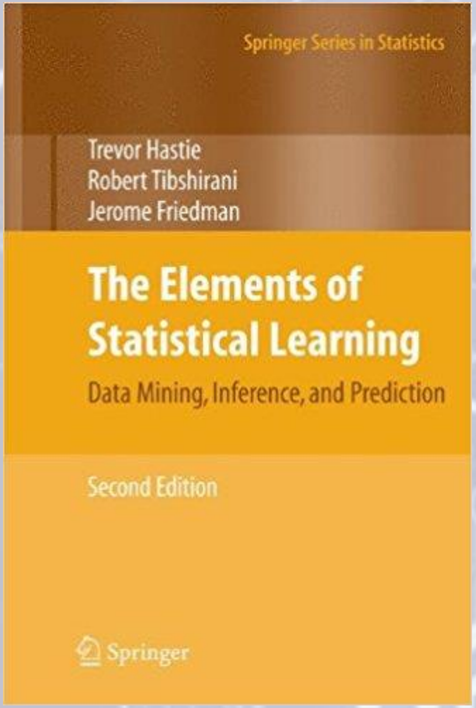
Introduction to Machine Learning

Tommaso Dorigo, INFN-Padova

XI ICNFP, Kolymbari 1/9/2022

Suggested reading

A couple of excellent textbooks which could be a good starting point for the interested student:



Springer Series in Statistics

Trevor Hastie
Robert Tibshirani
Jerome Friedman

The Elements of Statistical Learning

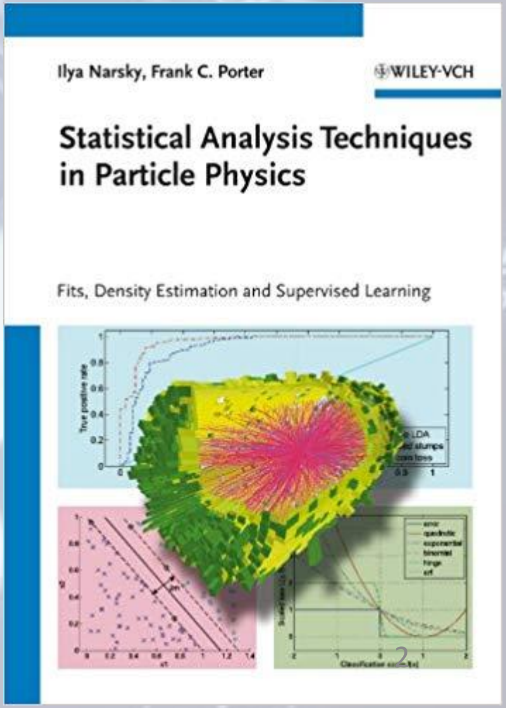
Data Mining, Inference, and Prediction

Second Edition

Springer

Hastie, Tibshirani, Friedman:
The elements of statistical learning
→ AVAILABLE ONLINE FOR FREE!

Narsky, Porter: Statistical Analysis techniques in Particle Physics, Wiley



Ilya Narsky, Frank C. Porter

WILEY-VCH

Statistical Analysis Techniques in Particle Physics

Fits, Density Estimation and Supervised Learning

Machine Learning is all around us

Areas of development for ML tools

More market-driven:

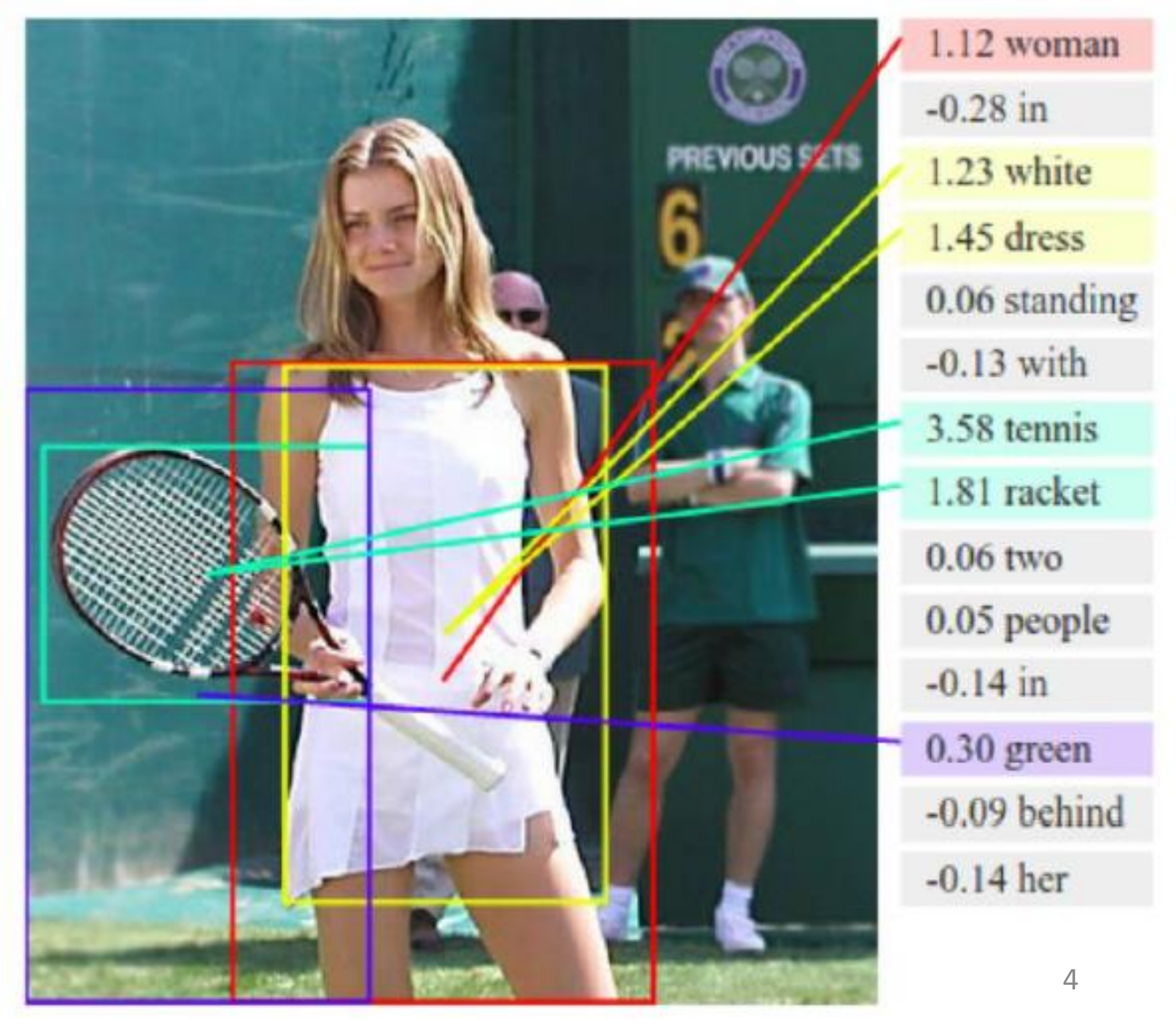
- Speech and handwriting
  - often the previous step
- Search engines, advertisi
  - ways to guess what you
- Stock market analysis, pr

More research-oriented:

- Master closed systems (g
- Natural language process
  - allows computers to ur
- Self-driving cars, object a
  - methods to give spatial
- AGI

→ Distinction more and mo

→ For fundamental physics



The path to Artificial Intelligence

- Alan Turing: machine shuffling 0's and 1's can simulate any mathematical deduction or computation

