

Statistics & Machine Learning for HEP 3

Harrison B. Prosper
Florida State University

11th CERN | 15 – 28 MARCH 2023

**LATIN-AMERICAN SCHOOL
OF HIGH-ENERGY PHYSICS**

San Esteban, Chile

<https://indico.cern.ch/e/clashep2023>

Topics

- **Lecture 1**
 - Frequentist Analysis (1)
- **Lecture 2**
 - Frequentist Analysis (2)
 - Bayesian Analysis
- **Lectures 3**
 - Machine Learning
 - A Brief History
 - Foundations
- **Lecture 4**
 - Simulation-based Inference with Machine Learning

Jupyter Notebooks

I encourage you to try out the [jupyter notebooks](#) at

<https://github.com/hbprosper/CLASHEP>

Also: <https://github.com/hbprosper/GSW>

Recommendation (for Windows, Linux and OSX)

1. Install miniconda. See instructions at:

<https://docs.conda.io/en/latest/miniconda.html>

2. Create a **miniconda** environment

```
conda create --name clashep
```

3. Activate environment

```
conda activate clashep
```

Jupyter Notebooks

4. Install root, python, numpy, ...

```
conda install -c conda-forge root
```

5. Install pytorch, matplotlib, scikit-learn, ...

```
conda install -c conda-forge pytorch
```

```
conda install -c conda-forge matplotlib
```

```
conda install -c conda-forge scikit-learn
```

```
conda install -c conda-forge pandas
```

```
conda install -c conda-forge sympy
```

```
conda install -c conda-forge imageio
```

```
conda install -c conda-forge jupyter
```

Jupyter Notebooks

6. Install **git** if it is not yet on your system, then download the **CLASHEP** package:

```
conda install -c conda-forge git
mkdir <working directory>
cd <working directory>
git clone https://github.com/hbprosper/CLASHEP
```

7. Open a new terminal window and run the jupyter notebook in that window (in blocking mode):

```
cd <working directory>
jupyter notebook
```

In your browser, navigate to the CLASHEP directory.

Outline

- A Very Incomplete History of ML/AI
- Foundations

A Very Incomplete History of ML/AI

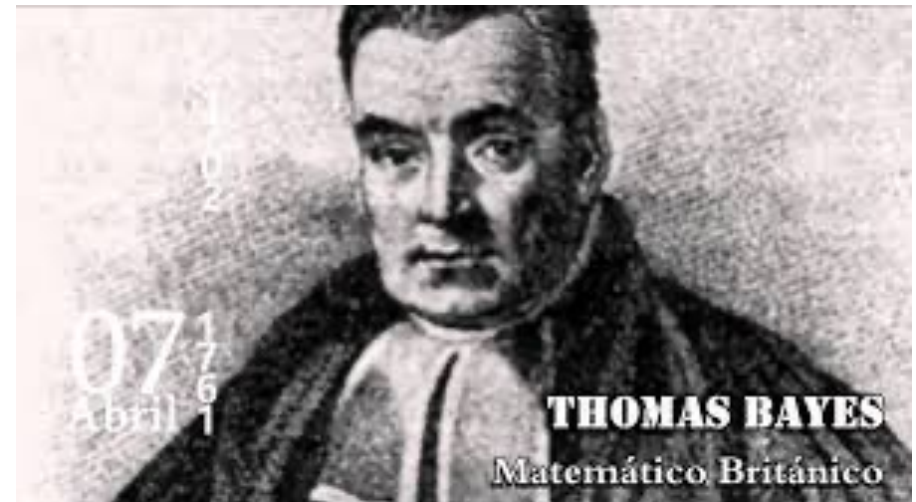
17th century

- Many philosophical ideas about knowledge, reason, and the nature of Man.

18th century

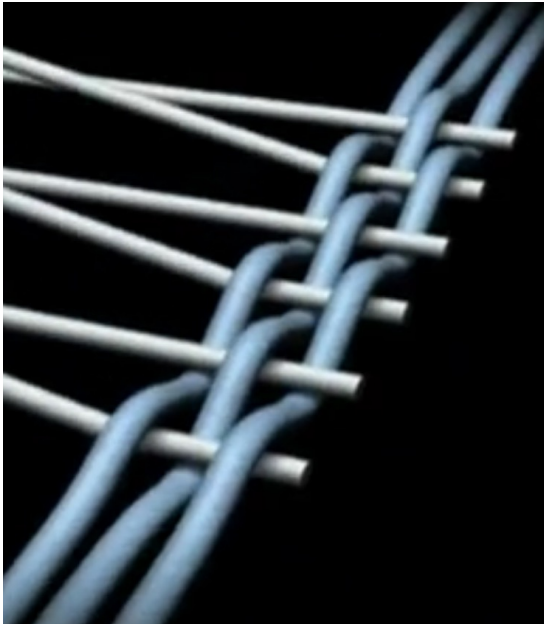
- **1763** – English cleric Thomas Bayes publishes important theorem.

