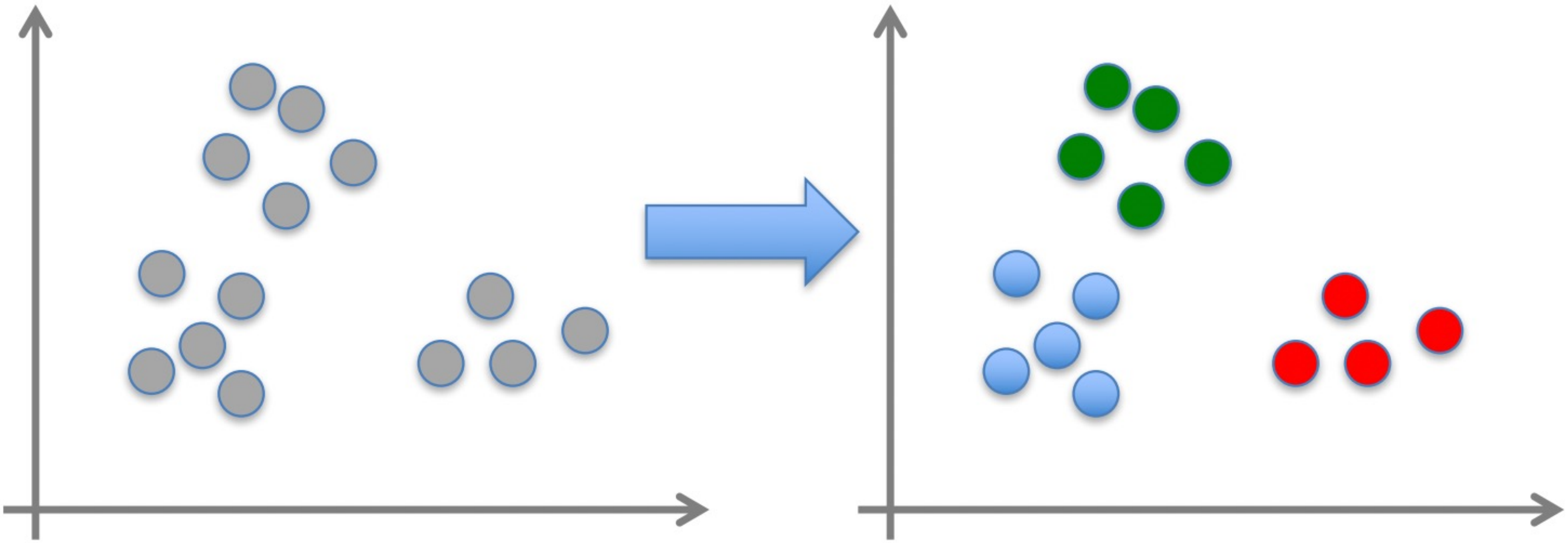
Supervised Learning



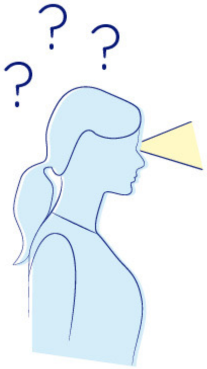
Supervised Learning



Unsupervised Learning



Top-down vs Bottom-up Information



What am I seeing?

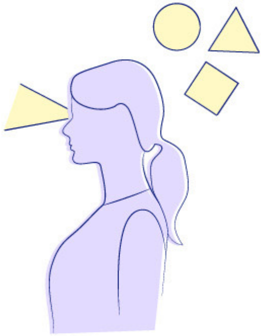
Top-up processing:

using models, ideas, and expectations to interpret sensory information

Is that something I've seen before?

Bottom-up processing:

taking sensory information and then assembling and integrating it



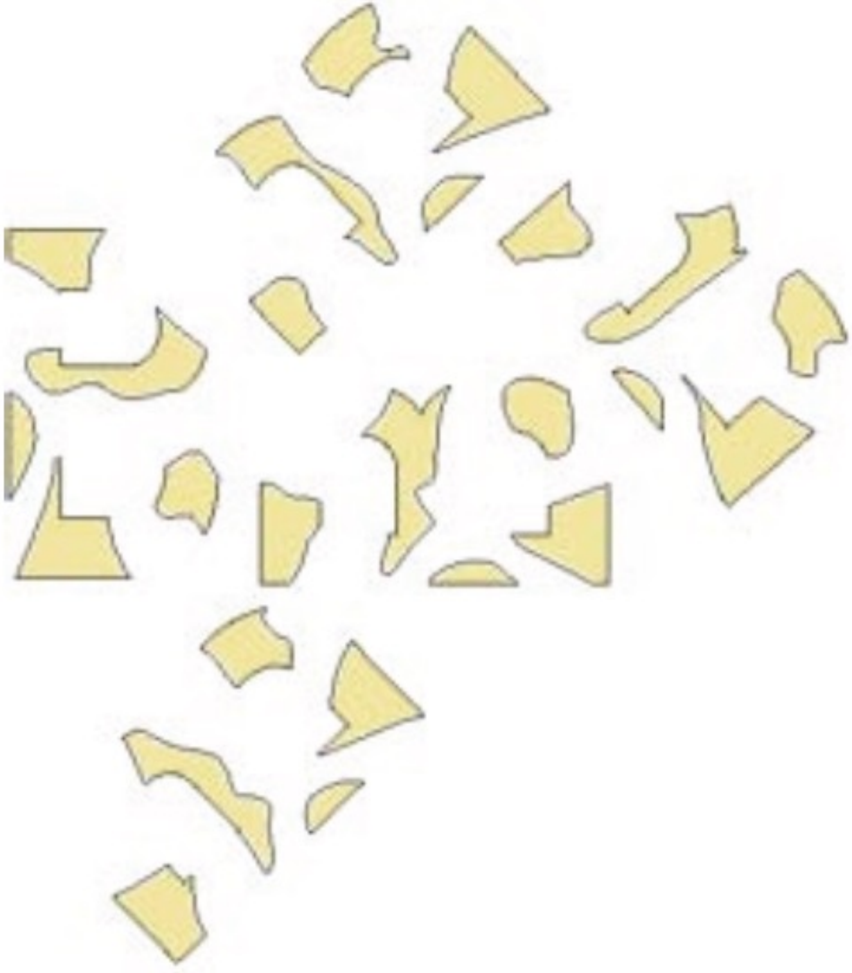
Jack Westin

robertpepperell.com

“ induce an indeterminate mental state”

investigates the nature of the conscious mind through painting and drawing, scientific experimentation and philosophical inquiry.

Jan Koenderink, The Clootcrans Press, Gestaltvision



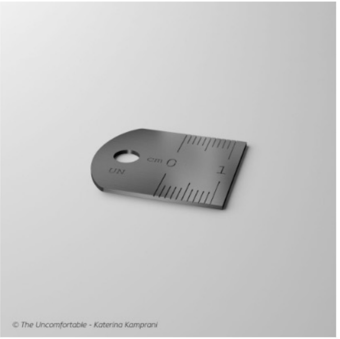
Reality subject to definition

- Feeds on mammals
- On earth for > 90 million years
- Fed even on dinosaurs
- Might wait for 20 years to meet the mammal
- **A Tick locates a mammal by**
    - Propane Carboxylic Acid
    - Warmth
- **Wikipedia defines mammal**
    - Air-breathing vertebrate animals characterized by the possession of endothermy, hair, three middle ear bones, and mammary glands functional in mothers with young.

A Tale of the Tick

“Object"

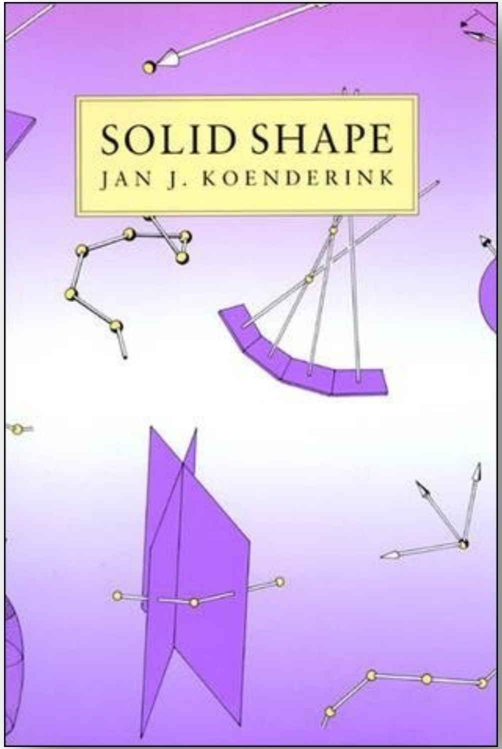
The Uncomfortable





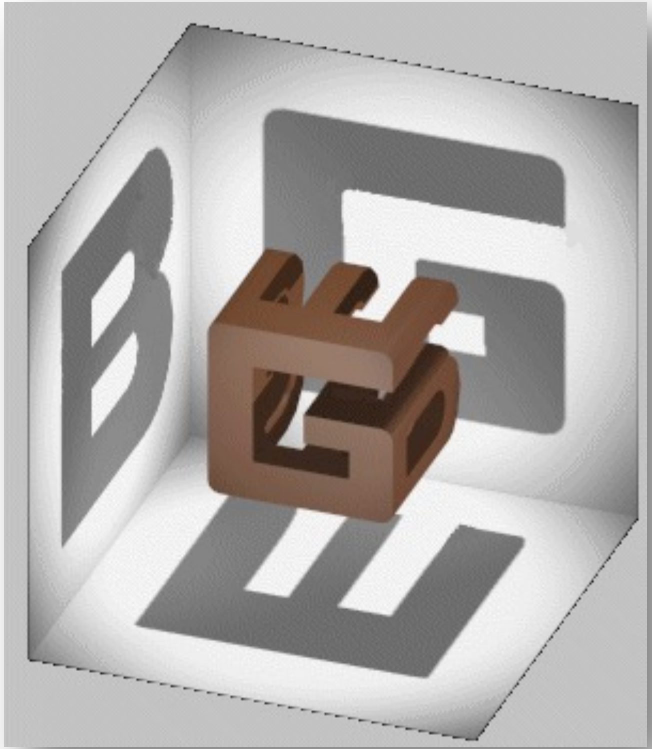






Solid Shape
Jan Koenderink

Consciousness
Christof Koch



Gödel, Escher, Bach
Douglas Hofstader

A Brief History

BC 350 : Aristotle & **Symbolic** Logic

15th century Gutenberg's invention of moveable type

Clocks (the measuring machines)

17th century:
- **Hobbes**: The Leviathan & Theory of Thinking
- **Descartes**: "Animals are complex machines."

18th century: Mechanical Turk mechanized "human" thinking in non-human objects

19th century:
- **Jacquard loom**: Punch card programmable machine
- **Babbage & Ada**: the Analytical Engine
- **Boole**: The Boolean logic
- **Frege**: Modern propositional logic

20th century (first half):
- **Principia Mathematica** (Russell, Wittgenstein, and Rudolf Carnap)
- **First computer game** (chess) – 1912
- **The Turing Machine** 1936-37
- The term **Cybernetics** first used (Wiener)

1950
- Introduction of the **Turing test** ["Computing Machinery and Intelligence" (A. Turing)]
- Claude Shannon: "Programming a computer to play chess"
- Asimov: Three laws of robotics

1956

John McCarthy: "**Artificial Intelligence**" as the topic of the Dartmouth Conference

MIT AI Lab founded.

computer vision begins

1960s
- Machines understand language: **NLP**.
- Minsky & Seymour, **Perceptrons**, on the limits of neural nets.

1970s
- **Star Wars** released.
- General intelligence hype
- **NLP** kicks in
- First **AI Winter**: '74-'80
- "**Connectionism**" was **shut**

Vapnik, 1995: SVM

first web search

1997

Deep Blue

2011-2014

AlexNet

Siri

Watson

2017

AlphaGo

Self Driving 2012-Present

2018

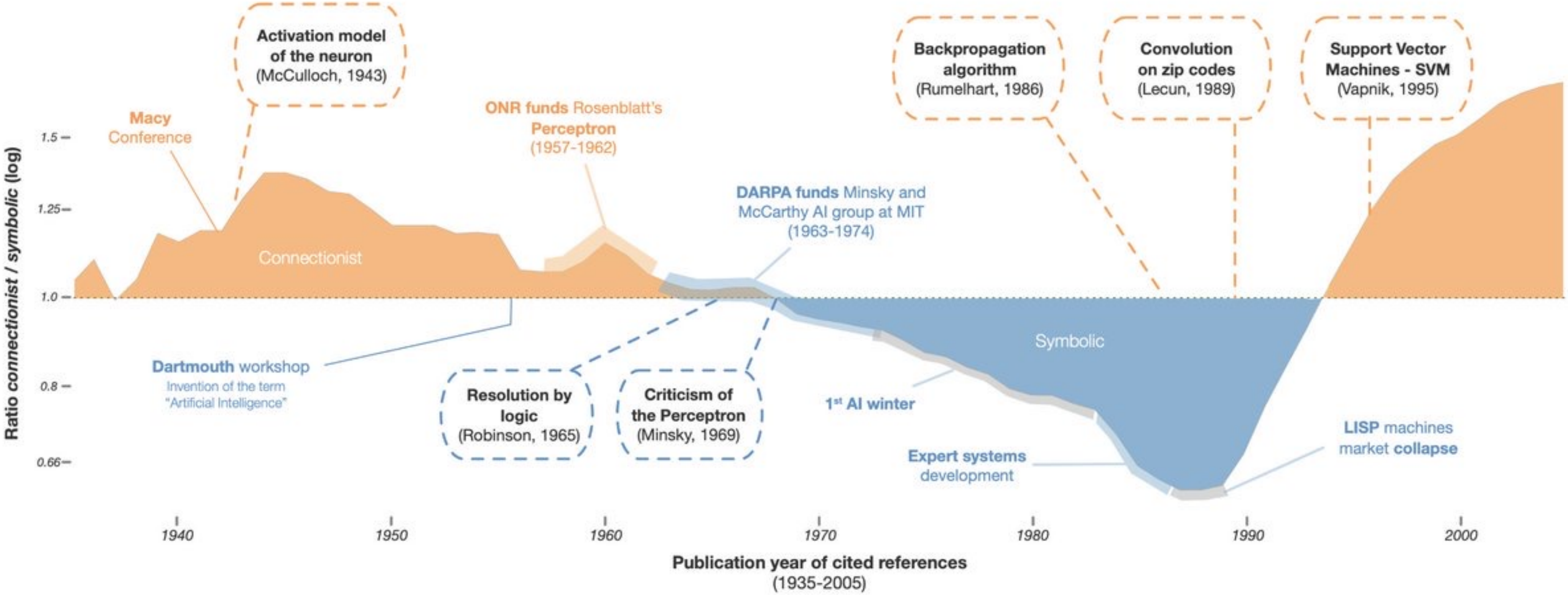
EU Guidelines for AI

Transformers

Theory of Deep Learning

Turing Award for Geoffrey Hinton Yoshua Bengio Yann Lecun

Connectionism vs Symbolism Battle

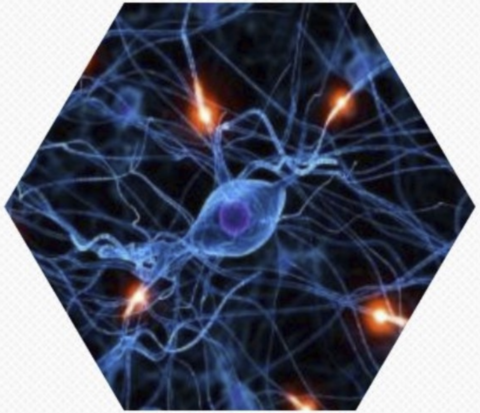


Re-Rise of AI

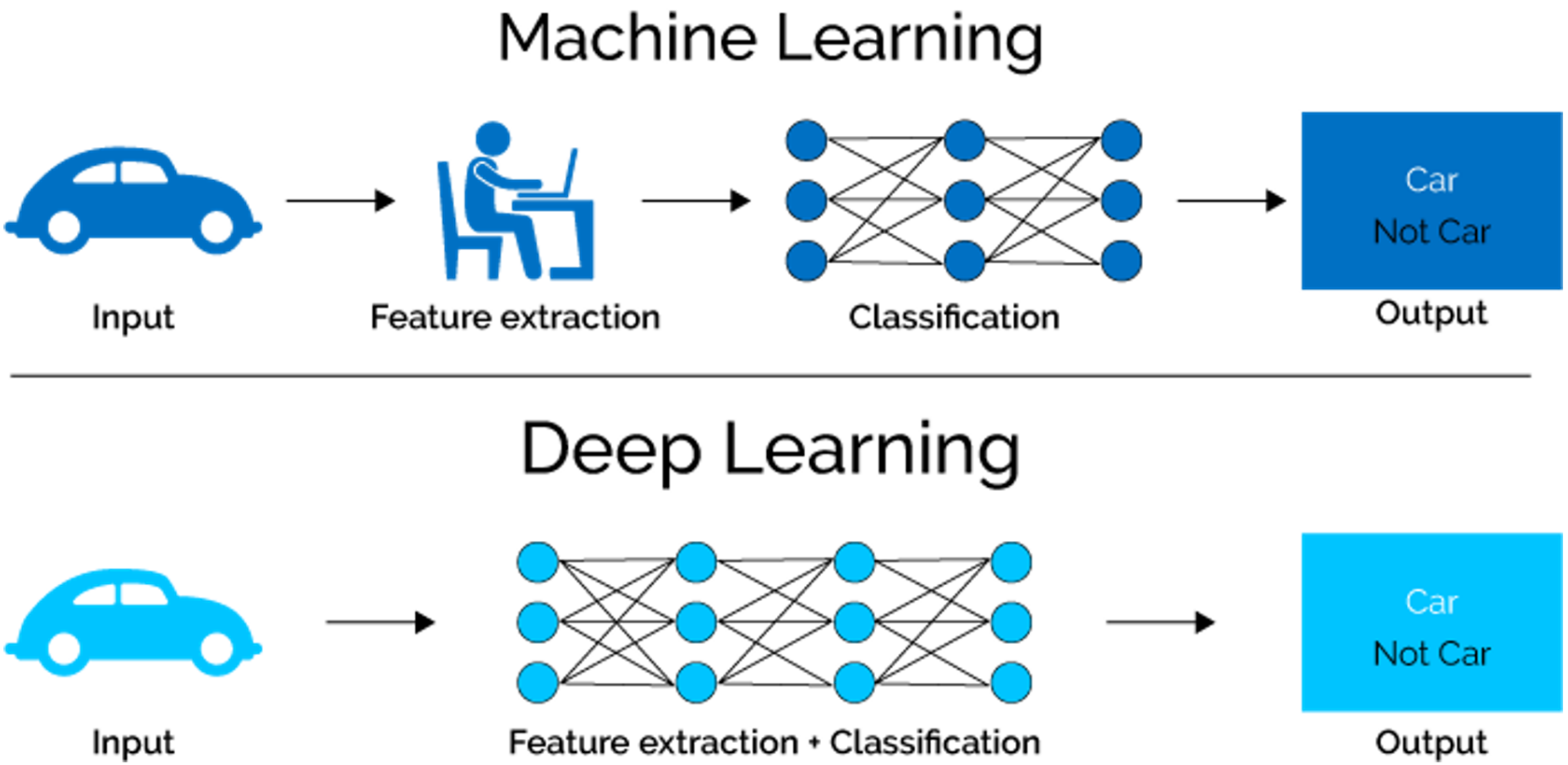
More Data

Better Models

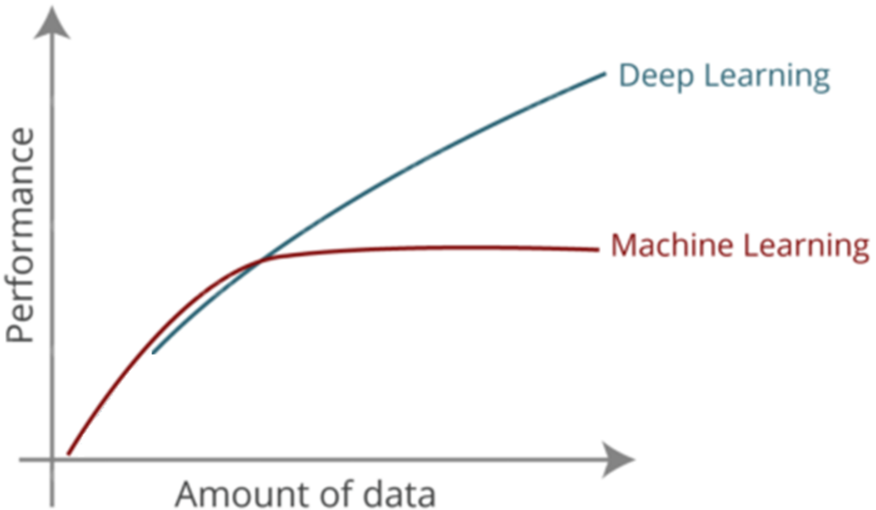
and Algorithms



Powerful GPU Accelerators


Machine Learning
Car
Not Car
Input
Feature extraction
Classification
Output
Deep Learning
Car
Not Car
Input
Feature extraction + Classification
Output