- Dartmouth conference, 1956 → many astonishing developments, checkers, automated theorem solvers
  - and funding from US defense budget

- In the seventies, research effort dampened
- Eighties: Expert systems
- Nineties: the second AI winter
- Development of Neural Networks
- Solutions to closed system problems; deep blue → alpha zero
- Toward AGI

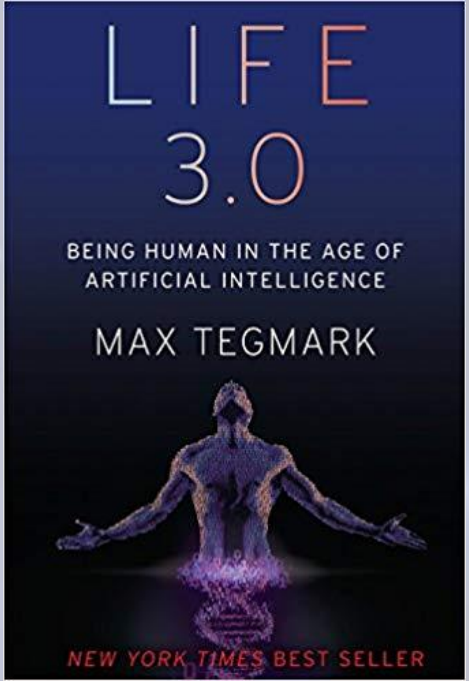
Is AI desirable?

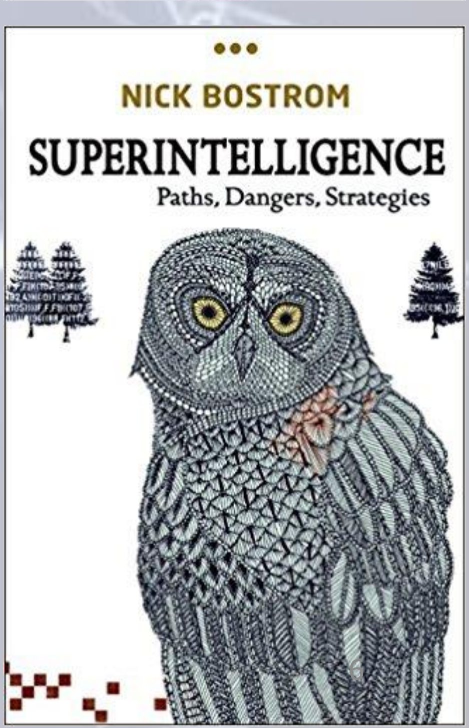
Humanity is going to face in the next few decades the biggest "growing pain" of its history: how to deal with the explosive force of artificial intelligence

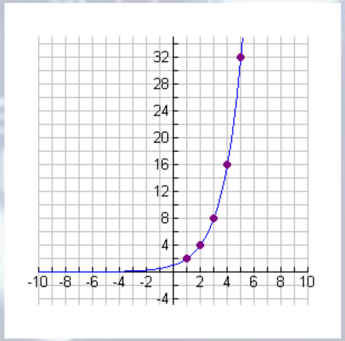
Once we create a superintelligence, there is no turning back – in terms of safeguards we have to "get it right the first time", or we risk to become extinct in a very short time

But we cannot stop AI research, no more than we could stop nuclear weapons proliferation

Interesting times ahead!


LIFE
3.0
BEING HUMAN IN THE AGE OF ARTIFICIAL INTELLIGENCE
MAX TEGMARK
NEW YORK TIMES BEST SELLER





Can we define Machine Learning?

It is useful to be clear what we are discussing here.
There are various options on how to define ML. Things I have heard around:

- *“[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.”* - Arthur Samuel (1959)
  - ok, but learning what?
- Wikipedia: "A scientific discipline concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data"
  - correct but slightly vague
- Mitchell (1997) provides a succinct definition: “*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E*"
- The fitting of data with complex functions
  - in the case of ML, we learn the parameters AND the function
- Mathematical models learnt from data that characterize the patterns, regularities, and relationships amongst variables in the system
  - more lengthy description of "fitting data"

Can we agree on what "Learning" is?

Maybe this really is the relevant question. Not idle to answer it, as by clarifying what our goal is we take a step in the right direction

Some **definitions of Learning**:

- the process of acquiring new, or modifying existing, knowledge, behaviors, skills, values, or preferences
- the acquisition of knowledge or skills through study, experience, or being taught.
- becoming aware of (something) by information or from observation.
- Also, still valid is Mitchell's definition in previous slide

In general: learning involves adapting our response to stimuli via a continuous inference. The inference is done by comparing new data to data already processed. How is that done? **By the use of analogy**.

What about analogies?

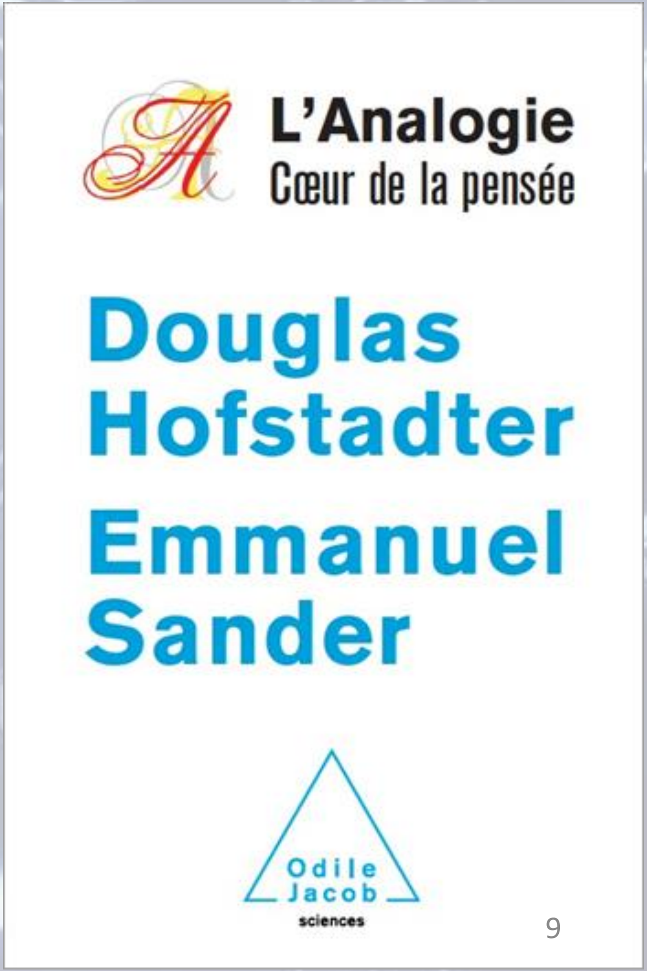
Analogies: arguably the building blocks of our learning process. We learn new features in a unknown object by analogy to known features in **similar** known objects

This is true for language, tools, complex systems – EVERYTHING WE INTERACT WITH

... But before we even start to use analogy for inference or guesswork, we have to **classify** unknown elements in their equivalence class!

(Also, one has to define a metric along which analogous objects align)

In a sense, **Classification** is even more "fundamental" in our learning process than is **Analogy.**
Or maybe we should say that classification is the key ingredient in building analogies.