$$P(\mathbf{H}|Data) = \frac{P(D|\mathbf{H})P(\mathbf{H})}{P(Data)}$$

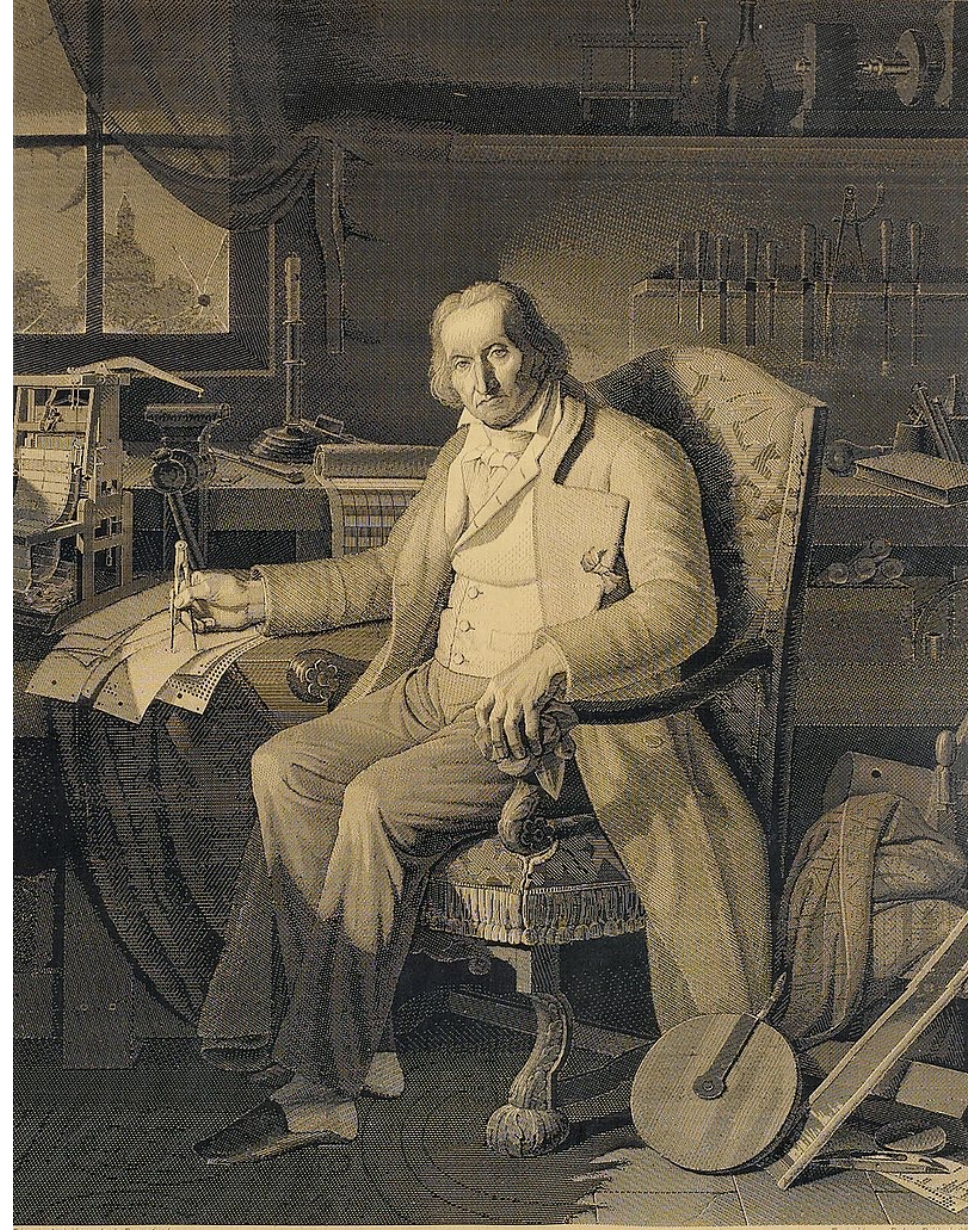


19th century

- 1801 – Joseph-Marie Jacquard invents first programmable machine.



Wikimedia commons



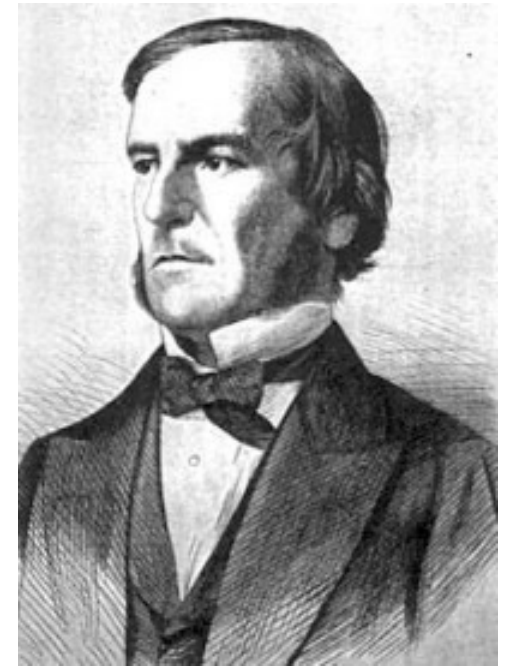
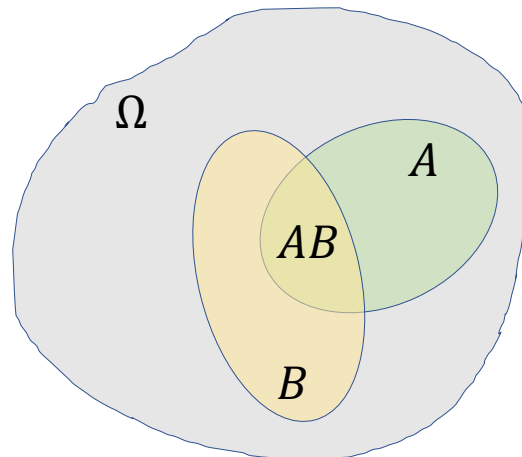
A LA MÉMOIRE DE J. M. JACQUARD.

Né à Lyon le 7 Juillet 1752. Mort le 7 Aout 1834.

A Very Incomplete History of ML/AI

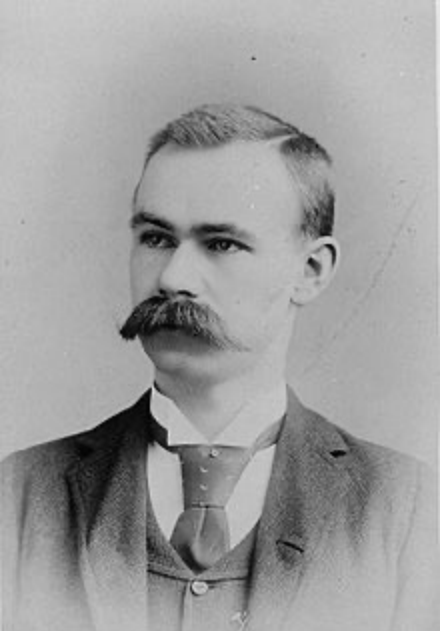
19th century

- 1832 – Charles Babbage designs first programmable calculator.
- 1854 – George Boole invents algebra of logic.



