3rd RED LHC workshop

6-8 May 2019
UAM, Madrid
Europe/Madrid timezone

Overview on Machine Learning (for the LHC)

Bryan Zaldívar
UAM & IFT

Outline

Brief introduction to Machine Learning

- Definition
- Paradigms
- Relevant models for the LHC

Examples of ML application to LHC physics

- Detector Simulation
- Object reconstruction
- Signal-to-background maximization
- ...among others

Brief Introduction to Machine Learning

Definition

- Definition is challenging, given the multidisciplinary character of the subject

Tom M. Mitchell (1997):

> *"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."*

An exhaustive definition of machine learning may as well contain, among others, the following key phrases:

- sub-field of artificial intelligence
- computational learning theory
- statistical inference
- study of algorithms and statistical models implemented in a computer
- generalize from experience
- pattern recognition

Main learning paradigms

1. Supervised Learning

data $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$

Input: $\dim(\mathbf{X}) = N \times D$

Output: $\dim(\mathbf{Y}) = N \times K$

N data points

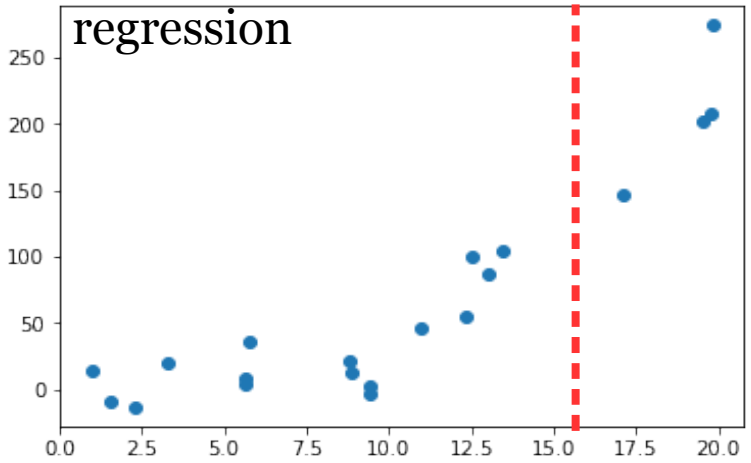
D input dim.

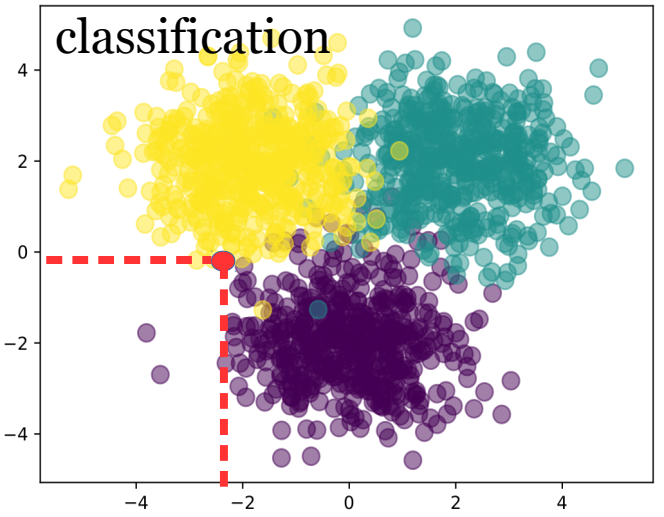
K output classes

Tasks:

1) Statistical inference of the distribution of data

2) Predict the output for a new point $\vec{x}_{\text{new}}$





Statistical inference of the distribution of data

- Frequentist approach (Max. Likelihood Estimator); overfitting control,

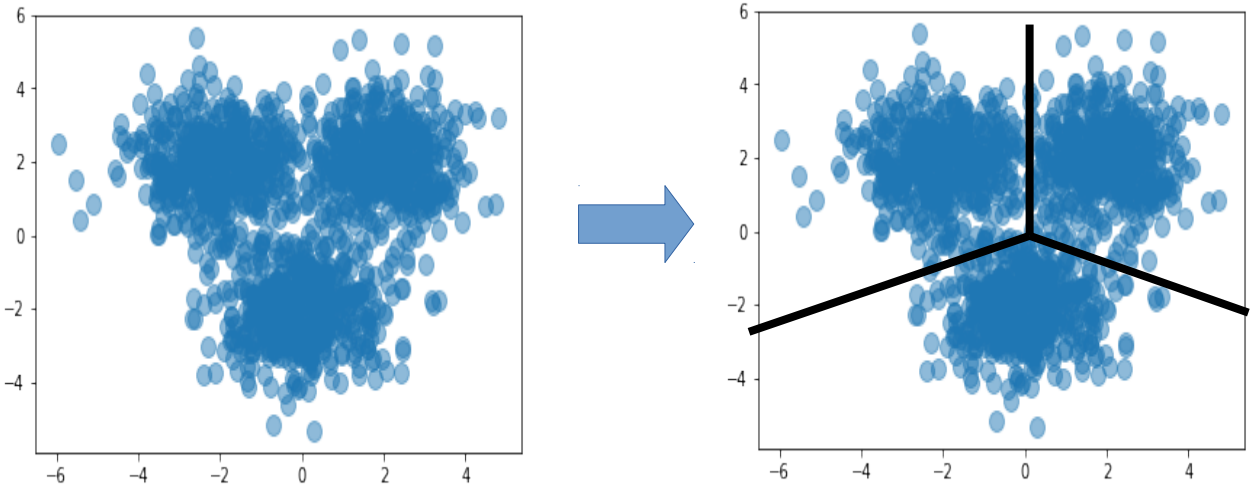
- Bayesian approach (Max. A Posteriori); approximations for often intractable integrals

Main learning paradigms

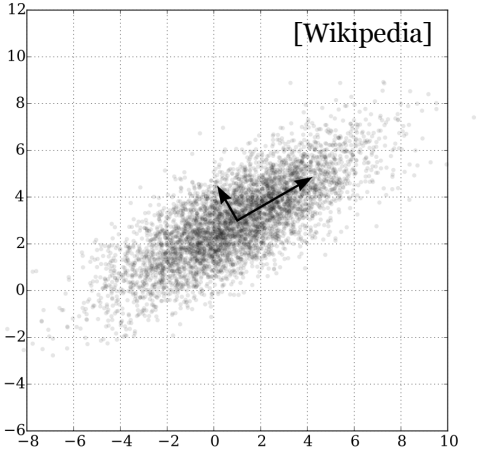
2. Unsupervised Learning

Data contains only inputs, not outputs (or labels)

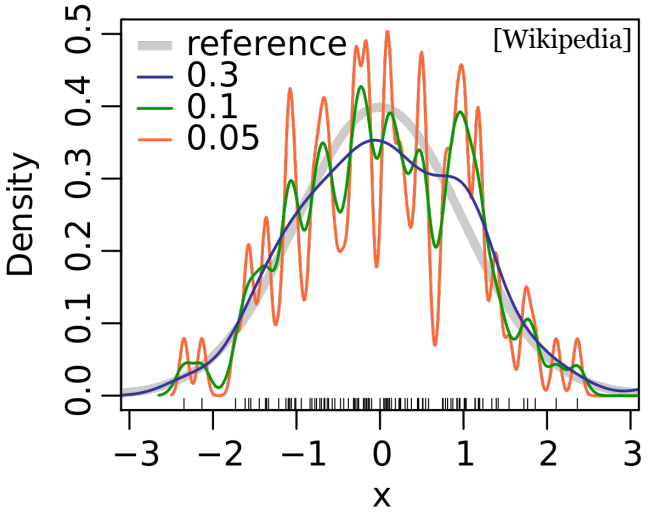
a) unlabelled data can clusterize, thus revealing structure in it



b) data can feature correlations, useful for dimensionality reduction or feature extraction



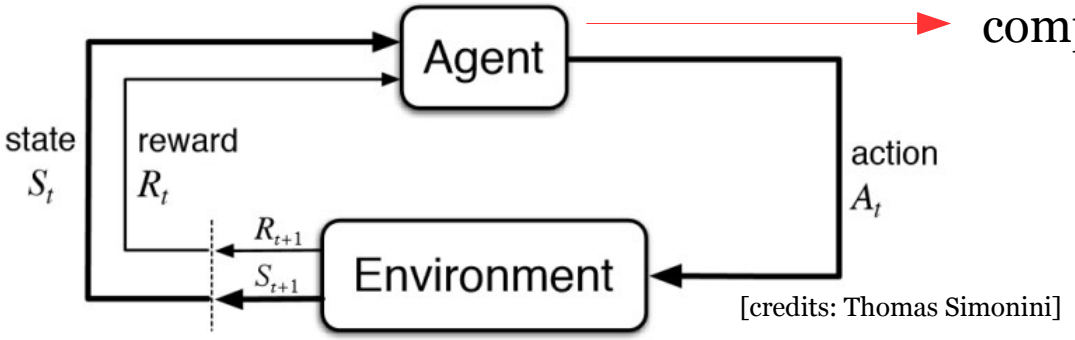
c) Probability density estimation



Main learning paradigms

3. Reinforcement Learning

Similar in spirit to the way humans learn from their environment



computer program

[credits: Thomas Simonini]

Typically used for learning to play games (hard to find an example in physics/maths)
-however see Phys. Rev. X 8, 031084 (2018)

Environment: Physical world in which the program operates
(e.g. a state in a tic-tac-toe game)

