Machine Learning

Lecture 1

School on Data Science in Fundamental Physics, IGFAE/USC, Spain

Dr. Pietro Vischia
pietro.vischia@cern.ch
@pietrovischia
https://vischia.github.io/

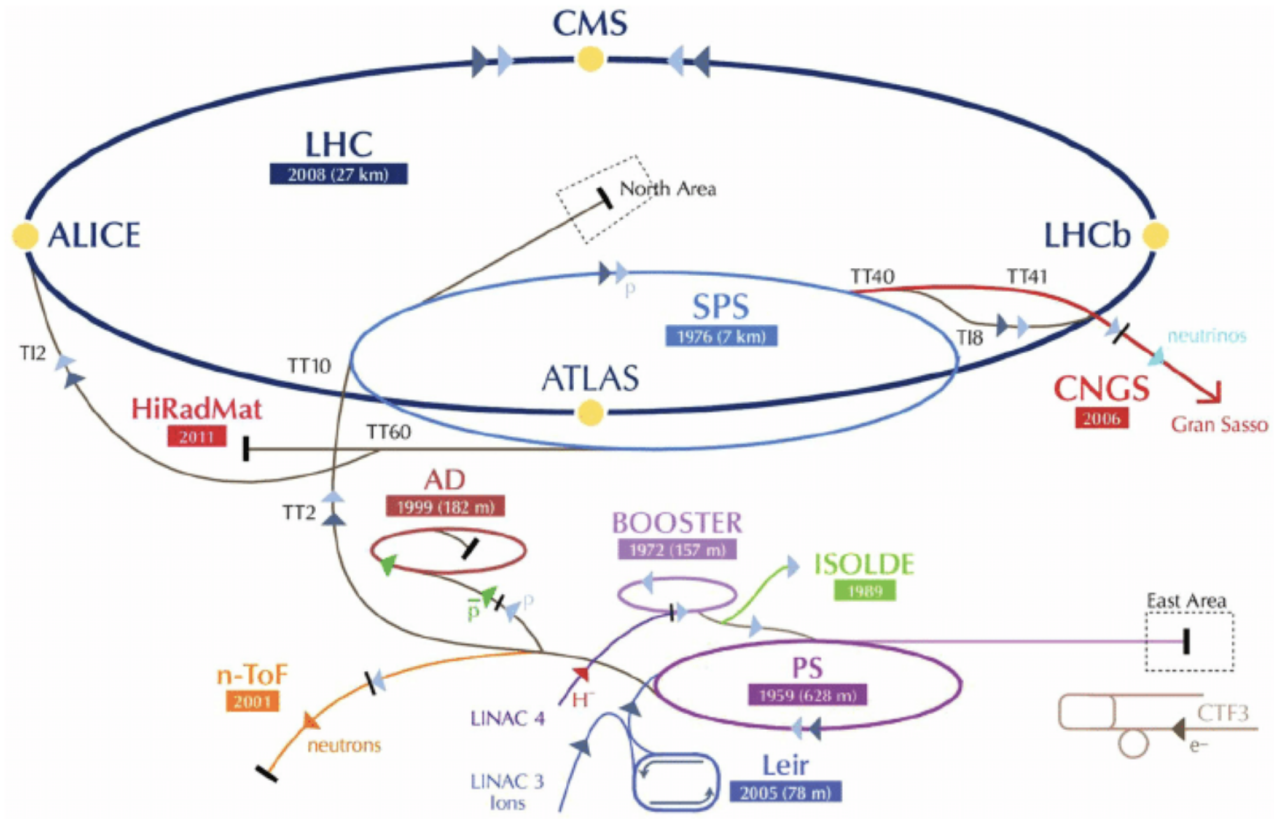
Supported by project
RYC2021- 033305-I
funded by

If you are reading this as a web page: have fun! If you are reading this as a PDF: please visit

https://www.hep.uniovi.es/vischia/persistent/2024-06-03to07_MachineLearningAtDataScienceSchoolIGFAE_vischia_1.html

to get the version with working animations

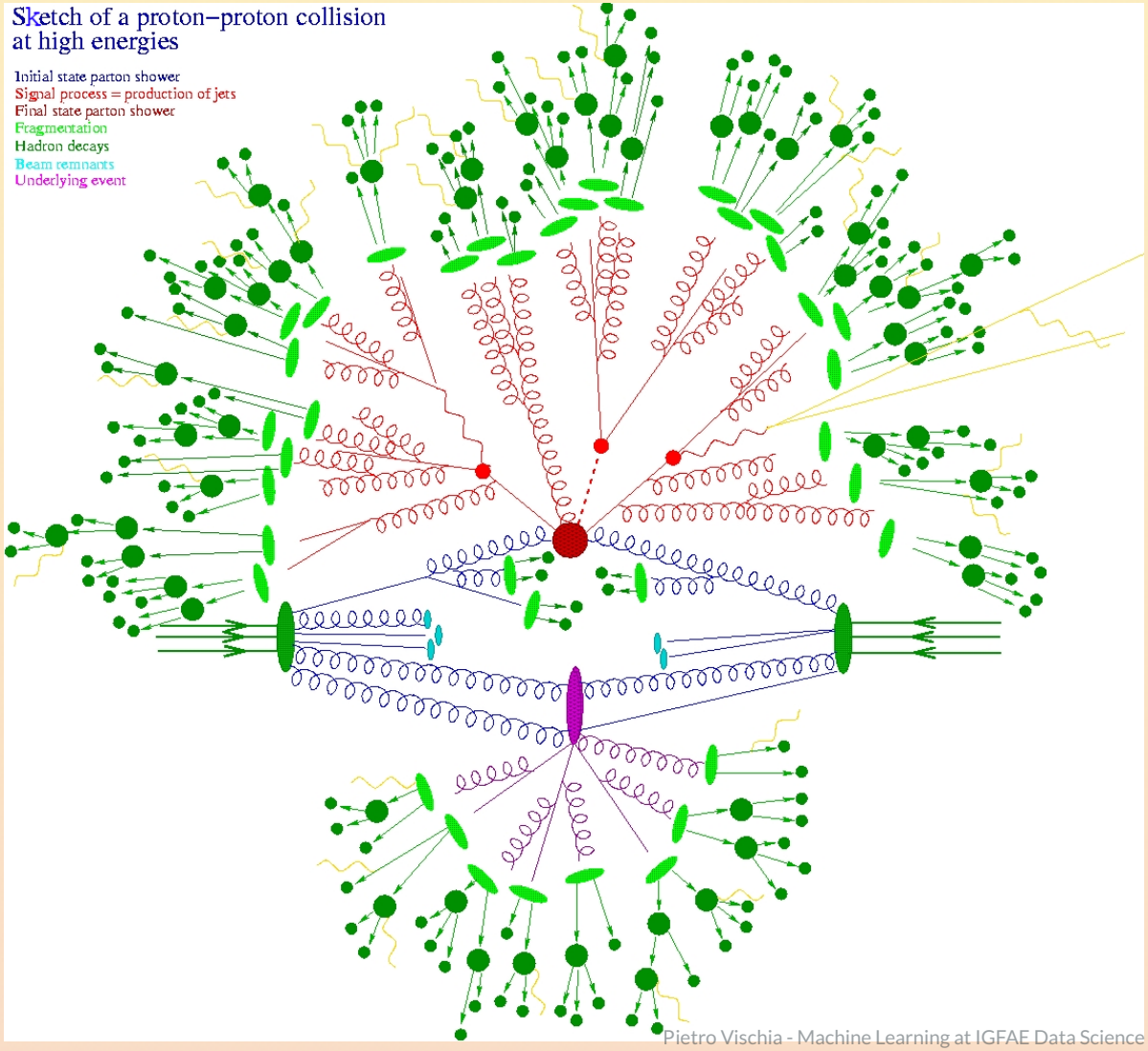
Complex accelerators



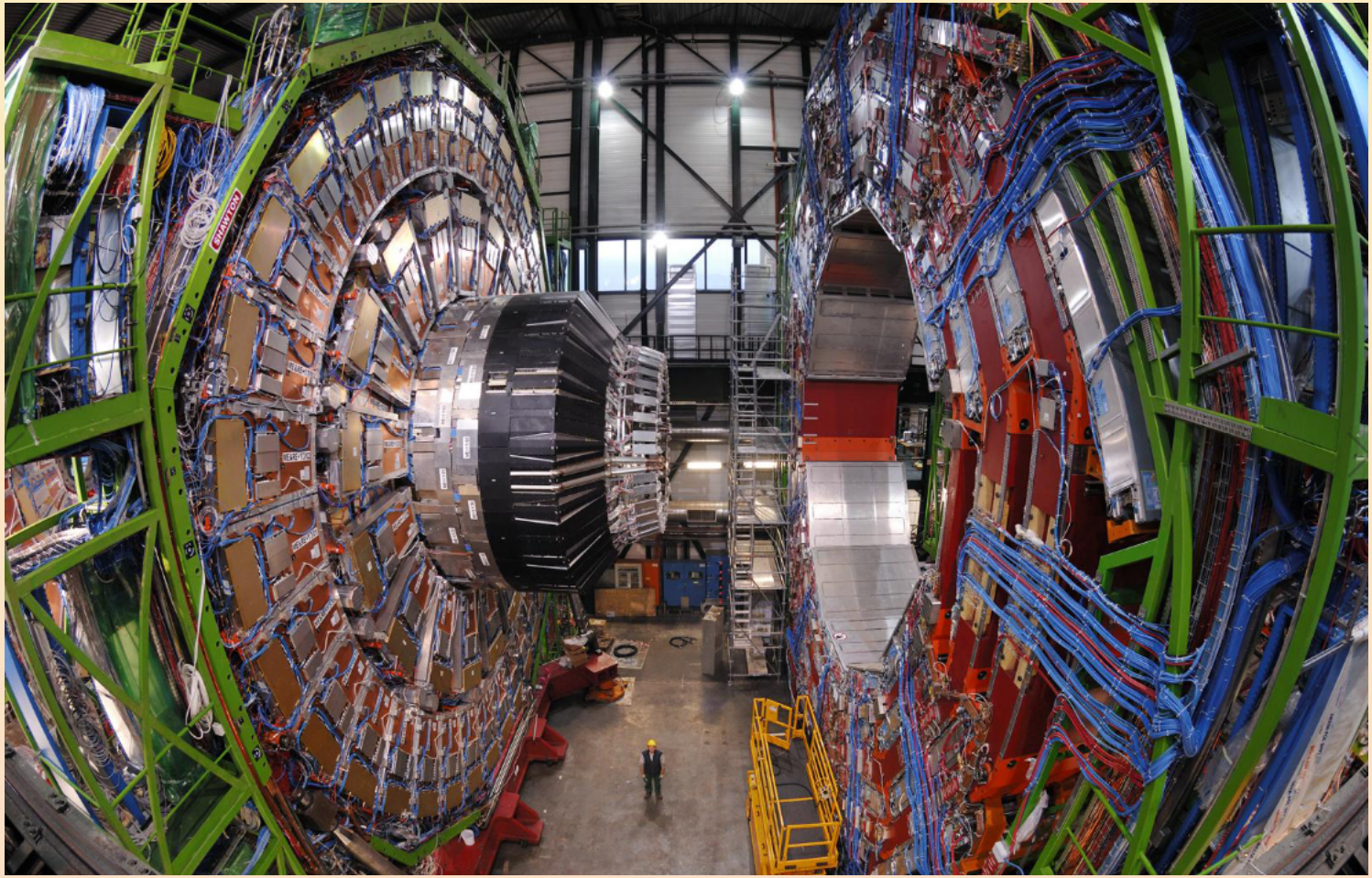
Existing CERN accelerator complex with Large Hadron Collider (LHC), Super Proton Synchrotron (SPS), Proton Synchrotron (PS), Antiproton Decelerator (AD), Low Energy Ion Ring (LEIR), Linear Accelerators (LINAC), CLIC Test Facility (CTF3), CERN to Gran Sasso (CNGS), Isotopes Separation on Line (ISOLDE), and neutrons Time of Flight (n-ToF).

—
Image from 10.1103/PhysRevSTAB.16.054801

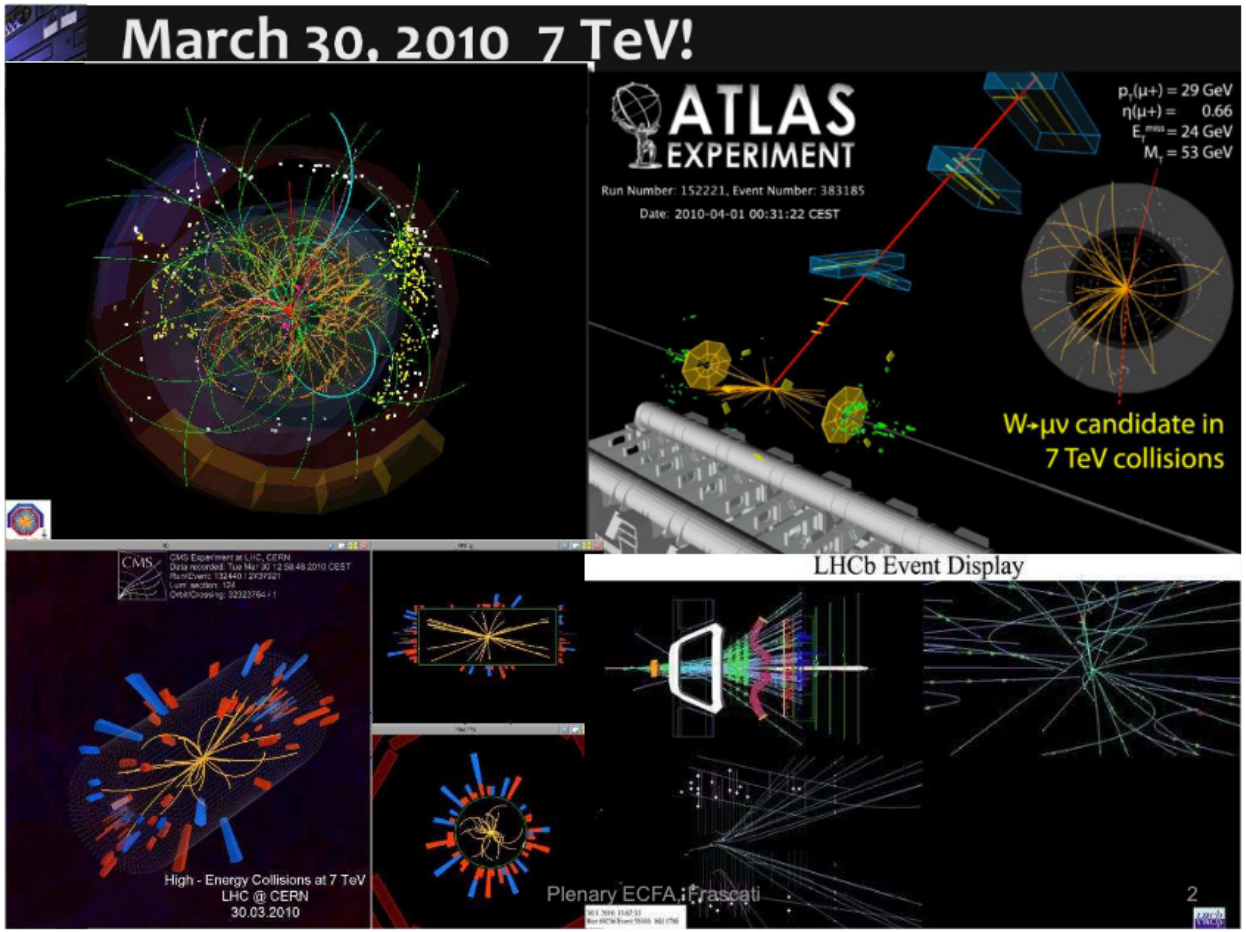
Complex phenomena



Complex Experiments



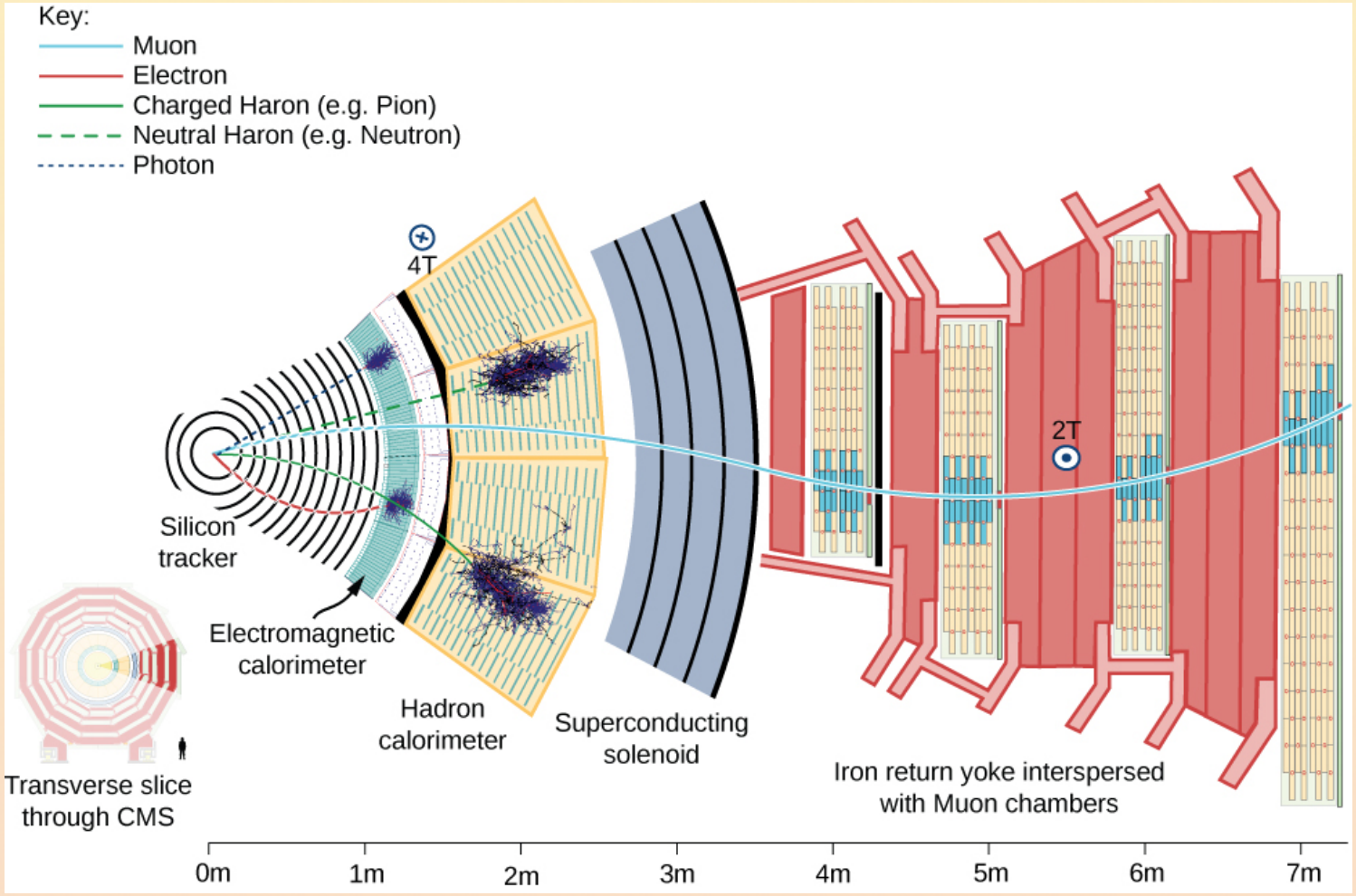
What we measure...