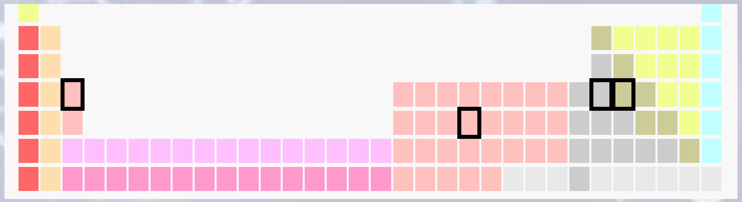
L'Analogie
Cœur de la pensée

Douglas Hofstadter
Emmanuel Sander

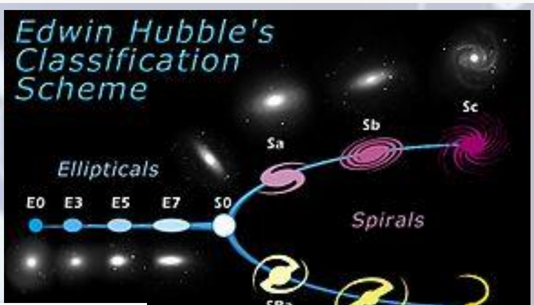
Odile Jacob
sciences

Classification and the scientific method

Countless examples may be made of how breakthroughs in the understanding of the world were brought by classifying observations



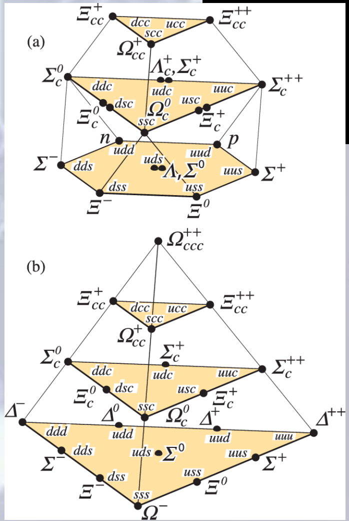
- Mendeleev's table, 1869 (150 years ago!)

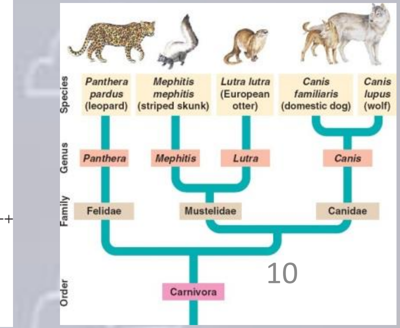


- Galaxies: Edwin Hubble, 1926

- Evolution of species: Darwin, 1859 (classification of organisms was instrumental in allowing a model of the evolution, although he used genealogy criteria rather than similarity)

- Hadrons: Gell-Mann and Zweig, 1964




Species
Genus
Family
Order
Panthera pardus (leopard)
Mephitis mephitis (striped skunk)
Lutra lutra (European otter)
Canis familiaris (domestic dog)
Canis lupus (wolf)
Panthera
Mephitis
Lutra
Canis
Felidae
Mustelidae
Canidae
Carnivora

Classification is at the heart of Decision Making

Finding similarities and distinctions between elements, creating classes, looks like an abstract task

Yet it is **at the heart of our decision making process**

→ in a game such as chess, **choosing the next move** entails a classification of the possible actions as good or bad ones, as a preprocessing step before going "deep" in the analysis

→ an **expert system** driving your vehicle must constantly decide how to react to external stimuli: it does so by classifying them (potentially dangerous → requires choice of proper reaction; irrelevant → can be ignored)

Decisions are informed by previous experience, and by information processing

That is how we "comprehend" the world and **assign it to some description**

Ultimately, an important component of intelligence is the continuous processing of our sensorial inputs, classifying them to react one way or the other to them

Classes of Statistical Learning algorithms

Supervised:

if we know the probability density of S and B, or if at least we can estimate it
→ E.g. we use "labeled" training events ("Signal" or "Background")
to estimate p(x|S), p(x|B) or their ratio

Semi-supervised:

it has been shown that even knowing the label for part of the data is usually sufficient to construct a classifier

Un-supervised:

if we lack an a-priori notion of the structure of the data, and we let an algorithm discover it without e.g. labeling classes → cluster analysis, anomaly detection, unconditional density estimation.

We may also single out, although use in HEP is scarce:

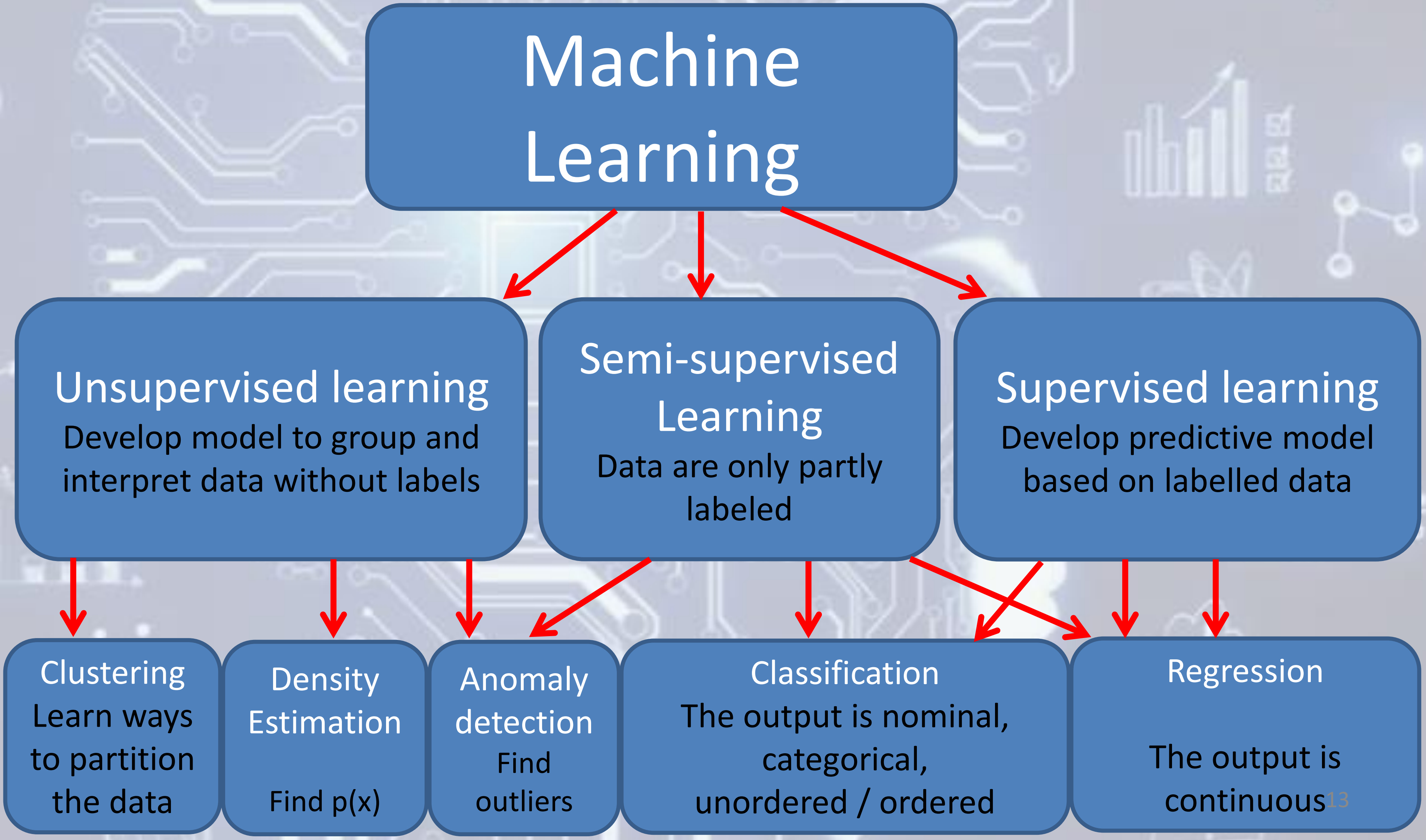
Reinforcement learning:

the algorithm learns from the success or failure of its own actions
→ E.g. a robot reaches its goal or fails