Boost



Left to Humans

A priori knowledge
Data selection
Data filtering & enhancing
Model selection
Learning technique choice
Experiment design
Avoiding brute force
Hybrid systems

A day in the life of an ML engineer

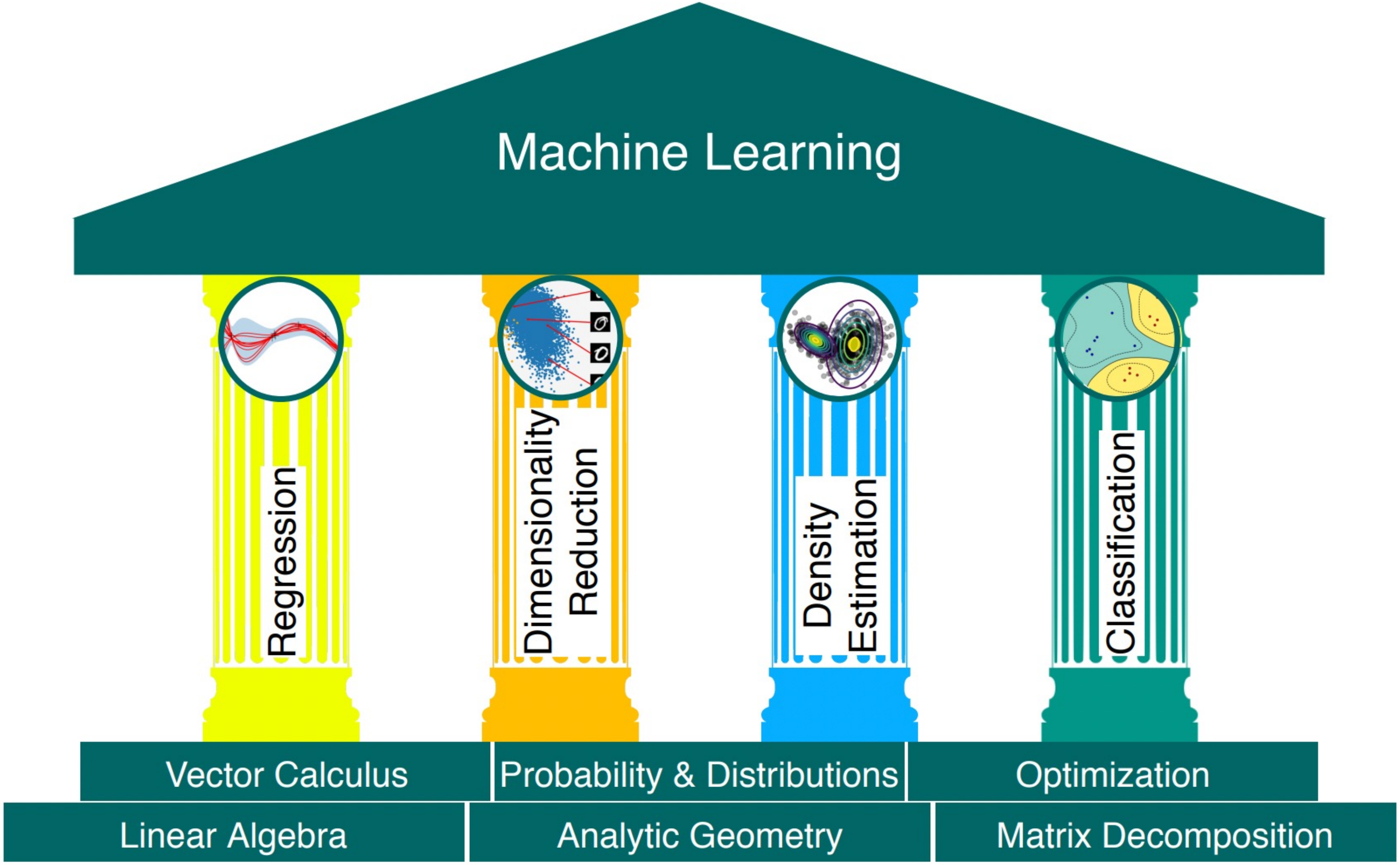
Optimizer
$W_0, W_1 \ldots W_k$

Final Weights

Input

Architecture

Loss Function


Machine Learning
Regression
Dimensionality Reduction
Density Estimation
Classification
Vector Calculus
Probability & Distributions
Optimization
Linear Algebra
Analytic Geometry
Matrix Decomposition

	Pazartesi	Salı	Çarşamba	Perşembe	Cuma
	Yapay Öğrenmenin Temelleri	**Üretici Modeller**	**Açıklanabilir Yapay Öğrenme**	**Geometrik Yapay Öğrenme**	**Güncel Araştırmalar**
09:00-12:00	Güncel uygulama örnekleri Yapay öğrenmede temel yöntemler Matematiksel temeller (vektör analizi, lineer cebir, olasılık teorisi, optimizasyon)	Koşullu ve marjinal olasılık Gizli değişken modelleri Gauss dağılımı Marjinal olabilirlik Veriye dağılım oturtma Gauss karışım modelleri	Global ve lokal açıklama Kendiliğinden açıklanabilir modeller Eğitilmiş model açıklama yöntemleri Kural tabanlı yapay öğrenme	Tensör analizi Diferansiyel geometri Manifoldlar	Grup teorisi ve eşvaryantlar (**Çağatay**) Geometrik yapay öğrenme ve topoloji (**Tolga**) Diferansiyel mahremiyet (**Sinan**)
	Tolga (Taylan, İlker)	**Sinan (Çağatay)**	**İlker (Özgür)**	**Tolga - Kamer**	**Herkes**
14:00-17:00	Ağ mimarileri Geri-yayılım Stokastik bayır inişi Yakınsaklık analizleri	Temel bileşenler ayrışımı Varyasyonel özkodlayıcı Gürültü giderici özkodlayıcı Difuzyon modelleri	Optimizasyon modelleri Kural kümeleri türetme Lokal doğrusal modeller Kuramsal altyapı	Izgaralar, gruplar, ayarlar, jeodezikler Simetri altında eşvaryant öğrenme Çizge sinir ağları Nokta bulutları	Yapay öğrenmede kısıtlı optimizasyon (**Figen**) Stokastik model kurma ile derin öğrenme modelleri eğitimi (**Özgür**) Gürbüz karşıolgusal açıklamalar (**İlker**)
	Özgür (Taylan, İlker)	**Çağatay (Sinan)**	**Hakan (Özgür)**	**Kamer - Tolga**	**Herkes**

Linear Things: Vectors, Maps, Tensors and All That.

Vector definition

Computer science: *vector* is a one-dimensional array of ordered real-valued scalars

Mathematics: *vector* is a quantity possessing both magnitude and direction, represented by an arrow indicating the direction, and the length of which is proportional to the magnitude

Vectors are written in column form or in row form

Denoted by bold-font lower-case letters

$$\mathbf{x} = \begin{bmatrix} 1 \\ 7 \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} 1 & 7 & 0 & 1 \end{bmatrix}^T$$

For a general form vector with n elements, the vector lies in the n-dimensional space $\mathbf{x} \in \mathbb{R}^n$

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

Euclidean Space

$$\mathbb{R}^n = (x_1 \ldots x_n)\ \forall x_i \in \mathbb{R}^1 = \mathbb{R}$$

$\mathbb{R}^1$: Line
$\mathbb{R}^2$: Plane
$\mathbb{R}^{3+}$: Space

$\mathbb{R}^n$ is a vector space:

$$\mathbf{x} + \mathbf{y} = (x_1 + y_1, \ldots, x_n + y_n)$$

$$a\mathbf{x} = (ax_1, \ldots, ax_2)$$

Hence, $\mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{R}^n$ are vectors.

Norm $\equiv$ *Length* of a vector:

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$$

Matrix is a rectangular array of real-valued scalars arranged in *m* horizontal rows and *n* vertical columns

Each element a_{ij} belongs to the i^{th} row and j^{th} column

The elements are denoted a_{ij} or $\mathbf{A}_{ij}$ or $[\mathbf{A}]_{ij}$ or $\mathbf{A}(\boldsymbol{i}, \boldsymbol{j})$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

For the matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, the size (dimension) is $m \times n$ or (m, n)

Matrices are denoted by bold-font upper-case letters

Addition or subtraction

$$\left(\mathbf{A}\pm\mathbf{B}\right)_{i,j}=\mathbf{A}_{i,j}\pm\mathbf{B}_{i,j}$$

$$\begin{bmatrix}1 & 3 & 1\\ 1 & 0 & 0\end{bmatrix}+\begin{bmatrix}0 & 0 & 5\\ 7 & 5 & 0\end{bmatrix}=\begin{bmatrix}1+0 & 3+0 & 1+5\\ 1+7 & 0+5 & 0+0\end{bmatrix}=\begin{bmatrix}1 & 3 & 6\\ 8 & 5 & 0\end{bmatrix}$$

$$\left(c\mathbf{A}\right)_{i,j}=c\cdot\mathbf{A}_{i,j}$$

Scalar multiplication

$$2\cdot\begin{bmatrix}1 & 8 & -3\\ 4 & -2 & 5\end{bmatrix}=\begin{bmatrix}2\cdot 1 & 2\cdot 8 & 2\cdot -3\\ 2\cdot 4 & 2\cdot -2 & 2\cdot 5\end{bmatrix}=\begin{bmatrix}2 & 16 & -6\\ 8 & -4 & 10\end{bmatrix}$$

$$\left(\mathbf{AB}\right)_{i,j}=\mathbf{A}_{i,1}\mathbf{B}_{1,j}+\mathbf{A}_{i,2}\mathbf{B}_{2,j}+\cdots+\mathbf{A}_{i,n}\mathbf{B}_{n,j}$$

Matrix multiplication *Defined only if the number of columns of the left matrix is the same as the number of rows of the right matrix*

$$\begin{bmatrix}\underline{2} & \underline{3} & \underline{4}\\ 1 & 0 & 0\end{bmatrix}\begin{bmatrix}0 & \underline{1000}\\ 1 & \underline{100}\\ 0 & \underline{10}\end{bmatrix}=\begin{bmatrix}3 & \underline{2340}\\ 0 & 1000\end{bmatrix}$$

Transpose of the matrix: $\mathbf{A}^T$ has the rows and columns exchanged

$$\left(\mathbf{A}^T\right)_{i,j} = \mathbf{A}_{j,i}$$

$$\begin{bmatrix}1 & 2 & 3\\ 0 & -6 & 7\end{bmatrix}^{\mathrm{T}} = \begin{bmatrix}1 & 0\\ 2 & -6\\ 3 & 7\end{bmatrix}$$

$$\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$$

$$(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \boldsymbol{B}^T$$

$$(\mathbf{A}^T)^T = \mathbf{A}$$

$$\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}$$

$$\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C}$$

$$(\mathbf{AB})^T = \boldsymbol{B}^T\mathbf{A}^T$$

Square matrix: has the same number of rows and columns

Identity matrix ($\mathbf{I}_n$): has ones on the main diagonal, and zeros elsewhere

$$\mathbf{I}_3 = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

Determinant of a matrix, denoted by det($\mathbf{A}$) or $|\mathbf{A}|$, is a real-valued scalar encoding certai
properties of the matrix

$$\det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = ad - bc$$

For larger-size matrices the determinant of a matrix id calculated as

$$\det(\mathbf{A}) = \sum_j a_{ij}(-1)^{i+j} det\left(\mathbf{A}_{(i,j)}\right)$$

Trace of a matrix is the sum of all diagonal elements

$$\mathrm{Tr}(\mathbf{A}) = \sum_i a_{ii}$$

A matrix for which $\mathbf{A} = \mathbf{A}^T$ is called a ***symmetric matrix***

To multiply two matrices $\mathbf{A} \in \mathbb{R}^{n \times k}$ and $\mathbf{B} \in \mathbb{R}^{k \times m}$

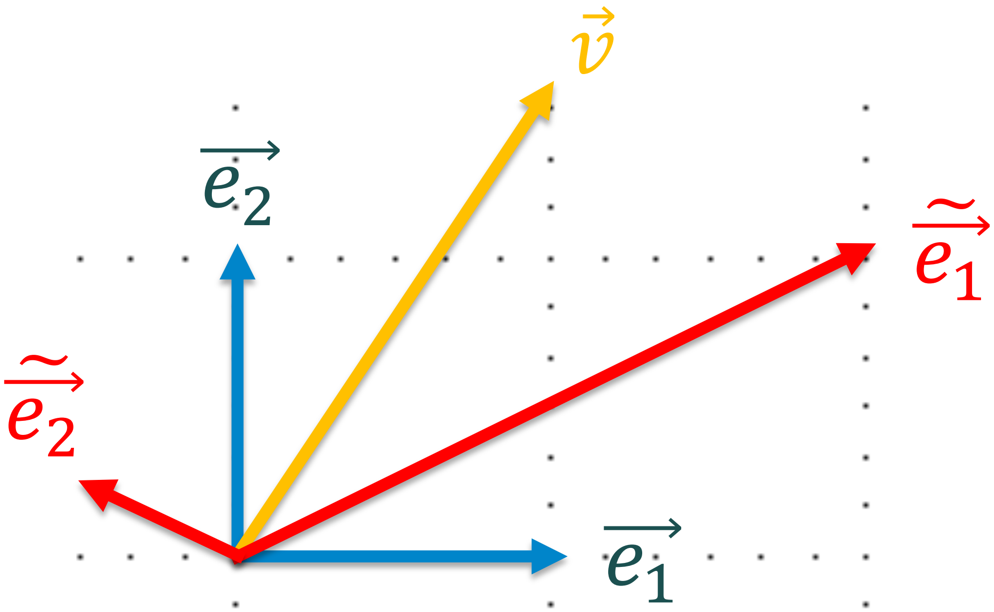
$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nk} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{k1} & b_{k2} & \cdots & b_{km} \end{bmatrix}$$

We can consider the matrix-matrix product as dot-products of rows in $\mathbf{A}$ and columns in $\mathbf{B}$

$$\mathbf{C} = \mathbf{A}\mathbf{B} = \begin{bmatrix} \mathbf{a}_1^\top \\ \mathbf{a}_2^\top \\ \vdots \\ \mathbf{a}_n^\top \end{bmatrix} \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_m \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1^\top \mathbf{b}_1 & \mathbf{a}_1^\top \mathbf{b}_2 & \cdots & \mathbf{a}_1^\top \mathbf{b}_m \\ \mathbf{a}_2^\top \mathbf{b}_1 & \mathbf{a}_2^\top \mathbf{b}_2 & \cdots & \mathbf{a}_2^\top \mathbf{b}_m \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{a}_n^\top \mathbf{b}_1 & \mathbf{a}_n^\top \mathbf{b}_2 & \cdots & \mathbf{a}_n^\top \mathbf{b}_m \end{bmatrix}$$

Size: $\mathbf{A}(n \times k) \cdot \mathbf{B}(k \times m) = \mathbf{C}(n \times m)$