These are the particles that hit the detector

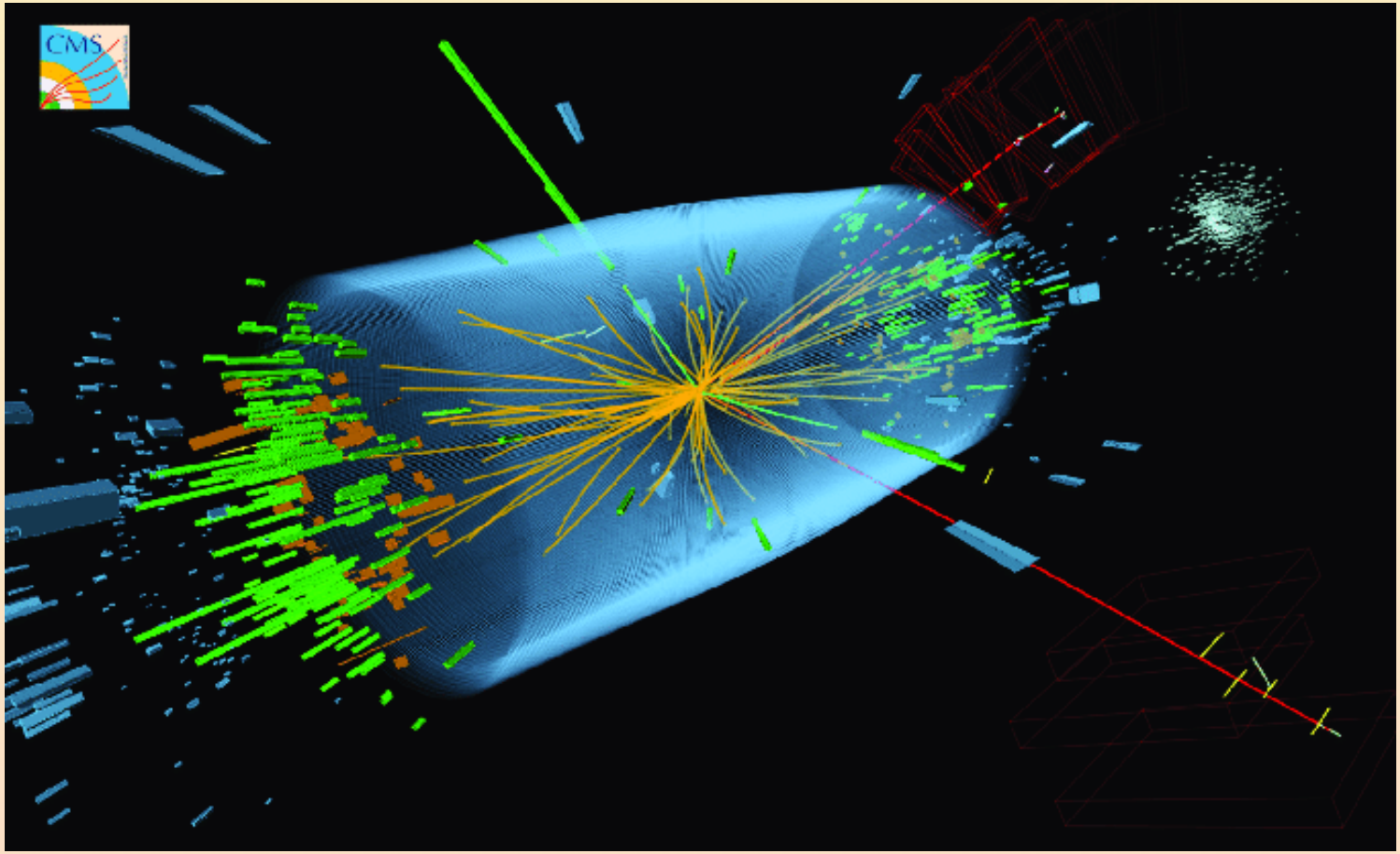
Pictorial representations by Torbjörn Sjöstrand

...40 Million Times per Second



Images from the CMS Collaboration

Complex Data



Likelihood and information

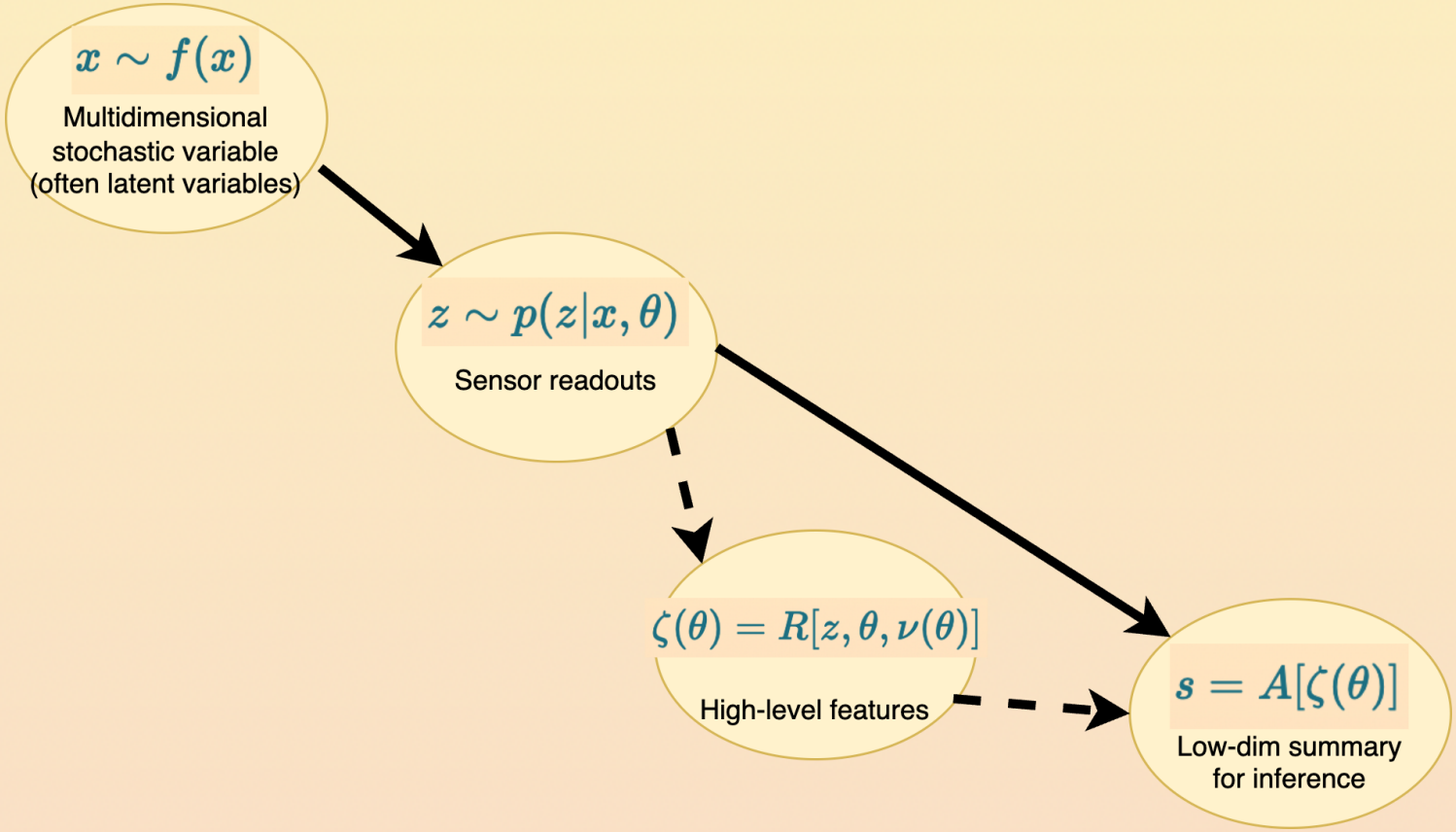
- Data sample X_{obs}

$$\mathcal{L}(X;\theta) := P(X|\theta)|_{X_{obs}}$$

- The Likelihood Principle: The likelihood function $L(\vec{x};\theta)$ contains all the information available in the data sample relevant for the estimation of θ
  - ✓ Bayesian statistics
  - ✗ Frequentist statistics

$$I(\theta) = -E\left[\left(\frac{\partial^2}{\partial\theta^2} lnL(X;\theta)\right)^2 |\theta_{true}\right]$$

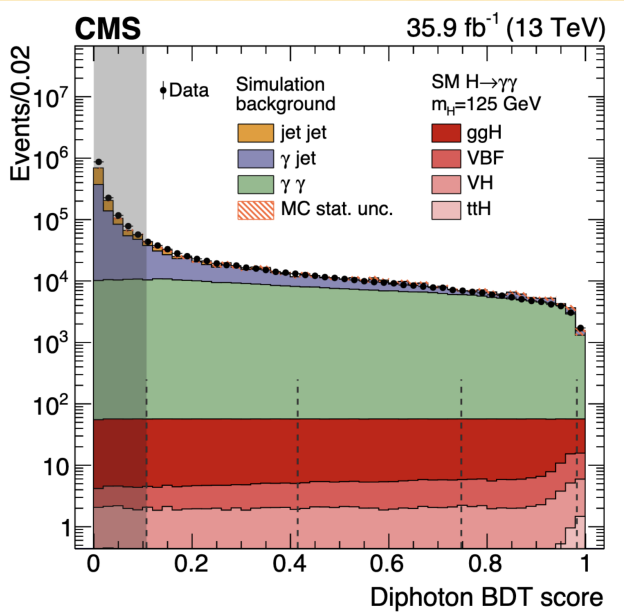
Typical analysis pipeline



–
Formulas from our white paper

We like low-dim summaries

- Discard uninteresting regions



- Physical observable for inference

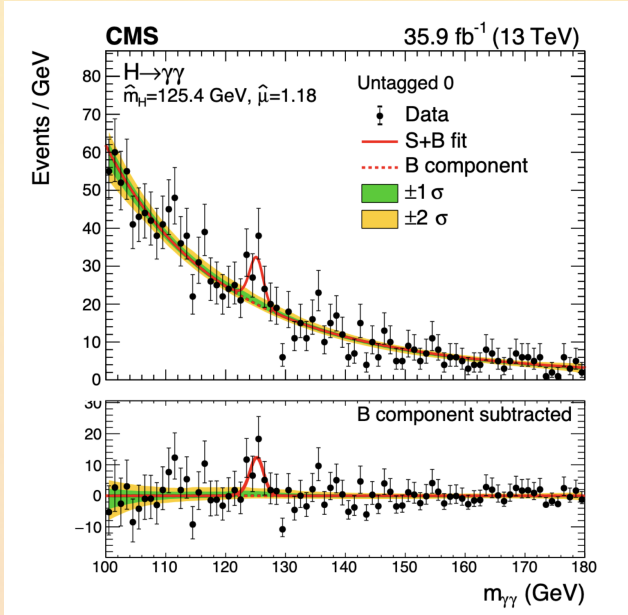
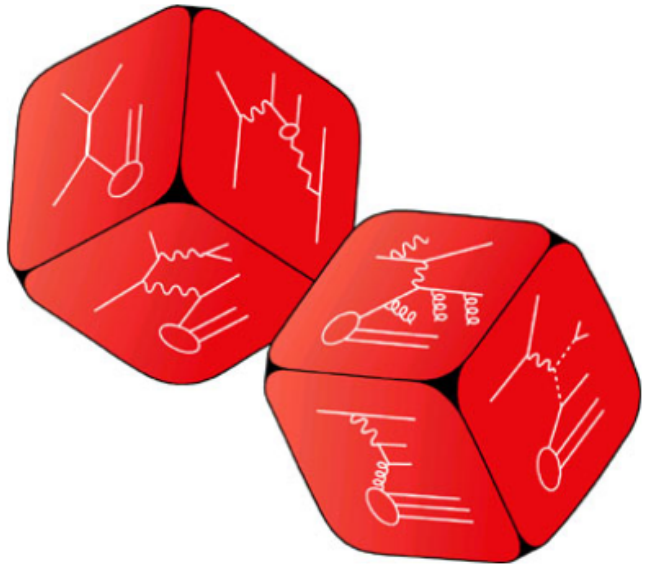


Figure from JHEP 11 (2018) 185

Quantum mechanics and probability



- Quantum mechanical amplitudes as probabilities
- Random numbers as a proxy for quantum mechanical choices

Pictorial representations by Torbjörn Sjöstrand

Efficiently sample from models

$$P(\mathbf{x}|\alpha) = \frac{1}{A_\alpha \sigma_\alpha} \int \mathrm{d}\Phi(y) \frac{\mathrm{d}x_1 \mathrm{d}x_2}{x_1 x_2 s} f(x_1) f(x_2) |\mathcal{M}_\alpha(y, x_1, x_2)|^2 W(\mathbf{x}|y)\, \epsilon_\alpha(y)$$

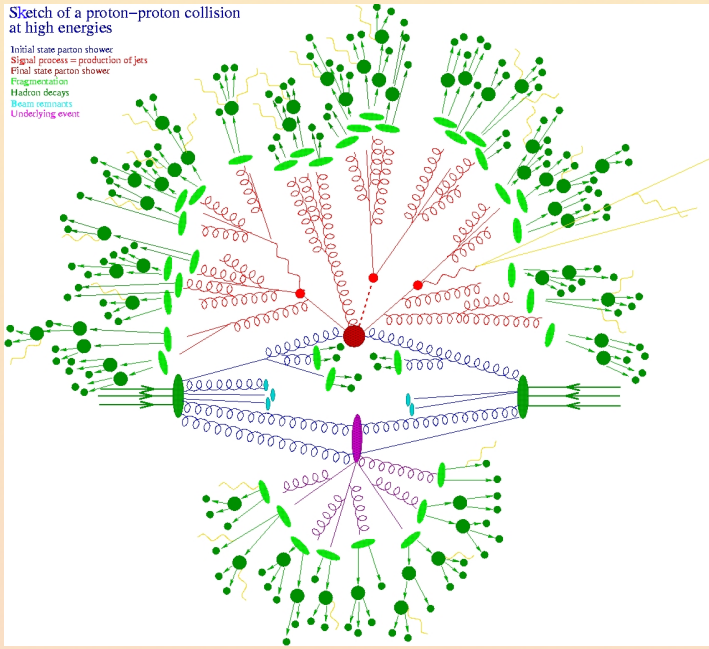
Normalization factor

We collide protons and that is a mess

Linear operator in Hamiltonian formalism, describes the physics process

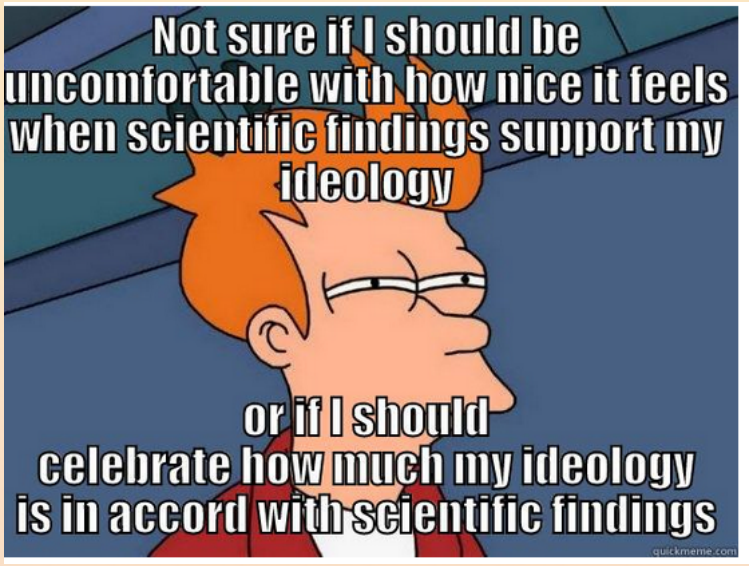
Detector response, experimental efficiencies

- Difficult integration
- Matrix element computed perturbatively
  - NLO (1st-order) solved in most processes
  - NNLO current challenge
- Costly MonteCarlo simulators
  - Hundreds-dimensional phase space



Generators as vehicles of ideology

- Study complex multiparticle physics (both experimentally and theoretically)
- Flexibility in physical quantities that can be addressed
  - Predict event rates and topologies (estimate feasibility)
  - Simulate possible backgrounds (devise analysis strategies)
  - Study detector requirements (can optimize detector/trigger design)
  - Study detector imperfections (evaluate acceptance corrections)



Meme from quickmeme.com. Text by Torbjörn Sjöstrand

Monte Carlo simulations

- We want to generate events as they are in Nature
  - average and fluctuations
  - random choices

$\sigma_{final\ state} = \sigma_{hard\ process} P_{tot,\ hard\ process \to final\ state}$

where $P_{tot} = P_{res} P_{ISR} P_{FSR} P_{MPI} P_{remnants} P_{hadronization} P_{decays}$

and $P_i = \prod_j P_{ij} = \prod_i \prod_k P_{ijk} = \dots$

- For each event, $\mathcal{O}(10)$ random choices (flavour, mass, momentum, spin, lifetime, ...)
- Typical LHC event: ~ 100 charged particles and ~ 200 neutral particles
- Several thousand choices

Likelihood intractability in HEP

$$p(x|\theta) = \int dz\, p(x,z|\theta) = \int dz\, p_x(x|\theta,z) \prod_i p_i(z_i|\theta, z_{<i})$$

- No access to the likelihood, or latent variables
  - Matrix element
  - Parton shower
  - Detector simulation
- Can always generate Monte Carlo samples from $x \sim p(x|\theta)$
- Histograms are likelihood-free!!!
  - Count events, assume Poisson per bin, global likelihood as product

Brain activity...

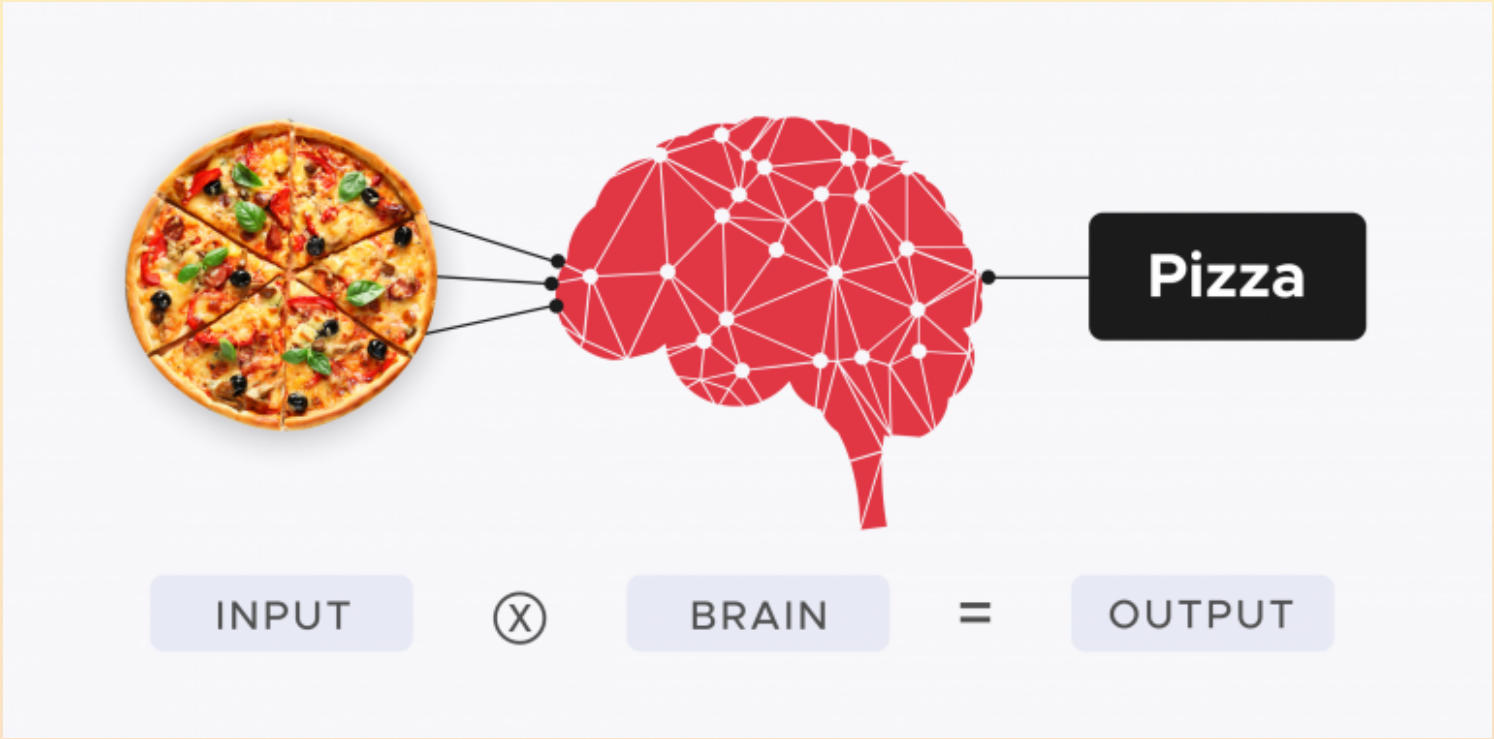


Image from zomato

...approximated...