Policy: Method to map the program's state to action

Action: done by the program to move to a new state

Reward: Feedback from the environment

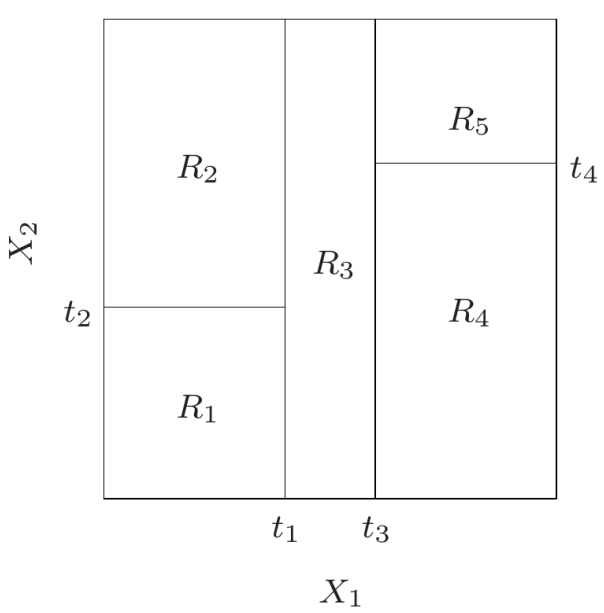
Models

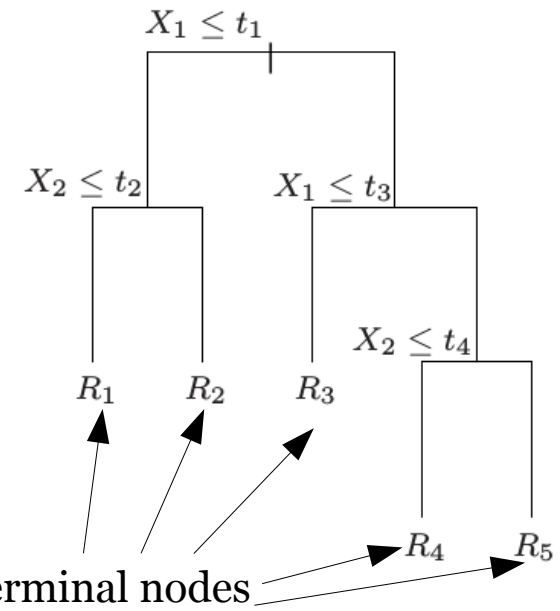
Decision trees

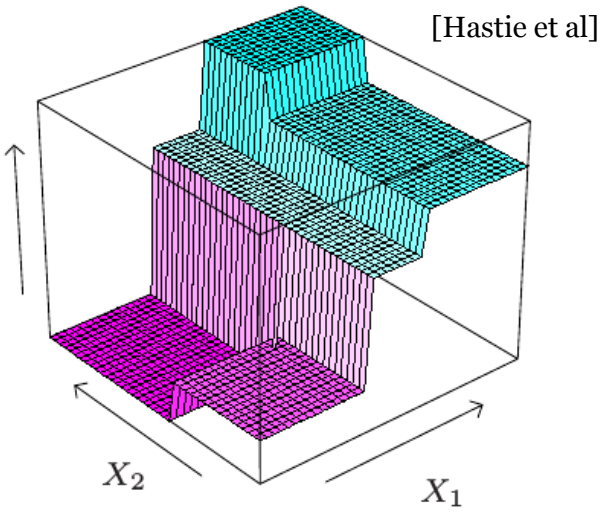
Decision Trees

A *greedy algorithm:*
At each step, choose the optimal variable (and its value) to make the split

Example for regression:







· Tree grown until some criterion is found
(e.g. min. node size, or min. classification error)

· In classification, several measures of node 'impurity'
(cross-entropy, Gini index, ...)

· Many important improvements (random forest, boosting, ...)

Random Forests & Boosting Trees

Combination of trees to avoid overfitting and improve the prediction power

Random Forests

- Create B bootstrap synthetic data from replacement (Bagging)
- Grow trees of all these B data and make prediction based on majority vote
- When growing a tree, select a random sample of m < p input variables at each step, to avoid having very similar trees

Paralellizable, easier to tune ☑ sometimes biased for categorical variables ☒

Boosting Methods

Combine "weak classifiers" (small trees) to produce a strong committee

- Start assigning equal weights to all the training observations

-`For m in range(M):`

a) Train a 1-level decision tree ("decision stump")
b) Assign weight to the tree from error rate (larger weights for smaller error)
c) Misclassified train. observations receive more weight in next iteration

-Combine all trees

Very good predictive power ☑ Sequential (slow), harder to tune ☒

Models

Neural Networks

Neural networks

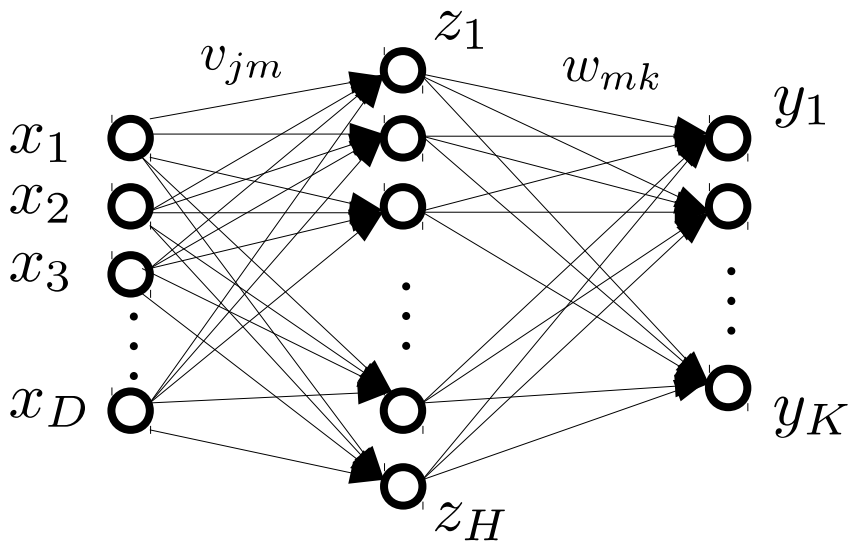
McCulloch and Pitts (1943) first attempts for mathematical modelling of neuronal activity

$$f(\vec{x}, \vec{w}) = g(\vec{w}^T \vec{\phi}(\vec{x})) \Rightarrow \text{Linear models} \begin{cases} g(a) = a & : \text{ regression} \\ g(a) \text{ valid activation} & : \text{ classification} \end{cases}$$

Neural nets follow the same functional form, with:

$$\vec{\phi}(\vec{x}) = h(\vec{v}^T \vec{x} + b)$$

$h(a)$: activation functions of hidden layers



Hidden layer

$$z_m = h(\sum_{j=1}^{D} v_{jm} x_j + b_m)$$

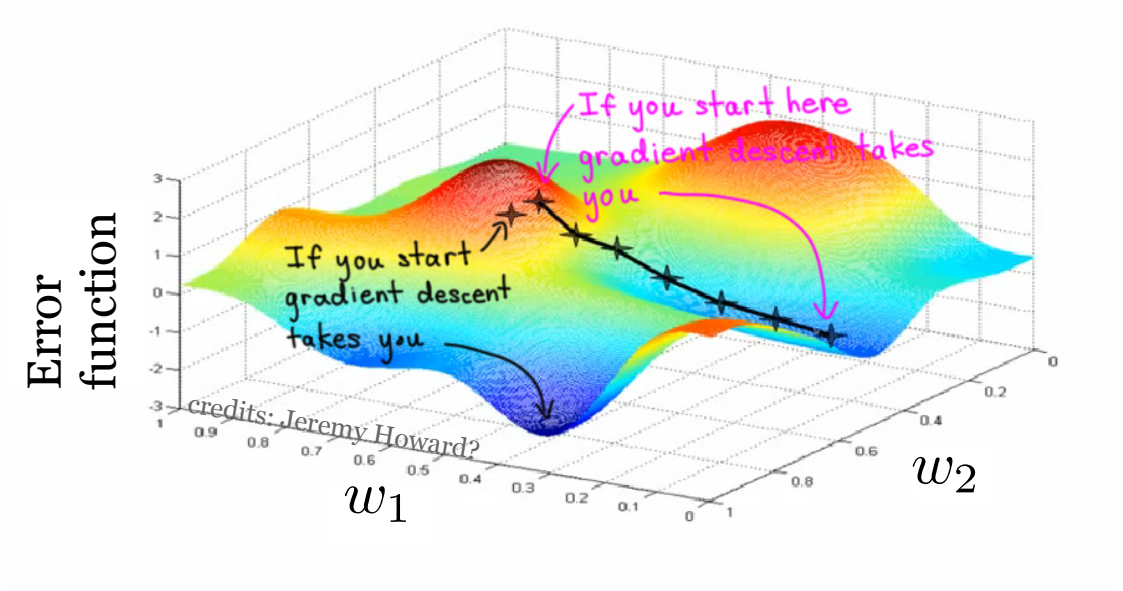
Output layer

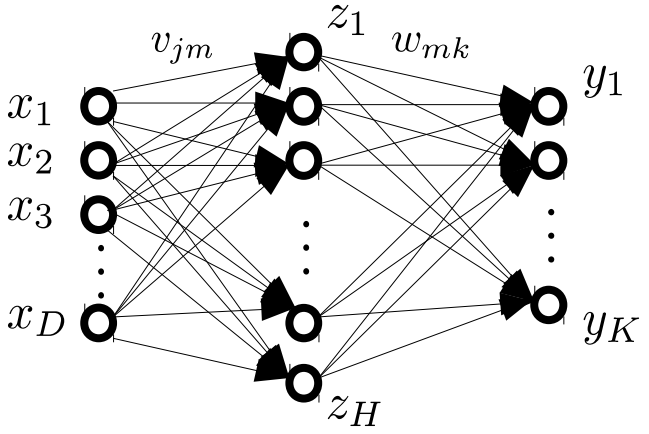
$$y_k = g(\sum_{m=1}^{H} w_{mk} z_m + c_k)$$