Vector



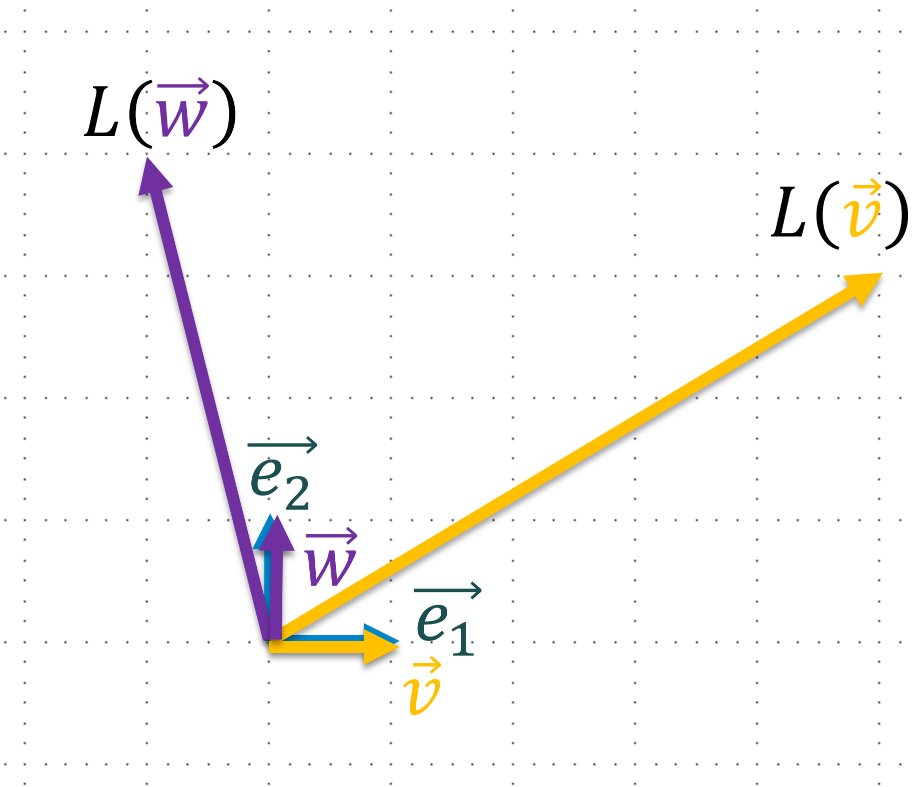
$$\vec{v} = 1\,\vec{e_1} + 1.5\,\vec{e_2}$$

$$\begin{bmatrix} 1 \\ 1.5 \end{bmatrix}_{\vec{e_i}}$$

$$\vec{v} = 1\,\vec{\tilde{e}_1} + 2\,\vec{\tilde{e}_2}$$

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix}_{\vec{\tilde{e}_i}}$$

Linear Maps

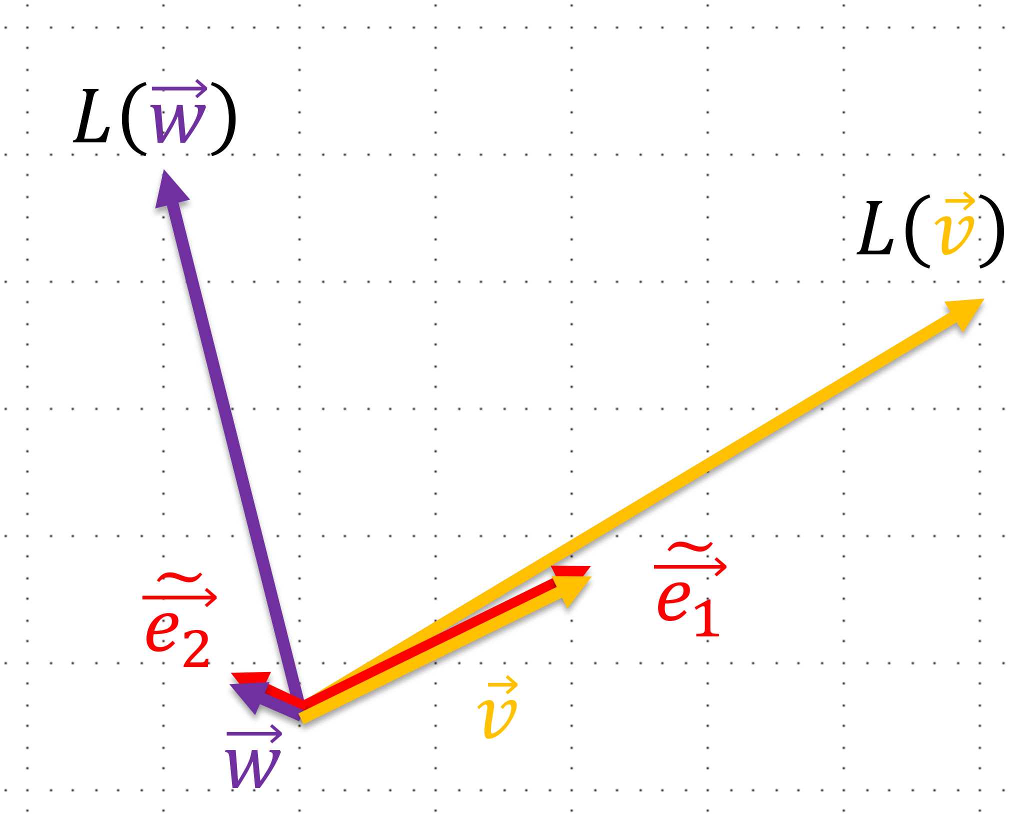


$$L(\vec{v}) = 5\vec{e_1} + 3\vec{e_2}$$

$$L(\vec{w}) = (-1)\vec{e_1} + 4\vec{e_2}$$

$$\begin{bmatrix} 5 & -1 \\ 3 & 4 \end{bmatrix}_{\vec{e_i}}$$

Linear Maps

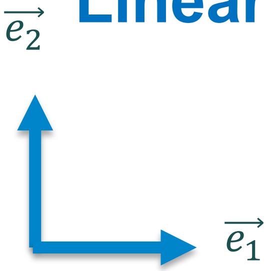


$$L(\vec{v}) = 2.75\widetilde{\vec{e_1}} + 1\widetilde{\vec{e_2}}$$

$$L(\vec{w}) = 1.75\widetilde{\vec{e_1}} + 9\widetilde{\vec{e_2}}$$

$$\begin{bmatrix} 2.75 & 1.75 \\ 1 & 9 \end{bmatrix}_{\widetilde{\vec{e_i}}}$$

Linear Maps



$$\begin{bmatrix} 1 \\ 1.5 \end{bmatrix}_{\vec{e_i}} \Rightarrow \begin{bmatrix} 1 \\ 2 \end{bmatrix}_{\tilde{\vec{e_i}}}$$

$$\begin{bmatrix} 4 & 2 \end{bmatrix}_{\vec{e_i}} \Rightarrow \begin{bmatrix} 10 & -1.5 \end{bmatrix}_{\tilde{\vec{e_i}}}$$

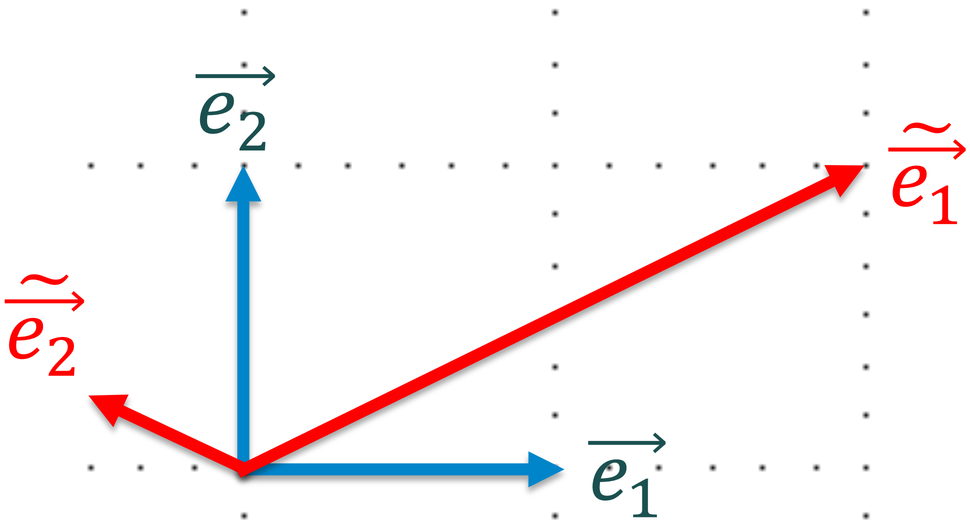
$$\begin{bmatrix} 5 & -1 \\ 3 & 4 \end{bmatrix}_{\vec{e_i}} \Rightarrow \begin{bmatrix} 2.75 & 1.75 \\ 1 & 9 \end{bmatrix}_{\tilde{\vec{e_i}}}$$

Old Basis: $\{\vec{e_1}, \vec{e_2}\}$

New Basis: $\{\vec{\widetilde{e_1}}, \vec{\widetilde{e_2}}\}$

$$\vec{\widetilde{e_1}} = 2\,\vec{e_1} + 1\,\vec{e_2}$$
$$\vec{\widetilde{e_2}} = -\tfrac{1}{2}\vec{e_1} + \tfrac{1}{4}\,\vec{e_2}$$

$$F = \begin{bmatrix} 2 & -\tfrac{1}{2} \\ 1 & \tfrac{1}{4} \end{bmatrix}$$



$$\vec{e_1} = \tfrac{1}{4}\,\vec{\widetilde{e_1}} + (-1)\vec{\widetilde{e_2}}$$
$$\vec{e_2} = \tfrac{1}{2}\,\vec{\widetilde{e_1}} + 2\,\vec{\widetilde{e_2}}$$

$$B = \begin{bmatrix} \tfrac{1}{4} & \tfrac{1}{2} \\ -1 & 2 \end{bmatrix}$$

Functions & Continuity

$f : \mathbb{R}^n \mapsto \mathbb{R}^m$ $\Rightarrow$ $f(x) \in \mathbb{R}^m$ is defined only for $x \in \mathbb{R}^n$

range domain

Composition: For $f : A \mapsto \mathbb{R}^m$ and $g : B \mapsto \mathbb{R}^p$ where $B \subset \mathbb{R}^m$,
$g \circ f(x) = g(f(x))$.

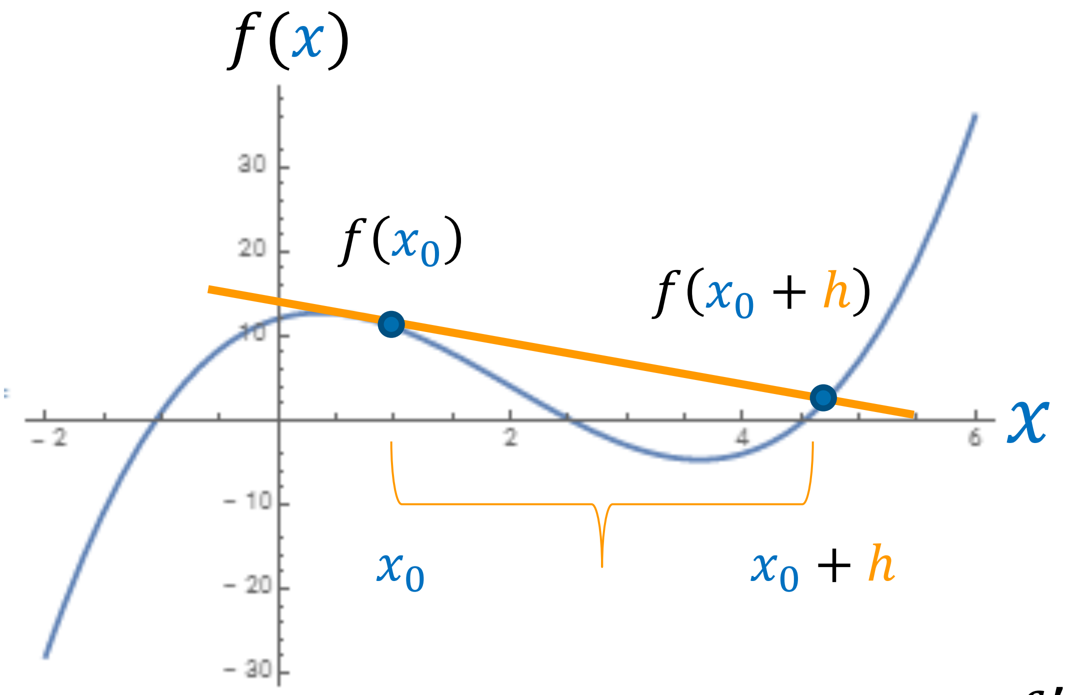
Domain of composition: The domain of $g \circ f = A \cup f^{-1}(B)$.

m-**component function**: $f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_m(\mathbf{x}))$

Projection function π_i: For an identity π, if $\pi(\mathbf{x}) = \mathbf{x}$ then $\pi_i(\mathbf{x}) = x_i$.

Continuity: $f : A \mapsto \mathbb{R}^m$ is **continuous at** $a \in A$ if $\lim_{x \to a} f(x) = f(a)$.
f is continuous if it is continuous at each $a \in A$.

Derivatives in 1D

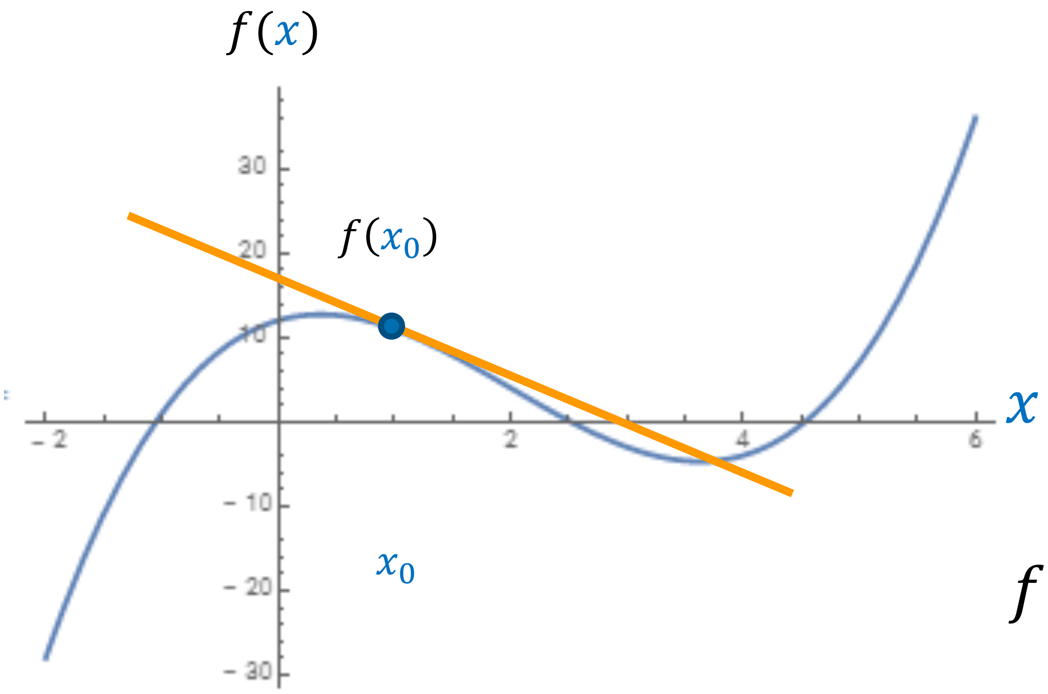


$$slope\ at\ x = f'(x) = \frac{df}{dx}$$

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}$$

$$f(x) = x^3 - 6x^2 + 4x + 12$$

Derivatives in 1D



$$\text{slope at } x = f'(x) = \frac{df}{dx}$$

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}$$

$$f(x) = x^3 - 6x^2 + 4x + 12$$

$$f'(x) = 3x^2 - 12x + 4$$

Power Rule:

$$\frac{d}{dx}x^n = nx^{n-1}$$

Exponential Rule:

$$\frac{d}{dx}e^x = e^x$$

Trig Rules:

$$\frac{d}{dx}\sin(x) = \cos(x)$$

$$\frac{d}{dx}\cos(x) = -\sin(x)$$

Sum Rule:

$$\frac{d}{dx}\big(f(x) + g(x)\big) = \frac{df}{dx} + \frac{dg}{dx}$$

Product Rule:

$$\frac{d}{dx}\big(f(x)g(x)\big) = \frac{df}{dx}g(x) + f(x)\frac{dg}{dx}$$

Chain Rule:

$$\boldsymbol{\frac{d}{dx}\Big(f\big(g(x)\big)\Big) = \frac{df}{dg}\frac{dg}{dx}}$$

Chain Rule

Chain Rule:

$$\frac{d}{dx}\left(f\left(g(x)\right)\right) = \frac{df}{dg}\frac{dg}{dx}$$

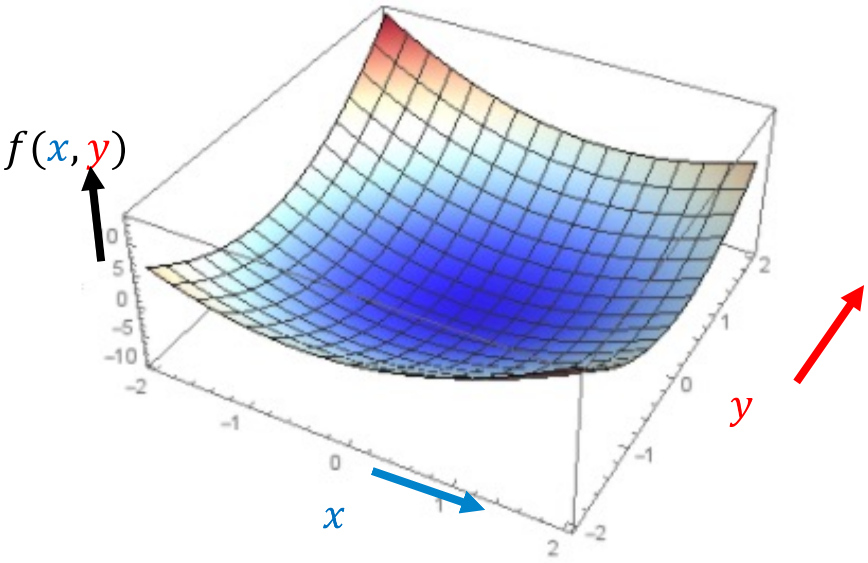
$$\frac{d}{dx}(\sin(2x))$$

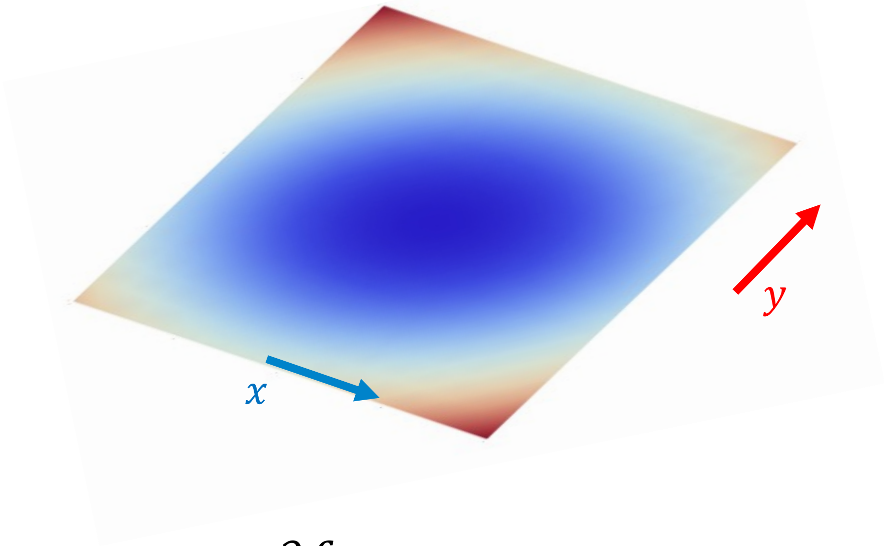
$$= \frac{d}{dg}(\sin(g)) \cdot \frac{d}{dx}(2x)$$