Image from zomato, elaborated with image from pixabay

...using computers

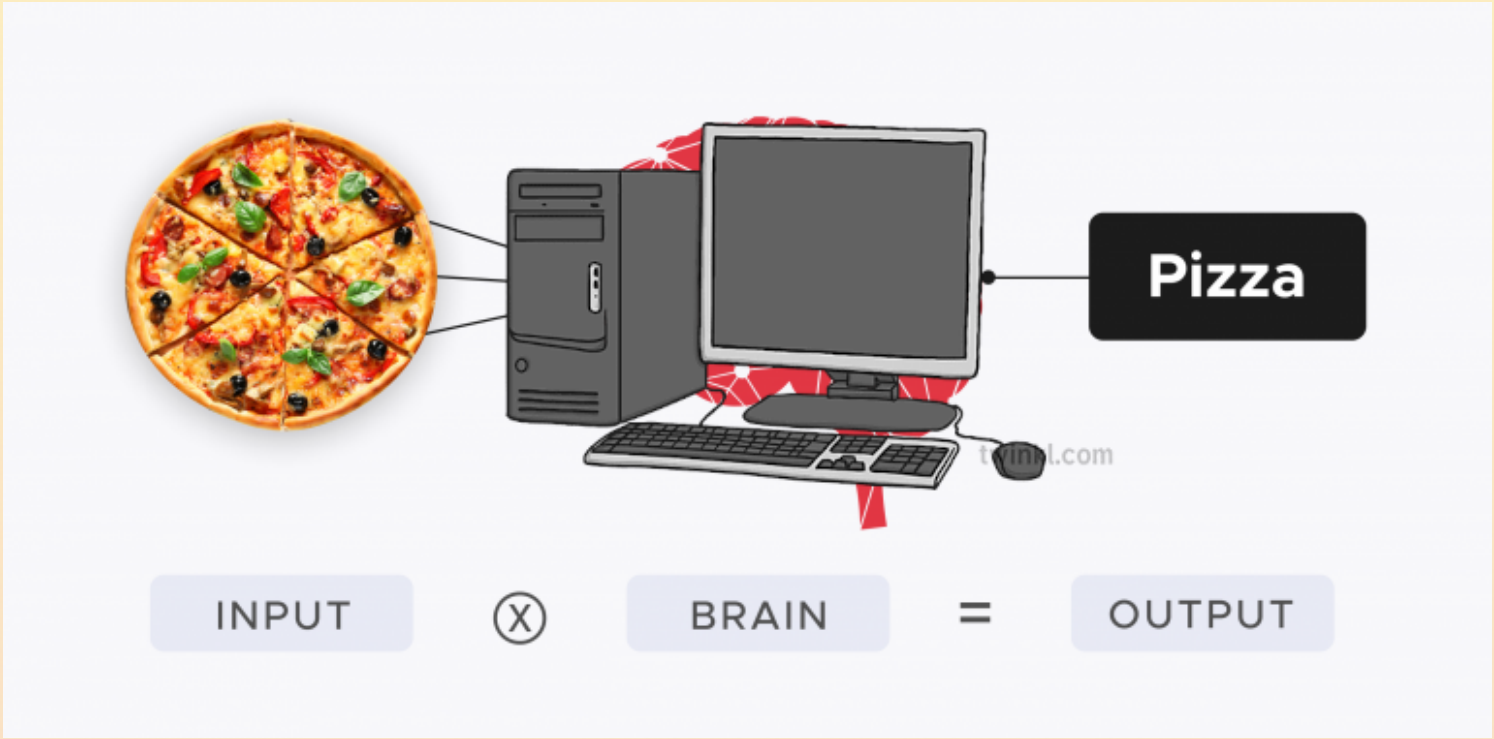


Image from zomato, elaborated with image from twinkl

Understanding Data

Vast amounts of data *are being generated in many fields, and the statistician's job is to make sense of it all: to* extract important patterns and trends*, and understand "what the data says." We call this* learning from data*.*
(Hastie, Tibshirani, Friedman, Springer 2017)

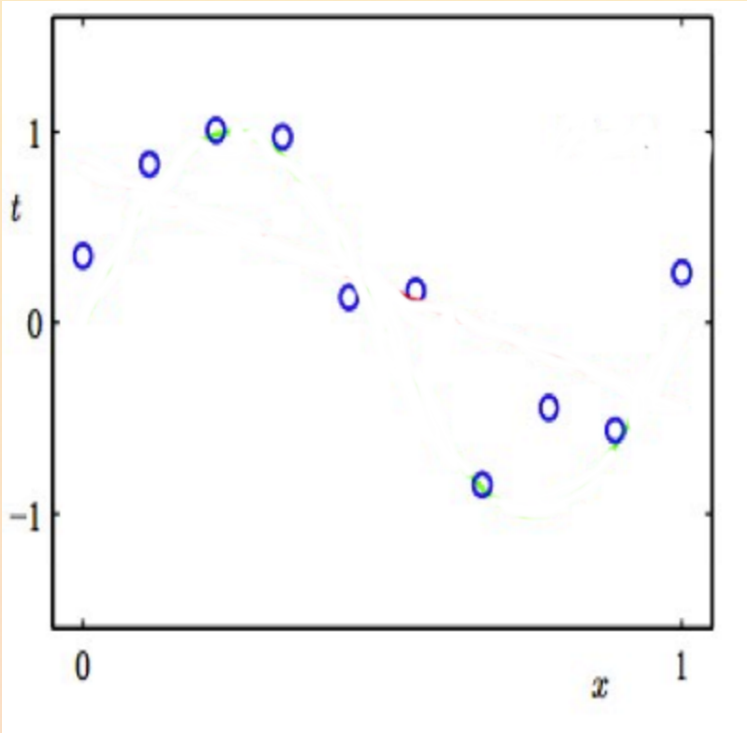


Image by Victor Lavrenko

Functions Describe the World

- Interpolation

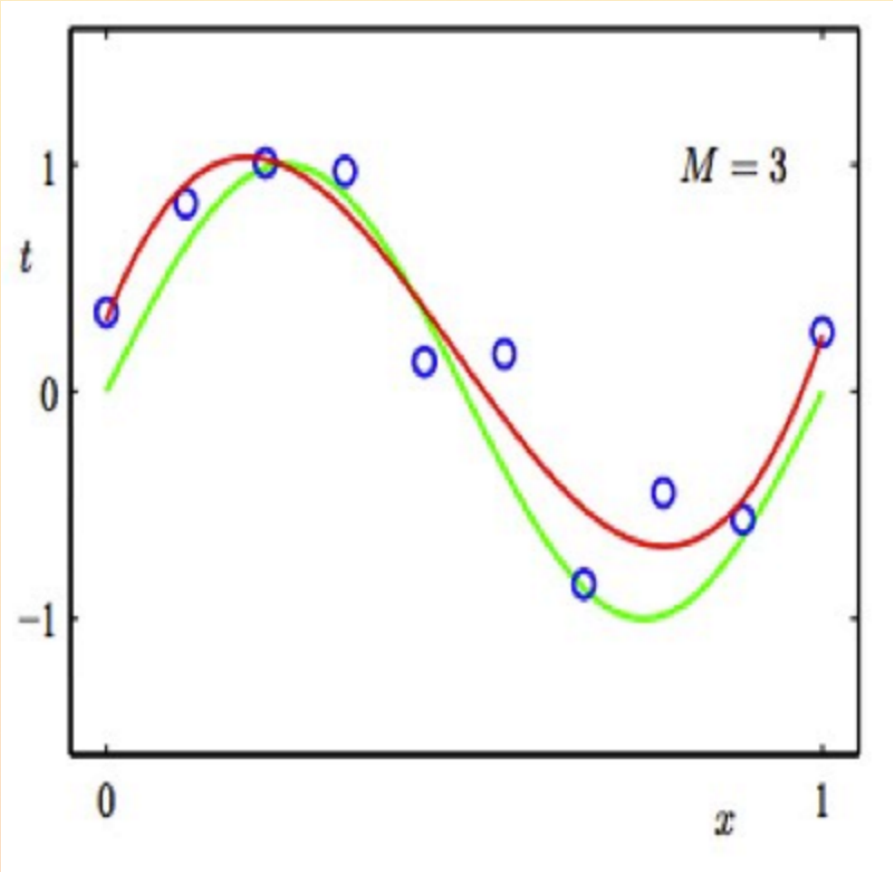


Image by Victor Lavrenko

Sometimes too well

- Generalization

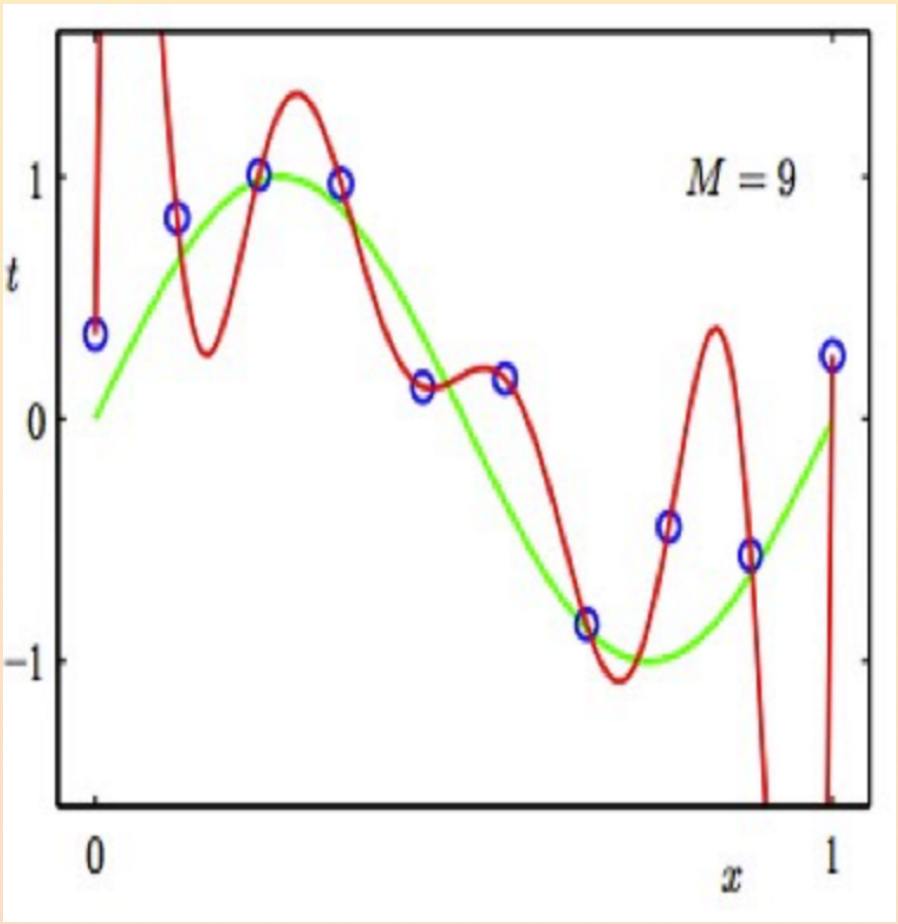


Image by Victor Lavrenko

Think in Millions of Dimensions

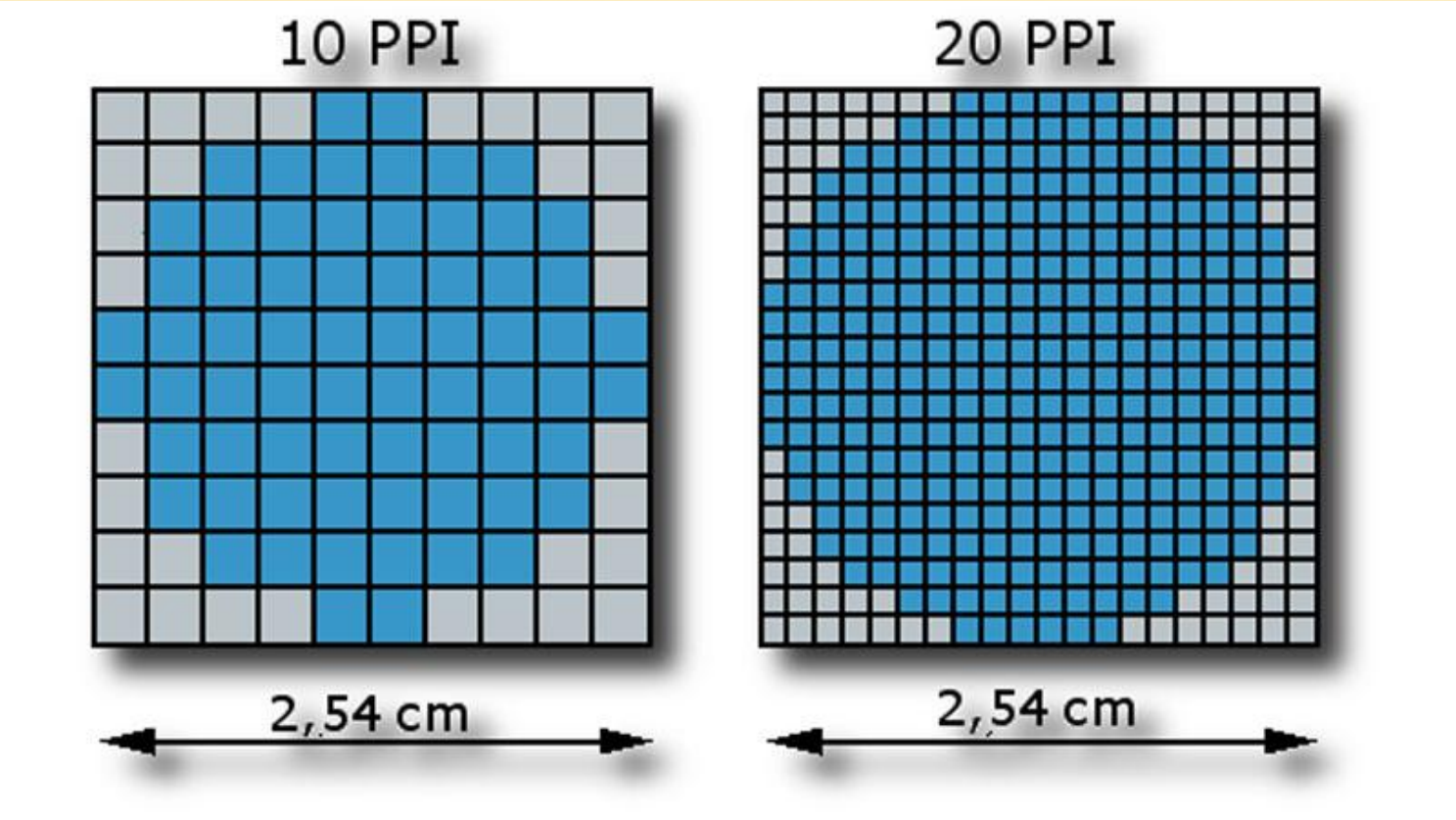


Image from scientiamobile.com

Easy or difficult?

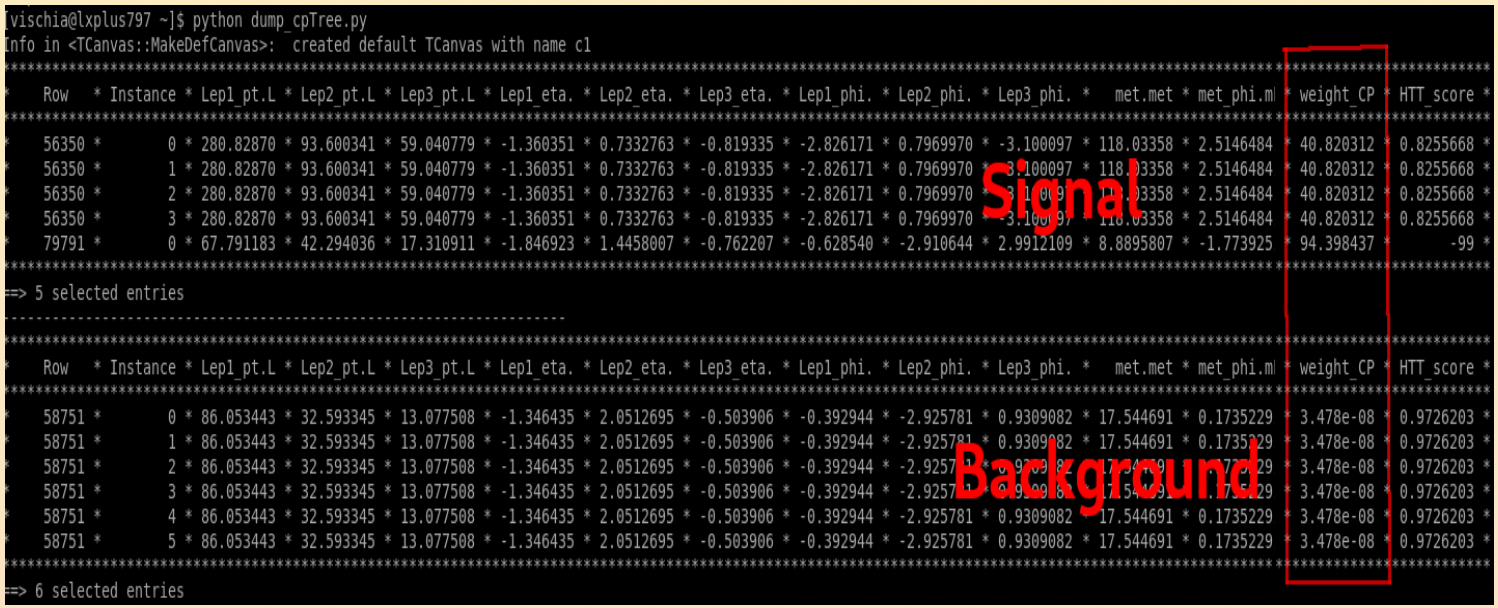


Image by Pietro Vischia

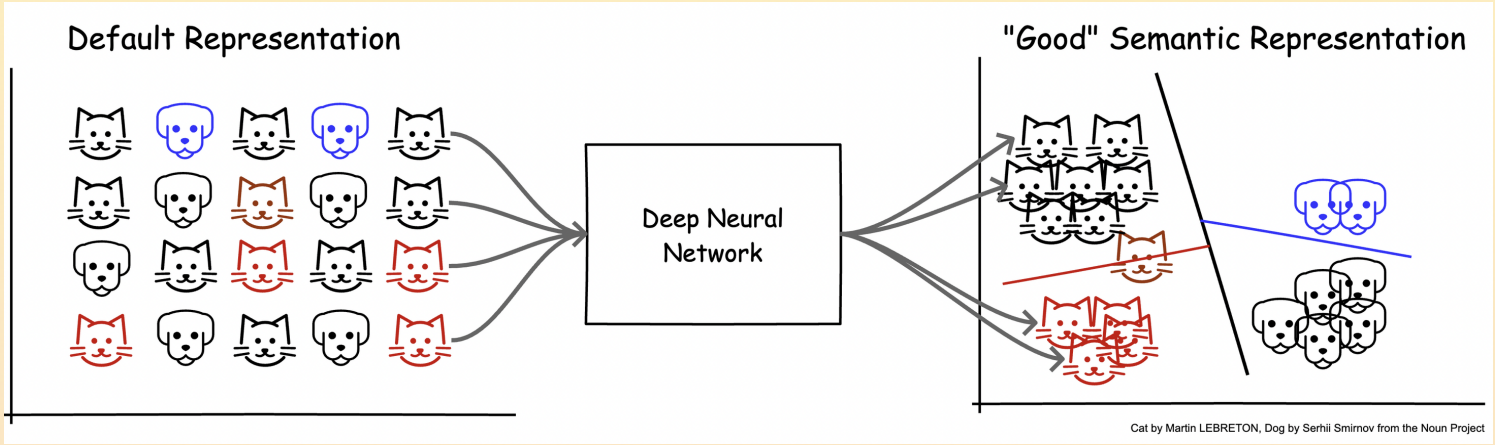
Easy or difficult?

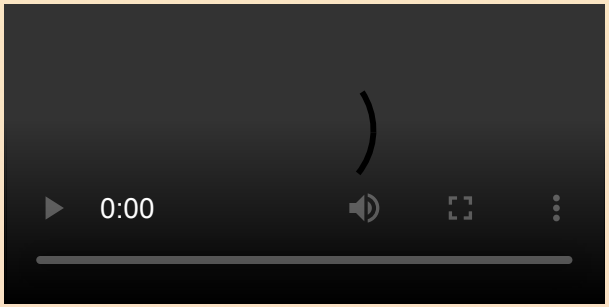
Image from indiatimes.com

Mapping Improves Understanding

Tabula Rogeriana by Al Idrisi (1154) 1929 reproduction

Representations Make Tasks Easier

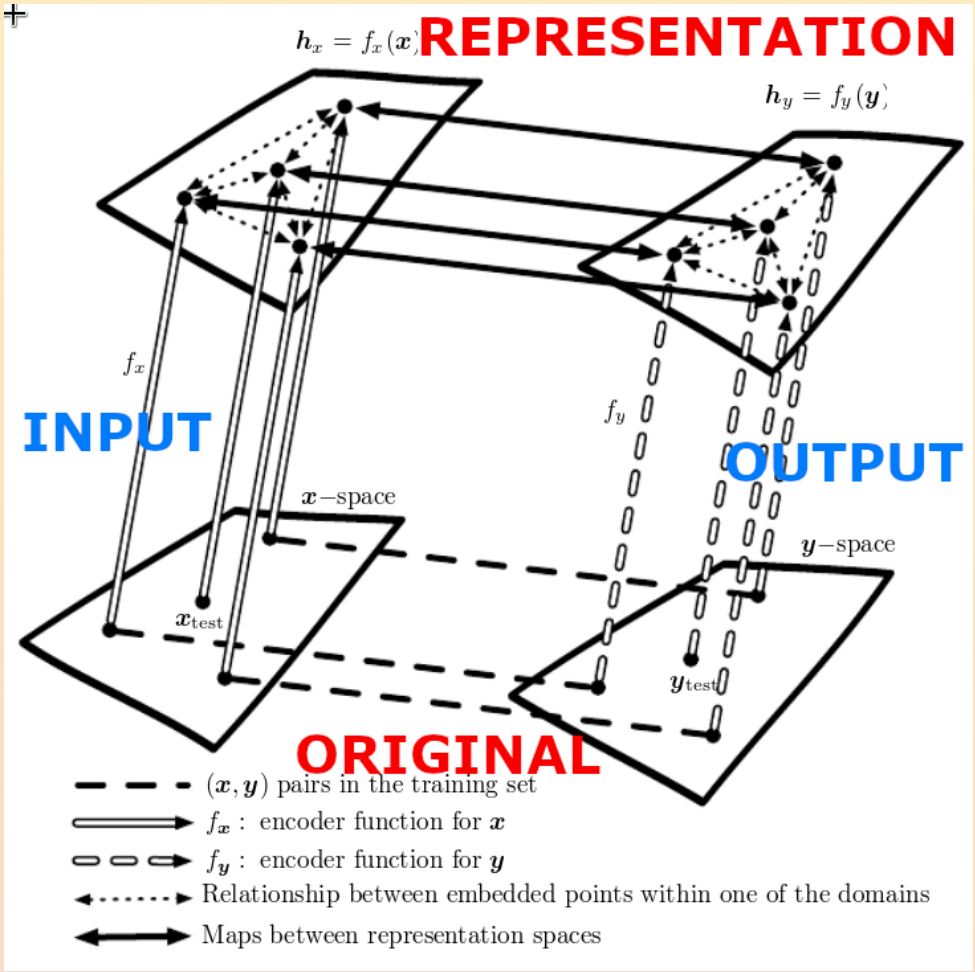




Animation and picture from FastForward Labs

Learn Representations

- Like AdS/CFT, but actually demonstrated 😜



Images from Goodfellow, Bengio, Courville, 2016

Learning from data:

the mathematical formalism

following Joan Bruna's formalization, see e.g. arXiv:1712.04741

Learn in different ways

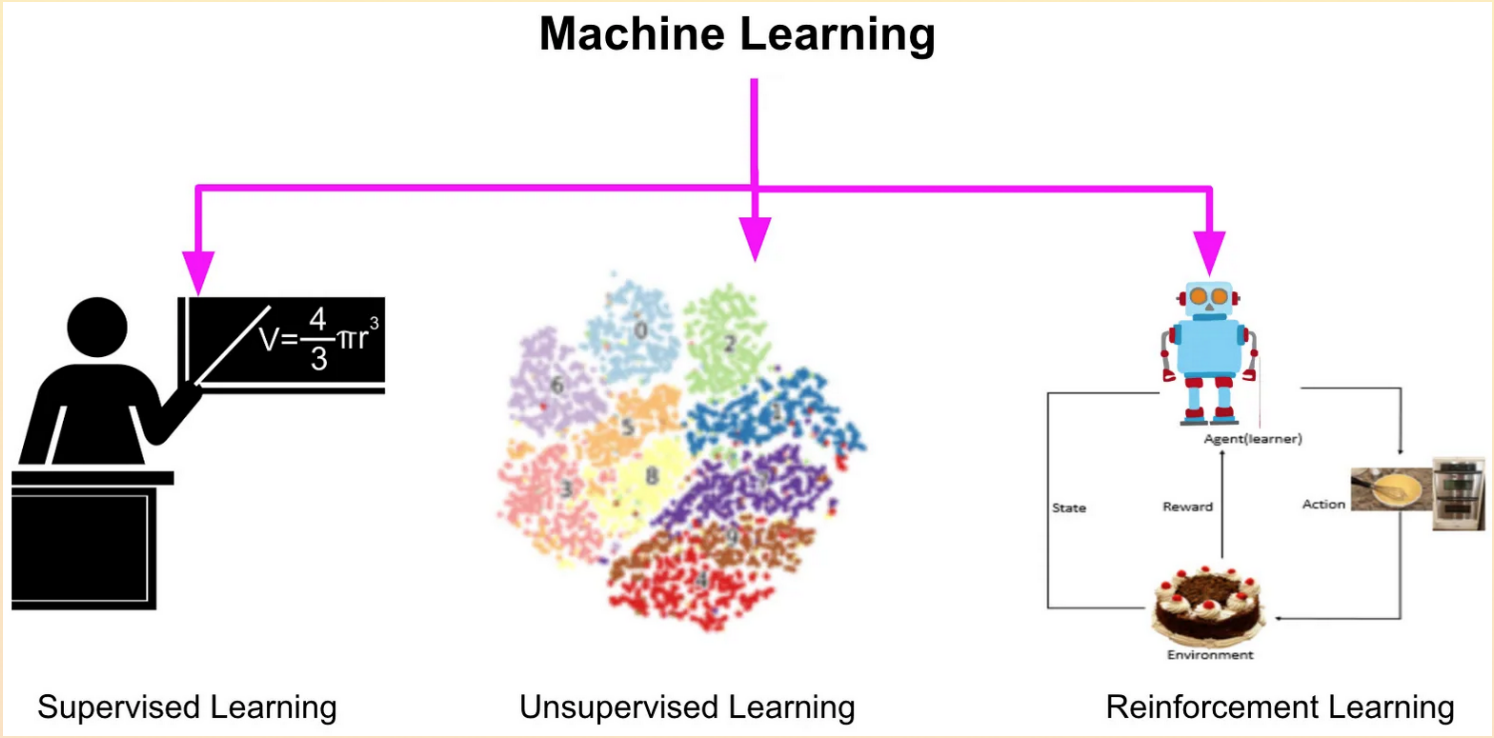


Image by Renu Khandelwal

Input space

- $\mathcal{X}$: a high-dimensional input space
  - The challenges come from the high dimensionality!
- If all dimensions are real-valued, $\mathbb{R}^d$
  - For square images of side $\sqrt{d}$, space is $\mathcal{X} = \mathbb{R}^d$, and $d \sim \mathcal{O}(10^6)$