1815 – 1864

1890 US Census



Herman Hollerith
(1860 – 1929)

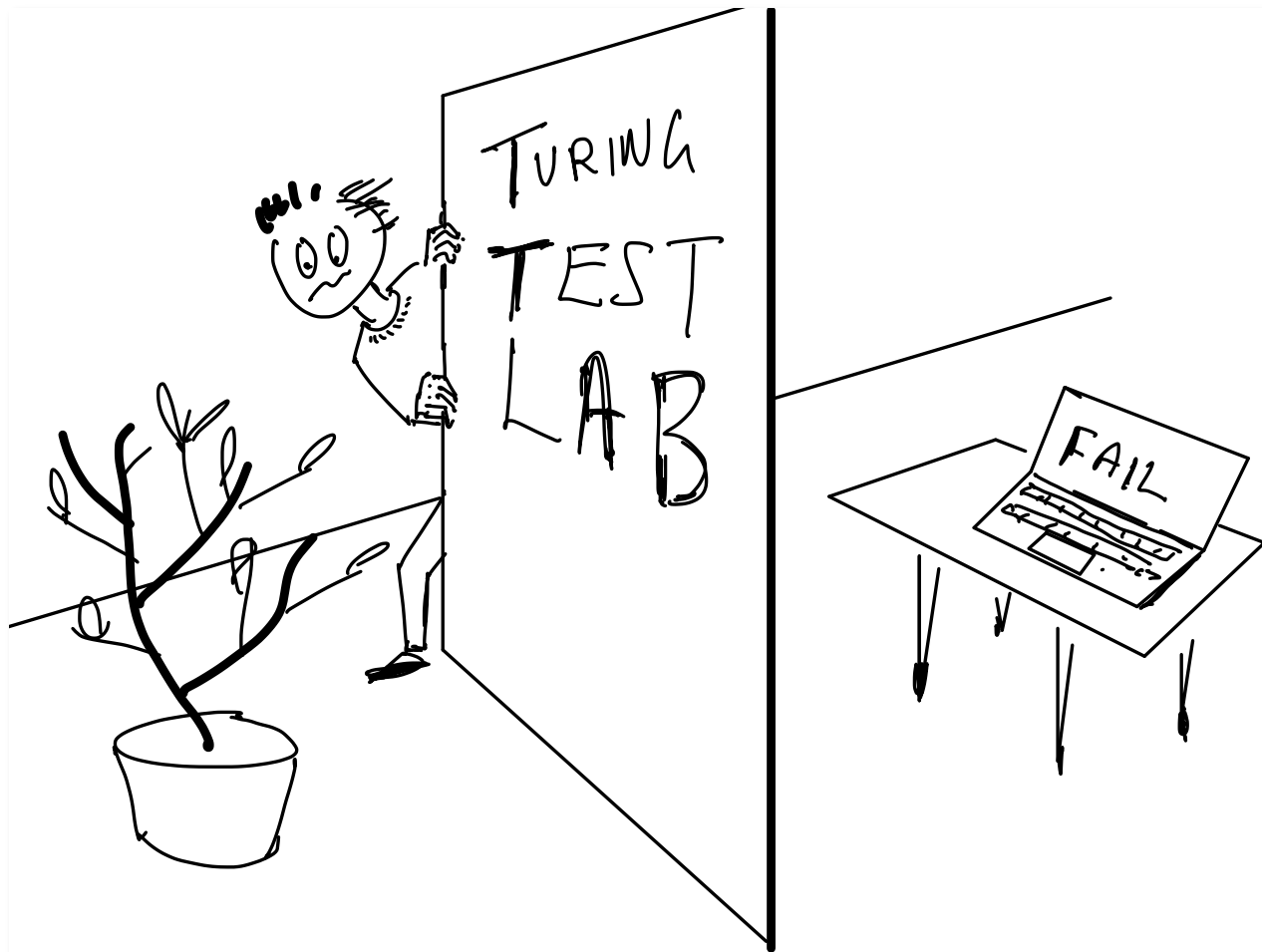


Wikimedia commons

A Very Incomplete History of ML/AI

20th century (1900 – 1950)

- 1936 – Alan Turing proposes a universal computing machine.
 - 1943 – Warren McCulloch and Walter Pitts invent *neural networks* (NN).
 - 1950 – Turing Test, an operational definition of an artificially intelligent agent.
-



“George rethinks his life after failing the Turing Test”

1997 World chess champion Gary Kasparov defeated by IBM's Deep Blue supercomputer.



Stan Honda/AFP/Getty Images

Computer Wins on ‘Jeopardy!’: Trivial, It’s Not
New York Times, Feb. 17, 2011

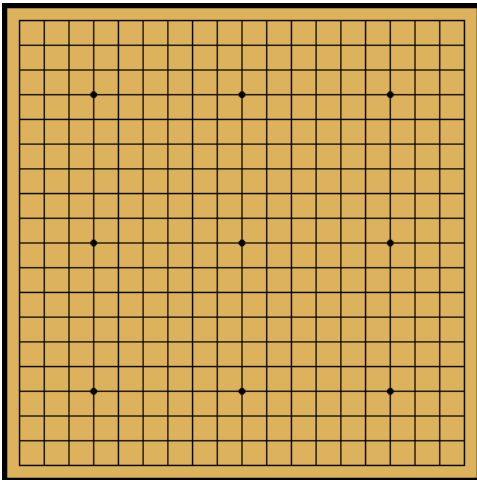


Carol Kaelson/Jeopardy Productions Inc., via Associated Press

Ken Jennings: “I felt obsolete”
TED Talk

AlphaGo 4, Homo sapiens 1!