Active learning:

the algorithm queries an “oracle” (the user, or other source knowing the truth) in a way that maximizes the improvement of the model

A map to clarify the players role



Machine Learning

Unsupervised learning

Develop model to group and interpret data without labels

Semi-supervised Learning

Data are only partly labeled

Supervised learning

Develop predictive model based on labelled data

Clustering

Learn ways to partition the data

Density Estimation

Find p(x)

Anomaly detection

Find outliers

Classification

The output is nominal, categorical, unordered / ordered

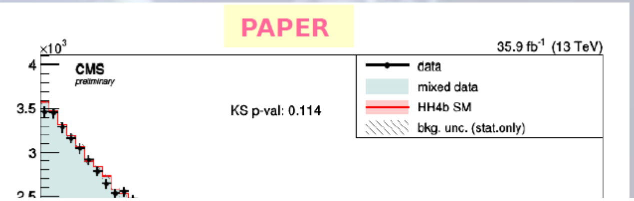
Regression

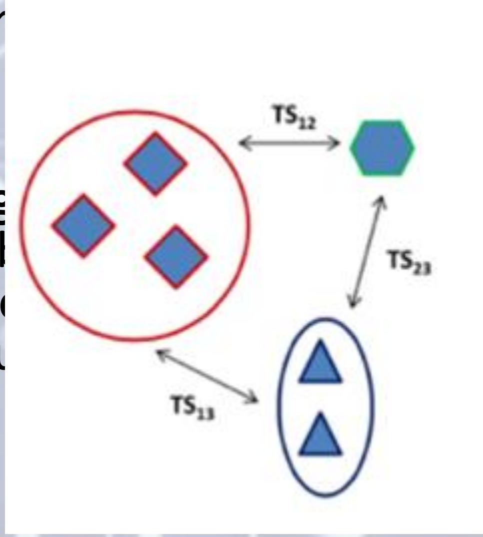
The output is continuous

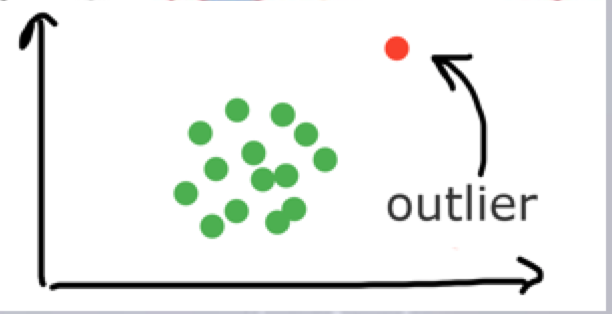
A more complete list of ML tasks /1

Classification and Regression are important tasks belonging to the "supervised" realm. But there are many other tasks:

- **Density estimation**: this is usually an ingredient of classification, but can be a task of its own.
- **Clustering**: find structures in the data, organize be a useful input to other tasks
- **Anomaly detection** (e.g. fr purchasing habits; or new
- **Classification with missing** from simple classification each mapping into the cat of missing components (bu handle it)

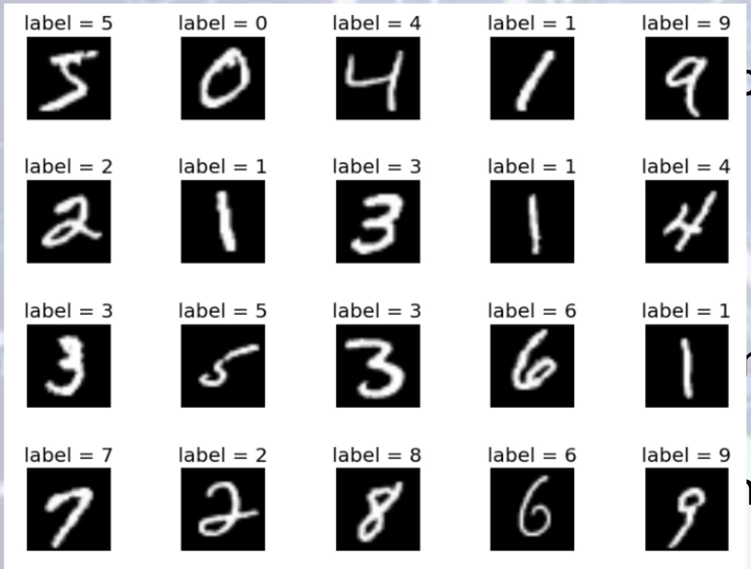






A more complete list of ML tasks /2

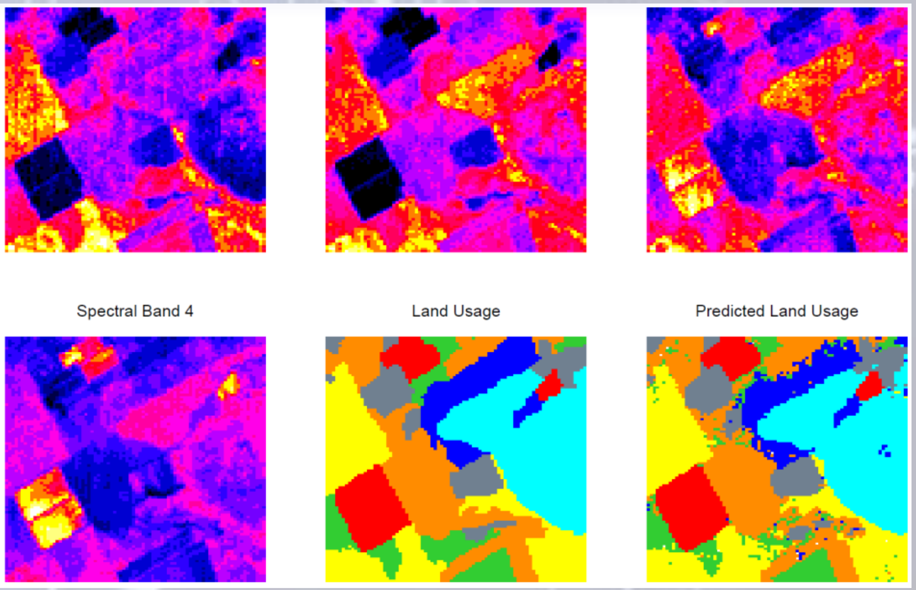
- **Structured output**:
  - **Transcription**: e.g. transform unstructured representation of data into discrete textual form; e.g. images of handwriting or numbers (Google Street view does it with DNN for street addresses), or speech recognition from audio stream
  - **Machine translation**
  - **parsing sentences** into grammatical structures
  - **image segmentation**, e.g. aerial pictures → road positions or land usage
  - **image captioning**