1-hidden layer, fully connected, feed-forward neural network

Neural networks. Training

Gradient descent with Backpropagation





weight update for epoch (iteration) $\tau + 1$

$$\vec{w}^{(\tau+1)} = \vec{w}^{(\tau)} - \eta^{(\tau)} \nabla E(\vec{w}^{(\tau)})$$

$\eta^{(\tau)}$: learning rate at epoch τ

$E(\vec{w})$: error function

$$\frac{\partial E}{\partial w_{mk}} = \sum_{i=1}^{N} \frac{\partial E_i}{\partial y_k} \frac{\partial y_k}{\partial w_{mk}} \qquad \frac{\partial E}{\partial v_{jm}} = \sum_{i=1}^{N} \frac{\partial E_i}{\partial y_k} \frac{\partial y_k}{\partial z_m} \frac{\partial z_m}{\partial v_{jm}}$$

chain rule

Neural networks

Deep Learning

1 hidden layer: $Y = g\big(h_1(XV_1 + B_1)W + C\big)$

2 hidden layers: $Y = g\Big(h_2\big(h_1(XV_1 + B_1)V_2 + B_2\big)W + C\Big)$

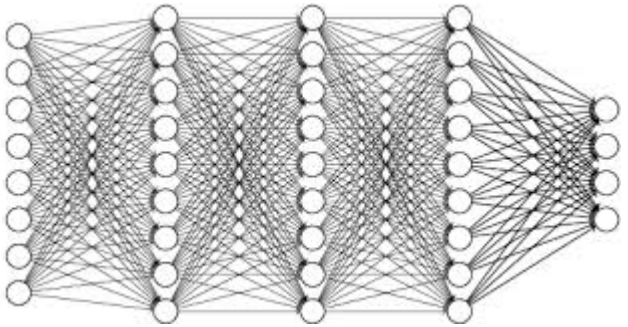
$\vdots$

L hidden layers: $Y = g\big(h_L(A_L)V + C\big)$

where

$$A_L = h_{L-1}(A_{L-1})V_L + B_L$$
$$A_{L-1} = h_{L-2}(A_{L-2})V_{L-1} + B_{L-1}$$
$$\vdots$$
$$A_2 = h_1(XV_1 + B_1)V_2 + B_2$$
$$A_1 = XV_1 + B_1$$

Fully-connected network



Other (more popular) Architectures:

Convolutional nets
+ O(10) others

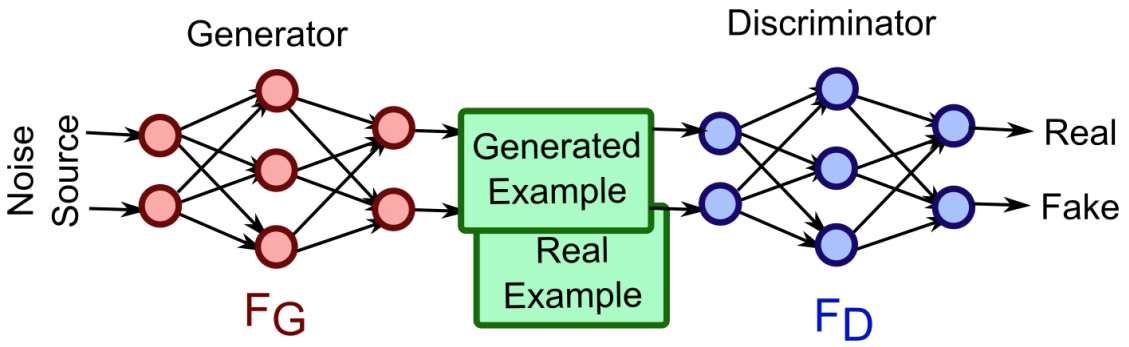
Examples in LHC physics

experimentalist phenomenologist

Example 1: **Detector simulation**

[Pagani, de Oliveira & Nachman, arXiv:1705.02355]

Idea: Accelerate showering simulation with Generative Adversarial Networks



GANs for EM calorimeter simulation,

Accuracy still not as desired (few % discrepancy)

But speed up by $\sim 10^5$

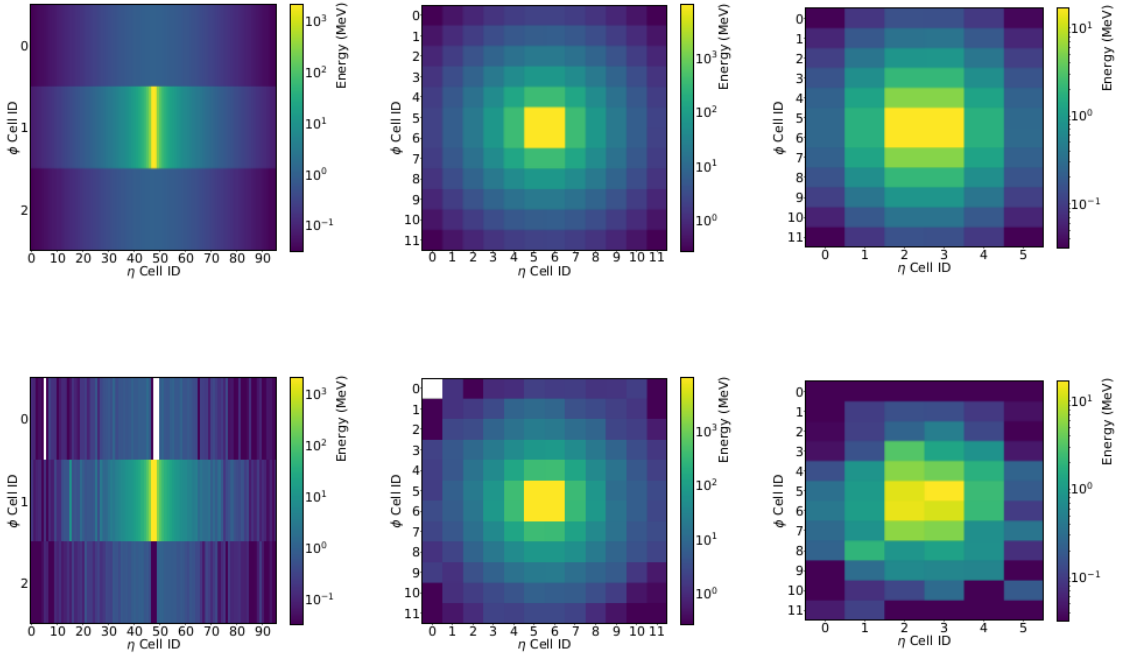


FIG. 1. Average γ GEANT4 shower (top), and average γ CALOGAN shower (bottom), with progressive calorimeter depth (left to right).

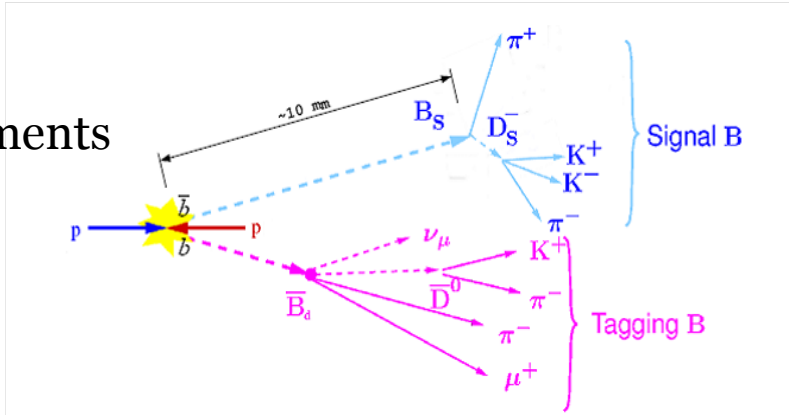
Example 2: **Triggering**

[Gligorov & Williams, arXiv: 1210.6861]

Idea: implement a high-level trigger using a variant of Boosted Decision Trees, then apply it to LHCb concerning B-meson decays

used nowadays by the LHCb collaboration itself

high-level trigger: using full-event info
vs.
first-level (L0) trigger: using local raw detector measurements



Input variables: 1) impact parameter,
2) vertex's sum over pT

Table 1. Performance on the toy model data of a cut-based, BDT-based and BBDT-based HLT algorithm. $\varepsilon_{n-\text{body}}$ are the efficiencies on the n-body signals. The instability is the increase in the rate under the imperfect online conditions (see text for details).