2016 – Google's *AlphaGo* program beats Go champion Lee Sodol.



Photograph: Yonhap/Reuters

AlphaZero

Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm

David Silver,^{1*} Thomas Hubert,^{1*} Julian Schrittwieser,^{1*}
Ioannis Antonoglou,¹ Matthew Lai,¹ Arthur Guez,¹ Marc Lanctot,¹
Laurent Sifre,¹ Dharshan Kumaran,¹ Thore Graepel,¹
Timothy Lillicrap,¹ Karen Simonyan,¹ Demis Hassabis¹

¹DeepMind, 6 Pancras Square, London N1C 4AG.

*These authors contributed equally to this work.

arXiv:1712.01815v1

.AIJ 5 Dec 2017

“Starting from **random play**, and given **no domain knowledge** except the game rules, *AlphaZero* achieved within 24 hours a **superhuman level of play** in the games of chess and shogi (Japanese chess) as well as Go, and convincingly defeated a world-champion program in each case.”

“Almost half the activities people are paid almost \$16 trillion in wages to do in the global economy have the potential to be automated by adapting currently demonstrated technology, according to our analysis of more than 2,000 work activities across 800 occupations.”

McKinsey & Company,

A FUTURE THAT WORKS: AUTOMATION, EMPLOYMENT, AND
PRODUCTIVITY

Executive Summary January 2017

A Very Brief History of ML/AI

“Profesores de la Universidad de Santiago protestan por su reemplazo por iPhone 9000s”

THE SANTIAGO TIMES

Thursday, March 20, 2053

Women in AI

Women have had a huge impact on computer science and AI, but, alas, they have not received the attention they deserve. Please take a look at these marvelous stories:

- <https://www.britannica.com/story/ada-lovelace-the-first-computer-programmer>
- <https://www.forbes.com/sites/markminevich/2022/03/10/the-9-inspirational-women-leaders-in-ai-shaping-the-21st-century/?sh=45c8c3672bb0>
- <https://imagga.com/blog/11-female-researchers-made-big-impact-artificial-intelligence/>

FOUNDATIONS

What is Machine Learning?

The art and science of creating statistical *models*

$f(x, \omega) \in F$ of data by minimizing a quantity called the *average loss*, or *empirical risk*,

$$R(\omega) = \frac{1}{N} \sum_{i=1}^N L(t_i, f_i)$$

where

$T = \{(t_i, x_i)\}$

f_i

$L(t_i, f_i)$,

are training data (*targets, inputs*),

is the model $f(x, \omega)$ evaluated at x_i , and

is the *loss function*, a measure of the loss incurred by choosing a function from F .

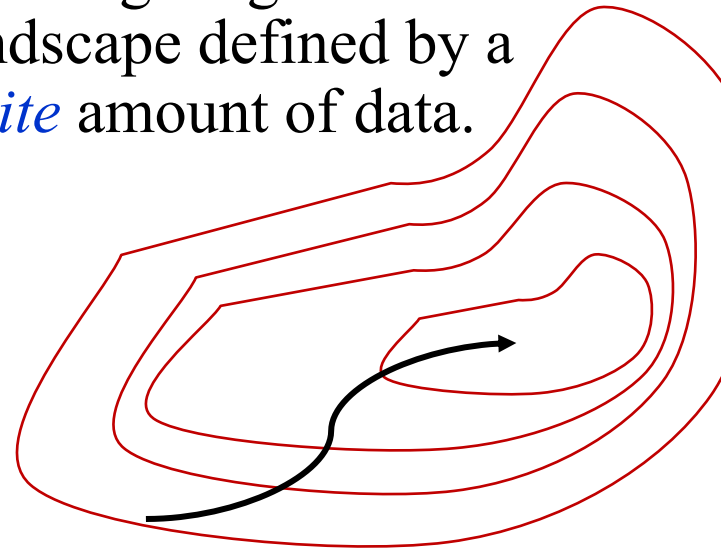
F = Function class

Minimizing the Average Loss

The average loss, $R(\omega)$, defines a “landscape” in the *parameter space* of the model $f(x, \omega) \in F$.



The Goal: find the lowest point in the landscape defined by an *infinite* amount of data by navigating the landscape defined by a *finite* amount of data.



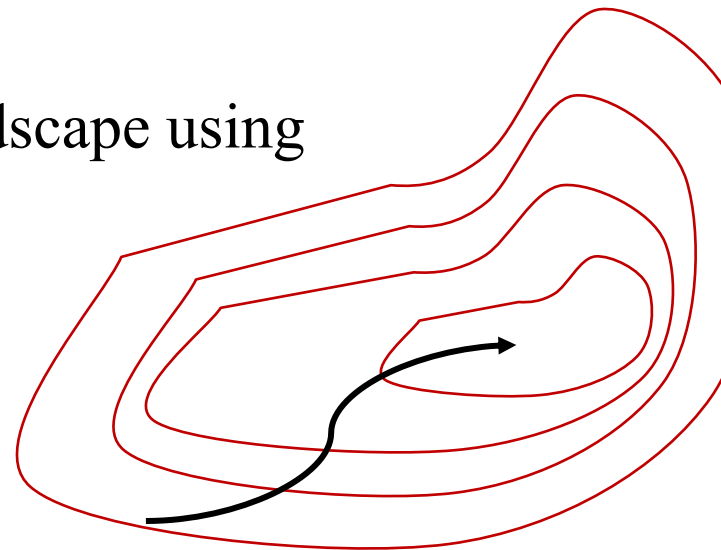
Minimizing the Average Loss

This is typically done by moving in the direction of steepest descent using **Stochastic Gradient Descent**.