peech sy
ch input

this can
nd custo

n exampl

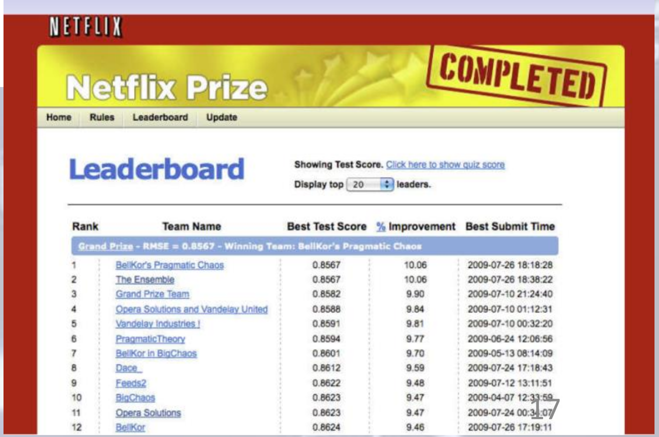


The supervised learning problem

- **Starting point**:
  - A vector of n predictor measurements X (a.k.a. inputs, regressors, covariates, features, independent variables).
  - One has training data $\{(x,y)\}$: events (or examples, instances, observations...)
  - The outcome measurement Y (a.k.a. dependent variable, or response, or target)
    - In classification problems Y can take a discrete, unordered set of values (signal/background, index, type of class)
    - In regression, Y has a continuous value
- **Objective**:
  - Using the data at hand, we want to predict y^* given x^*, when (x^*,y^*) does not necessarily belong to the training set.
  - The prediction should be **accurate**: $|f(x^*)-y^*|$ must be small according to some useful metric (see later)
- We would like to also:
  - understand what feature of X affects the outcome and how
  - assess the quality of our prediction

Example: the Netflix challenge

- competition started in October 2006. Training data is ratings for 18,000 movies by 400,000 Netflix customers, each rating between 1 and 5.
- training data is very sparse— about 98% missing.
- objective is to predict the rating for a set of 1 million customer-movie pairs that are missing in the training data.
- Netflix's original algorithm achieved a root MSE of 0.953. The first team to achieve a 10% improvement wins one million dollars.



Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace_	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11

The unsupervised learning problem

- **Starting point**:
  - A vector of n predictor measurements X (a.k.a. inputs, regressors, covariates, features, independent variables).
  - One has training data {x}: events (or examples, instances, observations...)
  - There is no outcome variable Y

- **Objective** is much fuzzier:
  - Using the data at hand, find groups of events that behave similarly, find features that behave similarly, find linear combinations of features exhibiting largest variation

- Hard to find a metric to see how well you are doing
- Result can be useful as a pre-processing step for supervised learning

Ideal predictions, a' la Bayes

For a regression problem, the best prediction we can make for Y based on the input X=x is given by the function

$f(x) = \text{Ave}\ (Y|X=x)$

This is the conditional expectation: you just derive the Y average for all examples having the relevant x.

- It is the best predictor if we want to minimize the average squared error,

  $\text{Ave}\ (Y-f(X))^2$.

- But it is NOT the best predictor if you use other metrics. E.g., if you wish to minimize $\text{Ave}\ |Y-f(X)|$ you should rather pick... Who can guess it?

  $f(x) = \text{Median}\ (Y|X=x)$

If we instead are after a qualitative output Y in {1...M} (a discrete one, as in multi-class classification tasks) what we can do is to compute

$P\ (Y=m|X=x)$

for each m: conditional probability of class m at position X=x; then we take as the class prediction

$C(x) = \text{Arg max}_j\ \{P(Y=j|X=x)\}$.

The above is the majority vote classifier.

Problem solved? Let us try and see how to implement these ideas.

Implementation

To predict Y at X=x*, collect all pairs (x*,y) in your training data, then

- For regression, get

  f(x*) = Ave (y|X=x*)

- For classification, get

  c(x*) = Arg max$_j$ {P(Y=j|X=x*)}

Alas, this would be good, but...

We usually have sparse training data, obtained by forward simulation. Our simulator gives p(x|y) but the process is stochastic. That means we cannot invert the simulator, extracting p(y|x)!

In most cases we have NO observations with X=x*.

Who you're gonna call ?

Density estimation methods!

DENSITY ESTIMATION

Density Estimation

Given a sample of data X, one wishes to determine their prior PDF p(X).

One can solve this with parametric or non-parametric approaches.

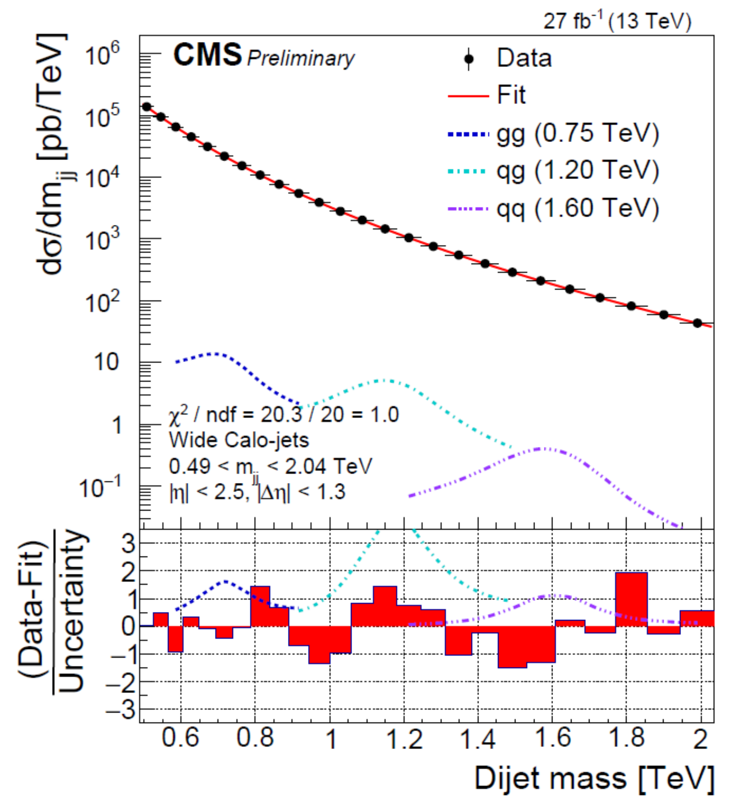
Parametric: find a model within a class, which fits the observed density

→ an assumption is necessary as the starting point

Non-parametric: sample-based estimators.

The most common is the histogram. NP density estimator have a number of attractive properties for experimental sciences:

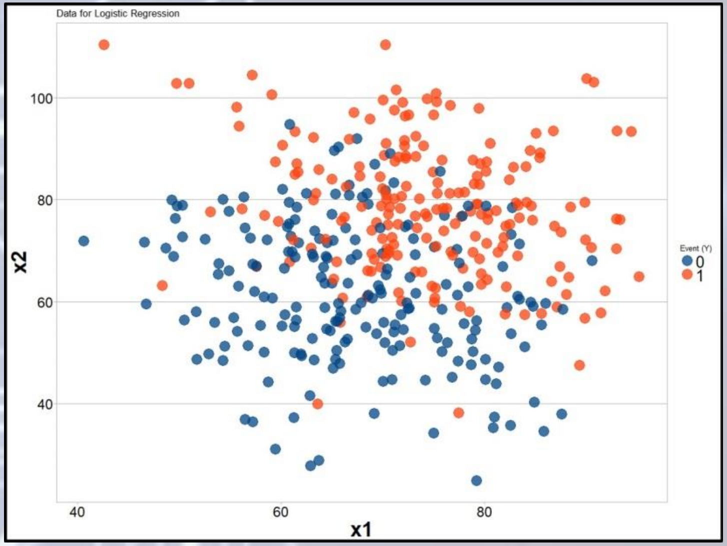
- are easy to use for two things dear to HEP/astro-HEP: efficiency estimates (e.g. from Bernoulli trials) and for background subtraction
- lend themselves to be good inputs to unfolding methods
- are an excellent visualization tool, both in 1 and 2D