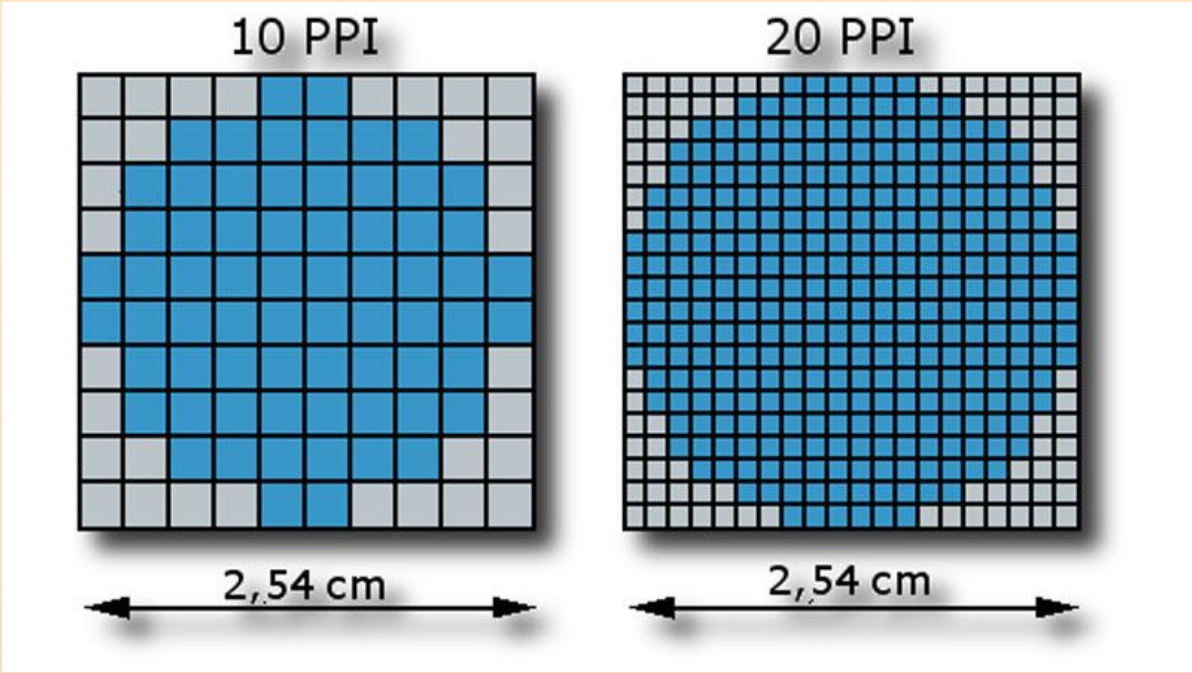


Image from scientiamobile.com

Data probability distribution

- ν: unknown data probability distribution
  - We can sample from it to obtain an arbitrary amount of data points
  - We are not allowed to use any analytic information about it in our computations

_
If you are interested, lecture on generators in the master's

The target function

- $f^* : \mathcal{X} \to \mathbb{R}$, unknown target function
  - In case of multidimensional output to a vector of dimension k, $f^* : \mathcal{X} \to \mathbb{R}^k$
  - Some loose assumptions (e.g. square-integrable with respect to the ν measure, i.e. finite moments, bounded...)

The loss functional

- $L[f] = \mathbb{E}_\nu \left[l\Big(f(x), f^*(x) \Big) \right]$
  - The metric that tells us how good our predictions are
- The function $l(\cdot, \cdot)$ is a given expression, e.g. regression loss, logistic loss, etc
  - In this lecture, typically it is the L^2 norm: $\| f - f^* \|_{L^2(\mathcal{X}, \nu)}$

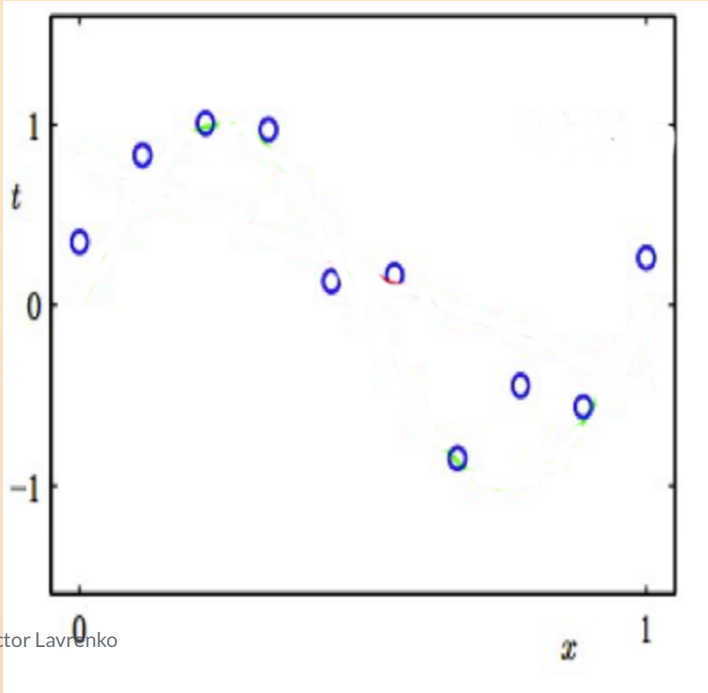
Loss function comes from inference

- Decision-theoretic approach (C.P. Robert, "The Bayesian Choice")
  - $\mathcal{X}$: observation space
  - Θ: parameter space
  - $\mathcal{D}$: decision (action) space
- Statistical inference take a decision $d \in \mathcal{D}$ related to parameter $\theta \in \Theta$ based on observation $x \in \mathcal{X}$, under $f(x|\theta)$
  - Typically, d consists in estimating $h(\theta)$ accurately

$$U(\theta, d) = \mathbb{E}_{\theta,d}\Big[U(r)\Big]$$

Learning goal

- Goal: predict \fstar from a finite i.i.d. sample of points sampled from ν
- Sample $\{x_i, f^*(x_i)\}_{i=1,\dots,n}, x_i \sim \nu$
  - For each of the points x_i, we know the value of the unknown function (our true labels)
  - We want to interpolate for any arbitrary x inbetween the labelled x_i...
  - ...in million of dimensions!



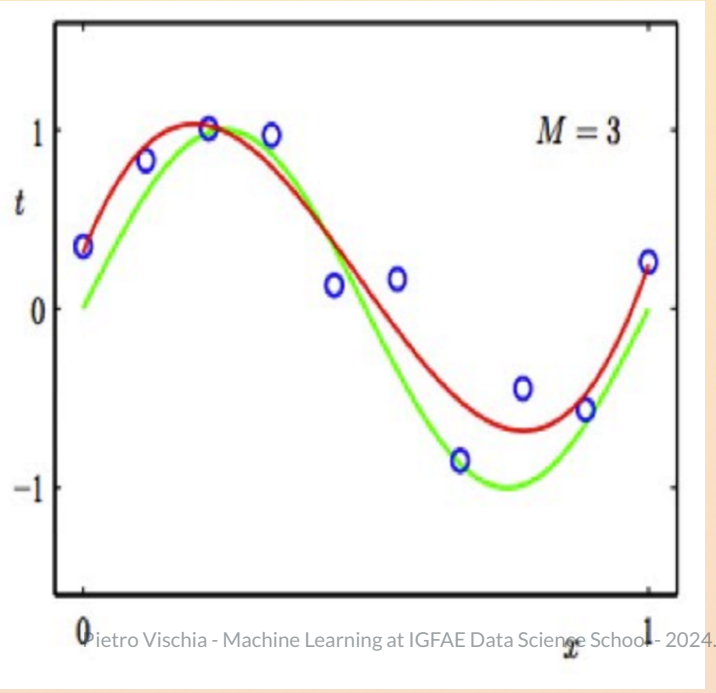


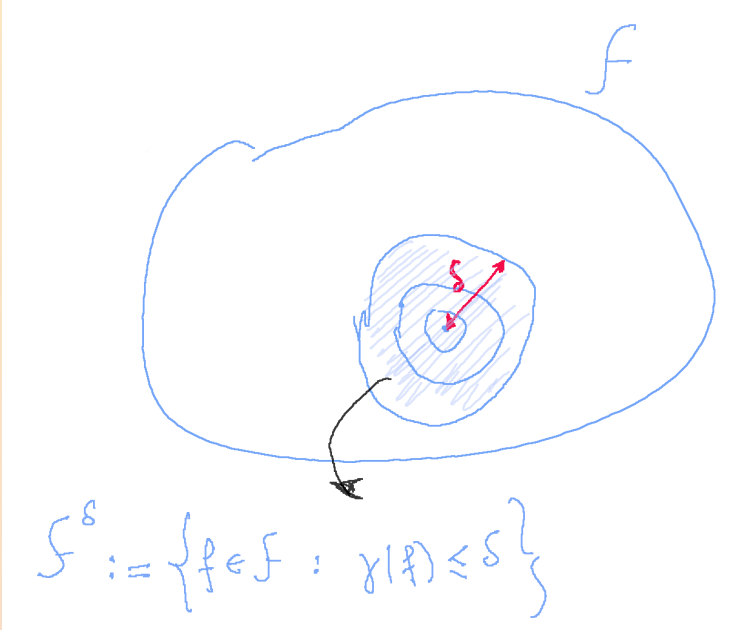
Image by Victor Lavrenko

The space of possible solutions

- The space of possible functional solutions is vast: $\mathcal{F} \subseteq \{f : \mathcal{X} \to \mathbb{R}\}$ (hypothesis class)
- We need a notion of complexity to "organize" the space
- $\gamma(f), f \in \mathcal{F}$: complexity of f
  - It can for example be the norm, i.e. we can augment the space $\mathcal{F}$ with the norm

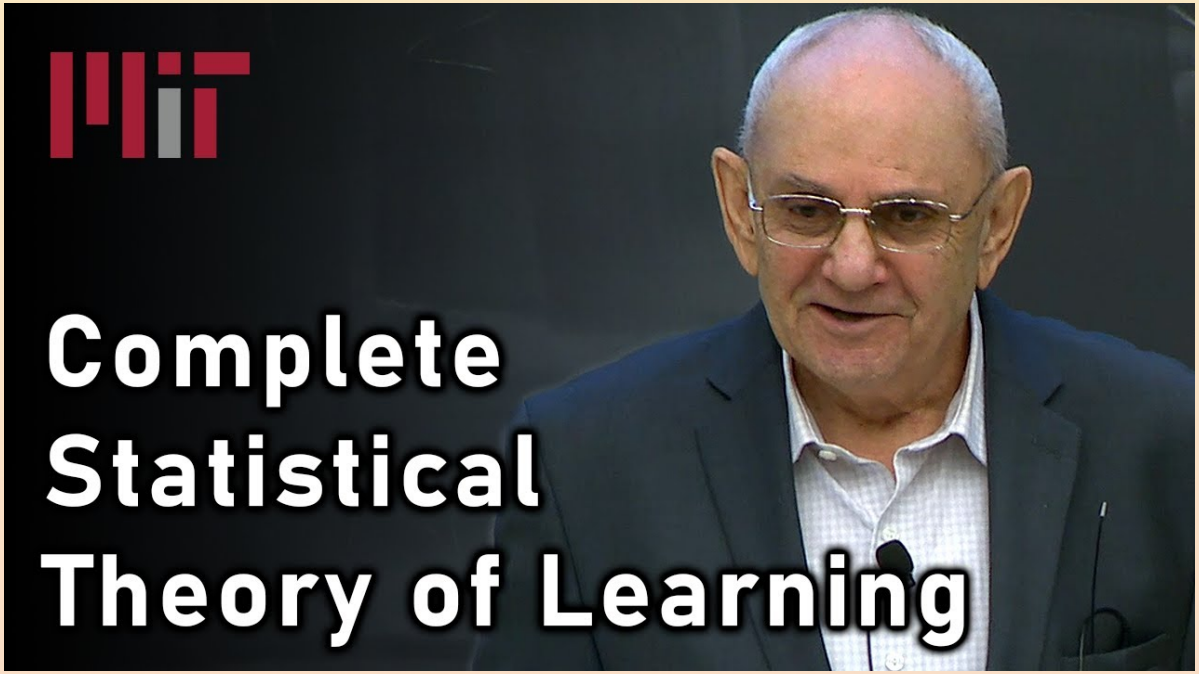
Organizing the space

- When the complexity is defined via the norm, $\mathcal{F}$ is highly organized: Banach space!
  - The simplest function according to the norm criterion is the 0 function
  - If we increase the complexity by increasing the norm, we obtain convex balls $\{f \in \mathcal{F}; \gamma(f) \leq \delta\} =: \mathcal{F}^\delta$
  - Convex minimization is considerably easier than non-convex minimization



Empirical Risk Minimization

- For each element of $\mathcal{F}$, a measure of how well it's interpolating the data
- Empirical risk: $\hat{L}(f) = \frac{1}{n} \sum_{i=1}^{n} |f(x_i) - f^*(x_i)|^2$
  - $|\cdot|$ is the empirical loss. If it's the norm, then $\hat{L}(f)$ is the empirical Mean Square Error
  - If you find an analogy with least squares method, it's because for one variable it's exactly that!



Vapnik's image from youtube

Formalizing the minimization of a functional in a given space}

- Constraint form: $\min_{f \in \mathcal{F}^\delta} \hat{L}(f)$.
  - Not trivial
- Penalized form: $\min_{f \in \mathcal{F}} \hat{L}(f) + \lambda \gamma(f)$.
  - More typical
  - λ is the price to pay for more complex solutions. Depends on the complexity measure
- Interpolant form: $\min_{f \in \mathcal{F}} \gamma(f)$ s.t. $\hat{L}(f) = 0 \iff f(x_i) = f^*(x_i) \qquad \forall i$
  - In ML, most of the times there is no noise, so $f(x_i)$ is exactly the value we expect there (i.e. we really know that x_i is of a given class, without any uncertainty)

Are the three forms equivalent?

- The interpolant form exploits the no-noise assumption (*"give me the least complex elements in $\mathcal{F}$ that interpolates"*)
- These forms are not completely equivalent. The penalized form to be solved requires averaging a full set of penalized forms, so it's not completely equivalent
- There is certainly an implicit correspondence between δ and λ
  - (the larger λ, the smaller δ and viceversa)

The Fundamental Theorem of Machine Learning

- We want to relate the result of the empirical risk minimization (ERM) with the prediction
  - Let's use the constraint form
- Let's assume we have solved the ERM at a precision ϵ (we are ϵ-away from...). We then have $\hat{f} \in \mathcal{F}^{\delta}$ such that $\hat{L}(\hat{f}) \leq \epsilon + \min_{f \in \mathcal{F}^{\delta}} \hat{L}(f)$
- How good is $\hat{f}$ at predicting f^*? In other words, what's the true loss?
  - Can use the triangular inequality

$$L(\hat{f}) - \inf_{f \in \mathcal{F}} L(f) \leq \inf_{f \in \mathcal{F}^{\delta}} L(f) - \inf_{f \in \mathcal{F}} L(f)$$

Approximation error

(how appropriate is my measure of complexity)

$$+2 \sup_{f \in \mathcal{F}^{\delta}} \left| L(f) - \hat{L}(f) \right|$$

Statistical error

(having the empirical loss instead of the true loss)

$$+\epsilon$$

Optimization error

The Error Market

- The minimization is regulated by the parameter δ (the size of the ball in the space of functions)
- Changing δ results in a tradeoff between the different errors
  - Very small δ makes the statistical error blow up
- We are better at doing convex optimization (easier to find minimum), but even then the optimization error ϵ will not be negligible
  - ϵ: how much are ou willing to spend in resources to minimize $\hat{L}(f)$
  - We kind of control it!
  - If the other errors are smaller than ϵ, then it makes sense to spend resources to decrease it
  - Otherwise, don't bother

–
Bottou and Bousquet, 2008, Shalev-Shwartz, Ben-David

The big questions

- Approximation: we want to design "good" spaces $\mathcal{F}$ to approximate f^* in high-dimension
  - Rather profound problem, on which we still struggle
- Optimization: how to design algorithms to solve the ERM in general
  - We essentally have ONE answer
  - Gradient Descent!

The Curse of Dimensionality

- How many samples do we need to estimate f^*, depending on assumptions on its regularity?

The Curse of Dimensionality

- How many samples do we need to estimate f^*, depending on assumptions on its regularity?
  - f^* constant $\rightarrow$ need only 1 sample
  - f^* linear $\rightarrow$ need d samples
- Space of functionals is $\mathcal{F} = \left\{ f : \mathbb{R}^d \rightarrow \mathbb{R}; f(x) =< x, \theta > \right\} \simeq \mathbb{R}^d$ (isomorphic)
  - It's essentially like solving a system of linear equations for the linear form $< x_1, \theta^* >$
  - d equations, d degrees of freedom
- The reason why it's so easy is that linear functions are regular at a global level
  - Knowing the function locally tells us automatically the properties everywhere

Locally linear functions

- Assume f^* locally linear, i.e. f^* is Lipschitz
  - $|f^*(x) - f^*(y)| \leq \beta \|x - y\|$
  - $Lip(f^*) = \inf \{\beta; |f^*(x) - f^*(y)| \leq \beta \|x - y\| \text{ is true}\}$
  - $Lip(f^*)$ is a measure of smoothness
- Space of functionals that are Lipschitz: $\mathcal{F} = \left\{f : \mathbb{R}^d \to \mathbb{R}; f \text{ is Lipschitz}\right\}$
- We want a normed space to parameterize complexity, so we convert to a Banach space
  - $\gamma(f) := max(Lip(f), |f|_\infty)$
  - The parameterization of complexity is the Lipschitz constant

Formalization of the prediction problem

- $\forall \epsilon > 0$, find $f \in \mathcal{F}$ such that $\|f - f^*\| \leq \epsilon$ from n i.i.d. samples
  - n: sample complexity, "how many more samples to I need to make the error a given amount of times smaller"
- If f^* is Lipschitz, it can be demonstrated that $n \sim \epsilon^{-d}$
  - Upper bound: approximate f with its value in the closest of the sampled data points, find out expected error $\sim \epsilon^2$, upper bound is exponential
  - Lower bound: maximum discrepancy (the worst case scenario): unless you sample exponential number of data points, knowing $f(x_i)$ for all of them doesn't let youwell approximate outside

PAC learning

DEFINITION 3.1 (PAC Learnability) A hypothesis class $\mathcal{H}$ is PAC learnable if there exist a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: For every $\epsilon, \delta \in (0,1)$, for every distribution $\mathcal{D}$ over $\mathcal{X}$, and for every labeling function $f : \mathcal{X} \to \{0,1\}$, if the realizable assumption holds with respect to $\mathcal{H}, \mathcal{D}, f$, then when running the learning algorithm on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d. examples generated by $\mathcal{D}$ and labeled by f, the algorithm returns a hypothesis h such that, with probability of at least $1 - \delta$ (over the choice of the examples), $L_{(\mathcal{D},f)}(h) \leq \epsilon$.

- ϵ ("approximately correct"): how far from optimality the model is
- δ ("probably"): how likely the model is to meet the accuracy requirement
- $m_{\mathcal{H}}$ determines sample complexity (how many examples to guarantee PAC?)