At every step:

1. Compute the local gradient of $R(\omega) = \frac{1}{n} \sum_{i=1}^n L(t_i, f_i)$ using a *batch* of training data with $n \ll N$.
2. Move to the next position in the landscape using

$$\omega_{j+1} = \omega_j - \eta \nabla R$$



Minimizing the Average Loss

Why does this algorithm

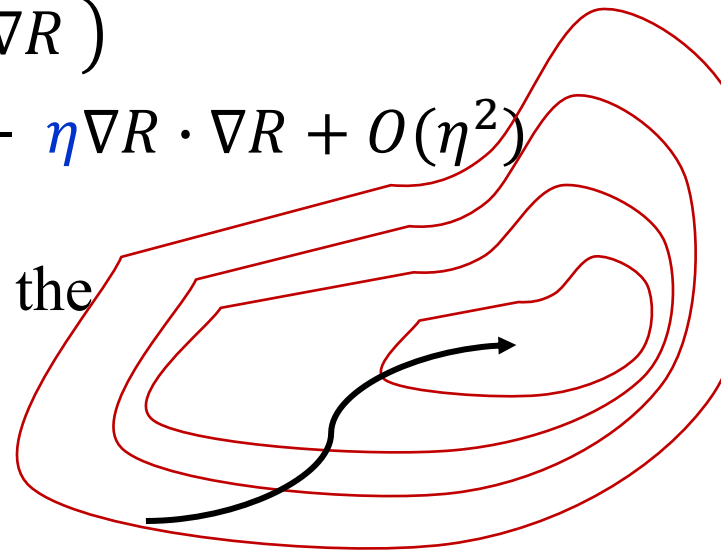
$$\omega_{j+1} = \omega_j - \eta \nabla R$$

work?

Here's why:

$$\begin{aligned} R(\omega_{j+1}) &= R(\omega_j - \eta \nabla R) \\ &= R(\omega_j) - \eta \nabla R \cdot \nabla R + O(\eta^2) \end{aligned}$$

If the $O(\eta^2)$ can be neglected, and since the $O(\eta)$ term is always negative, then $R(\omega_{j+1}) < R(\omega_j)$.



Minimizing the Average Loss

Why do we use batches of data with $n \ll N$?



To increase training speed and to add noise!

Minimizing the Average Loss

Since the goal, ideally, is to find the lowest point of the “landscape” for an *infinite* amount of training data, it’s instructive to consider the limit $N \rightarrow \infty$.

In that limit, the average loss $R(\omega)$ becomes the *functional*

$$R[f] = \int dx \int dt L(t, f) p(t, x)$$

which, given that $p(t, x) = p(t|x) p(x)$, can be written as

$$R[f] = \int dx p(x) \left[\int dt L(t, f) p(t|x) \right]$$

Minimizing the Average Loss

The *calculus of variations* shows that if $p(x) > 0$ for all values of x then the location of the minimum of $R[f]$, and hence the optimal function $f(x, \omega^*)$, is found by solving the equation

$$\frac{\delta R}{\delta f} = \int \frac{\partial L}{\partial f} p(t|x) dt = 0$$

The goal of a machine learning training algorithm is to find good approximations to solutions of the above equation using a (necessarily) finite training sample.

Common Loss Functions

Quadratic loss: $L(t, f) = (t - f)^2$

$$\int \frac{\partial L}{\partial f} p(t|x) dt = 0$$

Solution

$$f(x, \omega^*) = \int t p(t | x) dt$$

Very Important Point (VIP): The solution is independent of the details of the model f . The solution depends solely on the form of the loss function and the probability distribution, $p(t, x)$, associated with the training data.

Common Loss Functions

Binary cross entropy loss:

$$L(y, f) = -[t \log f + (1 - t) \log(1 - f)]$$

$$\int \frac{\partial L}{\partial f} p(t|x) dt = 0$$

Solution

$$f(x, \omega^*) = p(t = 1 | x) = \frac{p(x|t = 1)\epsilon}{p(x|t = 1)\epsilon + p(x|t = 0)}$$

where $t \in [0, 1]$ and $\epsilon = \frac{\pi(t=1)}{\pi(t=0)}$ is the ratio of training sample sizes for the two classes of objects labeled by $t \in [0, 1]$.

Common Loss Functions

Exponential loss:

$$L(y, f) = \exp(-wtf/2)$$
$$\int \frac{\partial L}{\partial f} p(t|x) dt = 0$$

Solution

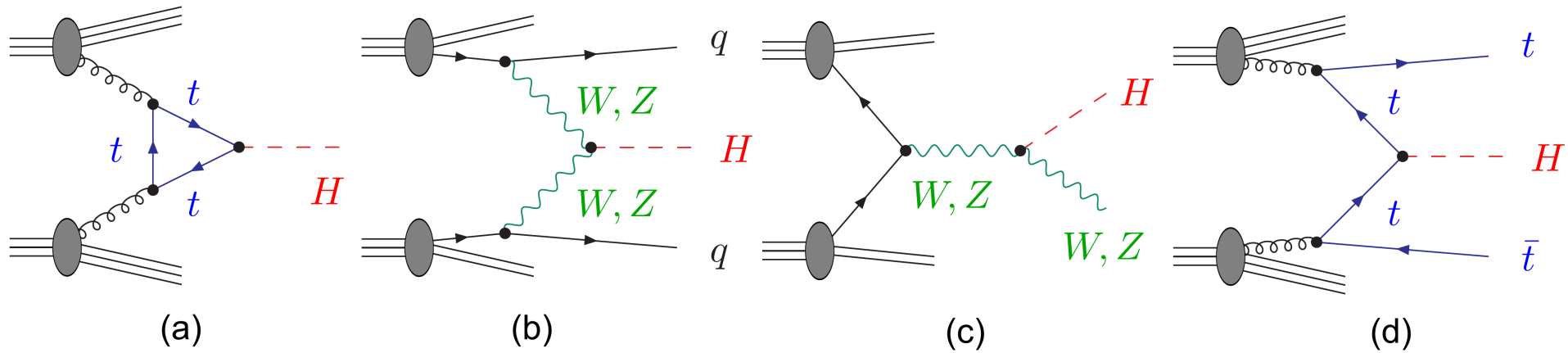
$$f(x, \omega^*) = \frac{1}{w} \log \left(\frac{p(x|t=1)}{p(x|t=-1)} \epsilon \right)$$

where $t \in [-1, 1]$ and $\epsilon = \frac{\pi(t=1)}{\pi(t=-1)}$ is the ratio of training sample sizes for the two classes labeled by $t \in [-1, 1]$.