$$= \cos(g) \cdot 2$$

$$= \cos(2x) \cdot 2$$

Derivatives in nD





$$\frac{\partial f}{\partial x} = 4x - y$$

$$\frac{\partial f}{\partial y} = -x + 6y$$

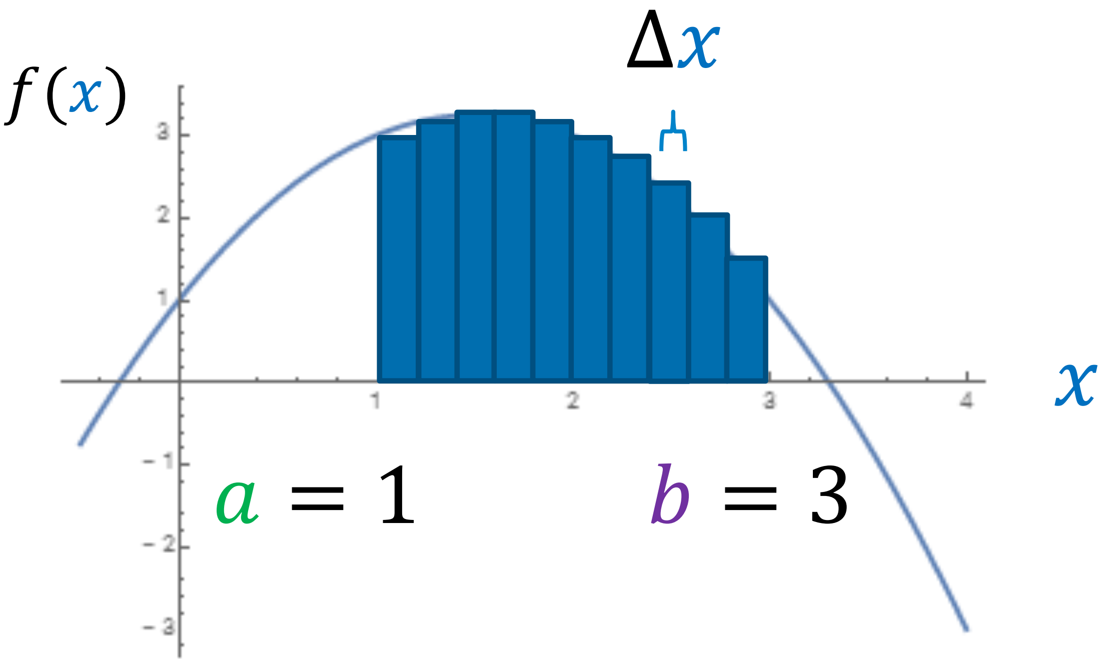
$$f(x, y) = 2x^2 - xy + 3y^2 - 10$$

“Smooth” Function

$f : \mathbb{R}^n \mapsto \mathbb{R}^m$ is **smooth** on the *open* interval (a, b) if $\frac{d^n f(x)}{dx^n}$ exists for all $n \geq 1$ and all $x \in (a, b)$.

In other words, a function is smooth if it is C^∞ (infinitely continuously differentiable).

Integrals

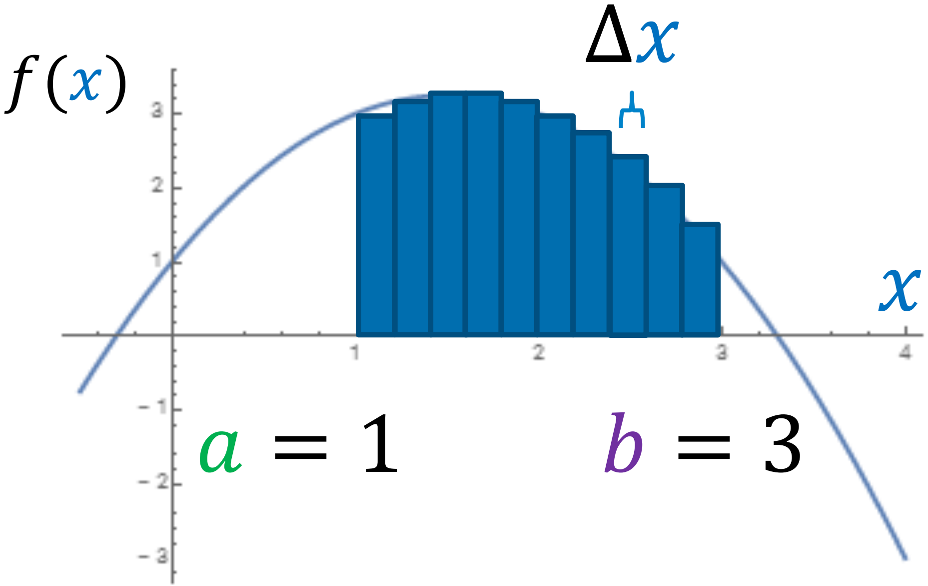


$$Area\ under\ curve$$

$$= \lim_{\Delta x \to 0} \sum_i f(x) \Delta x$$

$$f(x) = -x^2 + 3x + 1$$

Integrals



$f(x) = -x^2 + 3x + 1$

Area under curve

$$= \int_a^b f(x)dx = F(b) - F(a)$$

F is the anti-derivative of f

- $F'(x) = f(x)$

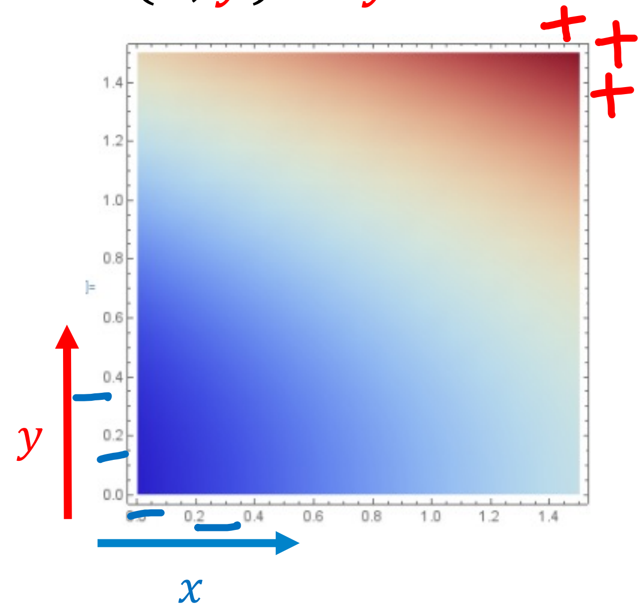
$$F(x) = -\frac{1}{3}x^3 + \frac{3}{2}x^2 + x + c$$

$$\int_1^3 f(x)dx = F(3) - F(1) = 5\frac{1}{3}$$

Chain Rule (multi-variable)

A. Temperature as a function of (2D) position

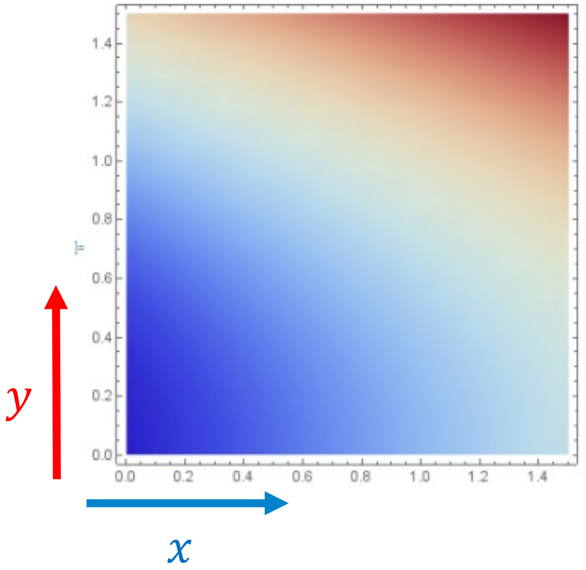
$$T(x, y) = y^2 + x - ½$$



Chain Rule (multi-variable)

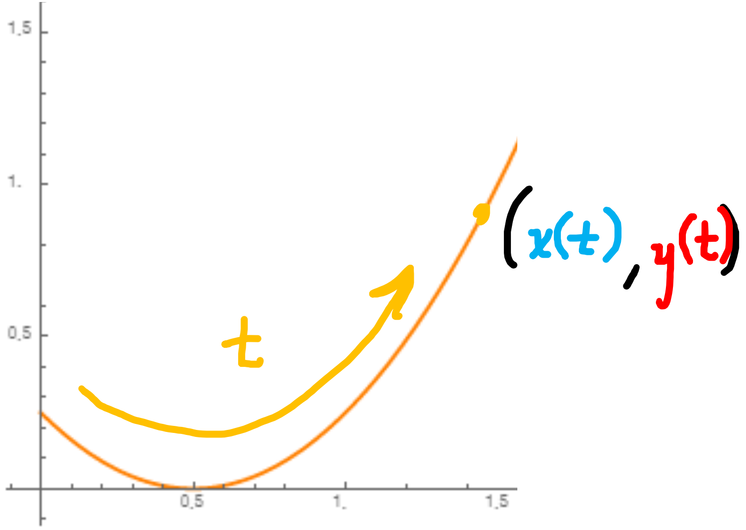
A. Temperature as a function of (2D) position

$$T(x, y) = y^2 + x - ½$$



B. Position (2D) as a function of time

$$(x(t), y(t)) = (t + 1, t^2 - t + ¼)$$

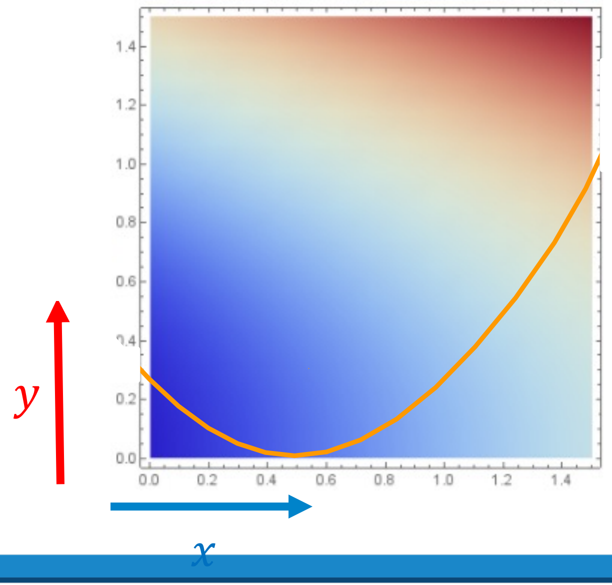


Chain Rule (multi-variable)

C. Temperature as a function of time

$$T(x, y) = y^2 + x - ½$$

$$(x(t), y(t)) = (t, t^2 - t + ¼)$$



$$T(t) = (t^2 - t + ¼)^2 + (t) - ½$$

$$\frac{dT}{dt} = 2(t^2 - t + ¼)(2t - 1) + 1$$

Chain Rule (multi-variable)

C. Temperature as a function of time

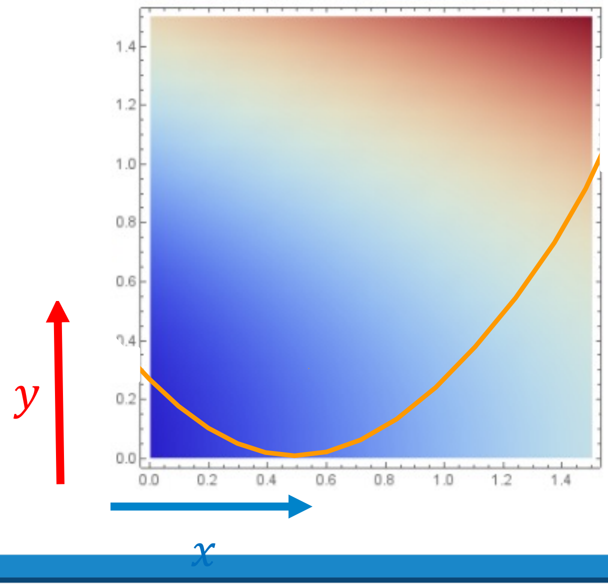
$$T(x, y) = y^2 + x - ½$$
$$(x(t), y(t)) = (t, t^2 - t + ¼\ )$$

$$\frac{dT}{dt} = \frac{\partial T}{\partial x}\frac{dx}{dt} + \frac{\partial T}{\partial y}\frac{dy}{dt}$$

$$= (1)(1) + (2y)(2t - 1)$$

$$= \quad 1 + (2(t^2 - t + ¼))(2t - 1)$$

$$= 2(t^2 - t + ¼)(2t - 1) + 1$$



Chain Rule (multi-variable)

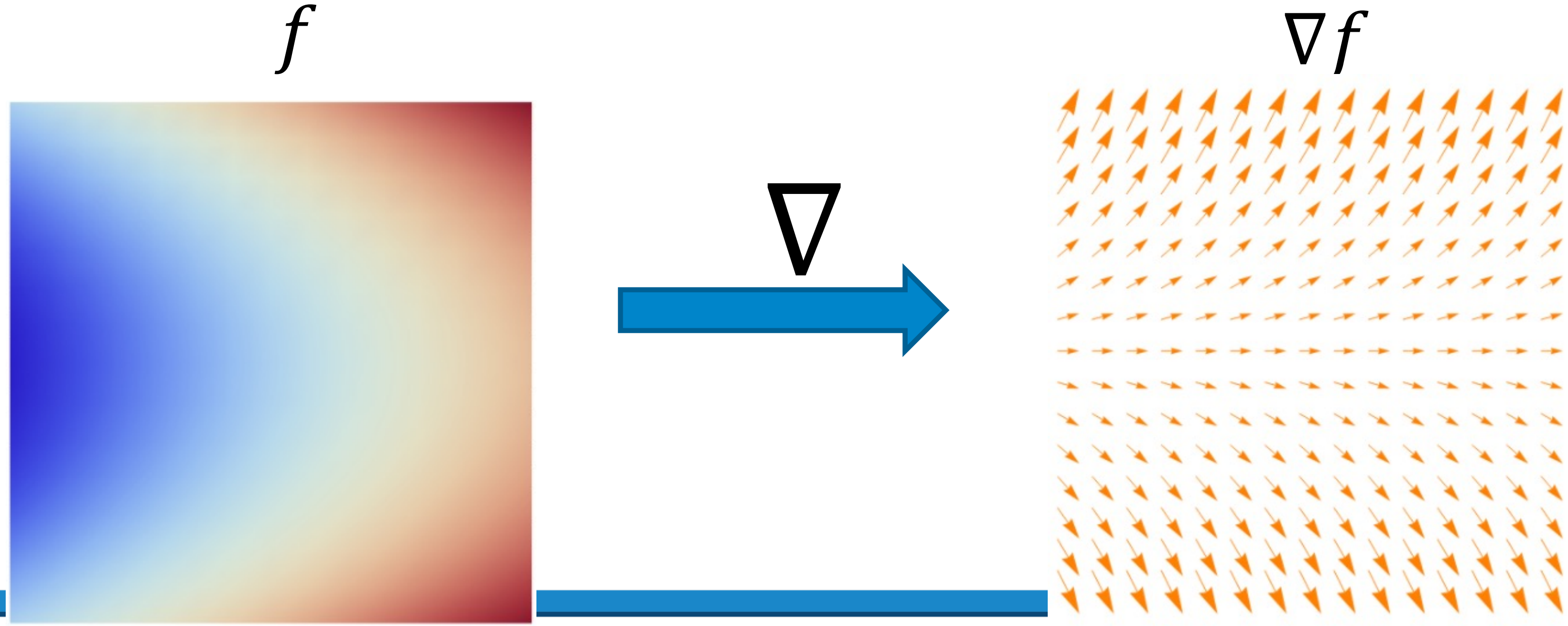
$$\frac{dT}{dt} = \frac{\partial T}{\partial x}\frac{dx}{dt} + \frac{\partial T}{\partial y}\frac{dy}{dt}$$

$$\frac{dT}{dt} = \sum_i \frac{\partial T}{\partial q^i}\frac{dq^i}{dt}$$

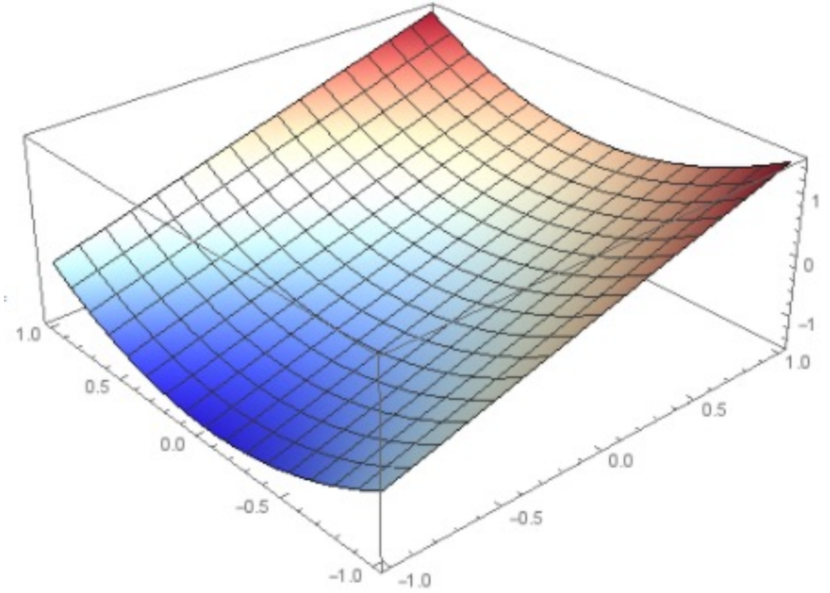
Gradient of a Function

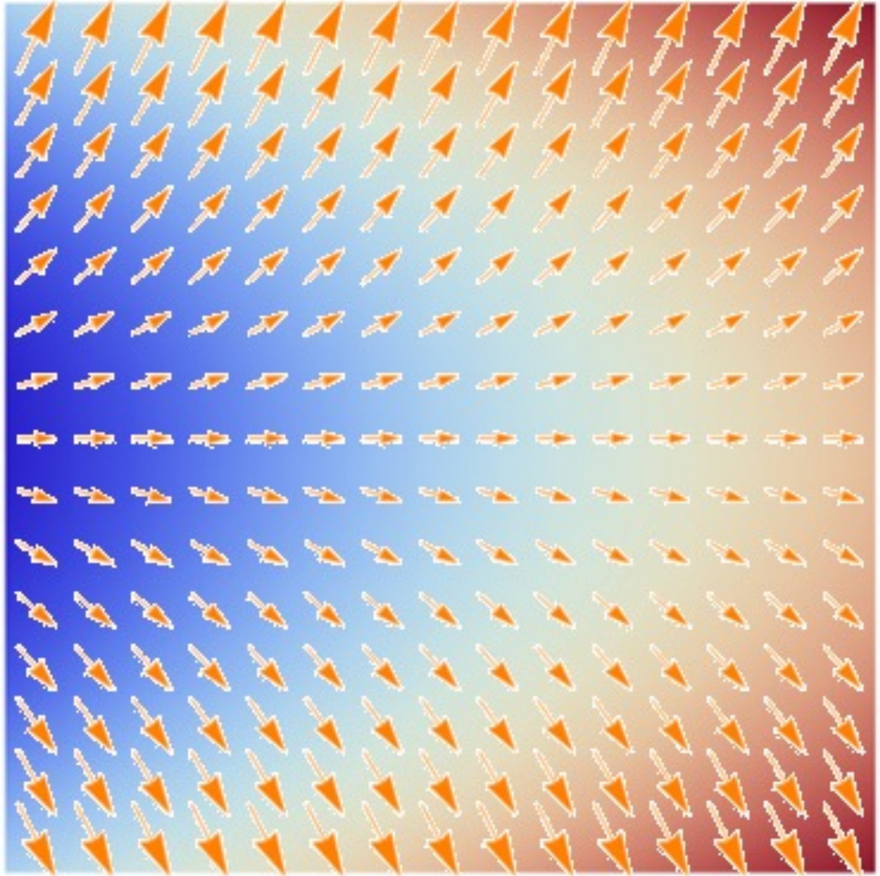
Gradient of a Function

- Gradient takes a function (i.e. scalar field) and produces a vector field
  - Vector points in direction of greatest increase
  - Vector magnitude is proportional to steepness (rate of increase)