COROLLARY 3.2 *Every finite hypothesis class is PAC learnable with sample complexity*

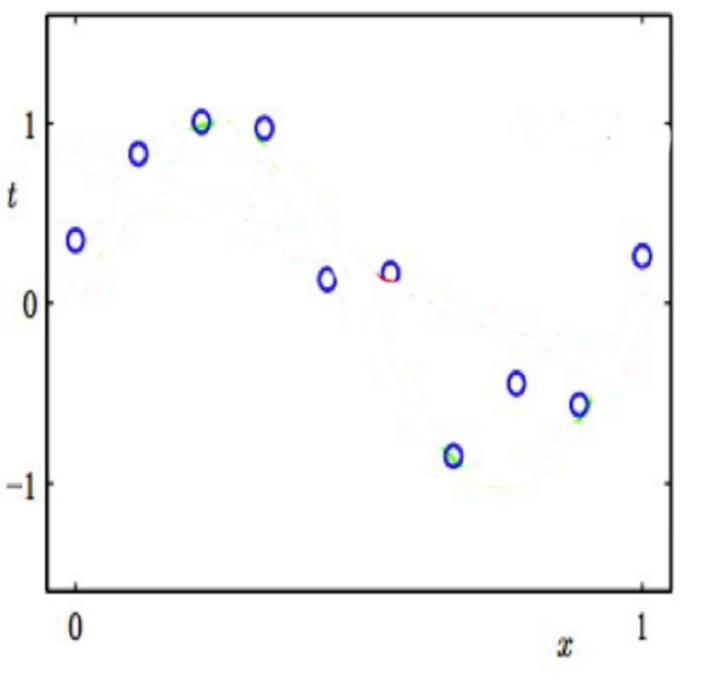
$$m_{\mathcal{H}}(\epsilon, \delta) \leq \left\lceil \frac{\log(|\mathcal{H}|/\delta)}{\epsilon} \right\rceil .$$

Images from Shalev-Shwartz and Ben-David (2014 Cambridge University Press)

Enough of the math?

What's the best function

To describe the data points?
(regression)



To separate into two classes?
(classification)

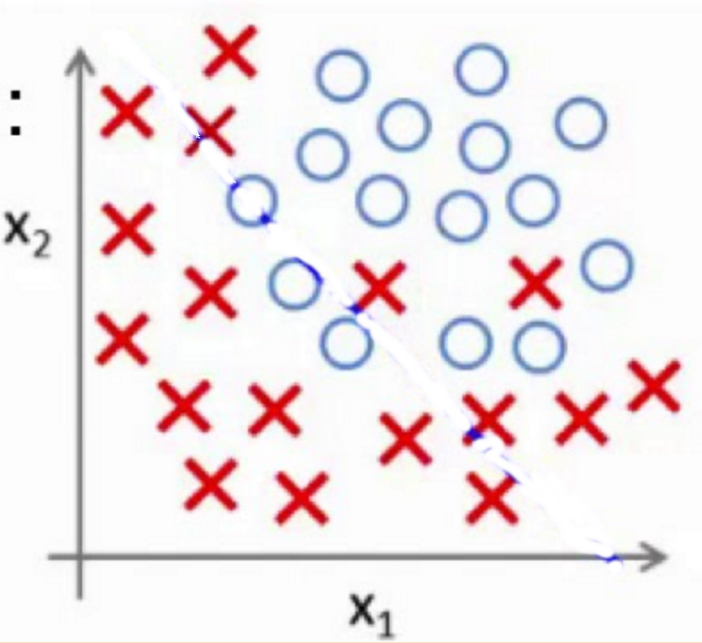
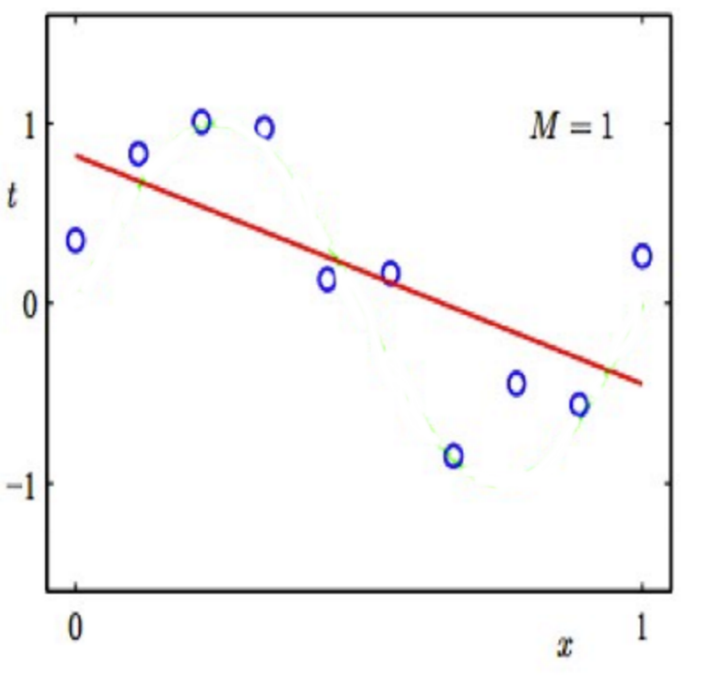


Image by Victor Lavrenko

What's the best function

To describe the data points?
(regression)



To separate into two classes?
(classification)

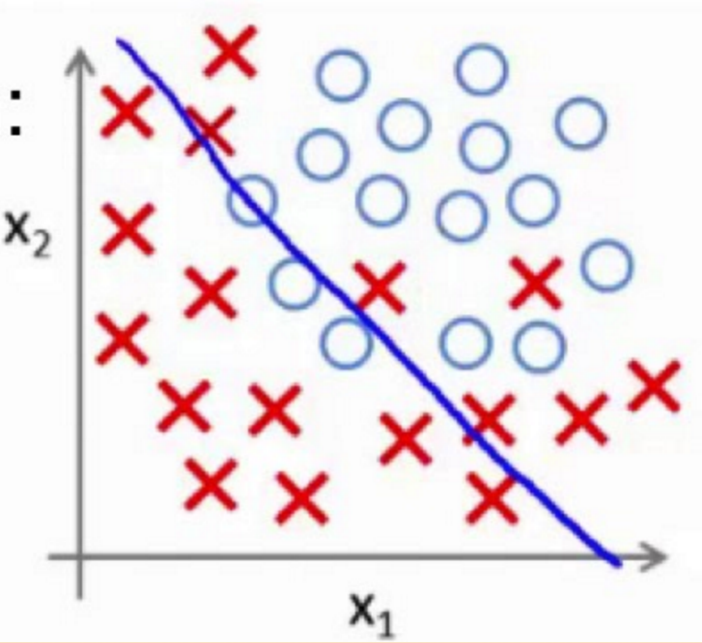
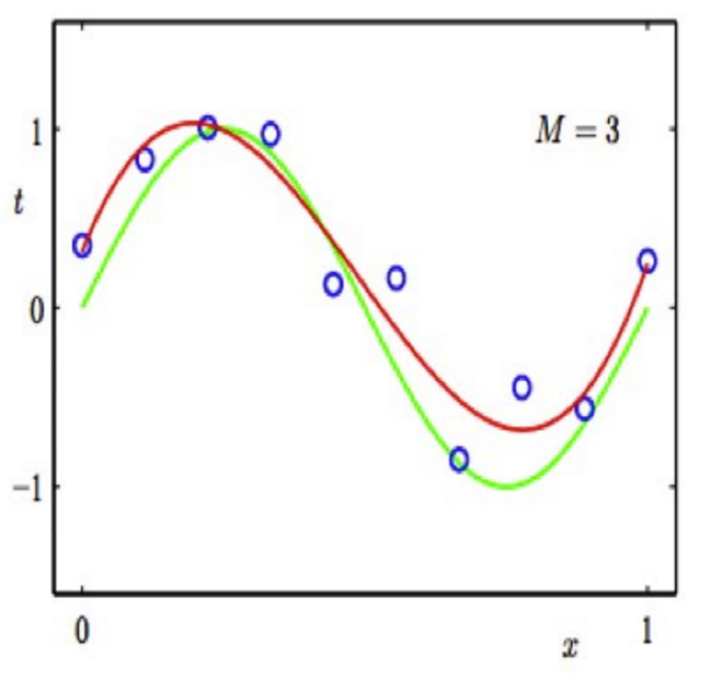


Image by Victor Lavrenko

What's the best function

To describe the data points?
(regression)



To separate into two classes?
(classification)

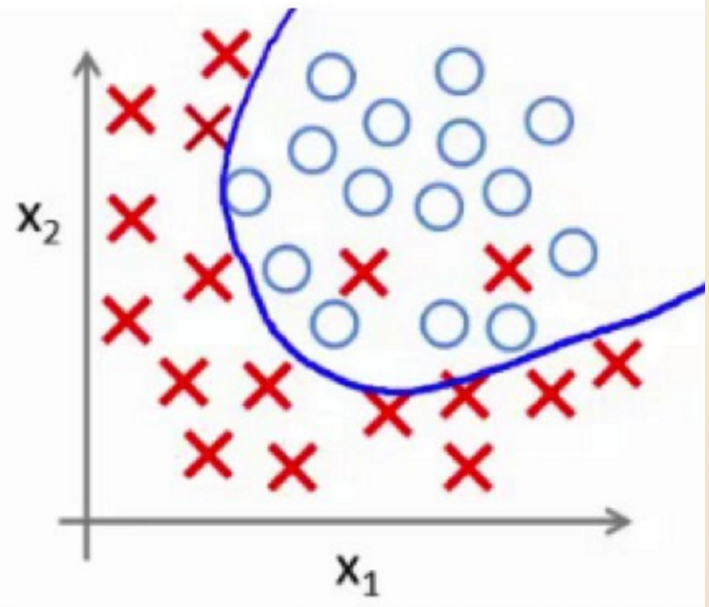
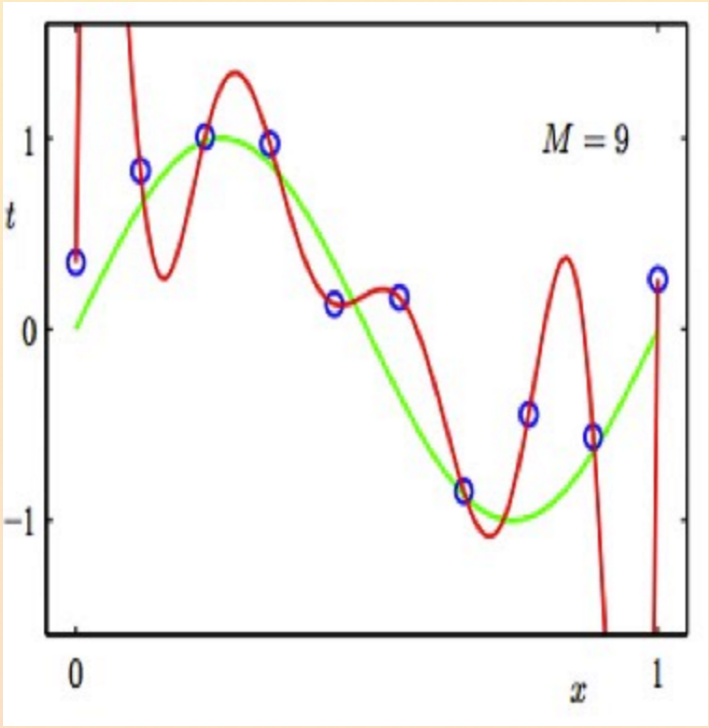


Image by Victor Lavrenko

What's the best function

To describe the data points?
(regression)



To separate into two classes?
(classification)

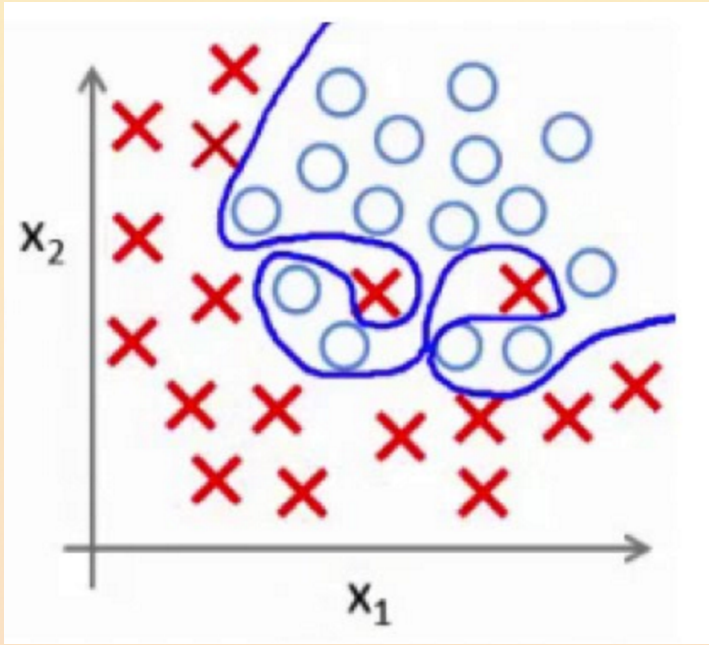
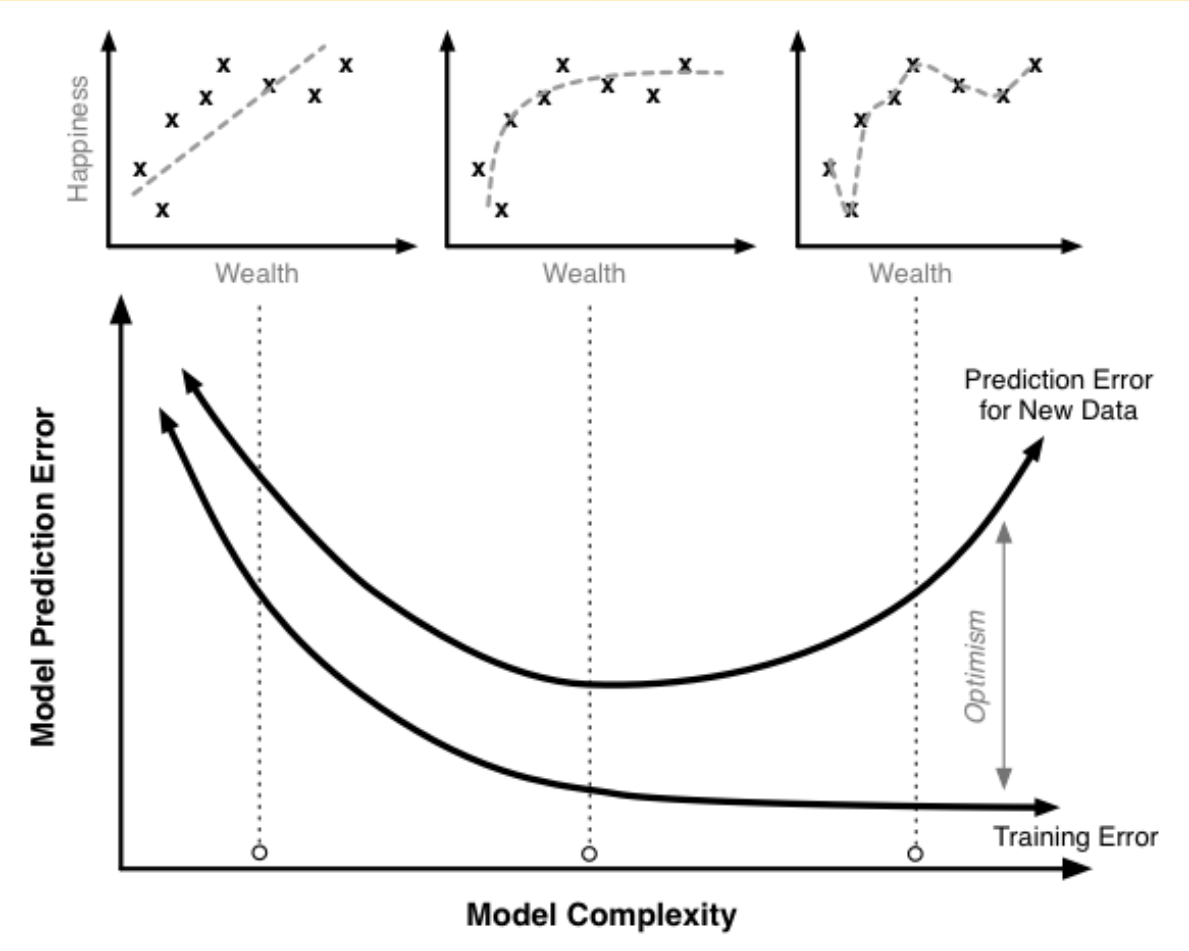


Image by Victor Lavrenko

Avoid overtraining



—
From can't remember where

Training a model

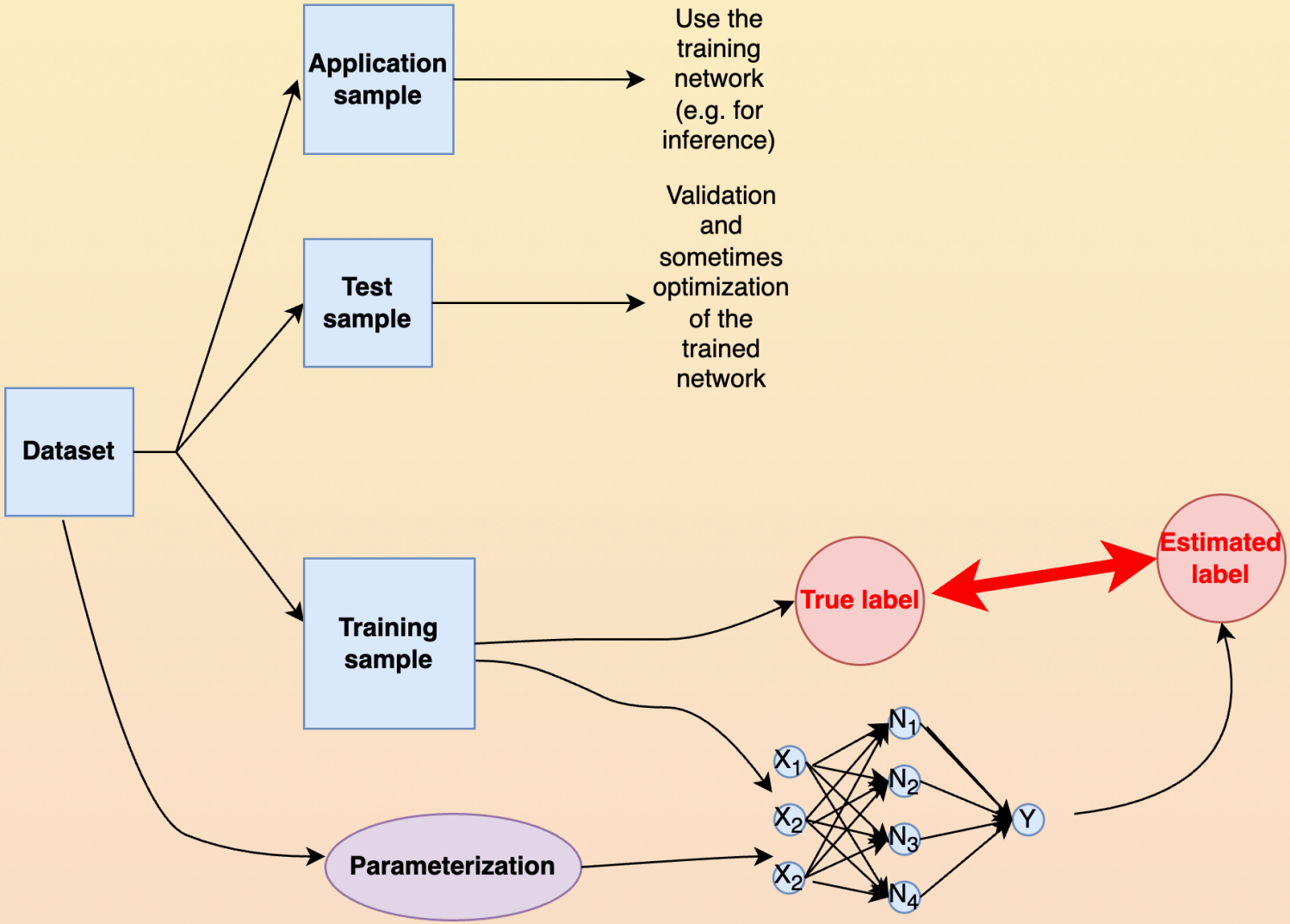


Illustration P. Vischia, 10.5281/zenodo.6373441

Data preparation: MinMax scaling

$$k = \frac{(x - min(x))}{abs(max(x) - min(x))}$$

$$z = k * (Max - Min) + Min$$

- Very sensitive to outliers (it scales them linearly)
- Scale to different variance and different mean