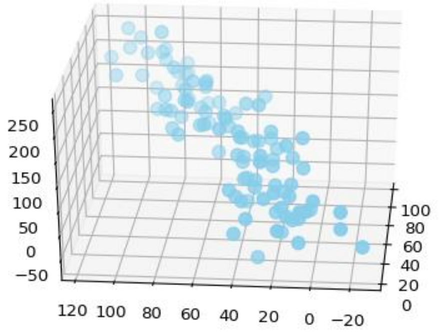
The empirical density estimate

With sparse data, the most obvious estimate of the density from which i.i.d. data $\{x_i\}$ are drawn is called "empirical probability density function" (EPDF), which can be obtained by placing a Dirac delta function at each observation x_i:

$$\hat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} \delta(x - x_i)$$



Of course, the EPDF is rarely useful in practice as a computation tool. However, it is a common way of visualizing 2D or 3D data (scatterplots)



Histograms

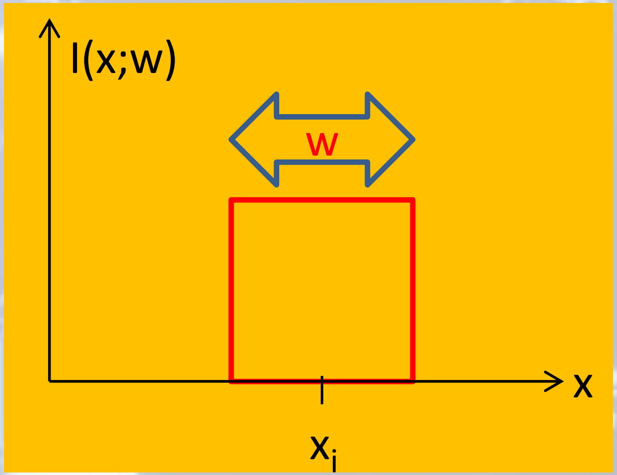
Histograms are a versatile, intuitive, ubiquitous way to get a quick density estimate from data points:

$$h(x) = \mathrm{Sum}_{i=1\ldots N}\ I\ (x-x_i;w)$$

where w is the width of the bin, and I is a uniform interval function (indicator function),

$I\ (x;w) = 1$ for x in [-w/2,w/2],

$= 0$ otherwise



The density estimate provided by h(x) is then

$$f(x) = h(x)/(Nw)$$

Yet **histograms have drawbacks**:

- they are *discontinuous*
- they *lose information* on the true location of each data points through the use of a "regularization", the bin width w
- *not unique*: there is a 2D infinity of histogram-based PDF estimates possible for each dataset, depending on binning and offset.

Kernel density estimation

A useful generalization of the histogram is obtained by substituting the indicator function with a suitable "kernel":

$$\hat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} k(x - x_i; w)$$

The kernel function k(x,w) is normalized to unity. It is typically of the form

$$\mathrm{k}(x - x_i; w) = \frac{1}{w} K(\frac{x - x_i}{w})$$

The advantage of using a kernel instead than a delta is evident: we obtain a continuous, smooth function. This comes, of course, at the cost of a modeling assumption.

A common kernel is the **Gaussian distribution**; the results however depend more on the smoothing parameter w than on the choice of the specific form of k.

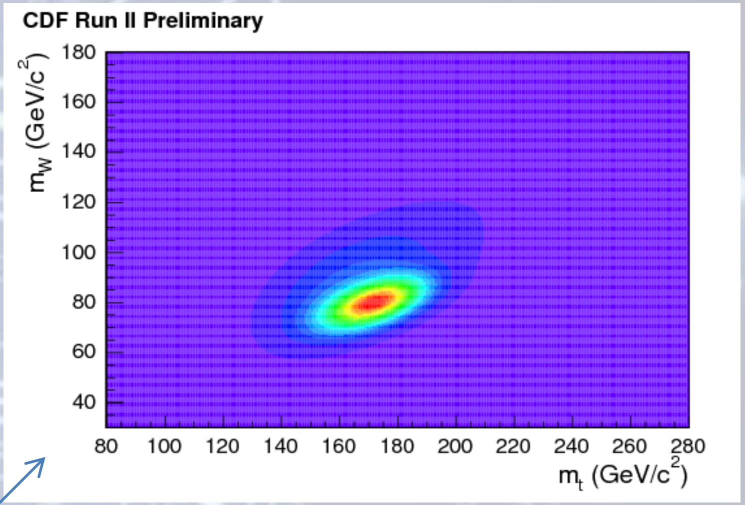
The "kernelization" of the data can be operated in multiple dimensions too. The kernels operating in each component are identical but may have different smoothing parameters:

$$\hat{f}(x, y) = \frac{1}{N w_x w_y}\sum_{i=1}^{N} K(\frac{x - x_i}{w_x}) K(\frac{y - y_i}{w_y})$$

A different extension of the KDE idea is to adapt the smoothness parameter w to reflect the precision of the local estimate of the data density → adaptive kernel estimation

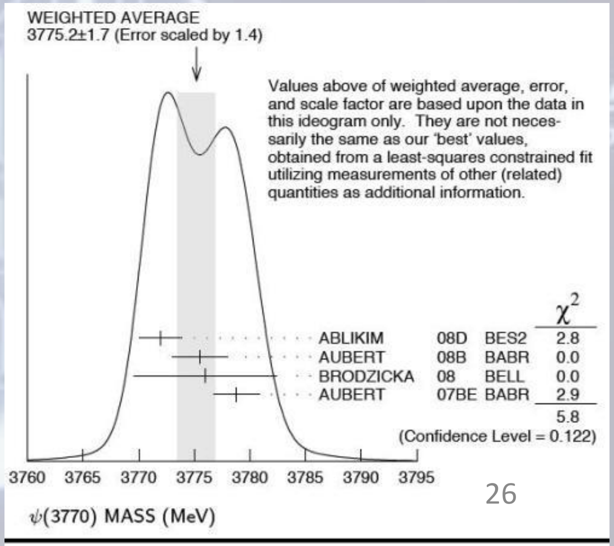
Ideograms

A further extension of KDE is the use of different kernels for different data points → **ideograms**. This may be very useful if the data coordinate x comes endowed with information on its measurement precision σ. In that case we may substitute the point with a Gaussian kernel, whose smoothing parameter w is equal to the precision σ. This improves the overall density estimate



The ideogram method is sometimes used in HEP, e.g. it has been employed in top quark mass measurements.

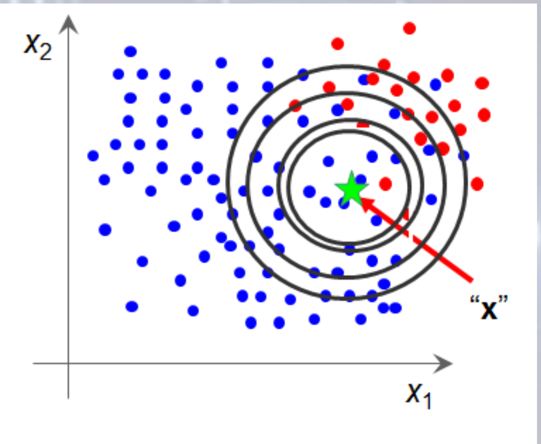
The PDG uses it to report the PDF of a unknown parameter when the individual estimates carry different uncertainty AND there is tension between the estimates (such that the weighted average is not necessarily the whole story)



Values above of weighted average, error, and scale factor are based upon the data in this ideogram only. They are not necessarily the same as our 'best' values, obtained from a least-squares constrained fit utilizing measurements of other (related) quantities as additional information.