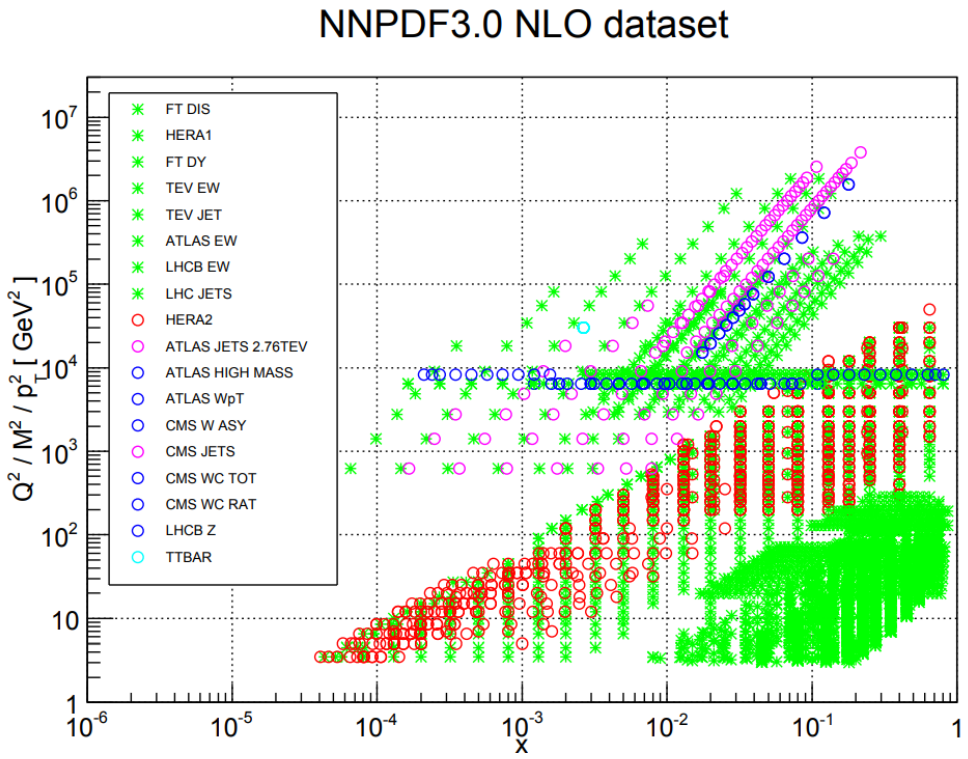
type	$\varepsilon_{4-\text{body}}$	$\varepsilon_{5-\text{body}}$	instability
cuts	63%	55%	9%
BDT	77%	68%	55%
BBDT	74%	69%	10%

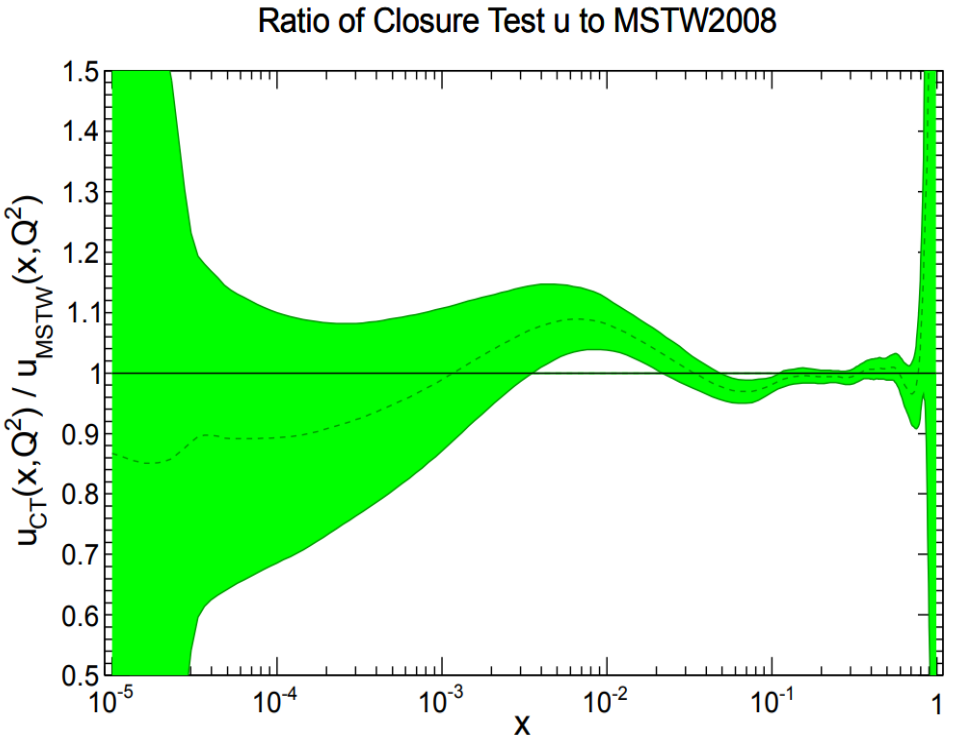
Example 3: **Parton Distribution Functions**

[The NNPDF collaboration, arXiv: 1811.05858]

Idea: use a 1HL neural network as fitting tool for extracting the PDFs from data

$$f_i(x, Q_0) = A_i \hat{f}_i(x, Q_0); \quad \hat{f}_i(x, Q_0) = x^{-\alpha_i}(1-x)^{\beta_i}\,\mathrm{NN}_i(x)$$





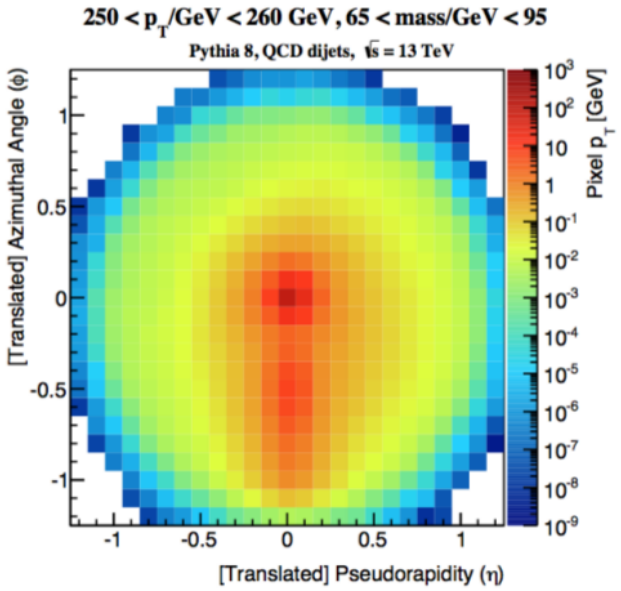
Example 4: **Object reconstruction**

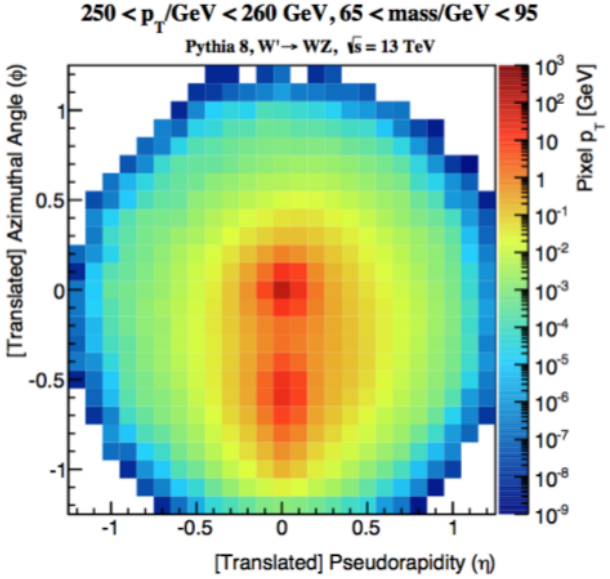
[Cogan, Kagan, Strauss & Schwartzman, arXiv: 1407.5675]

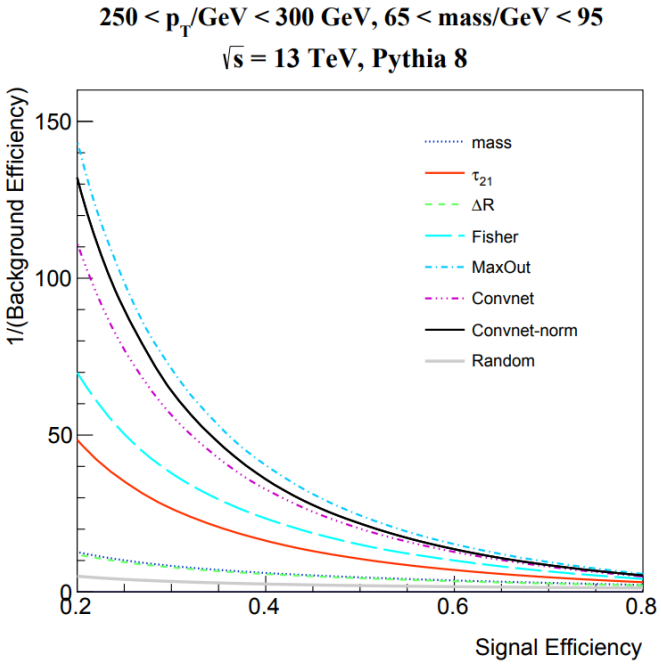
Idea: construct jet taggers by interpreting calorimeter towers as pixels of an image, Thus using computer vision techniques (here, linear Fisher discriminant)

[de Oliveira et al, JHEP 07:069 (2016)]: deep learning edition (CNNs)

"average images" of QCD vs W-initiated jets *after preprocessing*







Improved performance over single engineered variables (N-subjettines, Delta(R), ...)