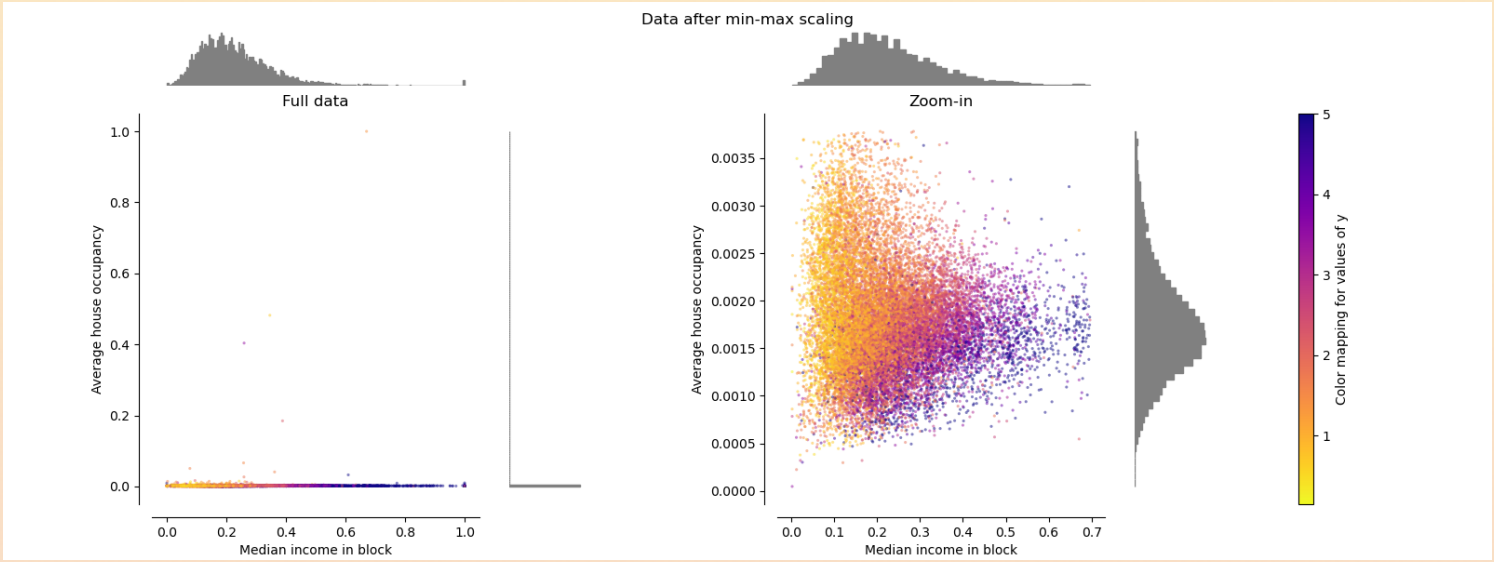


Image and code from scikit-learn

Data preparation: standardization

$$z = \frac{(x - mean(x))}{var(x)}$$

- Direct comparison between weights assigned to different features
- Easier, more effective numerical minimization, but still sensitive to outliers
- Scale to same variance and same mean (can also scale to same variance but different mean $z = \frac{x}{var(x)}$)

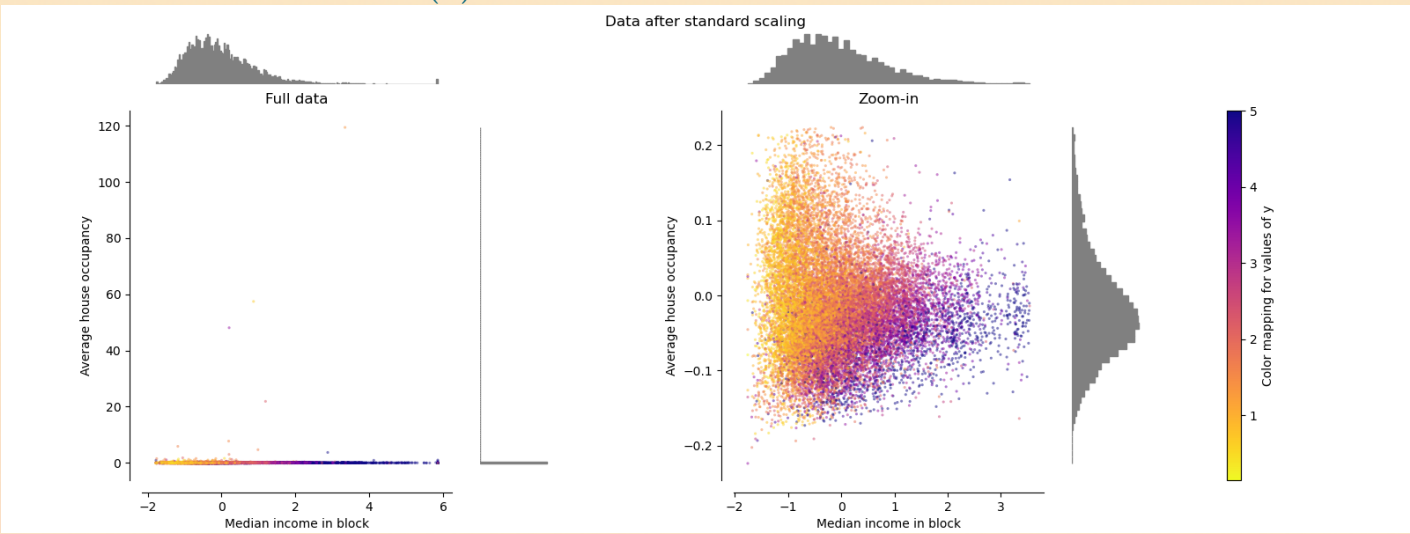


Image and code from scikit-learn

Data preparation: normalisation

- Normalize each data point to have unit norm
- Useful if using dot-product or other kernels to quantify similarity

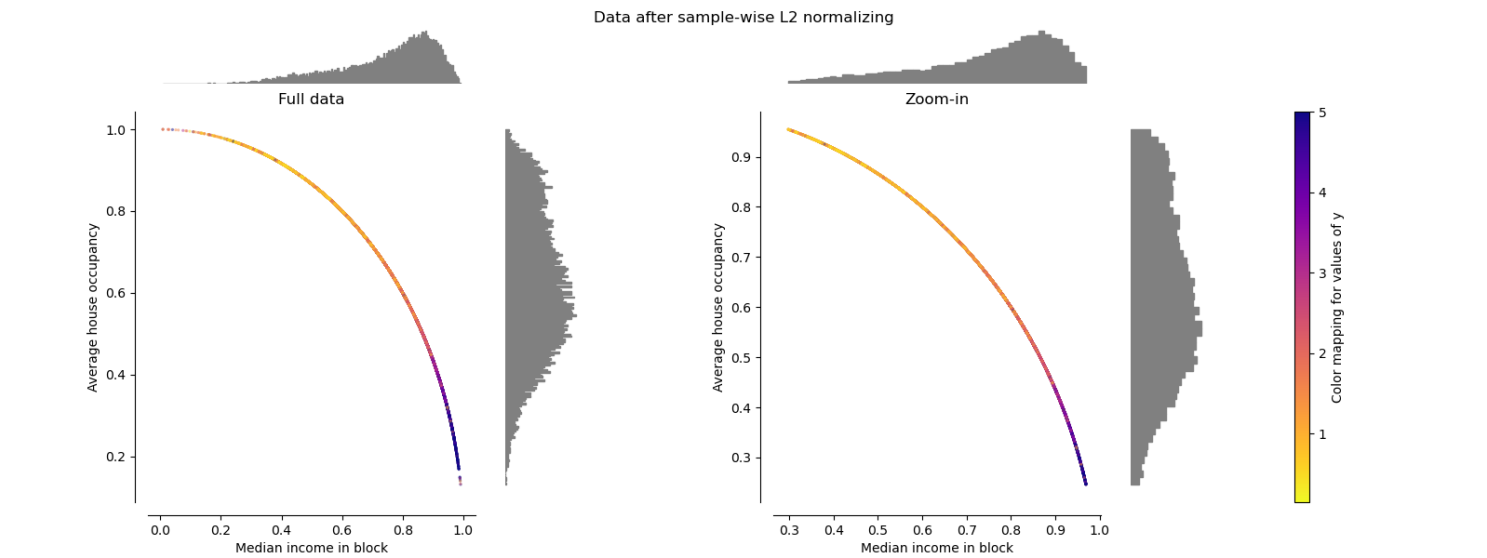


Image and code from scikit-learn

Data preparation: PCA

- Principal Component Analysis
  - Find iteratively the direction (linear combination of features) explaining the most variance

- Principal components are the eigenvectors of the data covariance matrix
  - Can be found by Singular Value Decomposition (SVD)
- Somehow analogous to finding axes of ellipsoid
  - Features with different units $\rightarrow$ arbitrariety (scale them first)
- Can retain a few dimensions: dimensionality reduction
  - Drop directions least explaining the variance

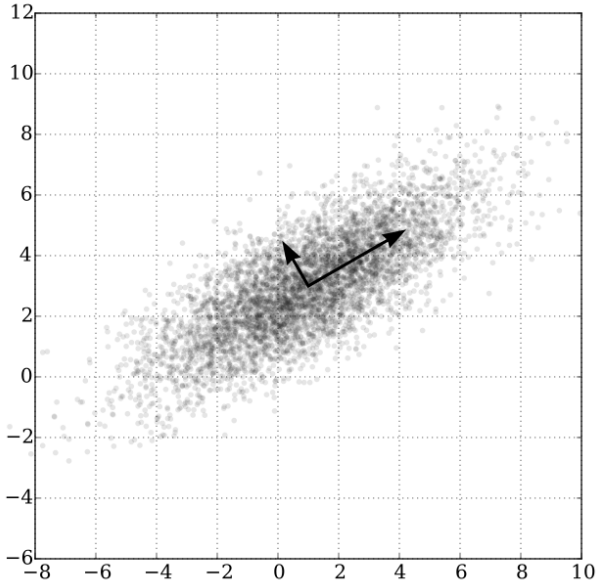
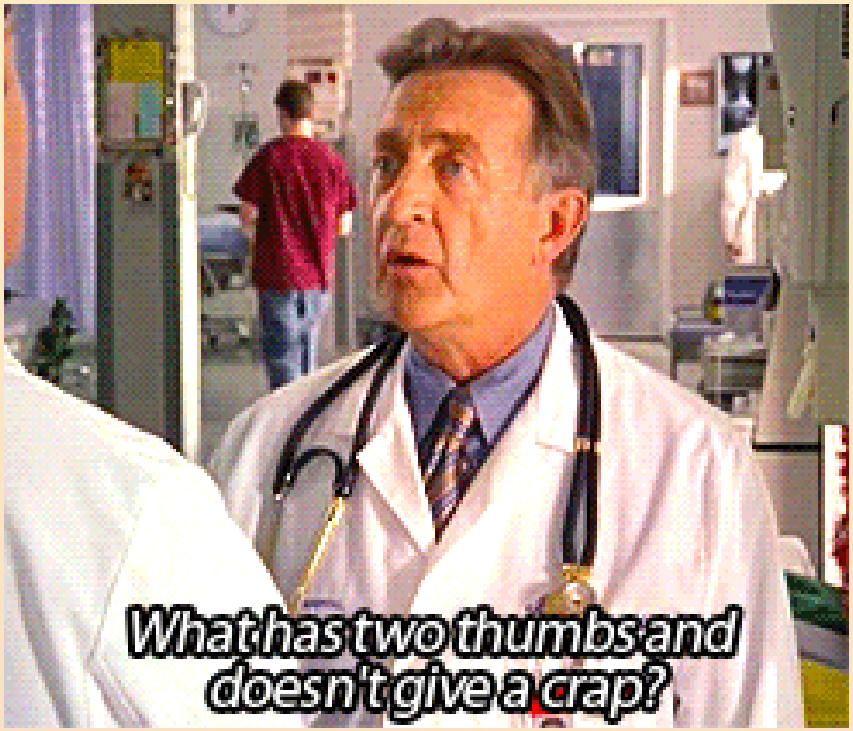


Image by Nicoguaro on Wikipedia

Data preparation: missing values

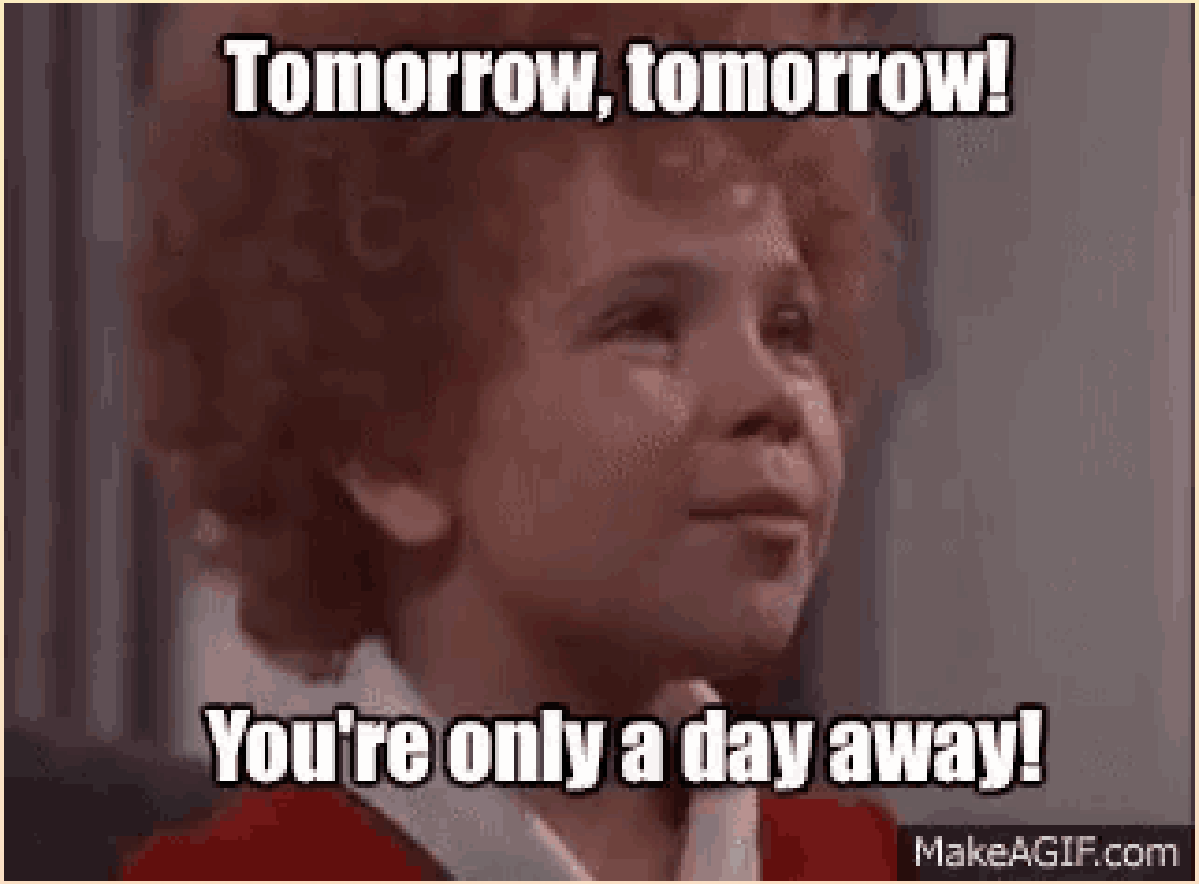
- LHC data are of extremely good quality (unlike e.g. medicine, social sciences)



  - Use proxy feature (pT of the two b jets $\longrightarrow$ pT of the two jets with highest b-tagging discriminator, in a region without b jets)
  - Use average over other data points (pT of the third jet $\longrightarrow$ **mean of the third jet pT for data points that have it, if some data points don't have three jets)

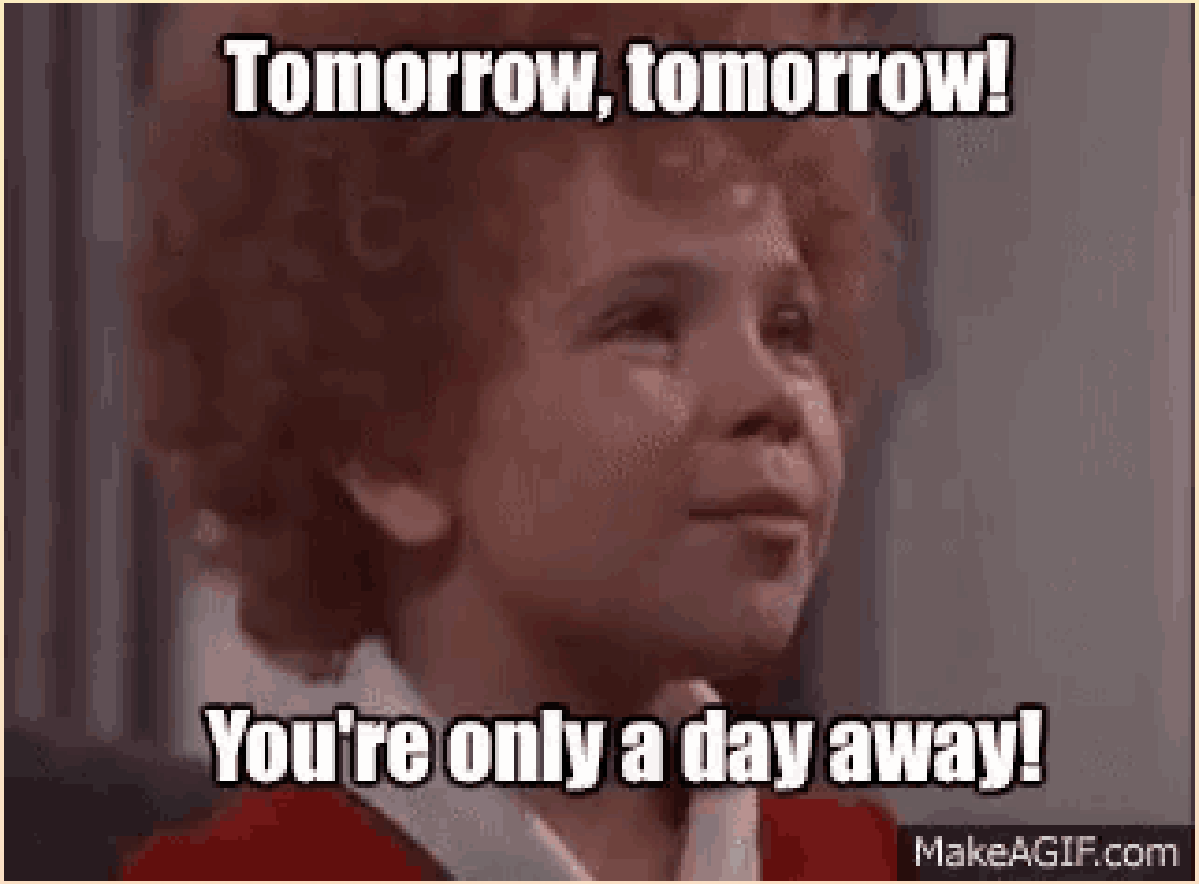
—
Gif from pinterest

Data preparation: feature encoding



—
Gif from pinterest

Class weights vs event weights



—
Gif from pinterest

Support Vector Machines (Vapnik)

- Similar to a "cut-based" analysis, but sophisticated strategy for optimal class separation
- Minimize empirical classification error + maximize geometric margin

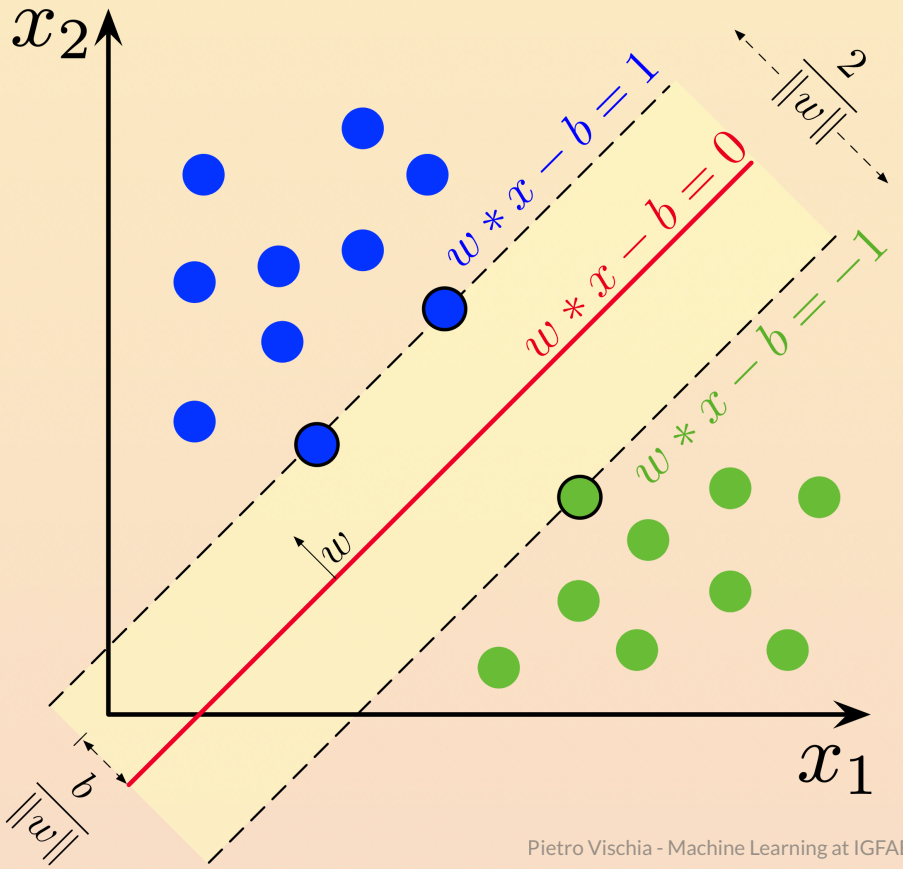


Figure by Larhmam on wikipedia

Support vector Machines

- Nonlinearity by kernel trick (deform the metric of the space until separation is linear)