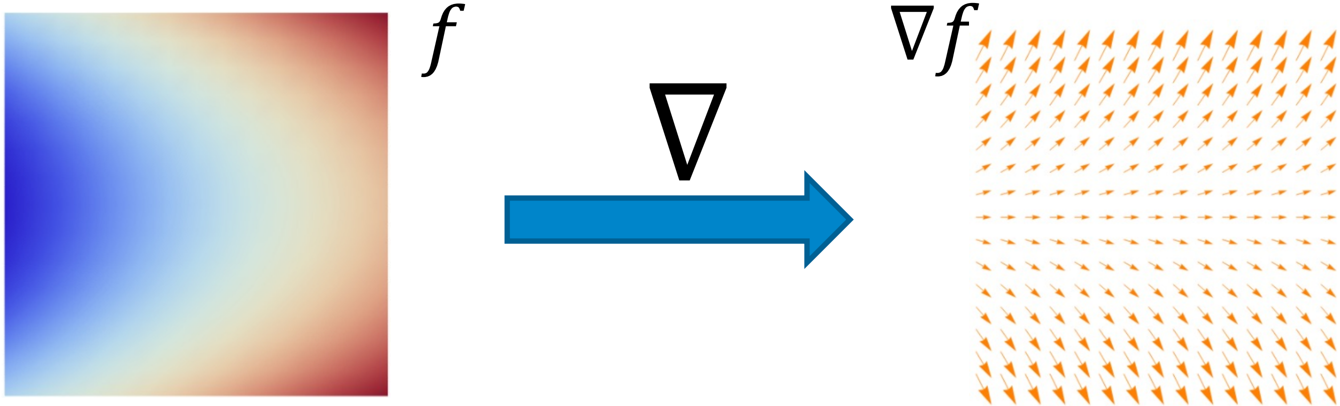


Gradient of a Function





Gradient of a Function



$$f(x, y) = y^2 + x - ½$$

$$\nabla = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix}$$

$$\nabla f = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} (y^2 + x - ½)$$

$$= \begin{bmatrix} \frac{\partial}{\partial x}(y^2 + x - ½) \\ \frac{\partial}{\partial y}(y^2 + x - ½) \end{bmatrix} = \begin{bmatrix} 1 \\ 2y \end{bmatrix}$$

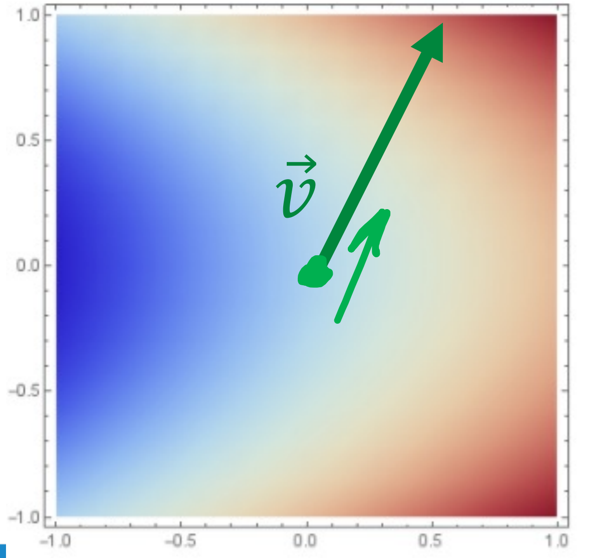
Directional Derivative

- Partial derivatives give us the rate of change in the x and y directions
- The directional derivative gives us the rate of change in any direction $\vec{v}$

$$\nabla_{\vec{v}} f = D_{\vec{v}} f = \frac{\partial f}{\partial \vec{v}}$$

$$\nabla_{\vec{v}} f(x, y) \equiv \lim_{h \to 0} \frac{f(x + hv_x, y + hv_y) - f(x, y)}{h}$$

f



$$\nabla_{\vec{v}} f = \vec{v} \cdot \nabla f$$

$$= \begin{bmatrix} ½ \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2y \end{bmatrix}$$

$$= ½ + 2y$$

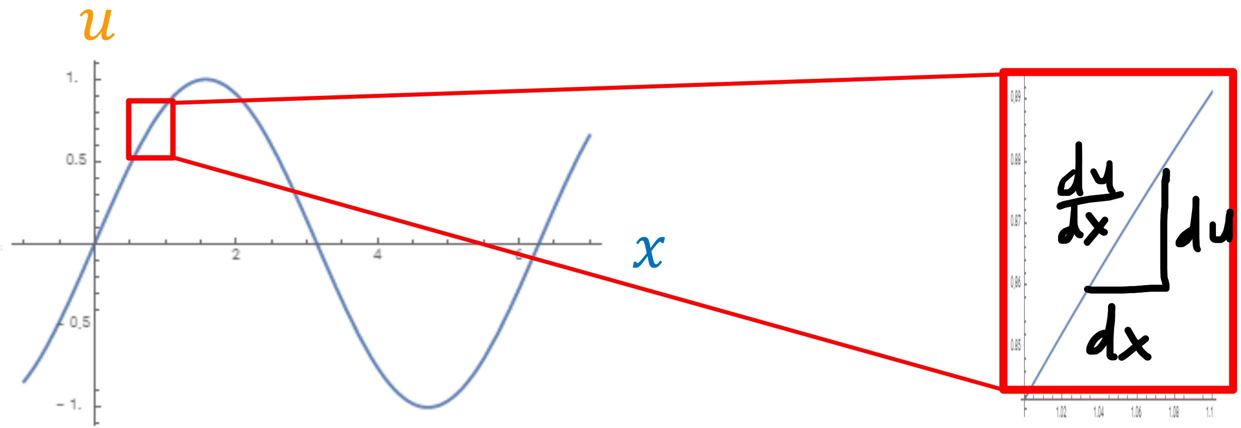
Differentials (multi-variable)

Change of Variables:

$$du = \frac{du}{dx}dx$$

$$u = \sin(x)$$

$$du = \frac{du}{dx}dx$$
$$= \cos(x)\, dx$$



$$\int_{x=0}^{x=\pi/2} (\sin(x))^2 \cos(x)\, dx \Rightarrow \int_{u=0}^{u=1} (u)^2\, du$$

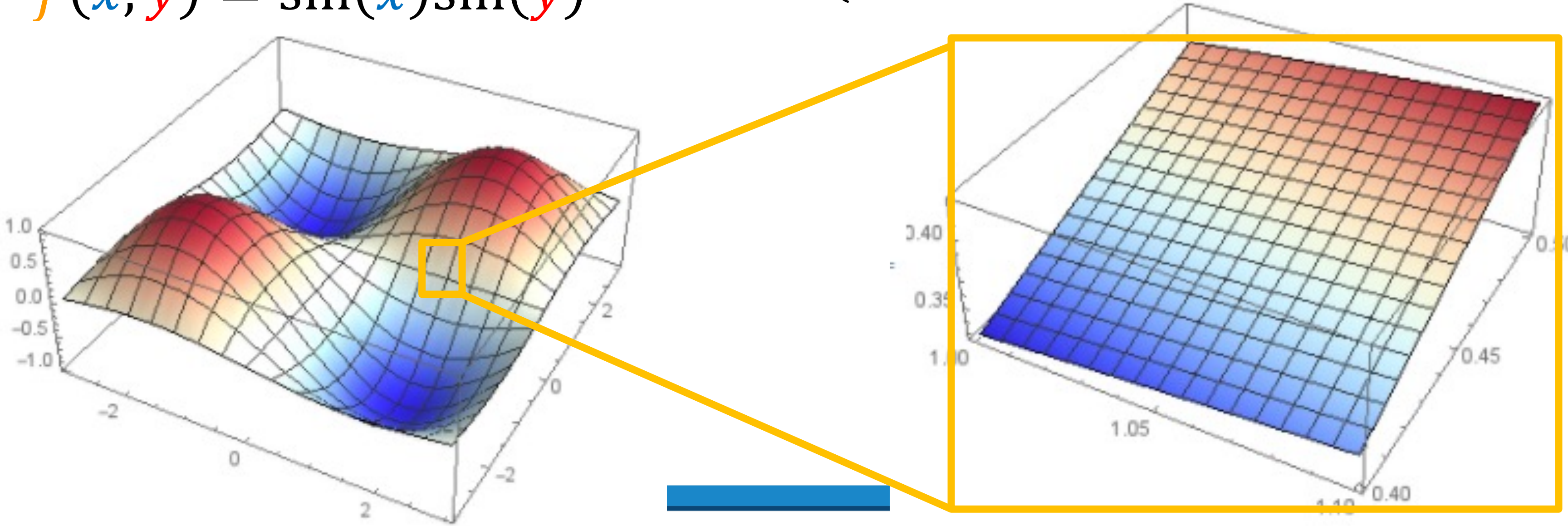
Differentials (multi-variable)

Differentials:
$$\boldsymbol{df = \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy}$$

$$f(x, y) = \sin(x)\sin(y)$$

$$df = \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy$$

$$df = \left(\cos(x)\sin(y)\right)dx + \left(\sin(x)\cos(y)\right)dy$$

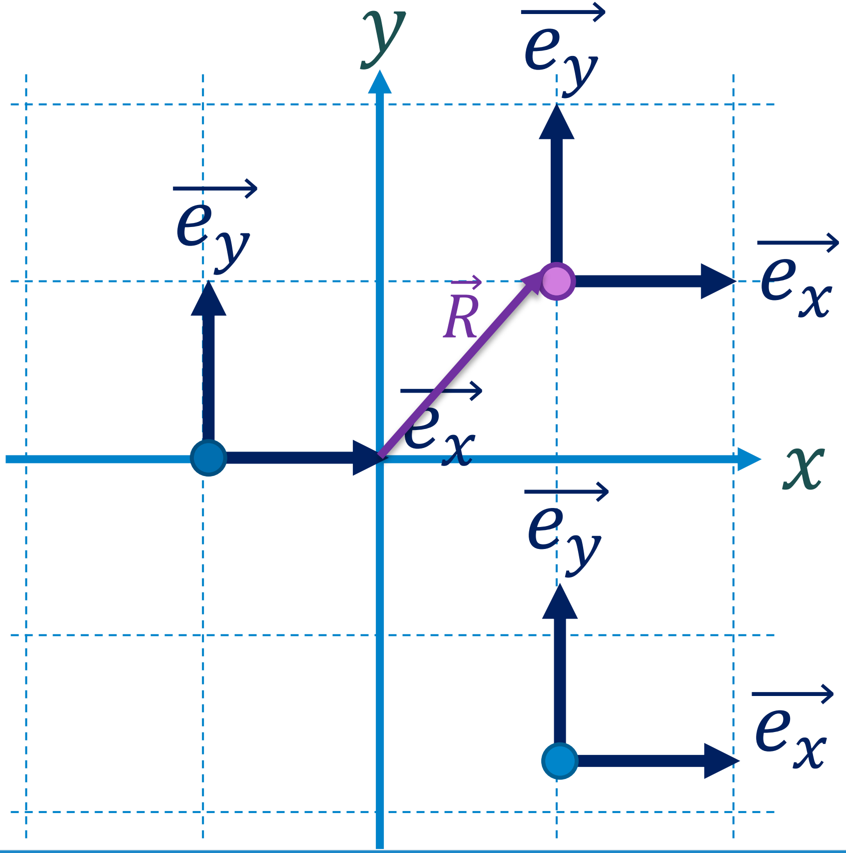


Differentials (multi-variable)

$$\boxed{\begin{gathered} df = \frac{\partial f}{\partial x} dx + \frac{\partial f}{\partial y} dy \\[2em] df = \sum_i \frac{\partial f}{\partial q^i} dq^i \end{gathered}}$$

Partial Derivatives as Basis Vectors

Partial Derivatives as Basis Vectors



$$\frac{\partial \vec{R}}{\partial x} = \overrightarrow{e_x}$$

$$\frac{\partial \vec{R}}{\partial y} = \overrightarrow{e_y}$$

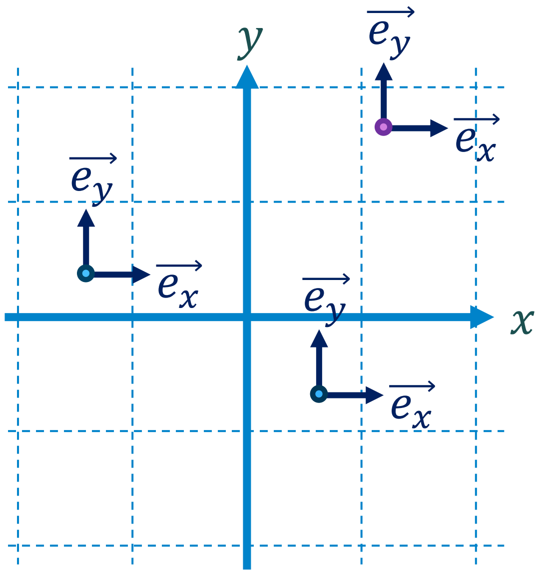
Basis Vectors = Partial Derivatives

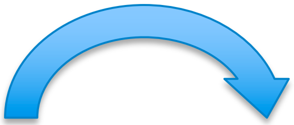
$$\overrightarrow{\widetilde{e_r}} = F_r^x \overrightarrow{e_x} + F_r^y \overrightarrow{e_y}$$

$$\overrightarrow{\widetilde{e_\theta}} = F_\theta^x \overrightarrow{e_x} + F_\theta^y \overrightarrow{e_y}$$

$$\frac{\partial \vec{R}}{\partial x} = \overrightarrow{e_x} \qquad \frac{\partial \vec{R}}{\partial y} = \overrightarrow{e_y}$$

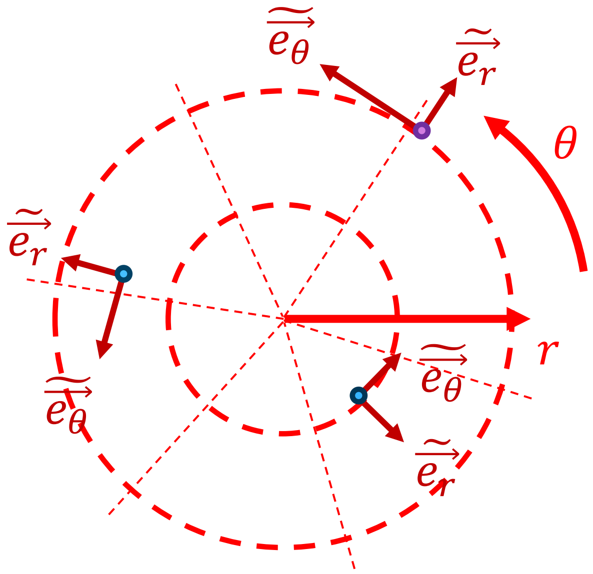
$$\frac{\partial \vec{R}}{\partial r} = \overrightarrow{\widetilde{e_r}} \qquad \frac{\partial \vec{R}}{\partial \theta} = \overrightarrow{\widetilde{e_\theta}}$$





$$x = r\cos\theta$$

$$y = r\sin\theta$$



Basis Vectors = Partial Derivatives

$$\overrightarrow{\widetilde{e_r}} = F_r^x \overrightarrow{e_x} + F_r^y \overrightarrow{e_y}$$

$$\overrightarrow{\widetilde{e_\theta}} = F_\theta^x \overrightarrow{e_x} + F_\theta^y \overrightarrow{e_y}$$

$$\frac{\partial \vec{R}}{\partial x} = \overrightarrow{e_x}$$

$$\frac{\partial \vec{R}}{\partial y} = \overrightarrow{e_y}$$

$$\frac{\partial \vec{R}}{\partial r} = \overrightarrow{\widetilde{e_r}}$$

$$\frac{\partial \vec{R}}{\partial \theta} = \overrightarrow{\widetilde{e_\theta}}$$

$$\frac{\partial \vec{R}}{\partial r} = \frac{\partial x}{\partial r}\frac{\partial \vec{R}}{\partial x} + \frac{\partial y}{\partial r}\frac{\partial \vec{R}}{\partial y}$$

$$\frac{\partial \vec{R}}{\partial \theta} = \frac{\partial x}{\partial \theta}\frac{\partial \vec{R}}{\partial x} + \frac{\partial y}{\partial \theta}\frac{\partial \vec{R}}{\partial y}$$

Basis Vectors = Partial Derivatives

$$\widetilde{\vec{e_r}} = F_r^x \vec{e_x} + F_r^y \vec{e_y}$$
$$\widetilde{\vec{e_\theta}} = F_\theta^x \vec{e_x} + F_\theta^y \vec{e_y}$$

$$\frac{\partial \vec{R}}{\partial x} = \vec{e_x}$$
$$\frac{\partial \vec{R}}{\partial y} = \vec{e_y}$$

$$\frac{\partial \vec{R}}{\partial r} = \widetilde{\vec{e_r}}$$
$$\frac{\partial \vec{R}}{\partial \theta} = \widetilde{\vec{e_\theta}}$$

$$\frac{\partial \vec{R}}{\partial r} = \cos\theta \frac{\partial \vec{R}}{\partial x} + \sin\theta \frac{\partial \vec{R}}{\partial y}$$

$$\frac{\partial \vec{R}}{\partial \theta} = (-r\sin\theta) \frac{\partial \vec{R}}{\partial x} + (r\cos\theta) \frac{\partial \vec{R}}{\partial y}$$