Example 5: **Object reconstruction**

[Bollweg, Haussmann, Kasieczka, Luchmann, Plehn & Thompson, arXiv: 1904.10004]

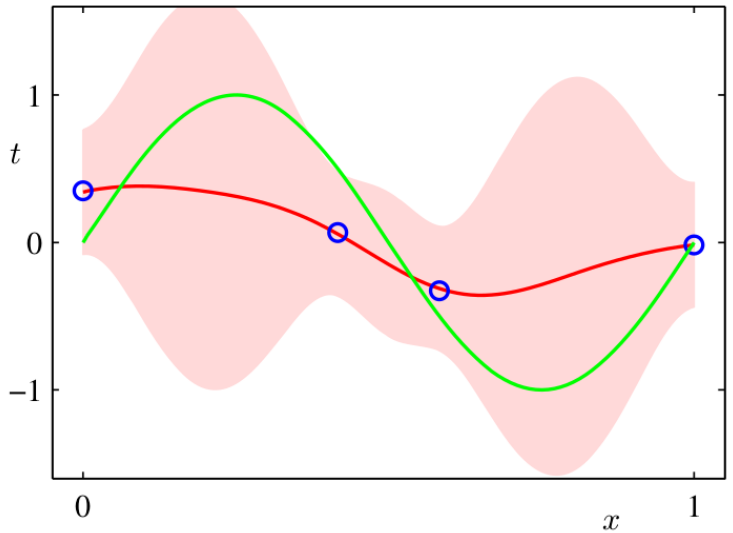
Idea: construct a top tagger using Bayesian learning with deep networks, based on high-level inputs

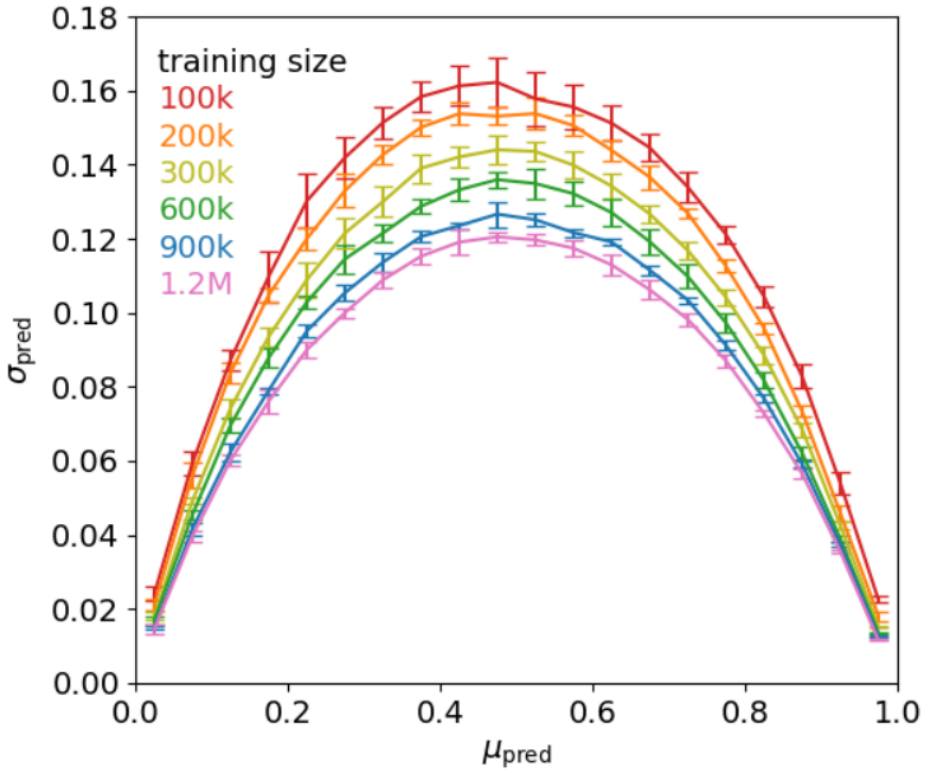
Binary classification: Top jet vs. QCD jet

Bayesian learning

Predictive distribution:

$$p(C_*|\mathbf{x}_*, \mathcal{D}_{\text{train}}) = \int p(C_*|\mathbf{x}_*, \mathbf{w}) \underbrace{p(\mathbf{w}|\mathcal{D}_{\text{train}})}_{\text{posterior}} d\mathbf{w}$$





AUC~0.98 performance,
Similar to frequentist NNs

Example 6: **Monte Carlo Generator tuning**

[Ilten, Williams & Yang, arXiv: 1610.08328]

Idea: Determining Pythia-8 parameters (~20) from data using Bayesian optimization

MC vs. Experimental data

distributions of observables

- metric between 2 distributions (e.g. two-sample chi2, or KS)

$f(\mathbf{x}, \vec{\theta}\,)$ very costly

Bayesian optimization

-treat f as a random function,
-place a prior over it (e.g. GP),
-obtain posterior $\mathcal{P}$ after obs.,
-build "acquisition function" $\alpha(\vec{\theta})$

$$\vec{\theta}_{\text{next}} = \underset{\vec{\theta}}{\text{argmax}}\ \alpha_{\mathcal{P}}(\vec{\theta})$$

Table 3. Tuning results for block 1.

Parameter	Monash Value	Tune Value	Range Considered
alphaSvalue	0.1365	0.1365 ± 0.0002	[0.06, 0.25]
pTmin	0.5	0.49 ± 0.02	[0.1, 2.0]
pTminChgQ	0.5	$1.89^{+0.10}_{-1.79}$	[0.1, 2.0]

Table 4. Tuning results for block 2.

Parameter	Monash Value	Tune Value	Range Considered
sigma	0.335	$0.333^{+0.001}_{-0.002}$	[0, 1]
bLund	0.98	$1.04^{+0.01}_{-0.02}$	[0.2, 2]
aExtraSQuark	0	$0^{+0.07}_{-0}$	[0, 2]
aExtraDiQuark	0.97	$1.48^{+0.15}_{-0.14}$	[0, 2]
rFactC	1.32	1.38 ± 0.06	[0, 2]
rFactB	0.855	0.887 ± 0.015	[0, 2]

[see 1404.5630 for the meaning of these params.]

Example 7: **signal-to-background maximization**

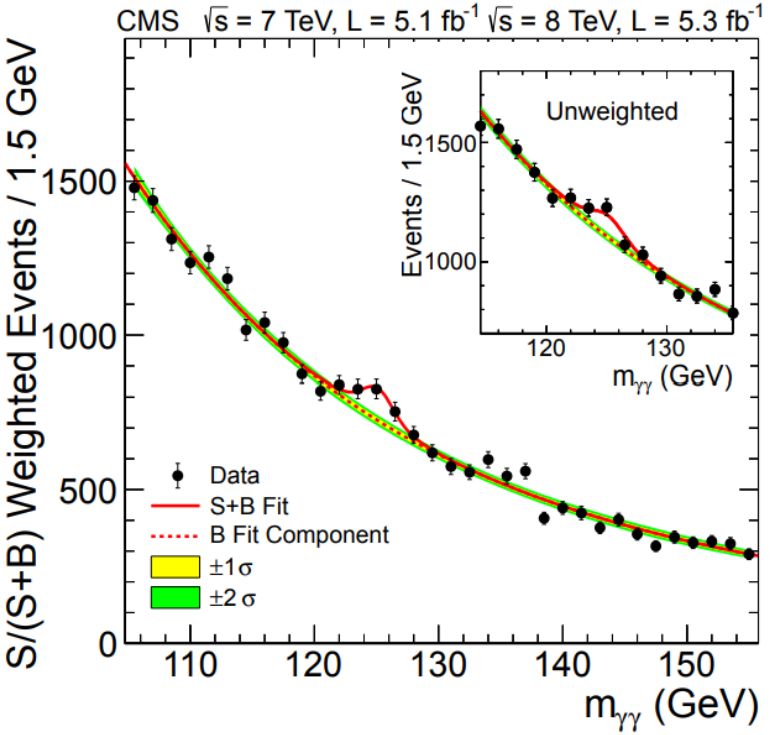
[CMS Collaboration, arXiv: 1207.7235] "discovery of the Higgs"

Idea: Training Boosted Decision Trees (BDT) to improve the signal-to-bckg ratio concerning the Higgs-to-diphoton channel

Input: photon quality, expected mass resolution, location of the vertex, kinematics

Background estimated from predefined fitting functions to data (no MC involved)

	Event categories	SM Higgs boson expected signal ($m_H = 125$ GeV)							Background $m_{\gamma\gamma} = 125$ GeV (events/GeV)
		Events	ggH	VBF	VH	ttH	σ_{eff} (GeV)	FWHM/2.35 (GeV)	
7 TeV, 5.1 fb^{-1}	BDT 0	3.2	61%	17%	19%	3%	1.21	1.14	3.3 ± 0.4
	BDT 1	16.3	88%	6%	6%	–	1.26	1.08	37.5 ± 1.3
	BDT 2	21.5	92%	4%	4%	–	1.59	1.32	74.8 ± 1.9
	BDT 3	32.8	92%	4%	4%	–	2.47	2.07	193.6 ± 3.0
	Dijet tag	2.9	27%	72%	1%	–	1.73	1.37	1.7 ± 0.2
8 TeV, 5.3 fb^{-1}	BDT 0	6.1	68%	12%	16%	4%	1.38	1.23	7.4 ± 0.6
	BDT 1	21.0	87%	6%	6%	1%	1.53	1.31	54.7 ± 1.5
	BDT 2	30.2	92%	4%	4%	–	1.94	1.55	115.2 ± 2.3
	BDT 3	40.0	92%	4%	4%	–	2.86	2.35	256.5 ± 3.4
	Dijet tight	2.6	23%	77%	–	–	2.06	1.57	1.3 ± 0.2
	Dijet loose	3.0	53%	45%	2%	–	1.95	1.48	3.7 ± 0.4



Example 8: **Likelihood-free inference**

[Cranmer, Pavez & Louppe, arXiv: 1506.02169]

Idea: Approximating Likelihood Ratios with ML classifiers

$$\lambda(\mathcal{D};\theta_0,\theta_1) = \prod_{\mathbf{x}\in\mathcal{D}} \frac{p(\mathbf{x}|\theta_1)}{p(\mathbf{x}|\theta_0)} \underset{\text{th.}}{=} \prod_{\mathbf{x}\in\mathcal{D}} \frac{p(u=s(\mathbf{x})|\theta_1)}{p(u=s(\mathbf{x})|\theta_0)}$$

$s(\mathbf{x})$ monotonic with $p(\mathbf{x}|\theta_1)/p(\mathbf{x}|\theta_0)$

- The optimal decision function of a ML classifier behaves like $s(\mathbf{x})$

- Here $p(u=s(\mathbf{x})|\theta)$ much easier density to estimate than $p(\mathbf{x}|\theta)$
(room for unsupervised learning)