Going multi-D: K-Nearest Neighbours

The kNN algorithm tries to determine the local density of multi-dimensional data by finding how many data points are contained in the **surroundings** of every point in feature space



One usually "weighs" data points with a suitable function of the distance from the test point

Problem: how to define the distance in an abstract space?

Also, your features might include real numbers, categories, days of the week, etcetera...
In general, it is useful to remove the dimensional nature of the features

General recipe: **standardization**

- for continuous variable x: find variance σ^2, obtain standardized variable $y^2 = x^2 / \sigma^2$
- for categorical variable c:
  - if target has large variance along c, can keep it as is (defines hyperplane, will only use data with c*=c to estimate local density)
  - otherwise may still try to standardize

More on kNN

Once the data is properly standardized one can construct an Euclidean distance:

$$D(y,y') = \sum_{i=1}^{D} (y - y')^2$$

kNN density estimates can be endowed with several parameters to improve their performance

Most obvious is k: how many events in the ball?

Rule of thumb: estimating a mean → if target varies a lot, can use small k; if variation small, k needs to allow precise estimate

Common to have k=20-50, but it of course depends on dimensionality D and size N of training data

kNN, continued

Also crucial to assess:

- **relative importance of variables**: assign larger weight to more meaningful components in the feature space (ones along which target has largest variance)
- **local gradients-aware**: one may try and adapt the shape of the "hyperellipsoid" to reflect how much target y(x) is variable in each direction, at test point x*

In general, kNN estimates suffer from a couple of shortcomings:

- The evaluation of local density requires to use all data for each point of feature space → **CPU expensive** (but there are shortcuts)
- The **curse of dimensionality**: for D>8-9, they become insensitive to local density (see next slide)

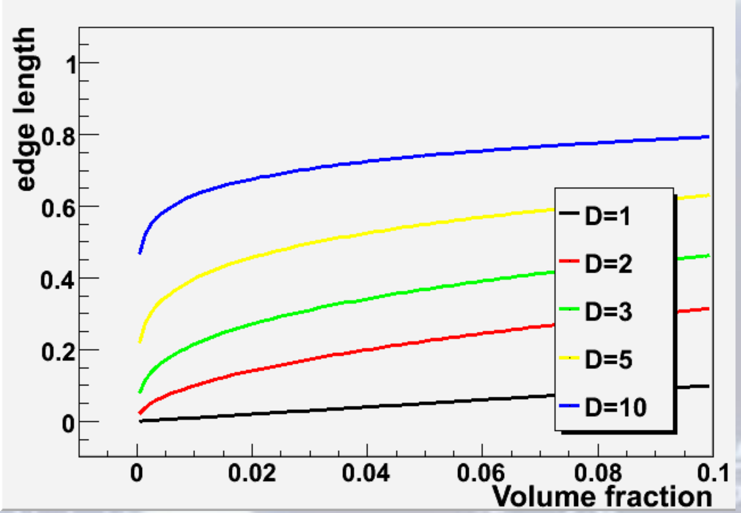
The Curse of Dimensionality

The estimate of density in a D-dimensional space is usually impossible, due to the lack of sufficient labeled data for the calculation to be meaningful. An adequate amount of data must grow exponentially with D for a good representation

For high problem dimensionality, the k "closest" events are in no way close to the point where an estimate of the density is sought:

$$\text{edge length} = (\text{fraction of volume})^{1/D}$$

This makes kNN impractical for D > 8-10 dimensions



In 10 dimensions, if a hypersphere captures 1% of the feature space, it has a radius of 63% in each variable span

HEP analyses often have >10 important variables so the kNN has limited use as a generative algorithm for S/B discrimination

But, one can apply dimensional reduction techniques to still use it, or resample subspaces

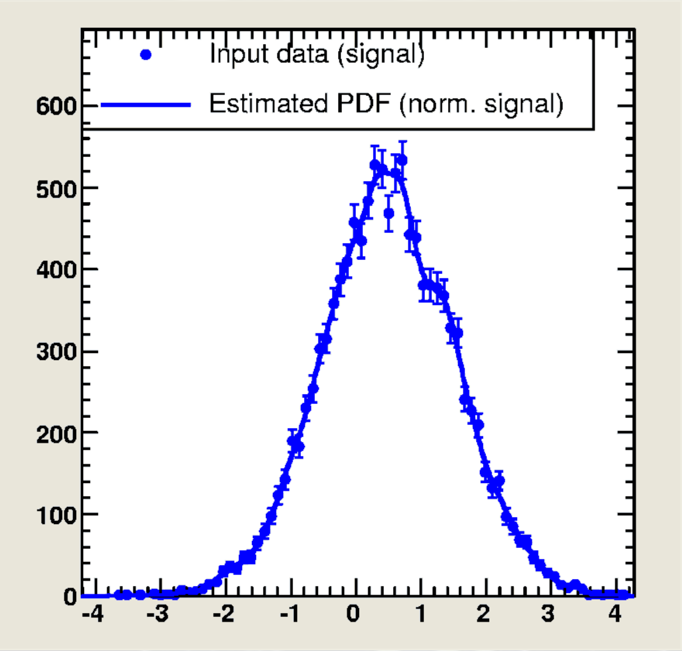
What If We Ignore Correlations?

All PDF-estimation methods similarly fail when D is large.
A "Naïve Bayesian" approach consists in ignoring altogether the correlations between the variables of the feature space

That is, one only looks at the "marginals": $P(\mathbf{x}) \cong \prod_{i=0}^{D} P_i(\mathbf{x})$

P is a product of "marginal PDFs" P_i

One usually models the P_i with a smoothing of histograms



The P(x) thus obtained can be used to construct a discriminant (a likelihood ratio), or more simply, to assign the class label to the highest p(x) given x

The method works if correlations are unimportant.
In HEP, however, this is not usually the case!

RESAMPLING TECHNIQUES

Resampling Techniques

Resampling techniques are pivotal for a number of ML tasks:

- hypothesis testing (construction of discriminative methods)
- estimation of bias and variance (optimization of predictors)
- cross-validation (estimate accuracy of predictors)

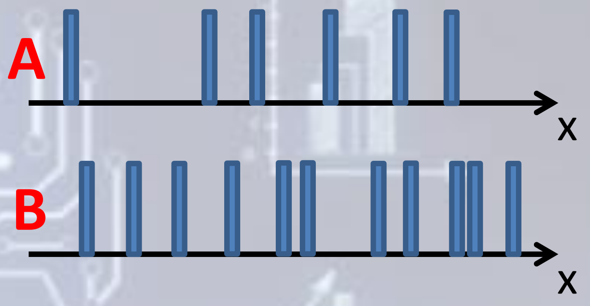
Resampling allows you to avoid modeling assumptions, as you construct a non-parametric model from the data themselves. The benefits can be huge (as measured e.g. by performance of boosting methods)

Here we only briefly flash the generalities of three basic ingredients:

- permutation tests
- bootstrap
- jacknife
- ~~cross-validation techniques~~ ← will treat later

The theoretical basis for resampling methods lies with their (usually very good) asymptotic properties of consistency and convergence (in probability)

Permutation Sampling

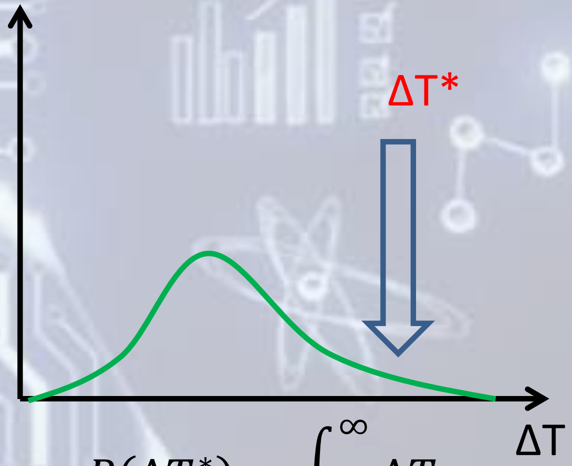


Permutation sampling is mainly used in hypothesis tests; usually the question is: are datasets A $\{x_1 \ldots X_{N_A}\}$ and B $\{x_1 \ldots x_{N_B}\}$ sampled from the same parent PDF?