EXAMPLE: DECISION TREES

pp → *H* → *ZZ* → *4l*

$pp \rightarrow H \rightarrow ZZ \rightarrow 4l$



Process

$\sigma \times BR$ (fb)

- (a) Gluon gluon fusion (ggF)
- (b) Vector boson fusion (VBF)
- (c) Associated production (VH)
- (d) Top anti-top fusion (ttH)

$pp \rightarrow H \rightarrow ZZ \rightarrow 4l$

We shall use *decision trees*
with the variables

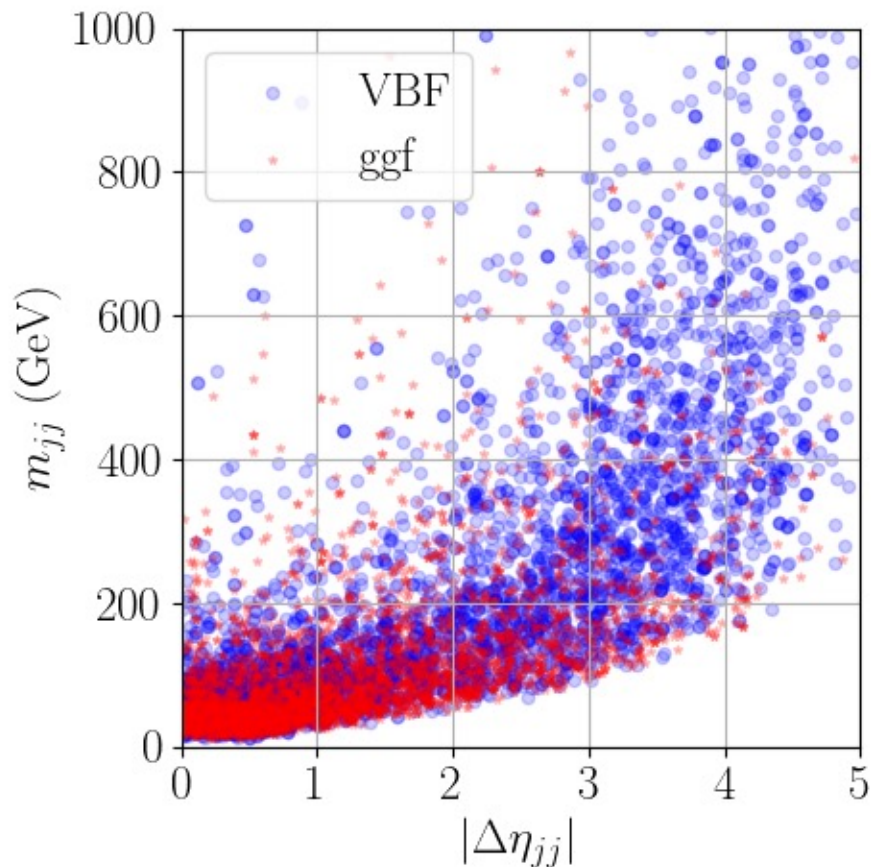
$$|\Delta\eta|_{jj}, \quad m_{jj}$$

to try to separate

$$VV \rightarrow H$$

from

$$gg \rightarrow H.$$



ATLAS Open Data

Decision Trees

A decision tree (DT) is a set of **if then else** statements that form a tree-like structure.

Algorithm: recursively partition the space into regions of diminishing *impurity*.

A common measure of impurity is the *Gini Index*:

$p(1 - p)$, where p is the *purity*

$$p = S / (S + B)$$

$p = 0$ or 1 : maximum purity

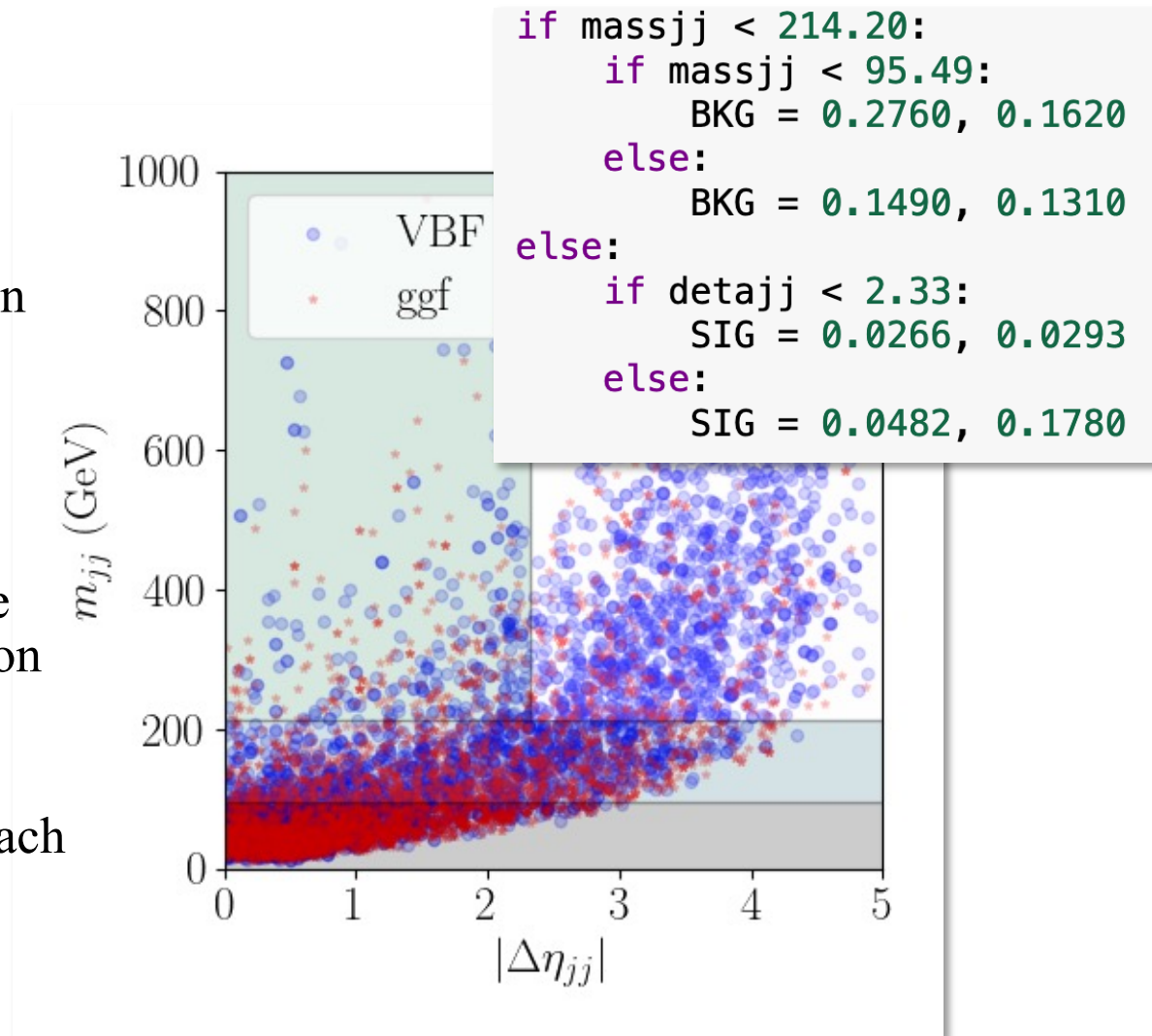
$p = 0.5$: maximum impurity



(Corrado Gini, 1884-1965)

Decision Trees

1. For each variable, find the partition (“cut”) that gives the greatest *decrease* in impurity.
2. Choose the *best partition* among all partitions and split the data along that partition into *two* subsets.
3. Repeat 1. and 2. for each subset of data.



Decision Trees (DT)

Unfortunately, decision trees are *unstable*!

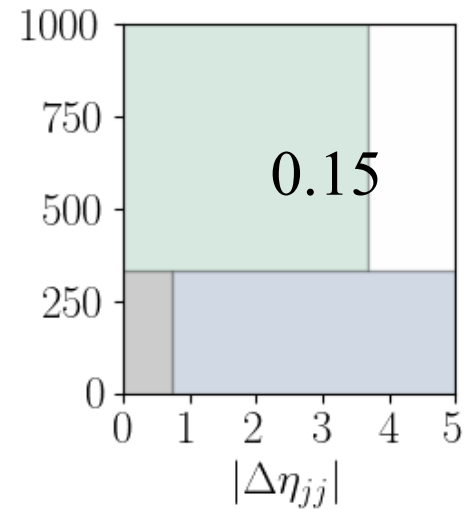
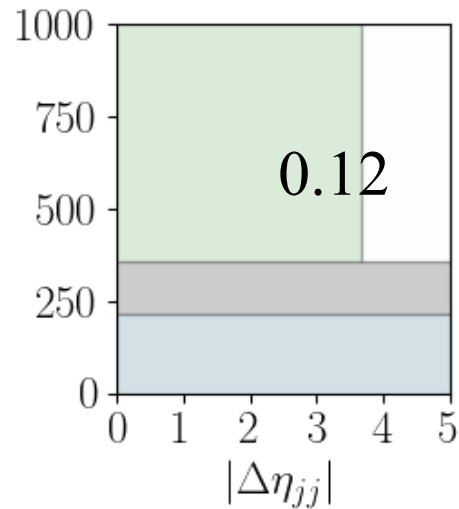
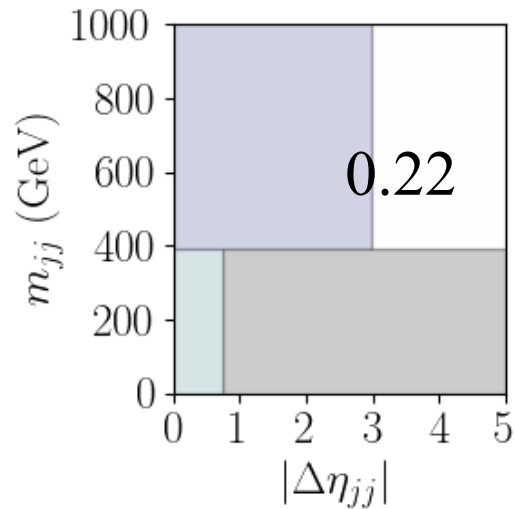
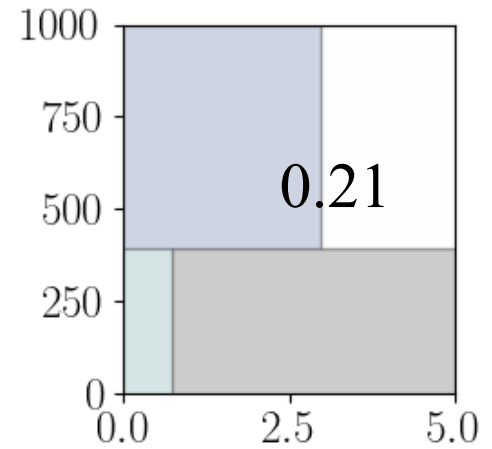
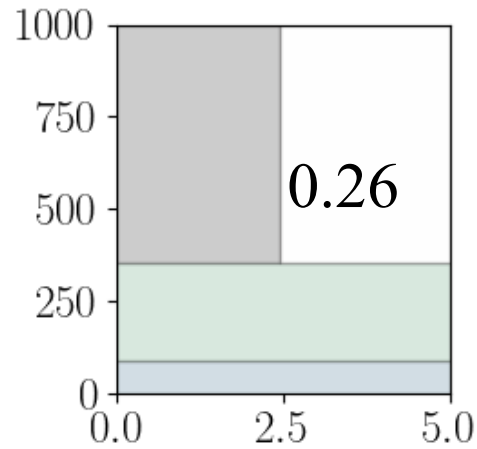
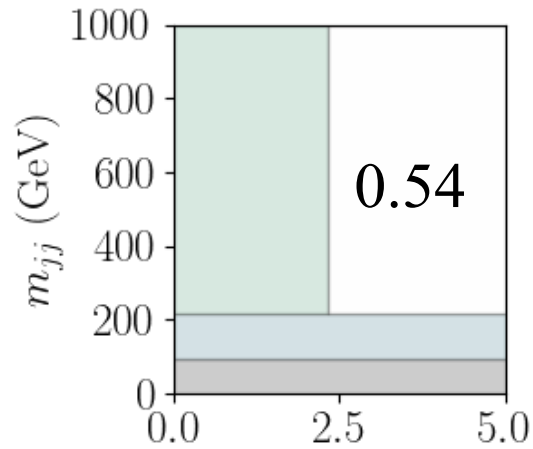
DT Averaging Methods