Basis Vectors = Partial Derivatives

$$\overrightarrow{\widetilde{e_r}} = F_r^x \overrightarrow{e_x} + F_r^y \overrightarrow{e_y}$$
$$\overrightarrow{\widetilde{e_\theta}} = F_\theta^x \overrightarrow{e_x} + F_\theta^y \overrightarrow{e_y}$$

$$\frac{\partial \vec{R}}{\partial x} = \overrightarrow{e_x}$$
$$\frac{\partial \vec{R}}{\partial y} = \overrightarrow{e_y}$$

$$\frac{\partial \vec{R}}{\partial r} = \overrightarrow{\widetilde{e_r}}$$
$$\frac{\partial \vec{R}}{\partial \theta} = \overrightarrow{\widetilde{e_\theta}}$$

$$\frac{\partial \vec{R}}{\partial r} = \cos\theta \frac{\partial \vec{R}}{\partial x} + \sin\theta \frac{\partial \vec{R}}{\partial y}$$

$$\frac{\partial \vec{R}}{\partial \theta} = (-r\sin\theta)\frac{\partial \vec{R}}{\partial x} + (r\cos\theta)\frac{\partial \vec{R}}{\partial y}$$

$$F = \begin{bmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{bmatrix} = J$$

$$F = \begin{bmatrix} x/r & -y \\ y/r & x \end{bmatrix} = J$$

The Jacobian Matrix: Forward Transform

$$F = \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} \end{bmatrix} = J$$

$$\overrightarrow{\widetilde{e_r}} = \frac{\partial x}{\partial r}\overrightarrow{e_x} + \frac{\partial y}{\partial r}\overrightarrow{e_y}$$

$$\overrightarrow{\widetilde{e_\theta}} = \frac{\partial x}{\partial \theta}\overrightarrow{e_x} + \frac{\partial y}{\partial \theta}\overrightarrow{e_y}$$

Inverse Jacobian Matrix: Backward Transform

$$B = \begin{bmatrix} \frac{\partial r}{\partial x} & \frac{\partial r}{\partial y} \\ \frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial y} \end{bmatrix} = J^{-1}$$

$$\vec{e_x} = \frac{\partial r}{\partial x}\overrightarrow{\widetilde{e_r}} + \frac{\partial \theta}{\partial x}\overrightarrow{\widetilde{e_\theta}}$$

$$\vec{e_y} = \frac{\partial r}{\partial y}\overrightarrow{\widetilde{e_r}} + \frac{\partial \theta}{\partial y}\overrightarrow{\widetilde{e_\theta}}$$

Change of Variables

Let $g : [a, b] \mapsto \mathbb{R}$ be continuously differentiable and $f : \mathbb{R} \mapsto \mathbb{R}$ continuous. Then:

$$\int_{x=g(a)}^{x=g(b)} f(x)dx = \int_{u=a}^{u=b} (f(g(u))) \cdot g'(u)du \rightarrow \int_{g(a)}^{g(b)} f = \int_{a}^{b} (f \circ g) \cdot g'.$$

Proof: Write $F' = f$. Then the left side is:

$$F(g(b)) - F(g(a)).$$

Also:

$$(F \circ g)' = (f \circ g) \cdot g'$$

The right hand side is also:

$$F \circ g(b) - F \circ g(a) = F(g(b)) - F(g(a)). \quad \square$$

Change of Variables

Let $A \subset \mathbb{R}^n$ be an open set and $g : A \mapsto \mathbb{R}^n$ be 1-1, continuously differentiable, such that $\det g'(x) \neq 0 \, \forall x \in A$. If $f : g(A) \mapsto \mathbb{R}$ is integrable, then:

$$\int\limits_{g(A)} f = \int\limits_{A} (f \circ g) |\det g'|.$$

The proof is more involved and will be omitted.

Change of Variables

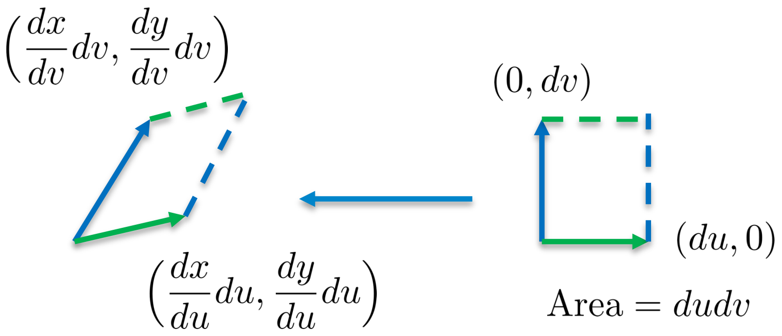
For $x = g(u,v)$ and $y = h(u,v)$, substitution yields:

$$\int\int_R f(x,y)dxdy = \int\int_G f(g(u,v),h(u,v))\underbrace{J(u,v)dudv}$$

Jacobian matrix and
'Jacobian
are different things.

where

$$J = \begin{vmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \end{vmatrix}$$



$$\text{Area} = J(u,v)dudv$$

Let’s Optimize / Train

Machine Learning: $f(x, w) \approx y \xrightarrow{\text{training}} w^\star$ (*Optimization*)

Model hypothesis: f

Parameters: w

Data space: $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$

Loss function: ℓ

$$w^\star = \arg\min_{w} \underbrace{\sum_{i=1}^{n} \ell\big(f(x_i, w), y_i\big)}_{\textbf{Empirical Risk}\ \hat{\mathcal{R}}\ \text{on}\ n\ \text{training samples}\ \mathcal{S} \in \mathcal{Z}^n}$$

$$\hat{\mathcal{R}}(w, S) := \frac{1}{n} \sum\nolimits_{i=1}^{n} \ell(w, z_i)$$

Learning

Machine Learning: $f(x, w) \approx y \xrightarrow{\text{training}} w^\star$ (*Optimization*)

Model hypothesis: f

Parameters: w

Data space: $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$

Loss function: ℓ

$$w^\star = \arg\min_w \underbrace{\sum_{i=1}^{n} \ell\big(f(x_i, w), y_i\big)}$$

Empirical Risk $\hat{\mathcal{R}}$ on n training samples $\mathcal{S} \in \mathcal{Z}^n$

$$\hat{\mathcal{R}}(w, S) := \frac{1}{n} \sum_{i=1}^{n} \ell(w, z_i)$$

Machine Learning: $f(x, w) \approx y \xrightarrow{\text{training}} \theta^\star$ $(Optimization)$

How do we optimize?

Training algorithm: $\mathcal{A}(\mathcal{S}, U)$

Algorithmic randomness

Algorithmic trajectories:

$$[\mathcal{A}(S, U)]_t = \mathrm{W}_t^{(S)} \qquad t \in [0, 1]$$

Training trajectory

Learning

Machine Learning: $f(x,\theta) \approx y \xrightarrow{\text{training}} \theta^\star$ $(Optimization)$

Model hypothesis: f

Parameters: w

Data space: $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$

Loss function: ℓ

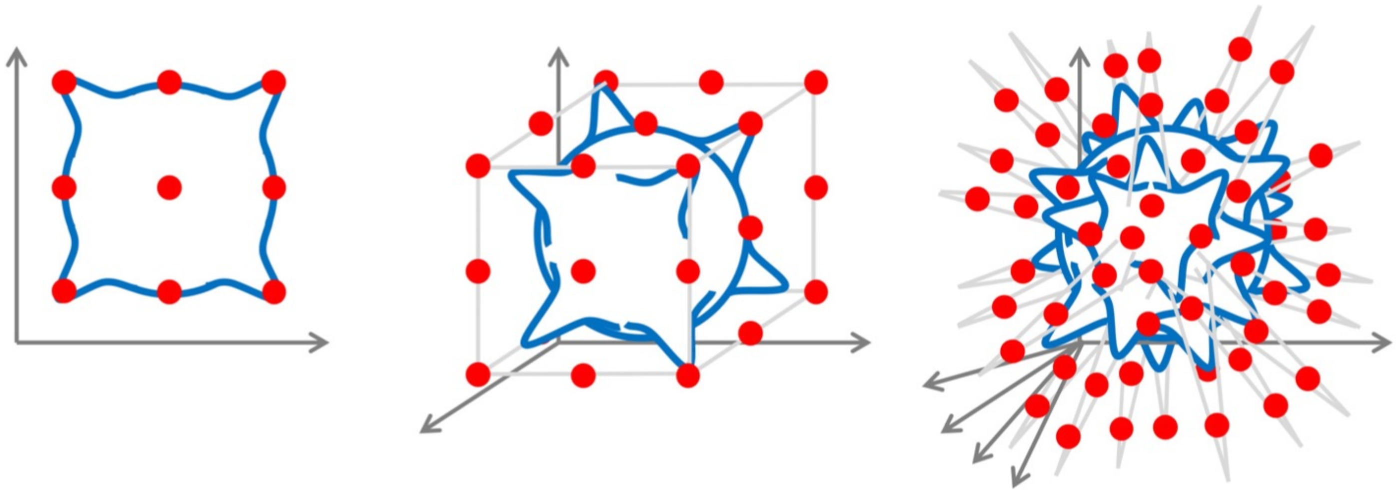
Inductive bias on the parameters

$$w^\star = \arg\min_w \sum_{i=1}^{n} \Big(\underbrace{\ell(f(x_i, w), y_i)}_{\text{data}} + \underbrace{h(w)}_{\text{regularizer}} \Big)$$

Inductive bias on the hypotheses class

$$\hat{f} = \arg\min_{f \in \mathcal{F}_\delta} \hat{\mathcal{R}}(f)$$

Curse of Dimensionality



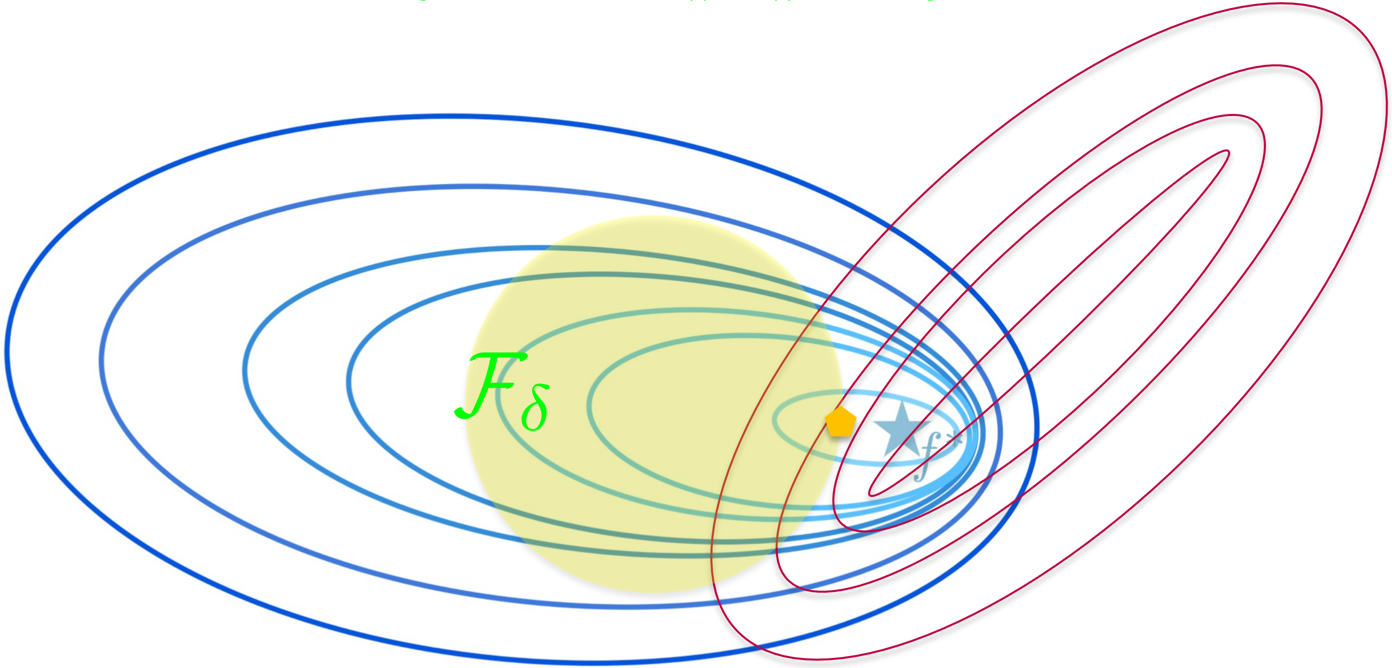
Population Loss (Risk): $\mathcal{R}(f) = \mathbb{E}[\ell(f(x), y)]$

Empirical Loss (Risk): $\hat{\mathcal{R}}(f) = \frac{1}{n} \sum_{i=1}^{n} [\ell(f(x_i), y_i)]$

Hypotheses Class: $\mathcal{F}_\delta = \{f = f_\delta : \|w\| \leq \delta\}$

Empirical Risk Minimization:

$\hat{f} = \arg\min_{f \in \mathcal{F}_\delta} \hat{\mathcal{R}}(f)$



Prefer Networks that "Generalize" Better

Generalization Gap

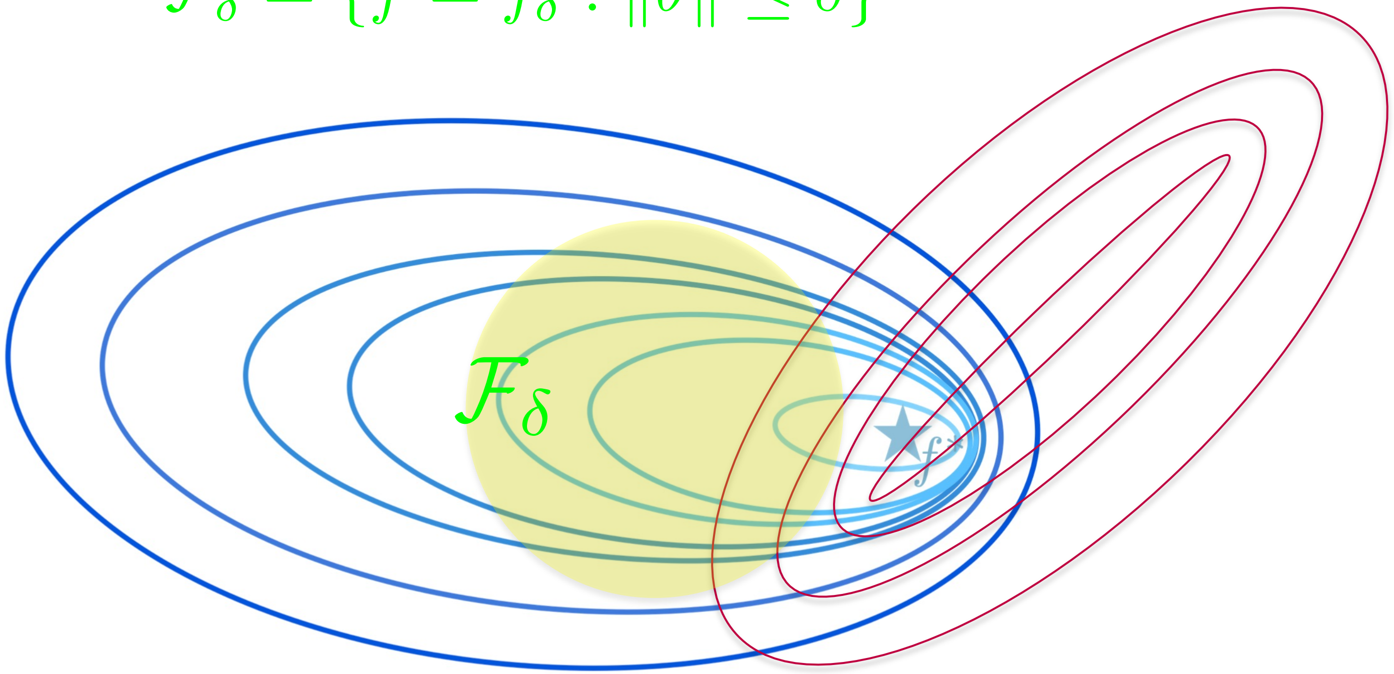
$$\sup_{w \in \mathcal{W}_S} |\hat{R}(w, S) - \underbrace{R(w)}_{\boxed{\mathcal{R}(w) := \mathbb{E}_z[\ell(w, z)]}}|$$

Population Loss (Risk): $\mathcal{R}(f) = \mathbb{E}[\ell(f(x), y)]$

Empirical Risk Minimization:

Empirical Loss (Risk): $\hat{\mathcal{R}}(f) = \frac{1}{n}\sum_{i=1}^{n}[\ell(f(x_i), y_i)]$

Hypotheses Class: $\mathcal{F}_\delta = \{f = f_\delta : \|\theta\| \leq \delta\}$



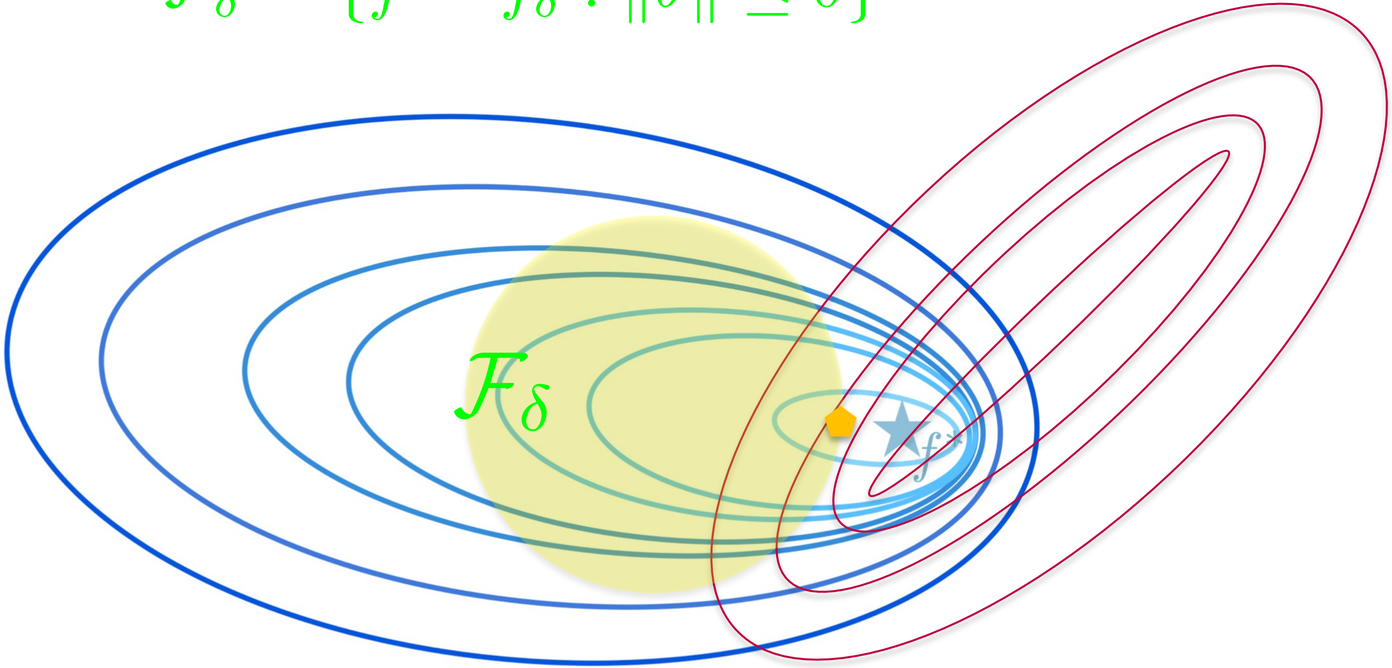
Population Loss (Risk): $\mathcal{R}(f) = \mathbb{E}[\ell(f(x), y)]$