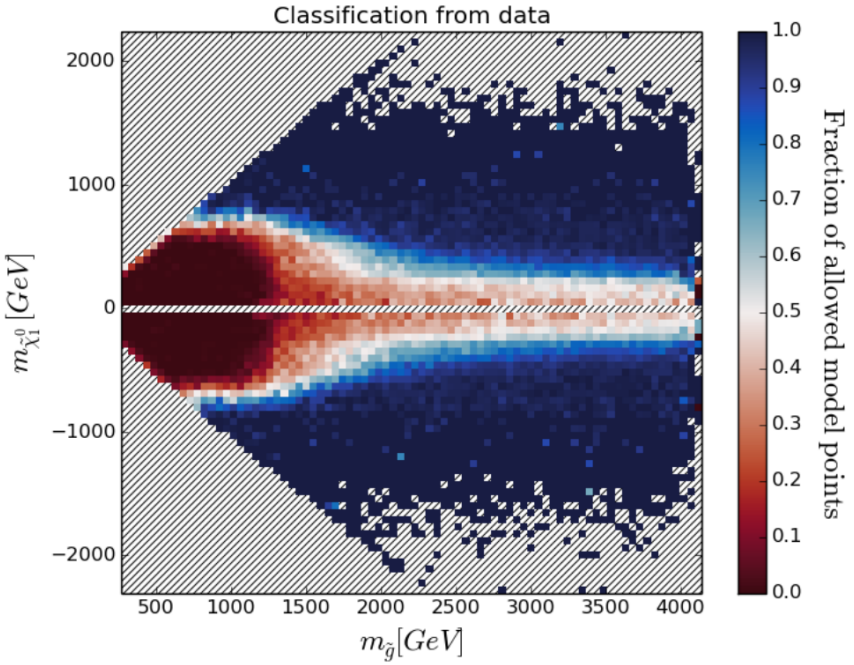
AFAIK, no concrete example for application to LHC physics (?!)
(however see [1810.09930] for a variant of this idea
in the context of dark matter Direct Detection)

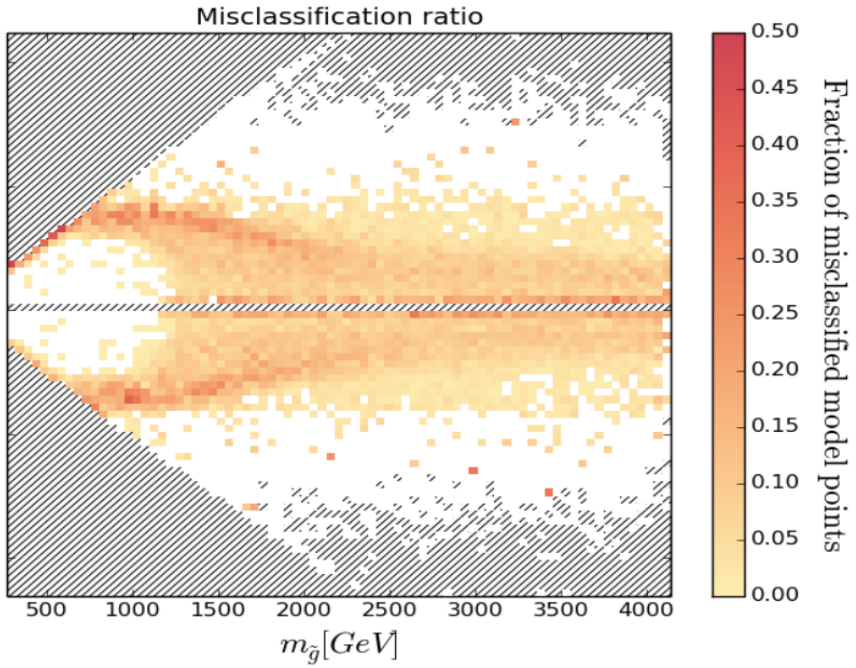
Example 9: **Parameter space interpolation**

[Caron, Kim, Rolbiecki, Ruiz-de-Austri & Stienen, arXiv: 1605.02797]

Idea: Train a classifier to identify points in complicated models which are excluded/allowed by a set of LHC analyses.

(Random forest with ~20 experimental searches applied to the pMSSM-19)





classification rate $\gtrsim 93\%$

Watch-out for absent features in training set!

Conclusions

- Machine Learning is a quite established technology in Particle Physics (actually even present at LEP times)

- Decision Trees and Neural Networks are by far the most popular ML models used for LHC physics

- Applications ranges from Detector Simulation to Parameter scans of complex BSM theories.

- Bayesian Learning increasing in popularity nowadays, many benefits, especially for predictions in physics

- Most of the analyses are close to being “study cases”,
so large room for improvement!

Thank you

Back-up

Important to note :

- A “model” in ML refers, a priori, to a specification for the distribution function

- *If the distribution is assumed* (not learned), then a ML model refers, commonly, to the **expected value** of the output variable (as we do e.g. in physics)

e.g. assuming a Gaussian distribution

$$y(x) \sim \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{[y(x) - \mu(x, \vec{w})]^2}{2\sigma^2}\right\}$$

determined by a given model, (e.g. the Standard Model of particle physics) respecting the laws of physics

$\mu(x, \vec{w})$

determined by a given ML model: (e.g. a neural network) respecting the laws of mathematics

Statistical inference of the distribution of data

$$\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\} \qquad \mathbf{X}: \text{input} \quad \mathbf{Y}: \text{output}$$

Two different approaches

1. Frequentist:

object of interest is Likelihood

$$\mathcal{L}_h(\vec{w}_h) := p(\mathcal{D}|\vec{w}_h, \mathcal{M}_h)$$

functional form *typically* fixed ad hoc

- catalog of models $\{\mathcal{M}_h\}_{h=1}^{M}$
- $\vec{w}_h$: parameters of the model $\mathcal{M}_h$

usual procedure is Maximum Likelihood Estimation

$$\vec{w}_h^{\text{MLE}} = \text{argmax}\ \mathcal{L}_h(\vec{w}_h)$$

in ML literature,
often we minimize the
"Error (or cost) function"

$$\mathcal{C} = -2 \ln \mathcal{L}$$

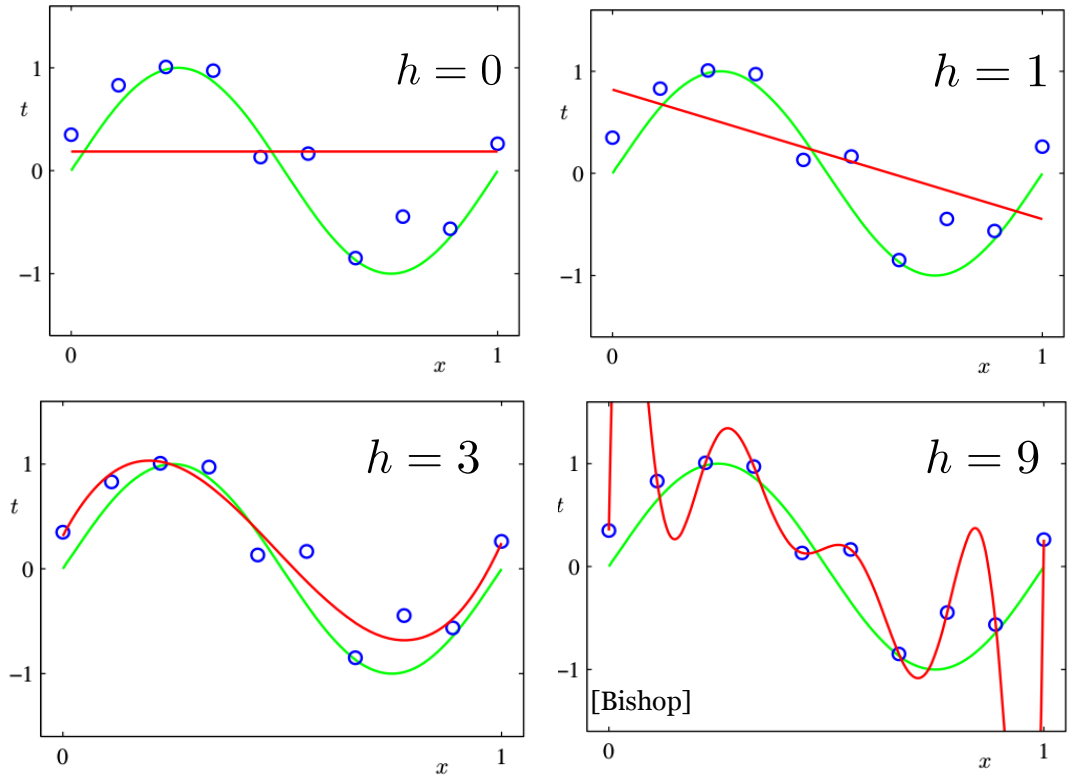
How do we choose which model is the best?