The most popular decision tree averaging methods are:

- **Bagging:** each tree is trained on a **bootstrap*** **sample** drawn from the training set
- **Random Forest:** bagging with **randomized** trees
- **Boosting:** each tree trained on a **different reweighting** of the training set

*A bootstrap sample is a sample of size N drawn, *with replacement*, from another of the same size. Duplicates can occur and are allowed.

First 6 Decision Trees



Boosted Decision Trees (BDT)

The contours are obtained from

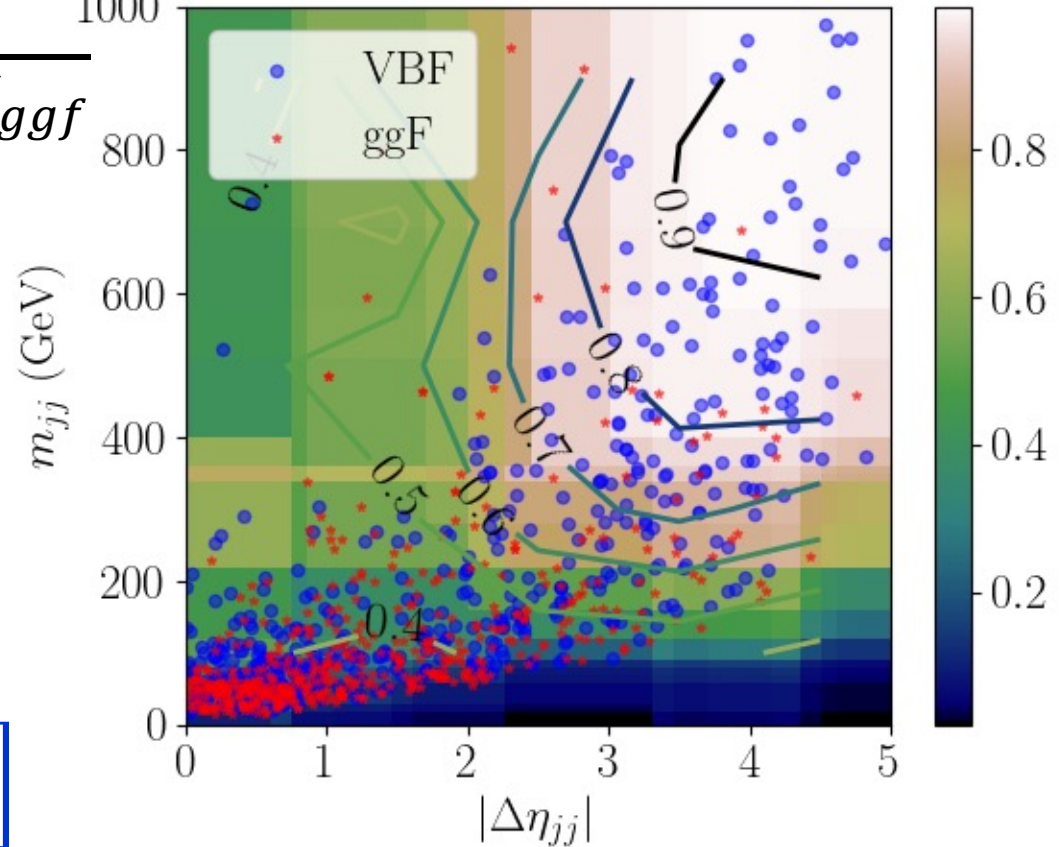
$$p(t = 1|x) \approx \frac{H_{VBF}}{H_{VBF} + H_{ggf}} \quad 1000$$

where H_{VBF} and H_{ggf} are histograms in the $x = |\Delta\eta_{jj}|, m_{jj}$ space.

A BDT minimizes

$$R[f] = E \left[\exp \left(-\frac{wtf}{2} \right) \right]$$

with $w = 2$, which yields $p(t = 1|x) = 1/(1 + \exp(-2f))$



Summary

- Machine learning (ML) models are *statistical models* trained, that is, fitted, to data by minimizing a given *average loss*.
- The mathematical quantity approximated by an ML model depends solely on the *form of the loss function and the probability distribution of the training data*. In particular, it does not depend on the details of the model!
- The quality of the approximation, however, does depend on the model, as well as the amount of data used, and the quality of the training.