Empirical Loss (Risk): $\hat{\mathcal{R}}(f) = \frac{1}{n}\sum_{i=1}^{n}[\ell(f(x_i), y_i)]$

Hypotheses Class: $\mathcal{F}_\delta = \{f = f_\delta : \|\theta\| \le \delta\}$

Empirical Risk Minimization:

$\hat{f} = \arg\min_{f \in \mathcal{F}_\delta} \hat{\mathcal{R}}(f)$



Your First Optimizer

Random Search (Rastgele Arama)

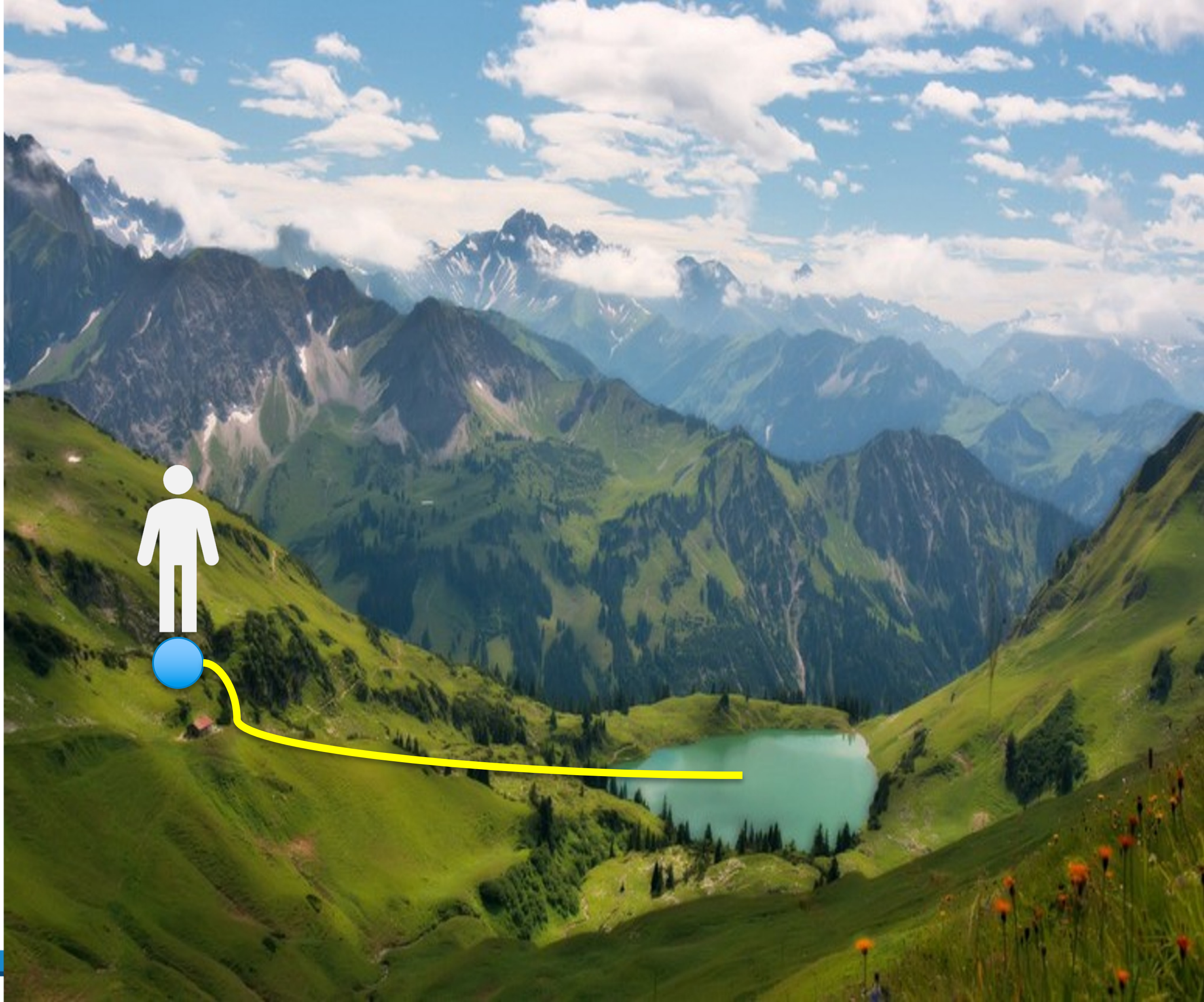
```
# assume X_train is the data where each column is an example (e.g. 3073 x 50,000)
# assume Y_train are the labels (e.g. 1D array of 50,000)
# assume the function L evaluates the loss function

bestloss = float("inf") # Python assigns the highest possible float value
for num in xrange(1000):
  W = np.random.randn(10, 3073) * 0.0001 # generate random parameters
  loss = L(X_train, Y_train, W) # get the loss over the entire training set
  if loss < bestloss: # keep track of the best solution
    bestloss = loss
    bestW = W
  print 'in attempt %d the loss was %f, best %f' % (num, loss, bestloss)

# prints:
# in attempt 0 the loss was 9.401632, best 9.401632
# in attempt 1 the loss was 8.959668, best 8.959668
# in attempt 2 the loss was 9.044034, best 8.959668
# in attempt 3 the loss was 9.278948, best 8.959668
# in attempt 4 the loss was 8.857370, best 8.857370
# in attempt 5 the loss was 8.943151, best 8.857370
# in attempt 6 the loss was 8.605604, best 8.605604
# ... (trunctated: continues for 1000 lines)
```

```python
# Assume X_test is [3073 x 10000], Y_test [10000 x 1]
scores = Wbest.dot(Xte_cols) # 10 x 10000, the class scores for all test examples
# find the index with max score in each column (the predicted class)
Yte_predict = np.argmax(scores, axis = 0)
# and calculate accuracy (fraction of predictions that are correct)
np.mean(Yte_predict == Yte)
# returns 0.1555
```

15.5% accuracy! not bad!
(State of the art is ~95%)



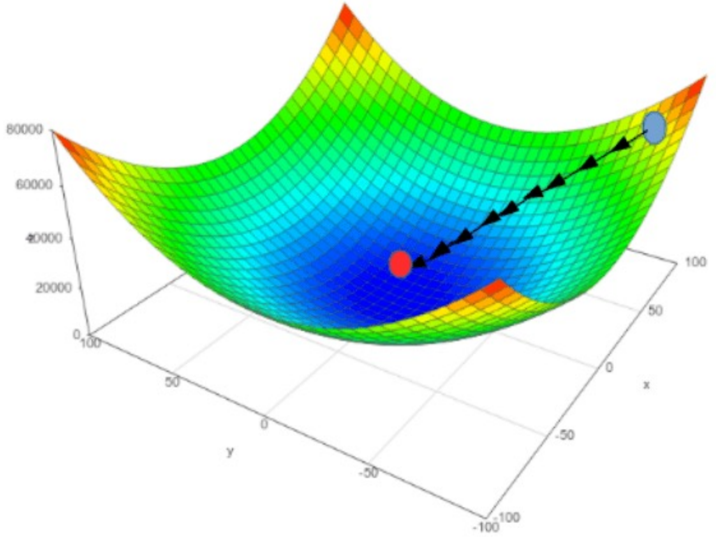
Algorithms: Descent Following

$$x_{k+1} = x_k + \Delta x_k = x_k + \tau_k x_k$$

movement

step

previous iterate

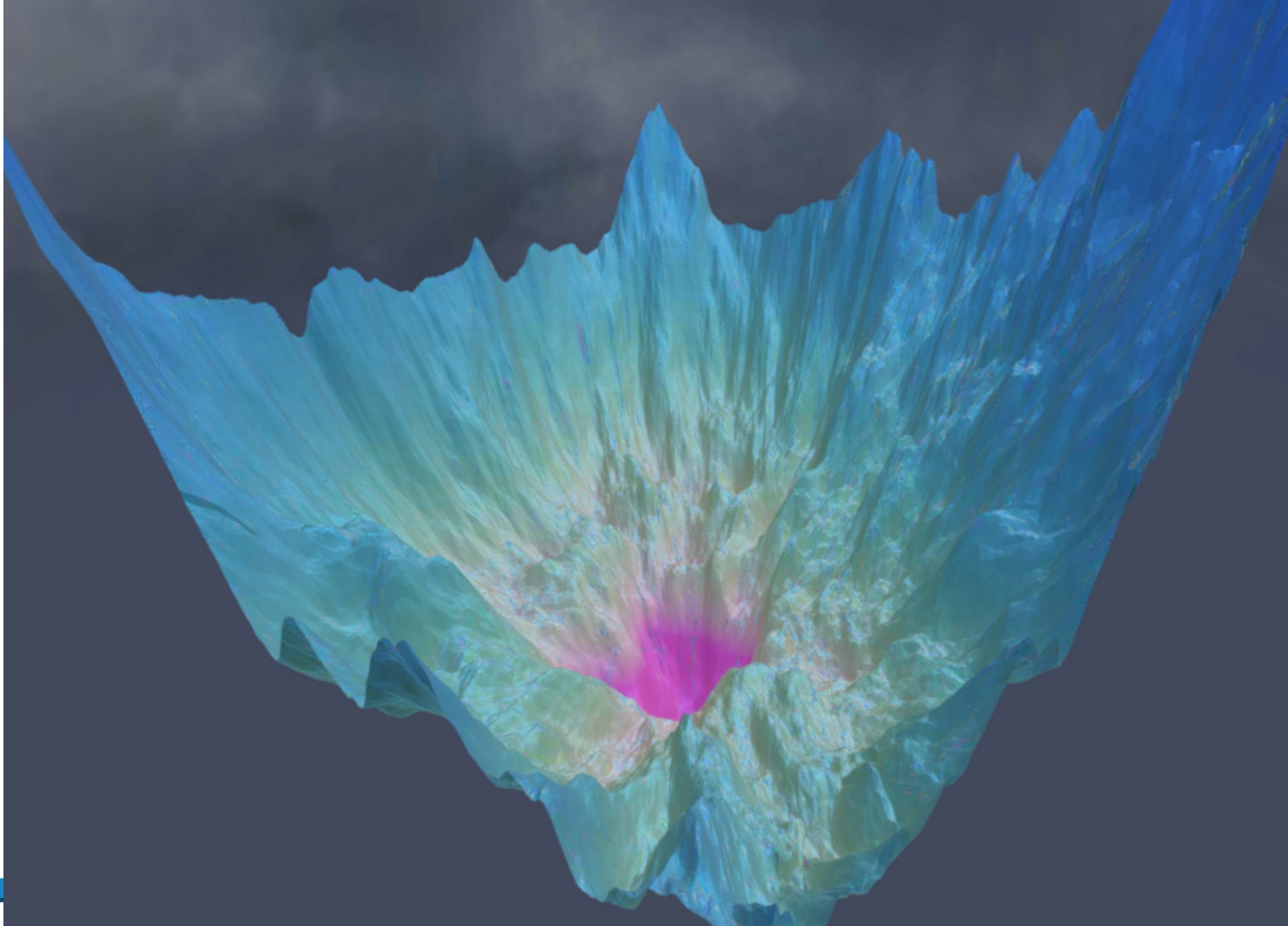


https://losslandscape.com/ by Javier Ideami

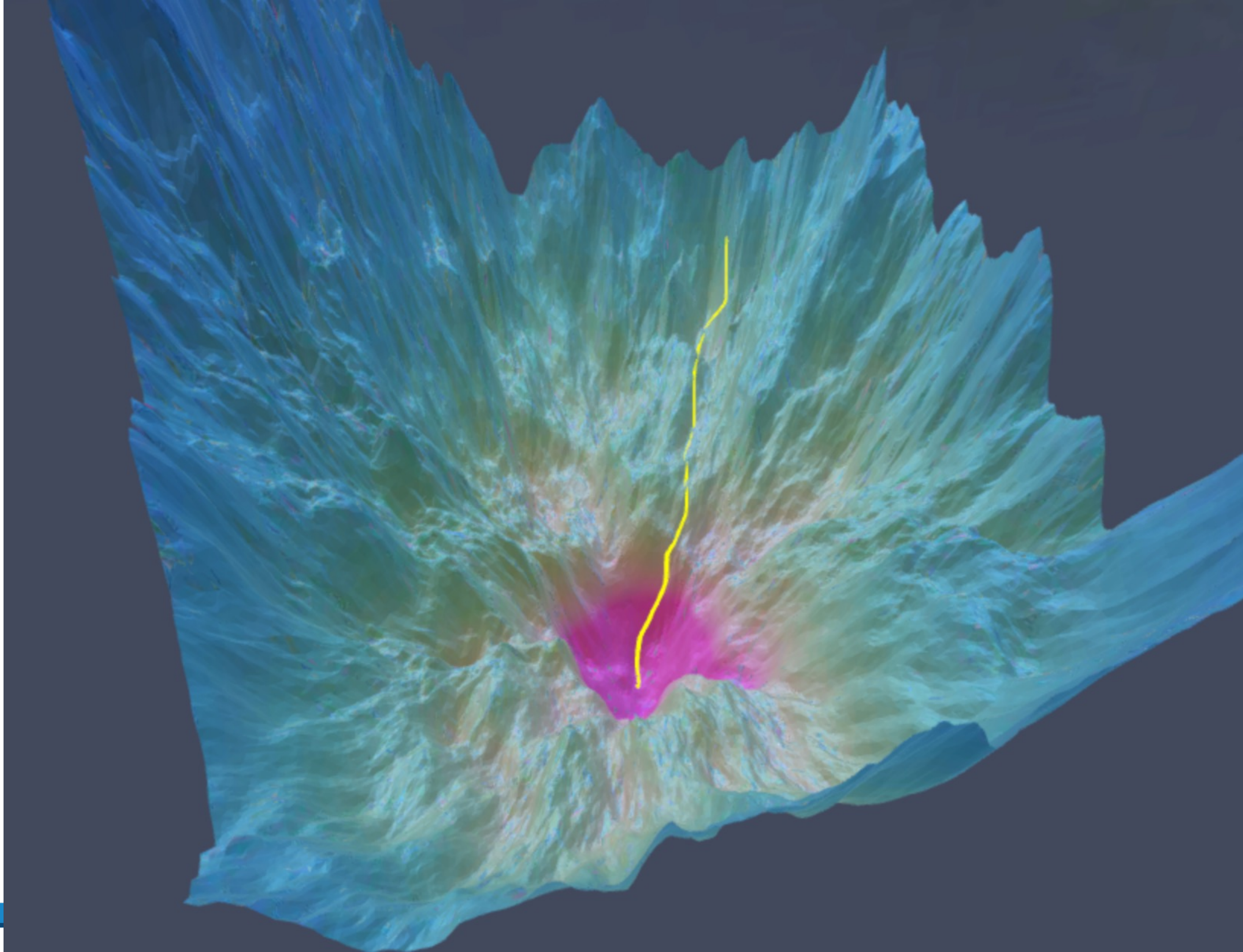
https://losslandscape.com/ by Javier Ideami

A lucky situation:

Convex up to noise

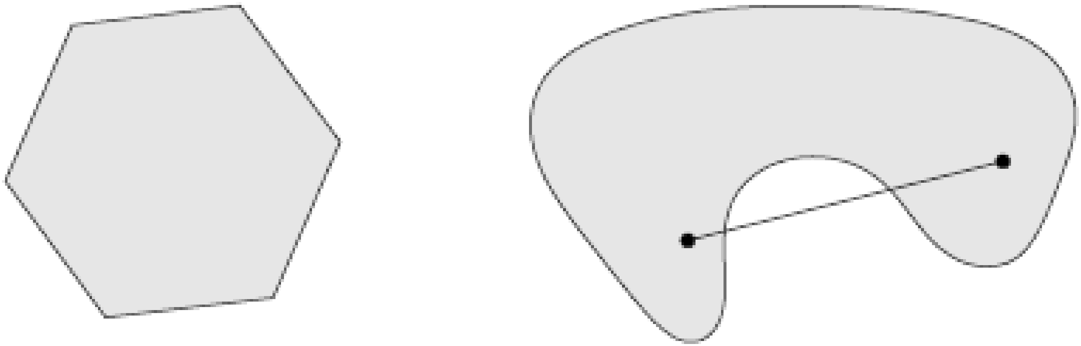


A Descent Trajectory



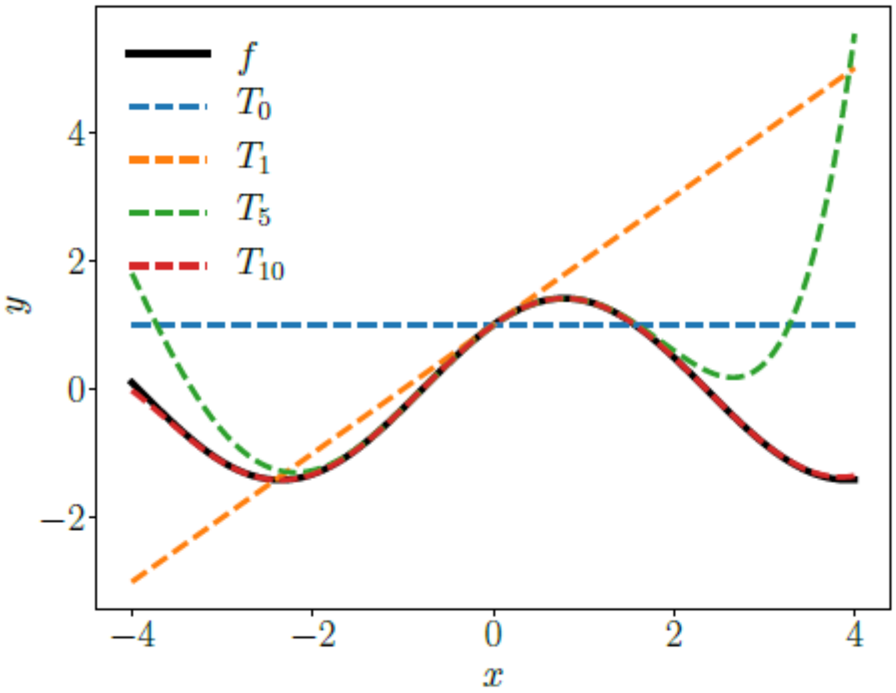
Convexity

A set C is convex if a line segment between any two points in C lies in C.