The way to answer is to form a sensitive statistic T (say: the mean of the observations) and **compare quantitatively** the difference between T_A and T_B

$$\Delta T^* = T_A - T_B$$

with what one would expect if the PDF were the same.



$$P(\Delta T^*) = \int_{\Delta T^*}^{\infty} \Delta T$$

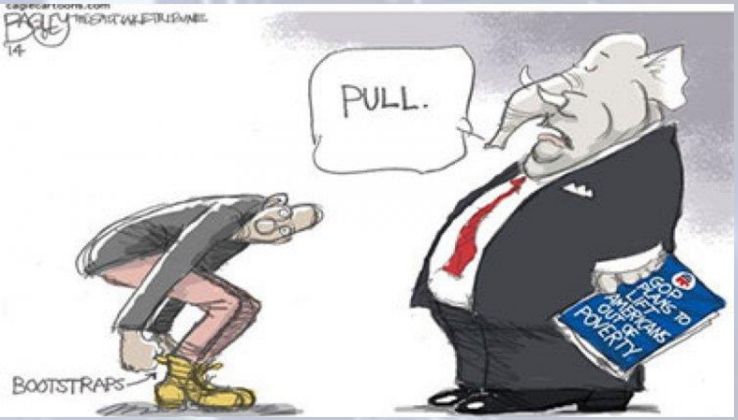
The comparison requires to know how T distributes under the null hypothesis (green curve).

Here permutation comes in: we assume A=B, merge A+B=C, and sample the N_A+N_B observations creating all possible pairs of splits A', B' still of size $\{N_A,N_B\}$, computing distances $\Delta T = T_{A'} - T_{B'}$.

This provides all the information in the data about the distribution of ΔT. The permutation test makes no model assumptions, so it is called an **exact** test.

The Bootstrap

The Bootstrap (B.Efron, 1979) is called this way because it allows to "*pull oneself up from one's own bootstraps*".


BAGLEY THE SALT LAKE TRIBUNE
'14
PULL.
BOOTSTRAPS
GOP PLANS TO LIFT AMERICANS OUT OF POVERTY

Motivating problem: get the variance of an estimator $\hat{\theta}(x)$ of a parameter θ, for a sample of i.i.d. observations $\{x_1...x_N\}$.

We may generate M replicas of the dataset X by repeatedly picking N observations at random from X, with replacement.

Then we estimate θ in each replica, and proceed to obtain a sample mean and variance with the $\widehat{\theta_i}(x)$,

$$\bar{\theta} = \frac{1}{M}\sum_{i=1}^{M}\theta_i$$

$$s^2{}_\theta = \frac{1}{M-1}\sum_{i=1}^{M}(\hat{\theta}_i - \bar{\theta})^2$$

You get an estimate of variance without assumptions of the distribution

Jackknife resampling

The jackknife is called after the large pocket knife one brings around in an excursion – it is indeed a versatile statistics tool.

Again the most common use is determining the bias and variance of an estimator. One works out n different estimates of the estimator by leaving out in turn each one of the n i.i.d. data points $\{x_1 \ldots x_n\}$.

$$\bar{x}_i = \frac{1}{n-1} \sum_{j=1, j \neq i}^{n} x_j, \qquad i = 1, \ldots, n$$

One may then get e.g. an estimate of the variance of the mean:

$$\mathrm{Var}(\bar{x}) = \frac{n-1}{n} \sum_{i=1}^{n} (\bar{x}_i - \bar{x})^2$$

Given an estimator θ, we may also compute its bias in a similar way: if the jacknife average is

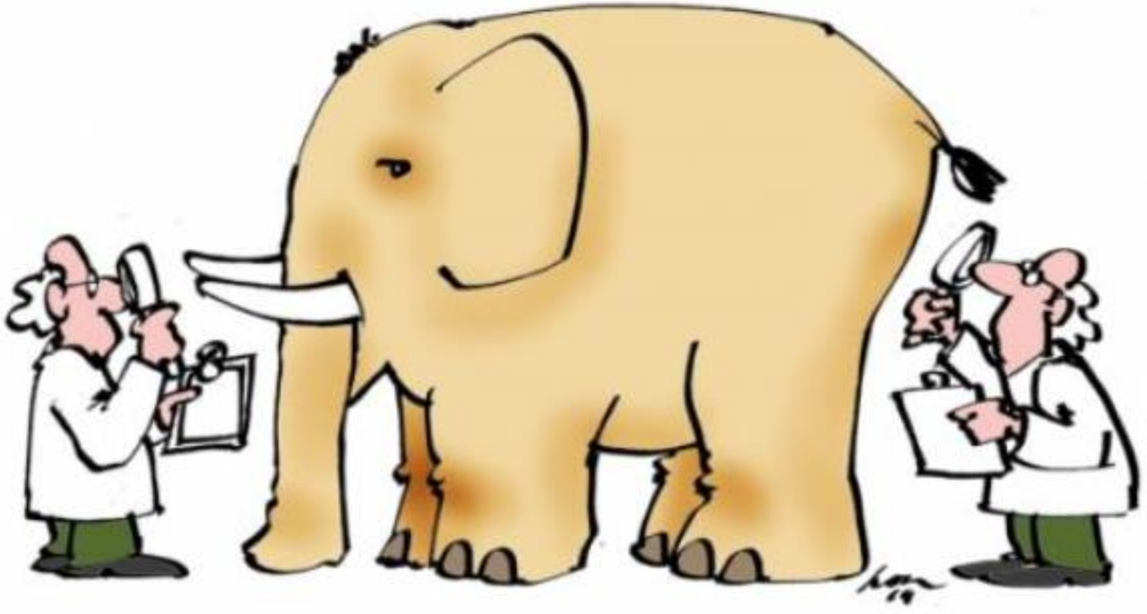
$$\hat{\theta}_{(.)} = \frac{1}{n} \sum_{i=1}^{n} \hat{\theta}_{(\mathrm{i})}$$

where $\hat{\theta}_{(i)}$ is the estimate obtained by leaving out the i-th value,

then the bias on the estimator is $$\widehat{\mathrm{Bias}}_{(\theta)} = (n-1)(\hat{\theta}_{(.)} - \hat{\theta})$$

What is better between bootstrap and jackknife? it depends on the problem! For smallish N jackknife is faster; however the bootstrap uses more information with non-linear estimators.

CONCLUSIONS



"Statistics: The only science that enables different experts using the same figures to draw different conclusions."
Evan Esar

Conclusions

- Machine learning has a large overlap with statistical learning, which has been around for much longer.
  - Emphasis is on large-scale applications, and on prediction accuracy (as opposed to emphasis on models and their uncertainty)

- Density estimation is an important ingredient of many ML methods
  - especially when they require pdfs as inputs

- Data preprocessing may be an essential step that pays dividends in performance later on

- In fundamental science we often deal with the lack of analytic likelihoods
  - ML methods can provide effective approximations to summary statistics to carry out the inference work