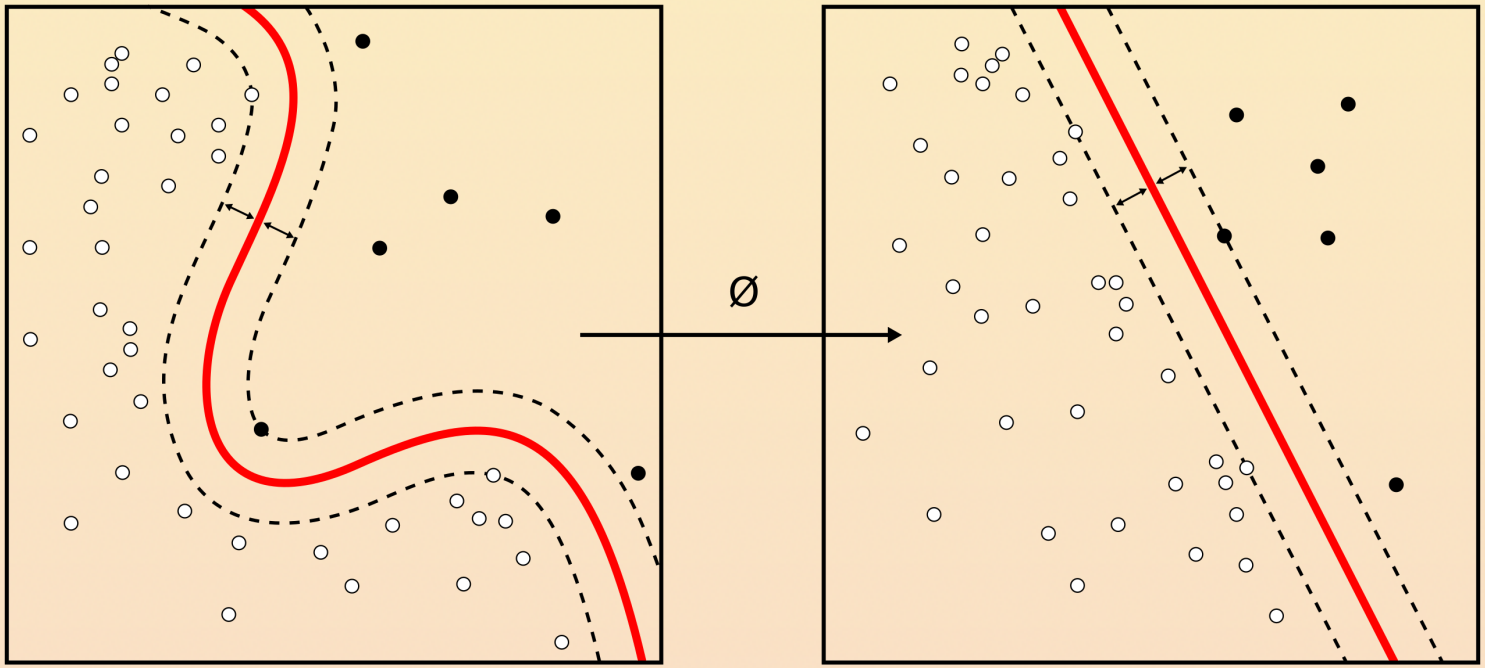


Figure by Alisneaky and Zirguezi on wikipedia

Decision Trees

- "Cut-based" analysis on steroids
- Ordering is key
  - 1) check stopping criterion
  - 2) sort according to each feature
  - 3) compute all separations
  - 4) if best separation improves, split
  - 5) back to 1

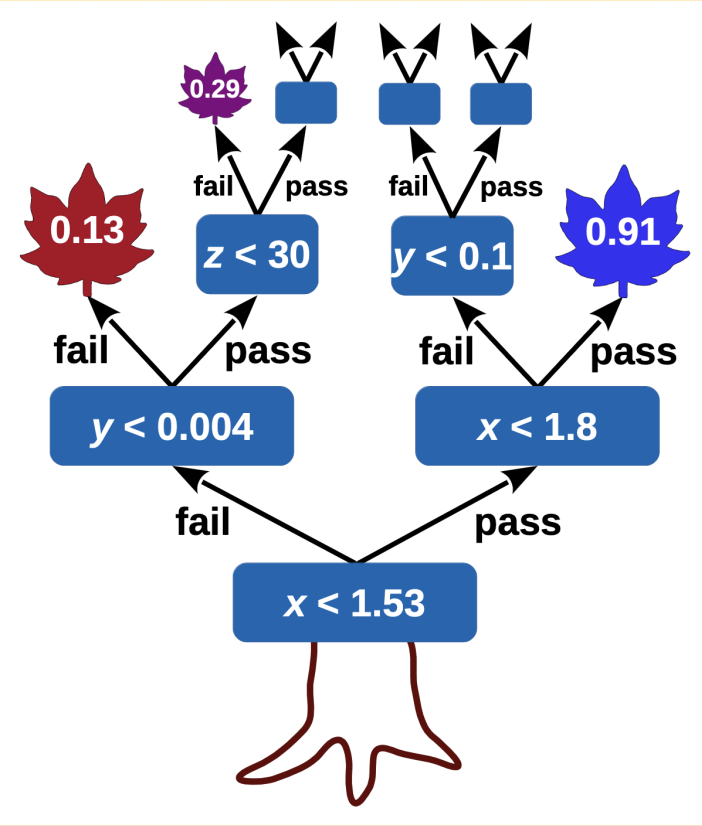


Image from 10.1142/9789811234033_0002

Decision Trees: hyperparameters

- Class balance: normalize classes $\sum_i w_i = \sum_j w_j, \forall(i,j)$ at root node
  - In practice, early splitting provides balance anyway
- Impurity $i(t)$, and stopping criterion
  - Minimum leaf (end node) size (maximum error $\sqrt{N_{min}}$ ensures significance of purity estimate)
  - Perfect separation
  - Insufficient improvement
  - Maximum tree depth (purely computational requirement)

Decision Trees: splitting

- Impurity decrease: $\Delta i(S,t) = i(t) - p_P i(t_p) - p_F i(t_F)$
- Optimization problem: $\Delta i(S^*,t) = \max_{S \in splits} \Delta i(S,t)$

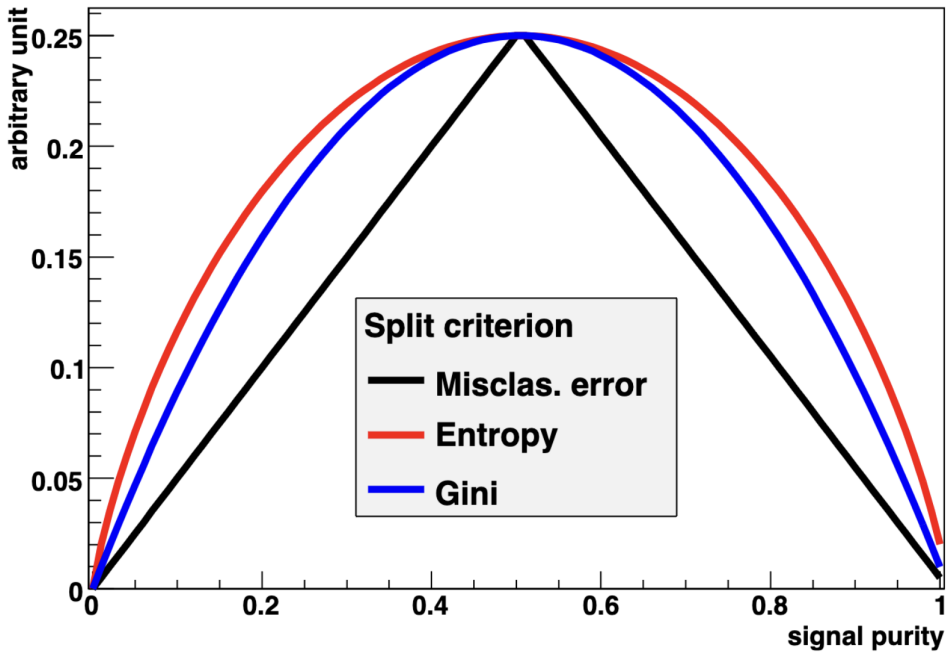


Fig. 5. Impurity measures as a function of signal purity.

Image from 10.1142/9789811234033_0002

Decision Trees: limitations

- Sensitivity to training sample
  - Decision trees are not robust: high variance for small changes in training sample
- Tree depth results in higher statistical uncertainty in the split purity
  - Can be mitigated by pruning
- Ensemble learning fixes all of this
  - Richer description when intersecting partitions
  - More accurate description when averaging partitions

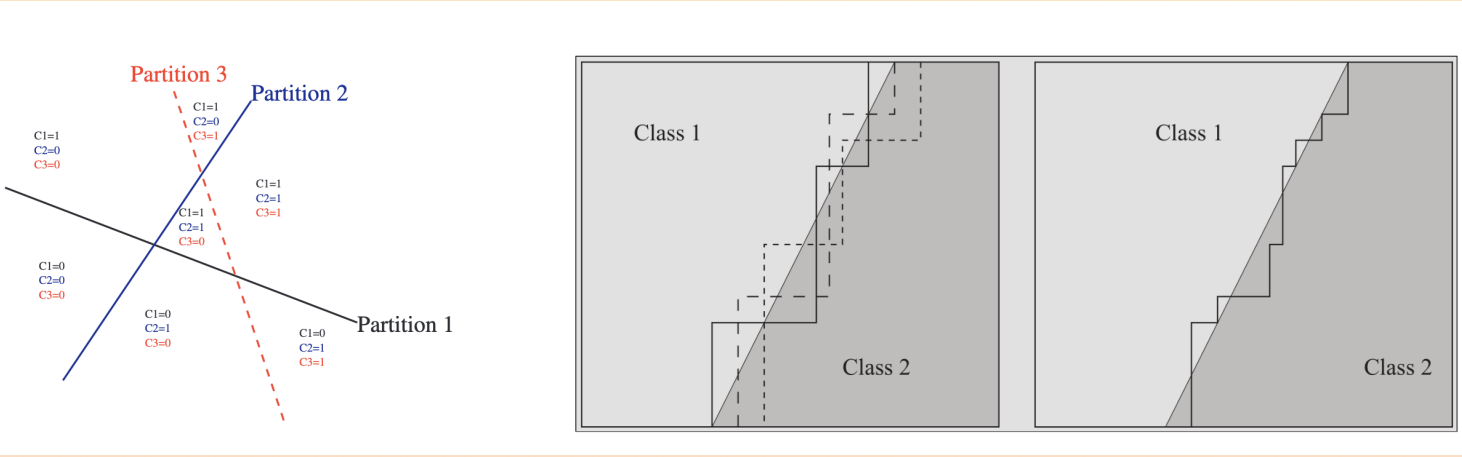
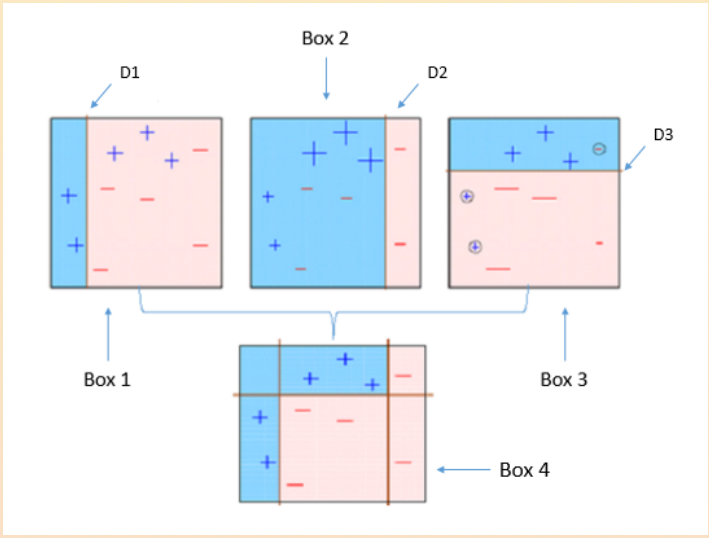


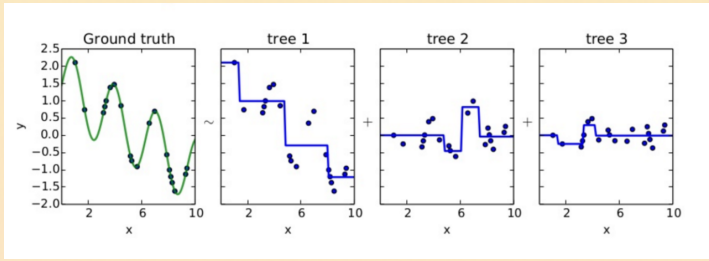
Image from 10.1142/9789811234033_0002

Boosted decision trees

- Ada(ptive) Boost
  - Increase at each iteration the importance of events incorrectly classified in the previous iteration



- Gradient Boost
  - fit the new predictor to the residual errors of the previous one



- Bagging: training trees on bootstrap replicas, average all trees
- Random forest: bagging + pick only a random subset of features at each step

I can't remember where I took the images from. Maybe the TMVA manual?

Performance (classification)

- Receiver operating characteristic (ROC) curve
  - Area $\in [0, 1]$, higher values are better

- For two classes (signal, background):
  - Signal efficiency
  - Background efficiency (rejection := 1-efficiency)
- Multiple classes: pairwise, one-vs-all

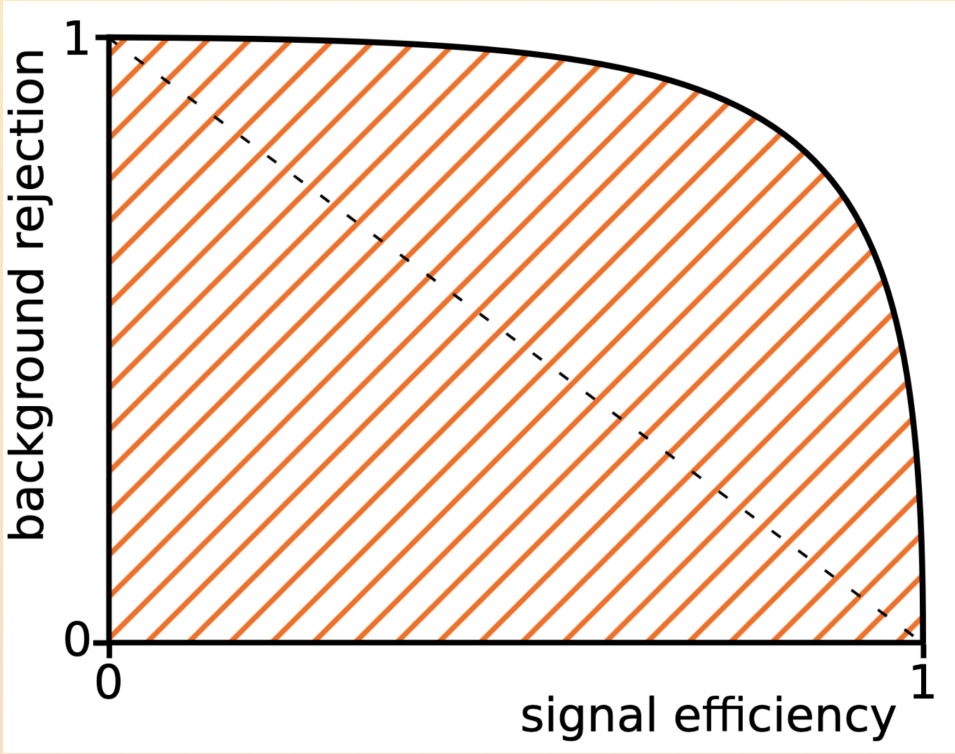


Image from 10.1142/9789811234033_0002

Two-dimensional scatter plot

- For regression problems
- Can compute linear pearson coefficient as an estimate of linearity
  - Formulas exist also for weighted events

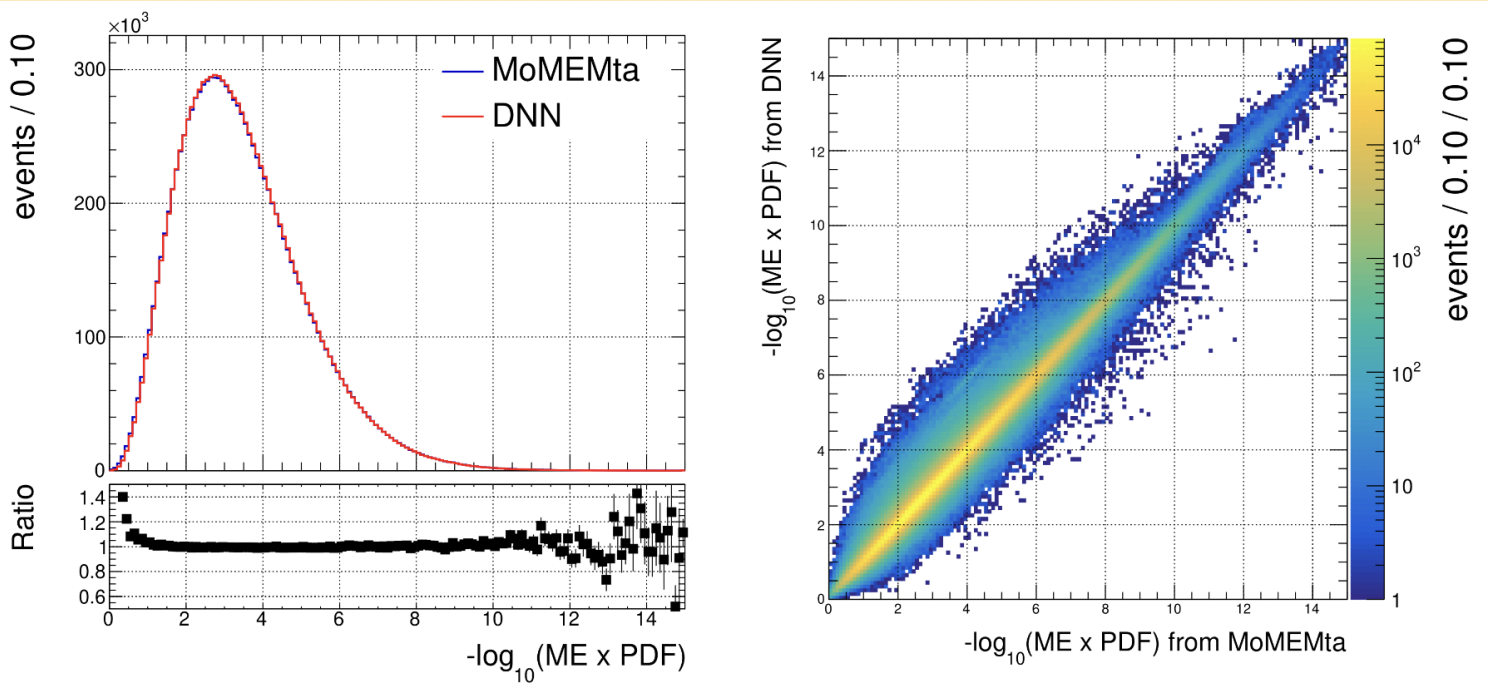


Image by Florian Bury, see also paper: 10.1007/JHEP04%282021%29020

Confusion matrix

- For classification, can also use it to discretize regression (e.g. in the case of histograms)
- Note the normalization (each true label row sums up to 1)

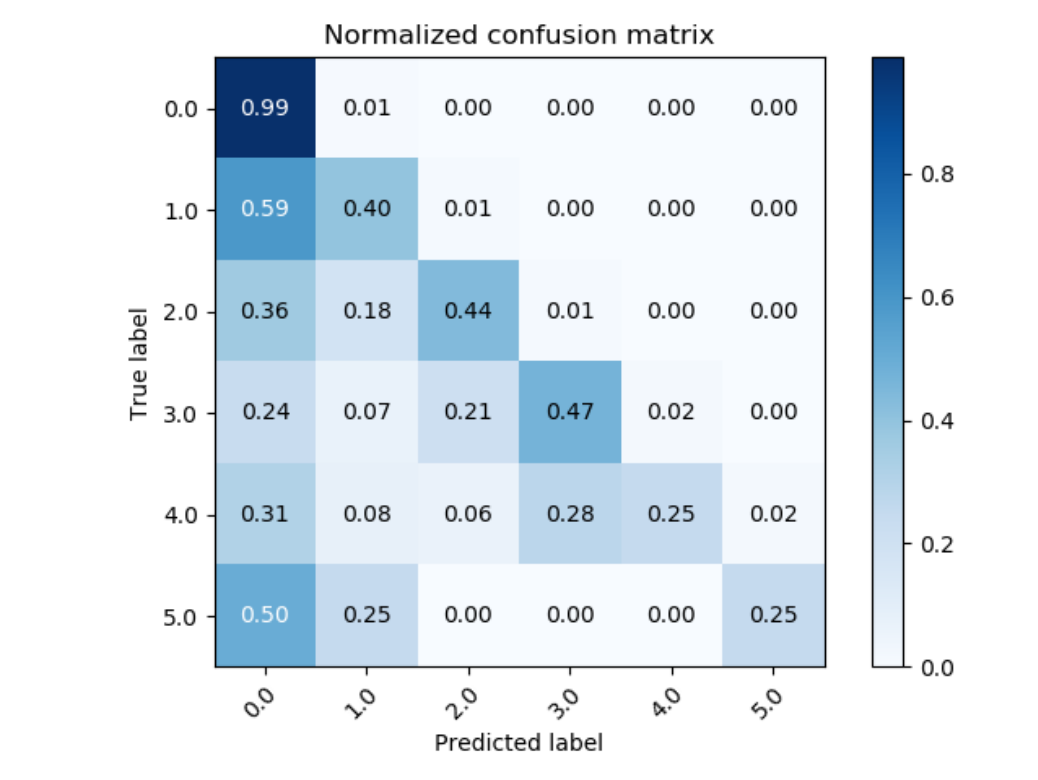


Image by seralouk on stackoverflow

Error rates (for reference)

		CONDITION determined by "Gold Standard"			
	TOTAL POPULATION	CONDITION POS	CONDITION NEG	PREVALENCE = CONDITION POS / TOTAL POPULATION	
TEST OUT-COME	TEST POS	True Pos TP	*Type I Error* False Pos FP	*Precision* Pos Predictive Value PPV = TP / TEST P	False Discovery Rate FDR = FP / TEST P
	TEST NEG	*Type II Error* False Neg FN	True Neg TN	False Omission Rate FOR = FN / TEST N	Neg Predictive Value NPV = TN / TEST N
	ACCURACY ACC ACC = (TP + TN) / TOT POP	*Sensitivity (SN), Recall* Total Pos Rate TPR TPR = TP / CONDITION POS	*Fall-Out* False Pos Rate FPR FPR = FP / CONDITION NEG	Pos Likelihood Ratio LR + LR + = TPR / FPR	Diagnostic Odds Ratio DOR DOR = LR + / LR -
		Miss Rate False Neg Rate FNR FNR = FN / CONDITION POS	*Specificity (SPC)* True Neg Rate TNR TNR = TN / CONDITION NEG	Neg Likelihood Ratio LR - LR - = TNR / FNR	