Statistical inference of the distribution of data. Overfitting

- Suppose data points coming from given function plus noise
- e.g. *expected value* of output modelled by

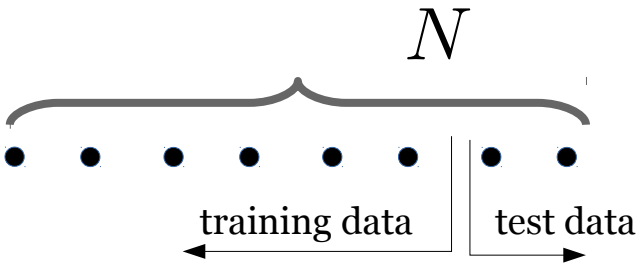
$$f(x, \vec{w}) = \sum_{h=0}^{M} w_h x^h$$

h: degree of the polynomial



$$\mathcal{L}_1 < \mathcal{L}_2 < \ldots < \mathcal{L}_M$$

- A popular procedure to avoid overfitting:



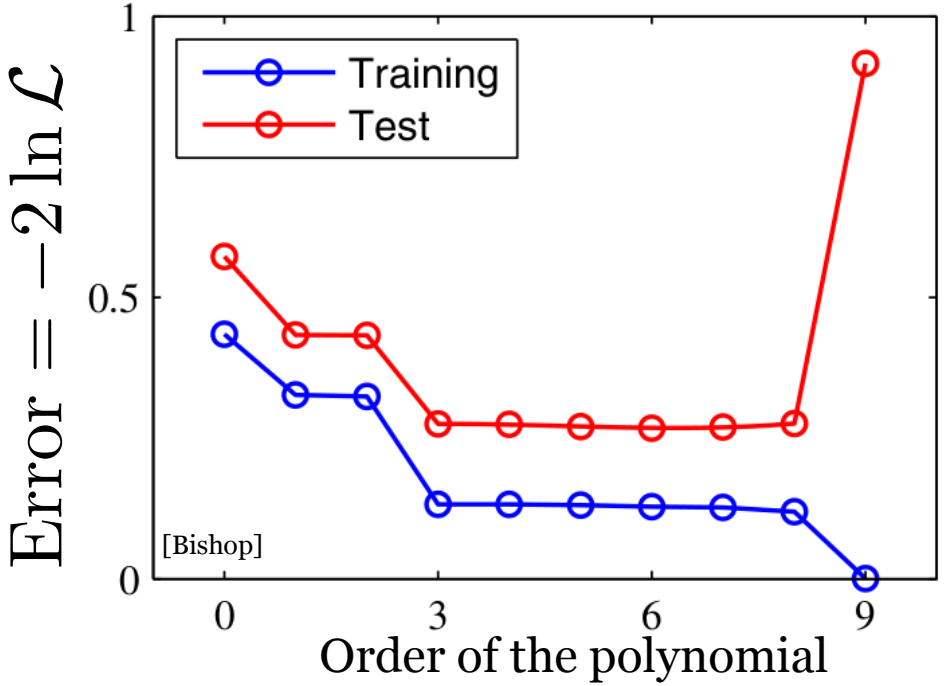
1- For each model, make the fit using only training data

$$\vec{w}_h^{\text{MLE}} = \text{argmax } \mathcal{L}_h^{\text{train}}$$

2- Choose the model that maximizes the Likelihood of the test data

$$\vec{w}^* = \text{argmax } \mathcal{L}^{\text{test}}(\vec{w}_h^{\text{MLE}})$$

Statistical inference of the distribution of data. Overfitting



Common form of error function: mean squared error (MSE):

$$E = \frac{1}{N_s} \sum_{i=1}^{N_s} (y_i - f(x_i, \vec{w}))^2$$

N_s : dim. of sample (train. or test)

Generic feature:

- Training error always goes down as the model complexity is increased
- Test error starts increasing when data is being overfitted

- What if data is scarce?

Training-test splitting may be an expensive luxury!

Ideally to use as much information as possible to build the best model

Statistical inference of the distribution of data

2. Bayesian approach:

Rules of probability:

Bayes theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$\underbrace{\mathcal{P}_h(\vec{w}_h|\mathcal{D},\mathcal{M}_h)}_{\text{posterior}} = \frac{\overbrace{p(\mathcal{D}|\vec{w}_h,\mathcal{M}_h)}^{\text{likelihood}}\overbrace{p(\vec{w}_h|\mathcal{M}_h)}^{\text{prior}}}{\underbrace{p(\mathcal{D}|\mathcal{M}_h)}_{\text{marginal likelihood}}}$$

prior guess about distrib. of $\vec{w}_h$ before seeing data

posterior distrib. of $\vec{w}_h$ after seeing data

marginal likelihood (or "evidence")

$\vec{w}_h^{\mathrm{MLE}}$ → $\vec{w}_h^{\mathrm{MAP}}$

"maximum a posteriori" estimate of the parameters

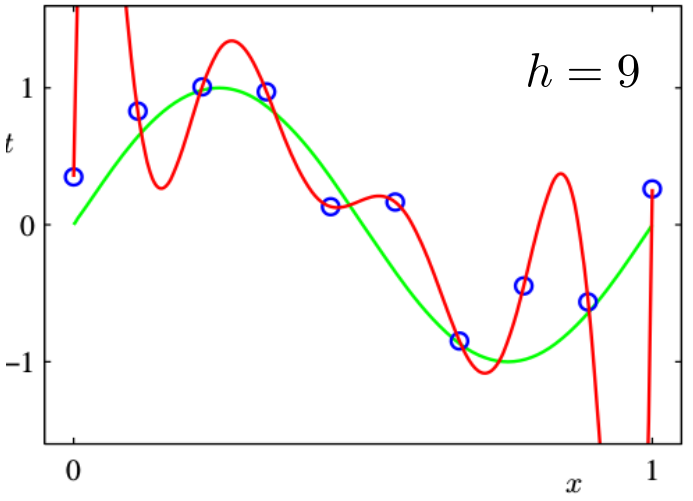
(don't need the evidence for that)

Statistical inference of the distribution of data

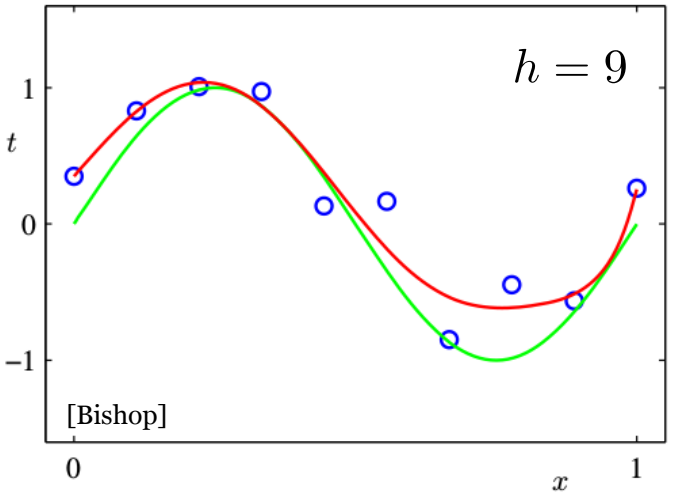
- What is the effect of the prior on the fit?

$h = 9$	no, or flat prior	standard normal prior
$w_0^\star$	0.35	0.35
$w_1^\star$	232.37	4.74
$w_2^\star$	-5321.83	-0.77
$w_3^\star$	48568.31	-31.97
$w_4^\star$	-231639.30	-3.89
$w_5^\star$	640042.26	55.28
$w_6^\star$	-1061800.52	41.32
$w_7^\star$	1042400.18	-45.95
$w_8^\star$	-557682.99	-91.53
$w_9^\star$	125201.43	72.68

no prior



std. normal prior



If no prior:
high fine tune among coefficients
to accommodate data noise

With the above prior:
Penalize large values, reduce overfitting

Statistical inference of the distribution of data

- Bayesian model comparison

probability of model *h* given the data

$$\mathcal{P}(\mathcal{M}_h|\mathcal{D}) \propto p(\mathcal{D}|\mathcal{M}_h)p(\mathcal{M}_h)$$

if model priors are similar $\forall h$

Bayes Factor favours one model over another $\dfrac{p(\mathcal{D}|\mathcal{M}_i)}{p(\mathcal{D}|\mathcal{M}_j)}$

Jeffreys scale ('61)

Bayes factor	"Strength of evidence"
< 1	Negative
1 – 3	Barely worth mentioning
3 – 10	Substantial
10 – 30	Strong
30 – 100	Very strong
> 100	Decisive

Caution: it may happen that for one $\mathcal{D}$, BF gets larger for the wrong model