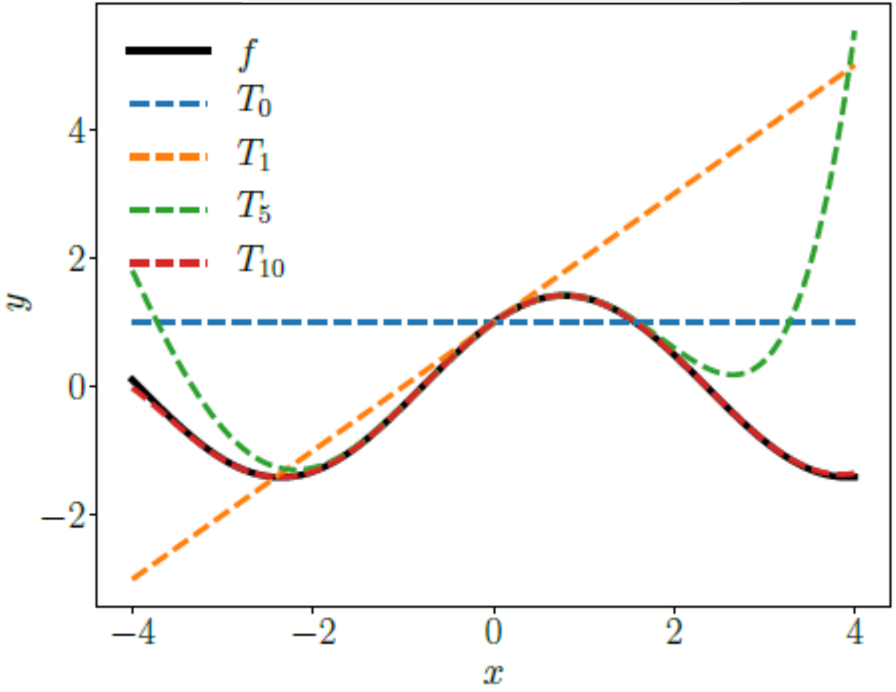
So, let’s start with Newton. But first: Taylor Series

$$T_\infty(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k$$



So, let’s start with Newton. But first: Taylor Series

$$f(x) = \sin(x) + \cos(x)$$



$$T_\infty(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k$$
$$= 1 + x - \frac{1}{2!}x^2 - \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \frac{1}{5!}x^5 - \cdots$$

So, let's start with Newton.

Based on the Taylor Series for $f(x+h)$:

$$f(x+h) = f(x) + hf'(x) + O(h^2)$$

To find a zero of f, assume $f(x+h) = 0$, so

$$h \approx -\frac{f(x)}{f'(x)}$$

And as an iteration:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

So, let's start with Newton.

For zeros of $f(x)$:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

At a local optima, $f'(x) = 0$, so we use:

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$$

If $f''(x)$ is constant (f is quadratic), then Newton's method finds the optimum in ***one step***. Generally, it has ***quadratic converge.***

So, let's start with Newton.

To find an optimum of a function $f(x)$ for high-dimensional x, we want zeros of its gradient: $\nabla f(x) = 0$

For zeros of $\nabla f(x)$ with a vector displacement h, Taylor's expansion is:

$$\nabla f(x+h) = \nabla f(x) + h^T H_f(x) + O(||h||^2)$$

Where H_f is the Hessian matrix of second derivatives of f. The update is:

$$x_{n+1} = x_n - H_f(x_n)^{-1} \nabla f(x_n)$$

Newton's Method

The Newton update is:

$$x_{n+1} = x_n - H_f(x_n)^{-1} \nabla f(x_n)$$

Converges very fast, but rarely used in DL. Why?

Too expensive: if x_n has dimension M, the Hessian $H_f(x_n)$ has dimension M^2 and takes $O(M^3)$ time to invert.

Too unstable: it involves a high-dimensional matrix inverse, which has poor numerical stability. The Hessian may even be singular.

$$x_{k+1} = x_k + \Delta x_k = x_k + \tau_k x_k$$

movement

step

previous iterate

Algorithms

- Steepest (Gradient) Descent:

update step size

$$x_{k+1} = x_k - \tau_k \nabla f(x_k)$$

- Newton's Method:

Hessian Gradient

$$x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$$

- Other methods to compute the direction and the step size.

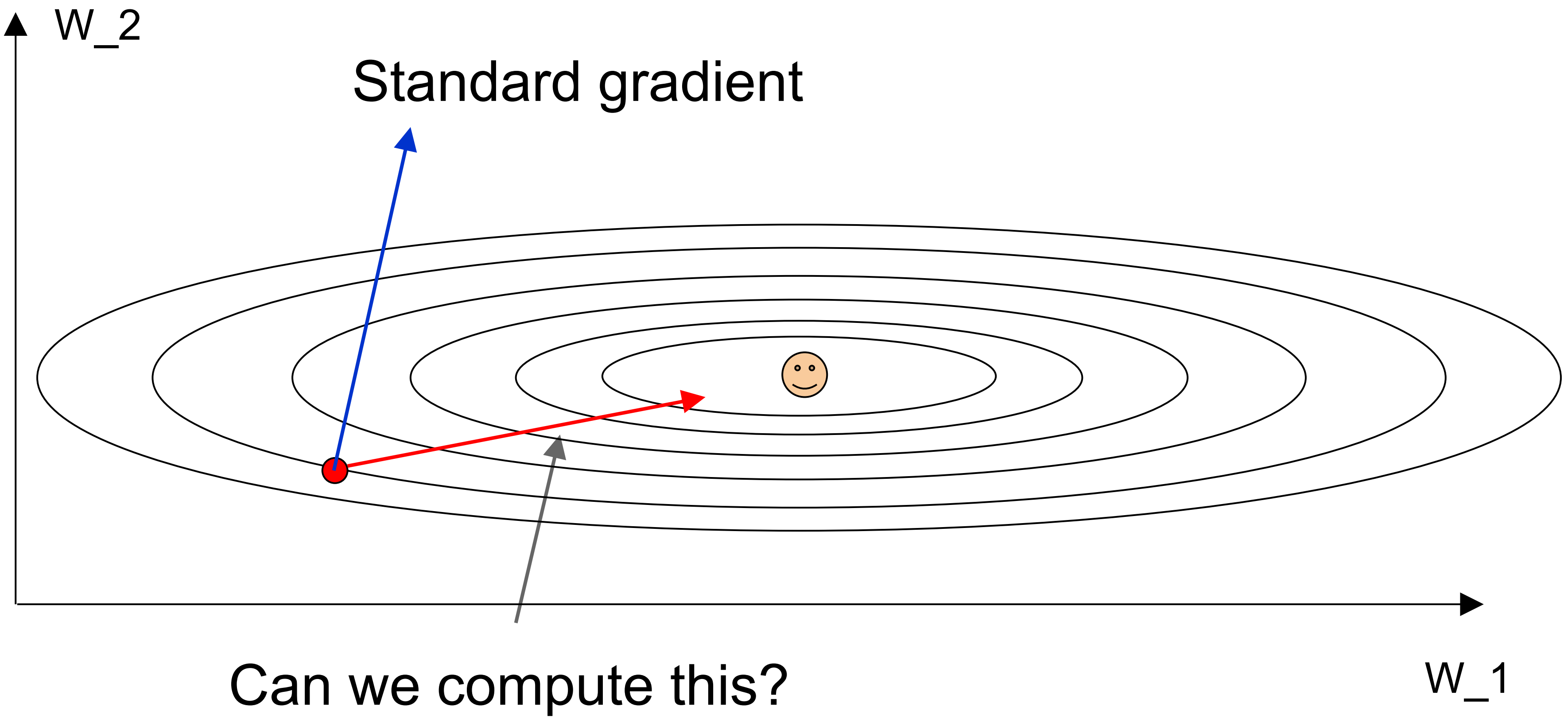
BFGS is Broyden-Fletcher-Goldfarb-Shanno

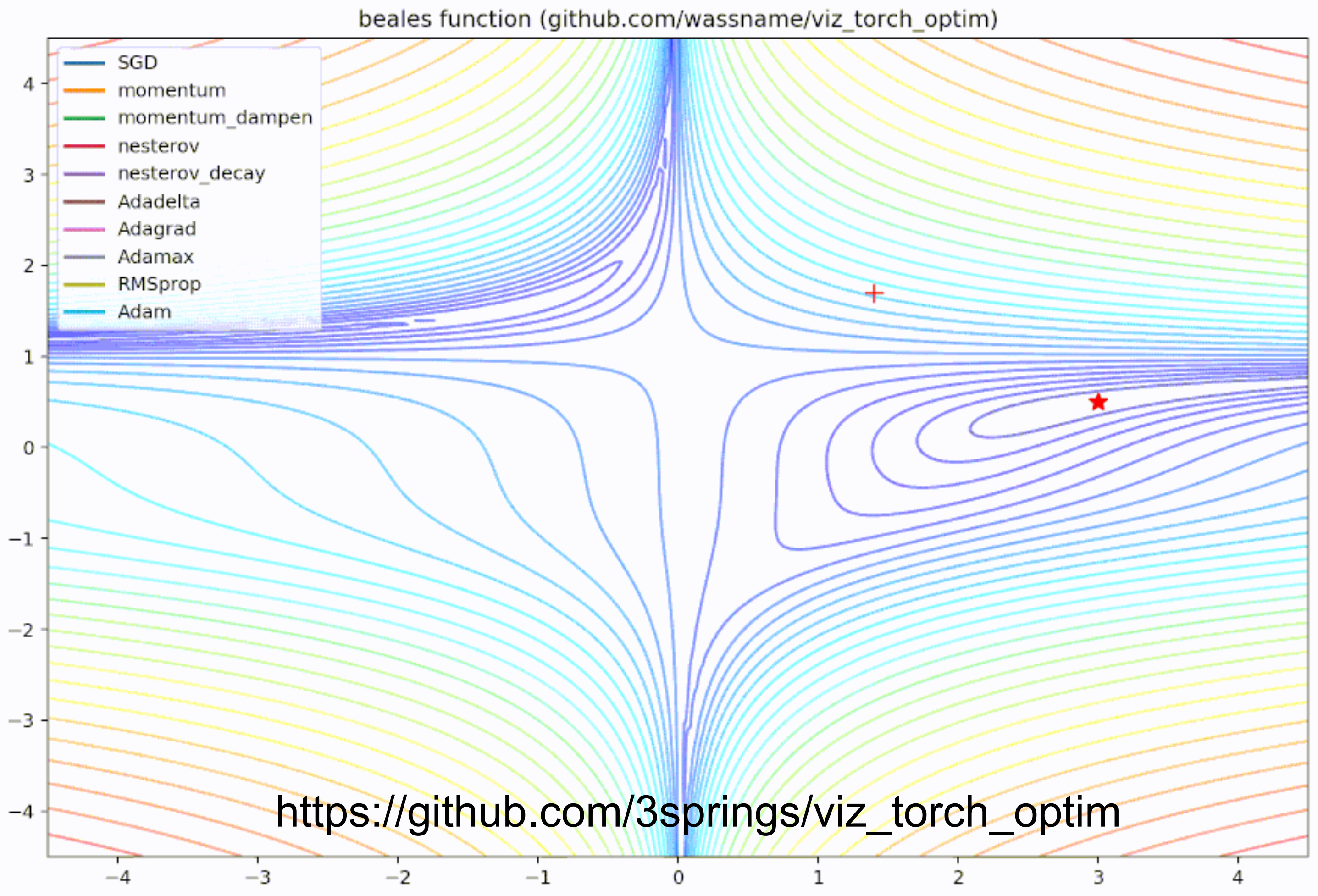
Idea: compute a low-dimensional approximation of $H_f(x_n)^{-1}$ directly.

Expense: use a k-dimensional approximation of $H_f(x_n)^{-1}$,
Size is O(kM), cost is O(k^2 M).

Stability: much better. Depends on largest singular values of $H_f(x_n)$.

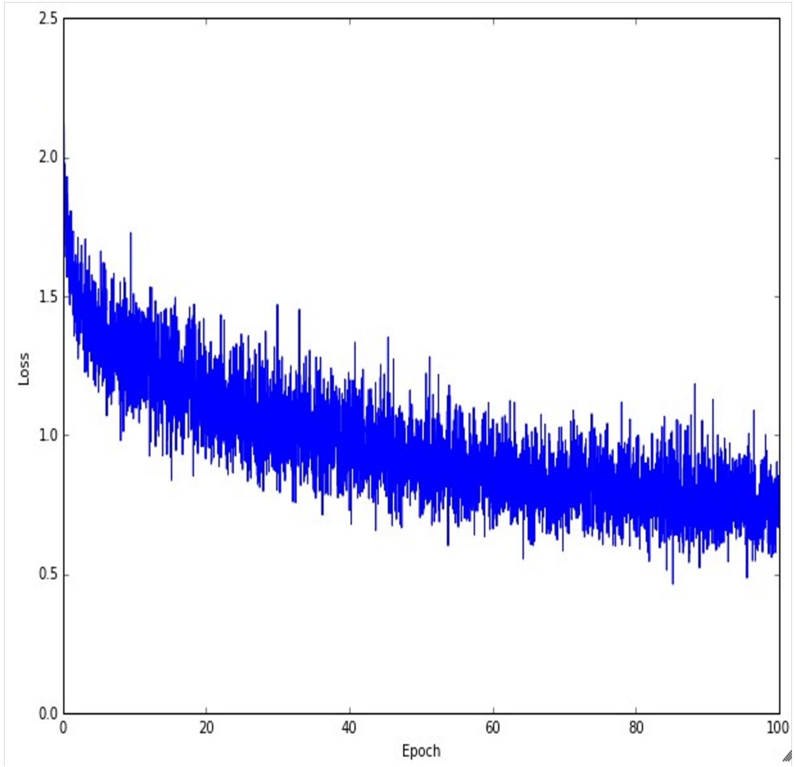
This question will lead to a never-ending human endeavor.

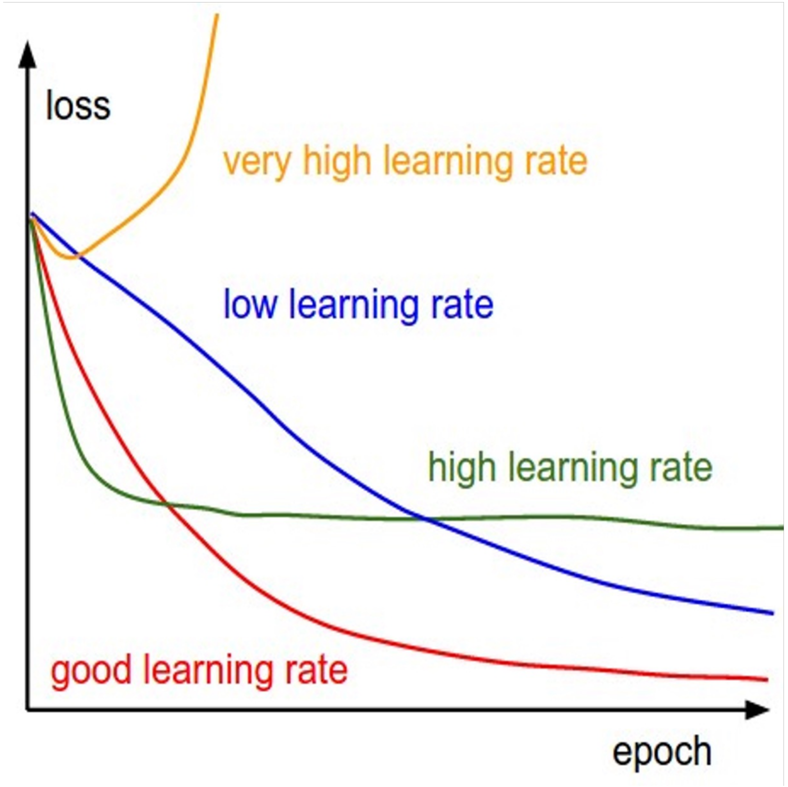




https://github.com/3springs/viz_torch_optim

“Learning Rate” / Step Size





Probabilistic View

$$\theta^\star = \arg\min_\theta \sum_{i=1}^{n} \Big(\underbrace{\ell(f(x_i,\theta), y_i)}_{\text{data}} + \underbrace{h(\theta)}_{\text{regularizer}} \Big)$$

$$\underbrace{p(\mathcal{D} \,|\, \theta)}_{\text{likelihood}} \propto \exp\left(-\frac{1}{n} \sum_{i=1}^{n} \Big(\mathcal{L}(f(x_i,\theta), y) \Big) \right)$$

$$\underbrace{p(\theta)}_{\text{prior}} \propto \exp\Big(-h(\theta) \Big)$$

Maximum A Posteriori (MAP) Estimate

$$\theta^\star = \arg\max_\theta \log \Big(\underbrace{p(\mathcal{D} \,|\, \theta)\, p(\theta)}_{\text{posterior}} \Big)$$

Probabilistic View

Bayesian Posterior:

$$p(\theta \,|\, \mathcal{D}) = \frac{p(\mathcal{D} \,|\, \theta)\, p(\theta)}{p(\mathcal{D})} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}$$

$$\downarrow$$

$$p(\mathcal{D}) = \int_{\theta} p(\mathcal{D} \,|\, \theta)\, p(\theta)\, d\theta$$

Cannot ignore.

Bayesian Inference is hard because **integration is hard**.

a, b, c	Scalar (integer or real)
$\mathbf{x}, \mathbf{y}, \mathbf{z}$	Vector (bold-font, lower case)
$\mathbf{A}, \mathbf{B}, \mathbf{C}$	Matrix (bold-font, upper-case)
A, **B**, **C**	Tensor ((bold-font, upper-case)
X, Y, Z	Random variable (normal font, upper-case)
$a \in \mathcal{A}$	Set membership: a is member of set $\mathcal{A}$
$\lvert\mathcal{A}\rvert$	Cardinality: number of items in set $\mathcal{A}$
$\lVert\mathbf{v}\rVert$	Norm of vector $\mathbf{v}$
$\mathbf{u} \cdot \mathbf{v}$ or $\langle \mathbf{u}, \mathbf{v} \rangle$	Dot product of vectors $\mathbf{u}$ and $\mathbf{v}$
$\mathbb{R}$	Set of real numbers
$\mathbb{R}^n$	Real numbers space of dimension *n*
$y = f(x)$ or $x \mapsto f(x)$	Function (map): assign a unique value $f(x)$ to each input x
$f: \mathbb{R}^n \to \mathbb{R}$	Function (map): map an *n*-dimensional vector into a scalar