Image from unite.ai

Model complexity

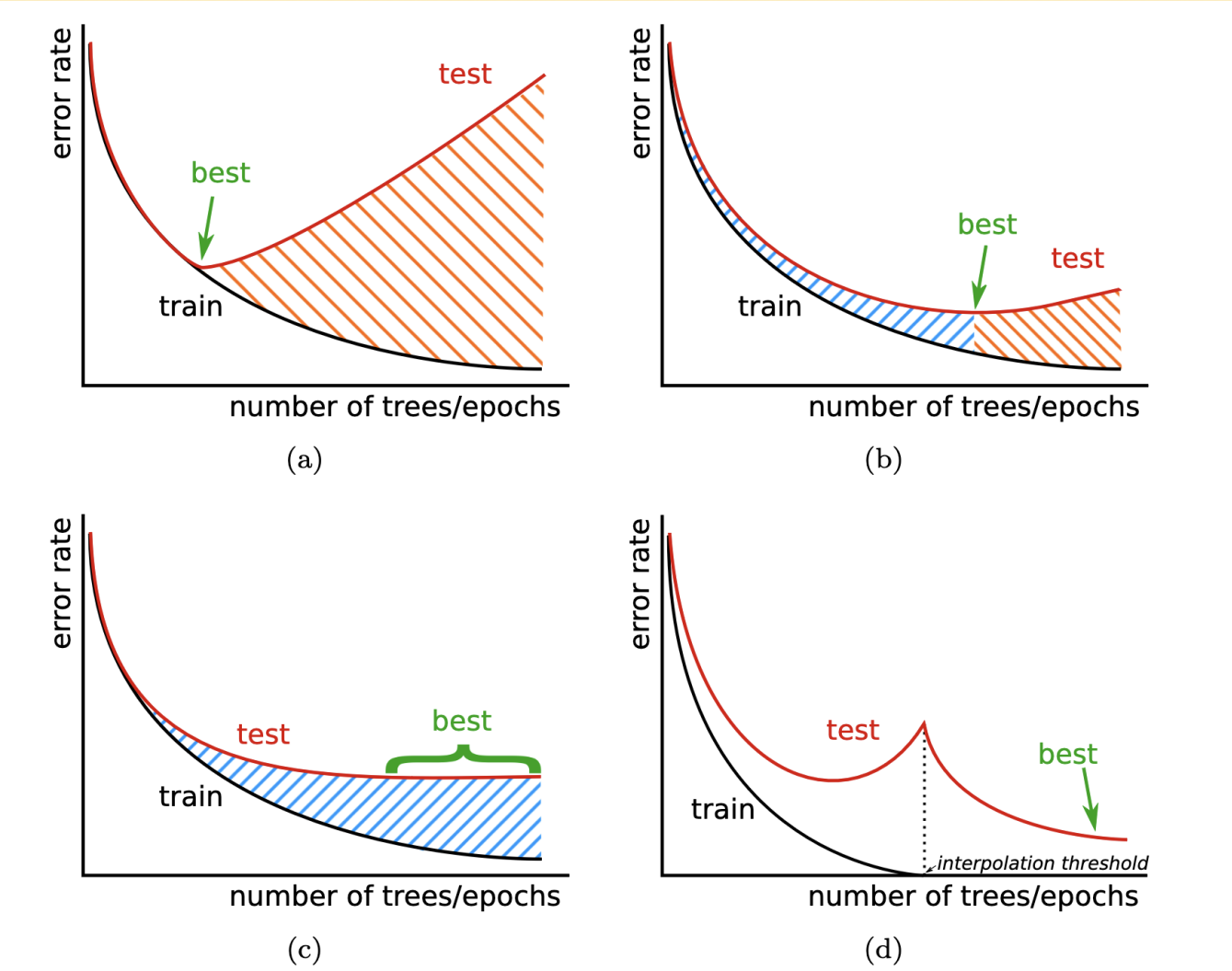
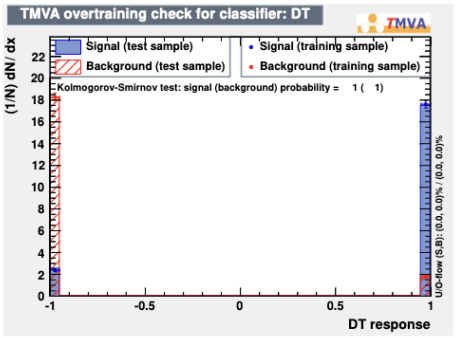


Image from 10.1142/9789811234033_0002

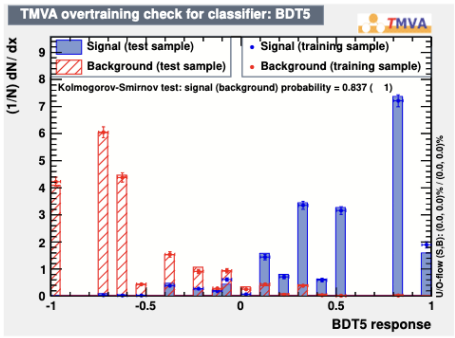
Overtraining check

- KS test done mostly for BDTs. For neural networks, much handier ways

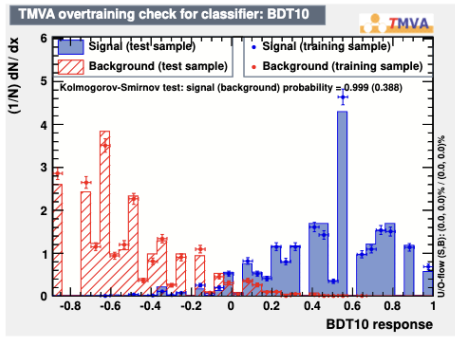
Do the training and test output distributions come from the same underlying p.d.f.?



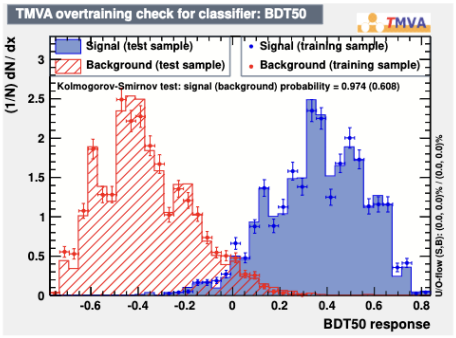
(a) Single decision tree



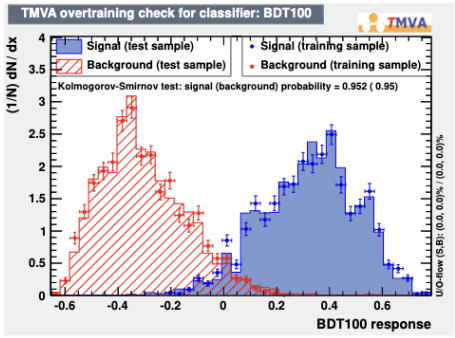
(b) 5 trees



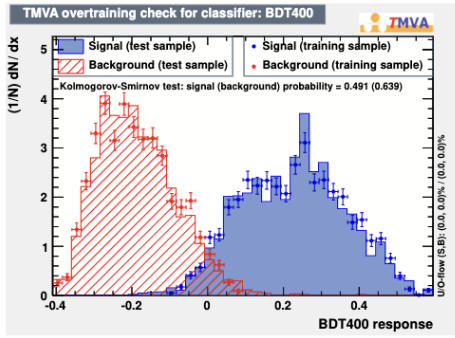
(c) 10 trees



(d) 50 trees



(e) 100 trees

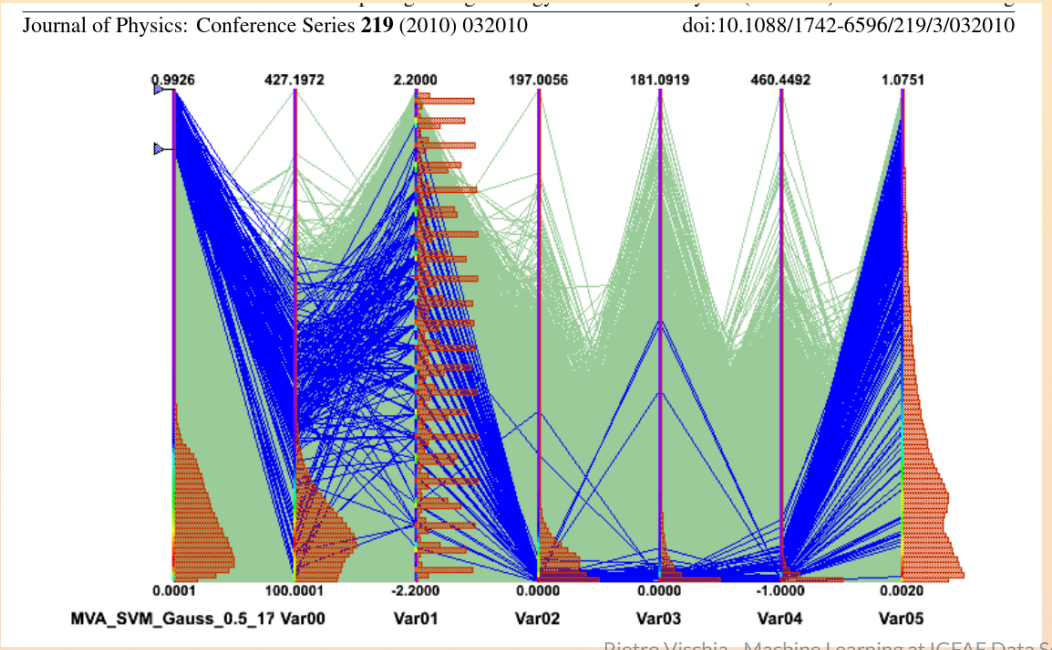


(f) 400 trees

Image from 10.1142/9789811234033_0002

Interpretability, explainability

- Permutation Importance: the decrease in a model score when a single feature value is randomly shuffled (see scikit-learn documentation) (akin to impacts for profile likelihood fits)
- Shapley Values: based on game theory (will use them this afternoon)
- Correlation-based: e.g. parallel coordinates in TMVA: look where each variable is mapped to/correlated with



Model assessment by comparing models

- Bayesian Information Criterion: $BIC = n_{free\,params} ln(n_{data})/2ln(\hat{L})$
  - Parameter θ predicted by two models M_0 and M_1: $P(\theta|\vec{x}, M) = \frac{P(\vec{x}|\theta,M)P(\theta|M)}{P(\vec{x}|M)}$
  - Apply Bayes theorem to Bayesian evidence (Model likelihood): $P(\vec{x}|M) = \int P(\vec{x}|\theta, M)P(\theta|M)d\theta$
  - Posterior odds: $\frac{P(M_0|\vec{x})}{P(M_1|\vec{x})} = \frac{P(\vec{x}|M_0)\pi(M_0)}{P(\vec{x}|M_1)\pi(M_1)}$
  - Can rewrite posteriors in terms of BIC, equivalent
- Minimum Description Length (MDL): Kolmogorov complexity (length of minimum program needed to describe the data)
  - $for\, i = 1\, to\, 2500; do\, print' 0001'; halt$
  - $print' 10100101010001011100100001000010111001110000101010010 1...'; halt$

Model assessment by comparing models

- Structural Risk Minimization: complexity as Vapnik-Chervonkensis class (largest number of shattered points)
  - Build a nested sequence of models with increasing VC complexity h
  - Write a probabilistic upper bound for the regression error: $err \leq f(h/N)$
  - Choose model with smallest value of the upper bound

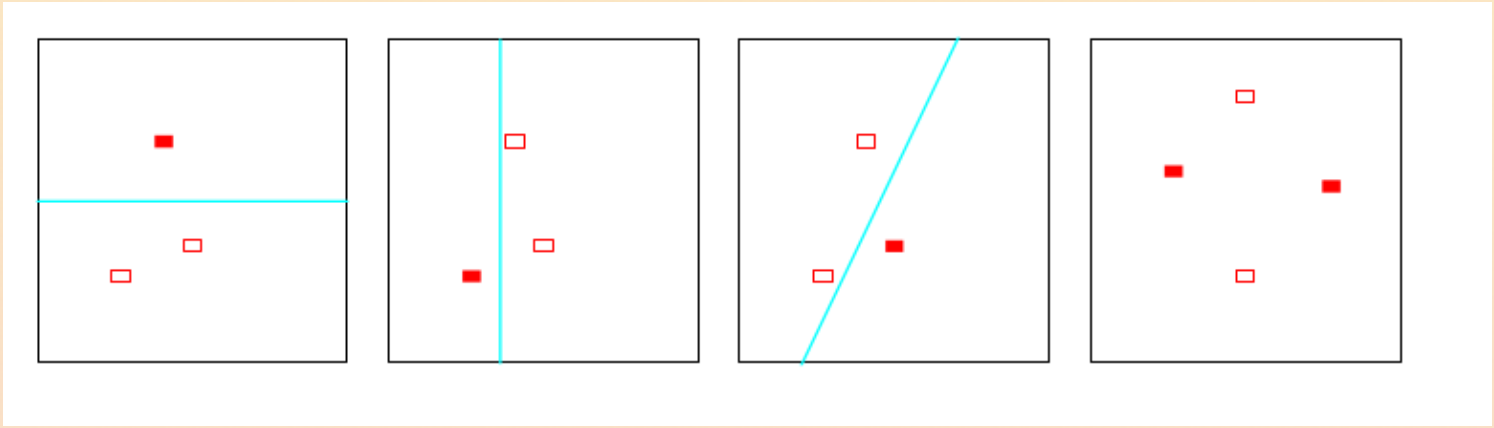
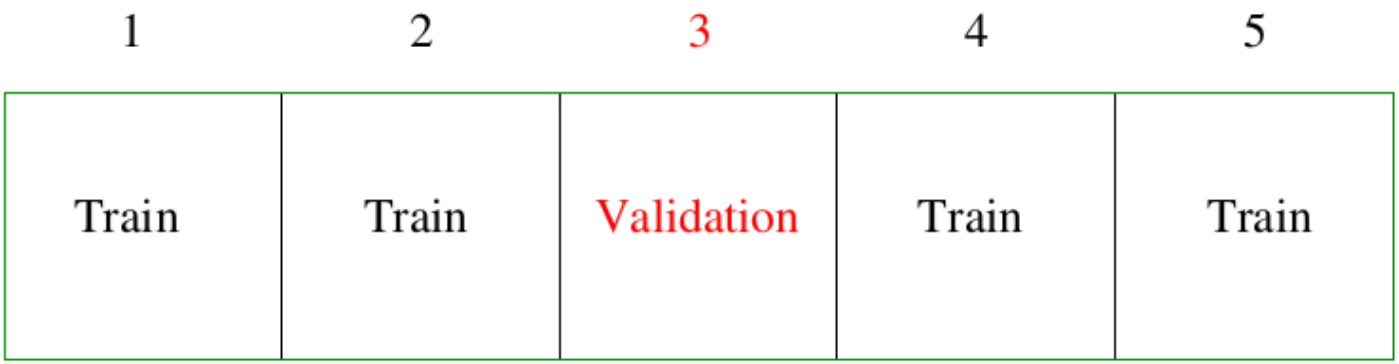


Image from Hastie, Tibshirani, Friedman

Model assessment: focus on prediction error

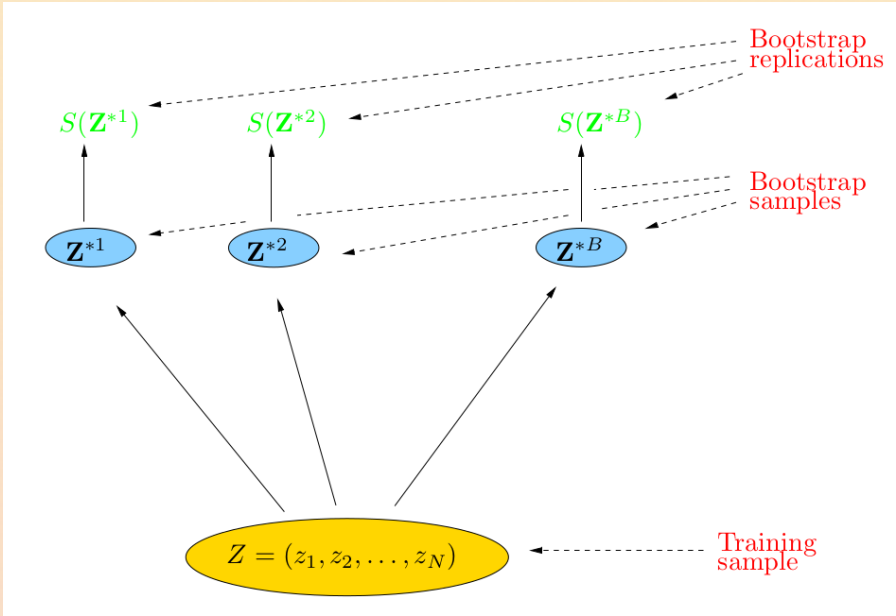
- Cross-validation: useful when data are scarce
  - Split the data into K parts ("folds")
  - For the kth part, fit the model to the other K-1 folds, and calculate test error as error on predicting the kth part data
  - Do this for all k, then combine the K estimates of the prediction error
  - Choose K
    - K=N (leave-one-out), unbiased but high variance (training sets are basically the same)
    - Low K (5--10): Lower variance, but maybe bias (folds not representative of the data set)



Images from Hastie, Tibshirani, Friedman

Model assessment: focus on prediction error

- Bootstrap: a general tool to assessing statistical accuracy
  - Estimate the variance on the statistic $S(Z)$ (Z are the data)
  - Can be used as model assessment tool, or to improve an estimator
  - Bagging to combine weak learners (ensemble learning)



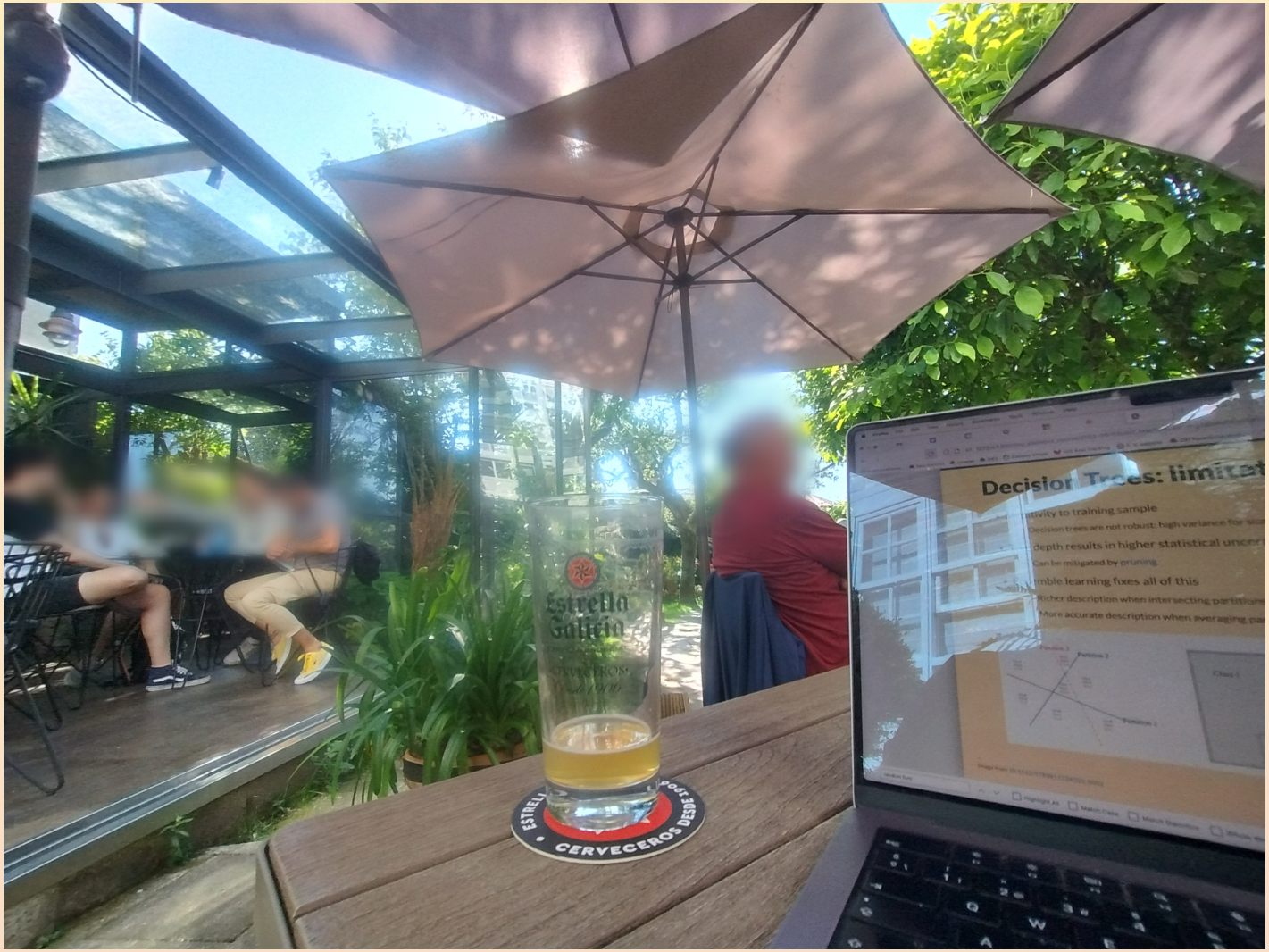
Images from Hastie, Tibshirani, Friedman

Exercises location

The public repository is at:
https://github.com/vischia/data_science_school_igfae2024

- The `README.md` contains instructions to run:
  - Locally on your machine (conda, virtualenv)
  - On Google Colab (in case you suspect your laptop may not be powerful enough for training neural networks in a timely manner (mostly relevant for tomorrow, today you should be fine)

Thank you!



—
Photo by me!