It can be shown that
average BF over datasets will always favour the true model

Statistical inference of the distribution of data

Predictions for a new input

1. Frequentist approach

given $\vec{w}^*, \vec{x}_{\text{new}}$

expected output modelled by $f(\vec{x}, \vec{w})$

prediction will be $f(\vec{x}_{\text{new}}, \vec{w}^*)$

No uncertainties

2. Bayesian approach

predictive distribution

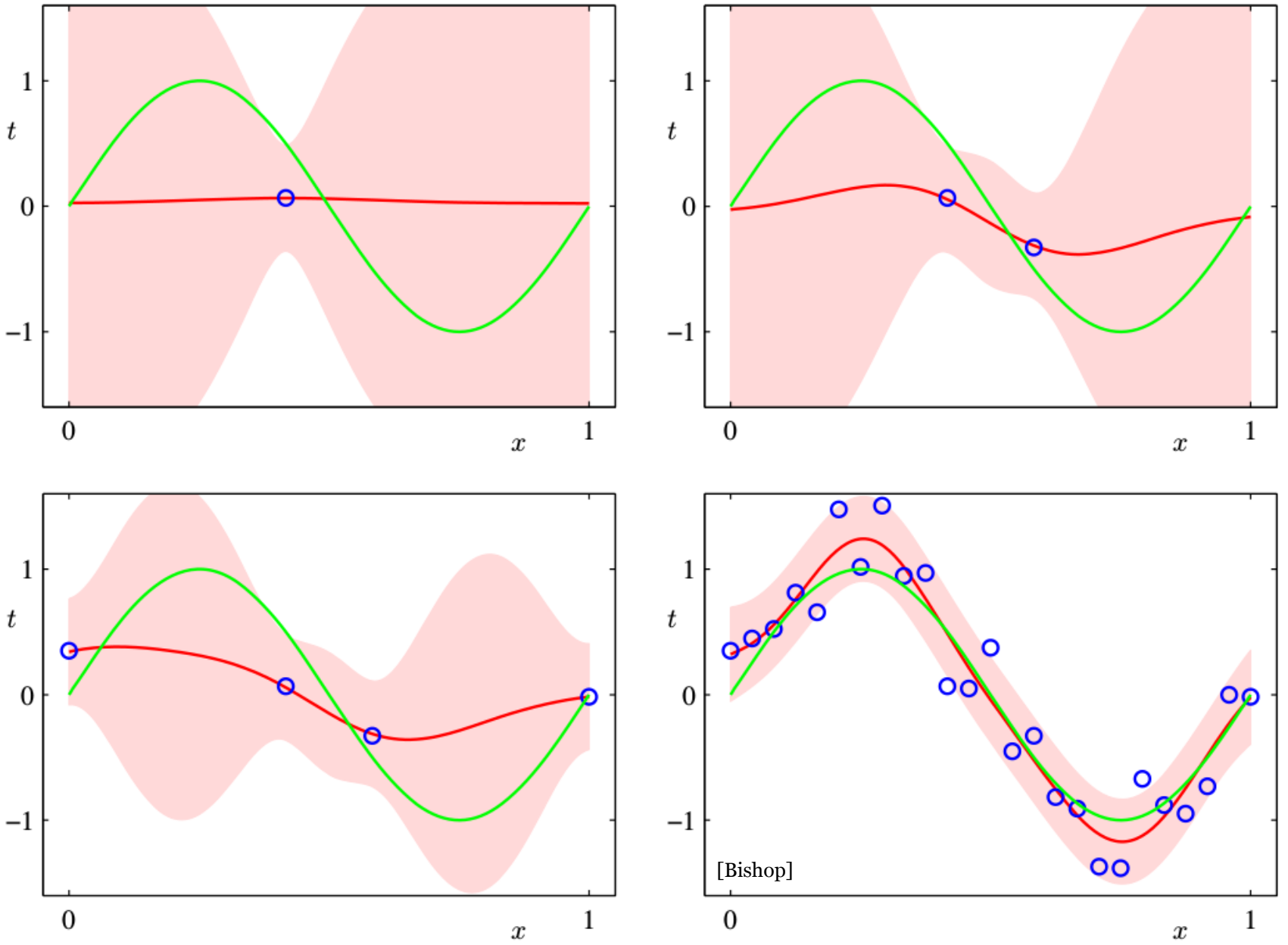
$$p(y_{\text{new}}|\vec{x}_{\text{new}}, \mathcal{D}) = \int d\vec{w}\ \underbrace{p(y_{\text{new}}|\vec{x}_{\text{new}}, \vec{w})}_{\text{Likelihood of new datum}} \mathcal{P}(\vec{w}|\mathcal{D})$$

prediction will be $f(\vec{x}_{\text{new}}, \vec{w}^{\text{MAP}})$

and uncertainties come automatically!

Statistical inference of the distribution of data

Predictive distribution



Classification

- Classes
  - · yellow/blue
  - · signal/background (HEP)
  - · dogs/cats/monkeys/humans

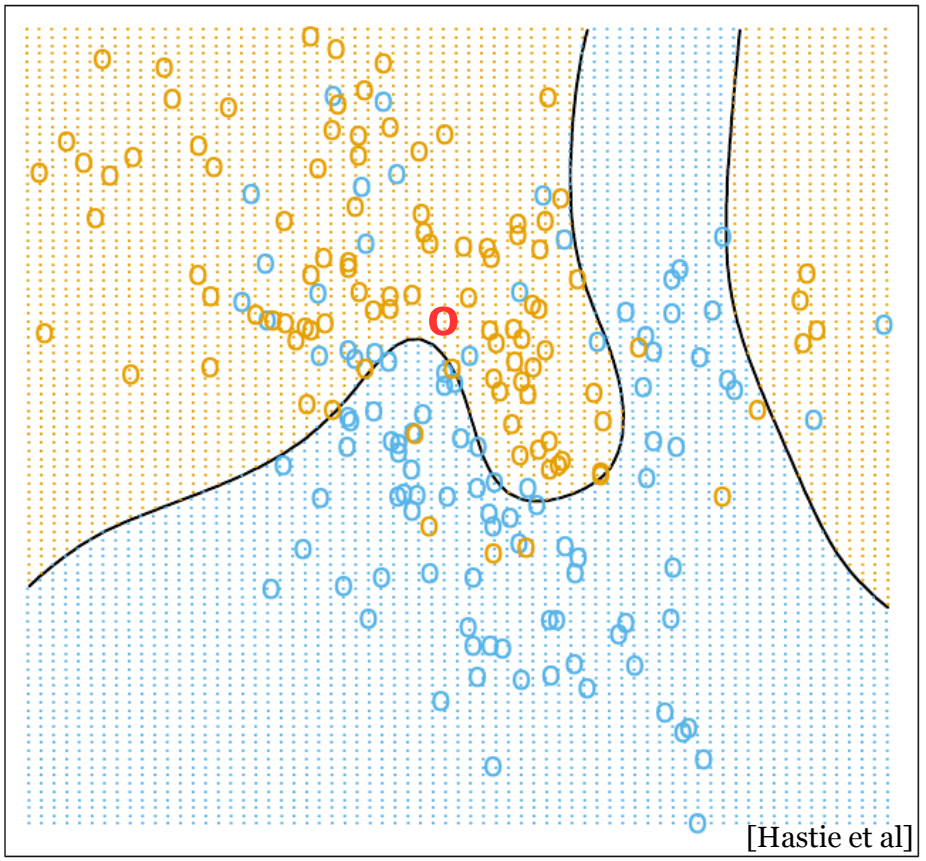
described by a **categorical variable**

$$t = \{1, 2, ..., K\} \quad \text{(output)}$$

following some discrete distribution

$$t \sim P(t|p_1, ..., p_K)$$

true (unknown) probabilities



- these are the objects we aim to predict

$$p(\mathcal{C}_k|\vec{x}_{\text{new}})$$

either in frequentist or Bayesian approach

Common form of error function:
cross-entropy

$$E = -\sum_{i=1}^{N}\sum_{k=1}^{K} t_{ik} \ln p(C_k|\vec{x}_i, \vec{w})$$

Curse of dimensionality

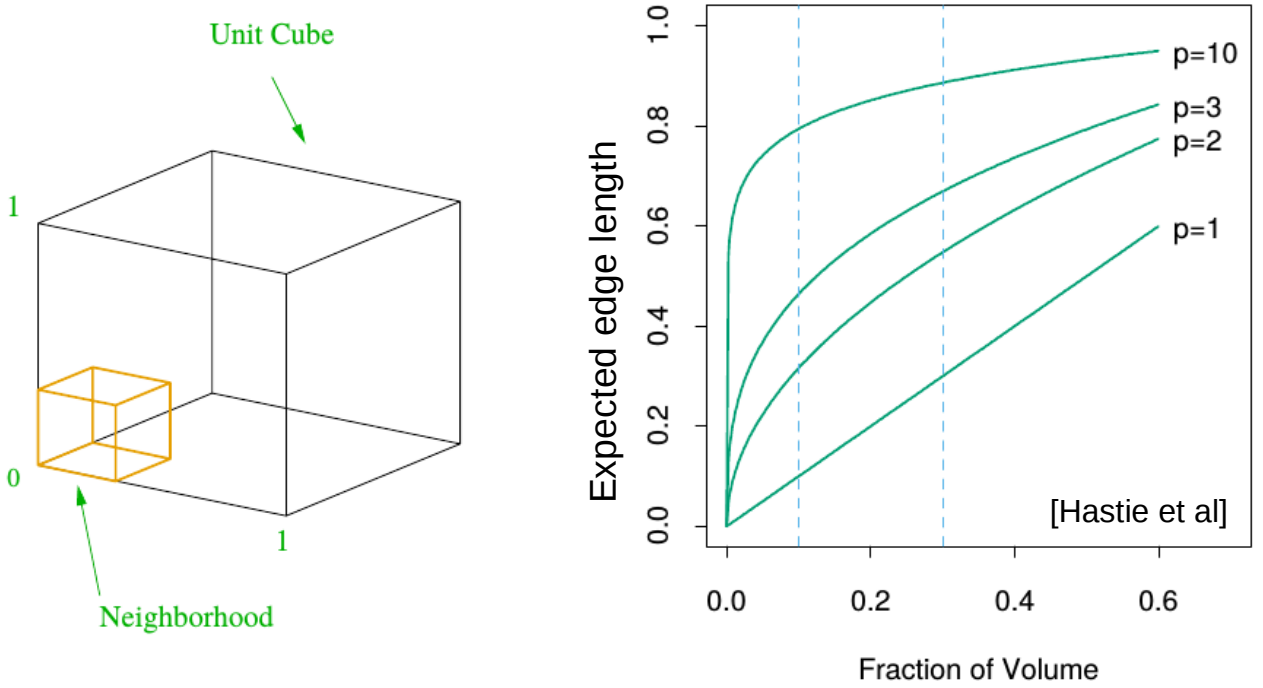


FIGURE 2.6. *The curse of dimensionality is well illustrated by a subcubical neighborhood for uniform data in a unit cube. The figure on the right shows the side-length of the subcube needed to capture a fraction r of the volume of the data, for different dimensions p. In ten dimensions we need to cover 80% of the range of each coordinate to capture 10% of the data.*

$$e_p(r) = r^{1/p}$$

p: dim. of data

Curse of dimensionality

- Polynomial model: D^M

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{i=1}^{D} w_i x_i + \sum_{i=1}^{D}\sum_{j=1}^{D} w_{ij} x_i x_j + \sum_{i=1}^{D}\sum_{j=1}^{D}\sum_{k=1}^{D} w_{ijk} x_i x_j x_k$$

- Decision trees:
  Depth of tree goes linear with D, but no. of branches goes exponentially
  With the number of states in each dimension

- 1HL Neural Net

$$Y = g\big(h_1(XV_1 + B_1)W + C\big)$$

no. of params = DxH + H + HxK + K , so linear in D

- Density methods
  For kernel method, as D increases,
  V should be fixed to a large fraction of the total volume