UNIVERSITY OF GENOA

MASTER THESIS

Efficient Machine Learning for new physics discoveries

Author:
Gianvito LOSAPIO

Supervisors:
Prof. Lorenzo ROSASCO
Marco LETIZIA, PhD

Examiner:
Prof. Stefano ROVETTA

A thesis submitted in fulfillment of the requirements for the MSc in Computer Science

in the

Department of Informatics, Bioengineering, Robotics and Systems Engineering (DIBRIS)

March 7, 2022

Declaration of Authorship

I, Gianvito LOSAPIO, declare that this thesis titled, "Efficient Machine Learning for new physics discoveries" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Money make life comfortable. People and ideas make it worth living."

UNIVERSITY OF GENOA

Abstract

Department of Informatics, Bioengineering, Robotics and Systems Engineering (DIBRIS)

MSc in Computer Science

Efficient Machine Learning for new physics discoveries

by Gianvito LOSAPIO

Searching for new physics, i.e., physical laws that go beyond the reference models, is the absolute priority of the field. In this thesis, a novel machine learning approach for model-independent new physics searches is presented, based on the work by D'Agnolo and Wulzer, 2019. The core algorithm of our proposal is powered by recent large scale implementations of kernel methods, non-parametric models that can approximate any continuous function given enough data. The model evaluates the compatibility between experimental data and a reference model, by implementing a hypothesis testing procedure based on the likelihood ratio. Model-independence is enforced by avoiding any prior assumption about the presence or shape of new physics components in the measurements. We show that our solution has dramatic advantages compared to neural network implementations in terms of training times and computational resources, while maintaining comparable performances. Our algorithm has been tested on High Energy Physics datasets of increasing dimensionality.

Acknowledgements

First of all I would like to thank my supervisor Prof. Lorenzo Rosasco for letting me work on this project when it was still in its infancy. Having the chance to work with him has been my main ambition since the beginning of the Master. I have a profound respect on him and I regard with large esteem his role as professor, researcher and as educator.

I would like to thank Marco Letizia who has been a fantastic supervisor. He has been a reference I could definitely count on whenever I needed help. Although mostly in videocalls, time spent with him has been great and made of stimulating conversations.

I would like to thank also Marco Rando who has helped giving a boost to this project. Discussions with him have always been interesting and fruitful.

Aside from the technical work, I would like to thank all the people who have contributed to make my life special in the last two years. A huge thanks goes to my family and to Daniela who have helped me finding again the beauty of light in the deep darkness and make myself a renewed, stronger person. Their love is to me the only thing that makes sense in this irrational world. Without them, I would not have been able to make it.

A special thanks to my friends because friendship is one of the greatest gifts of life. To Domenico, my brother who can understand me even if I don't speak, together we can laugh without a smile, we can cry without tears. To Larbi, who has been a true friend from the beginning, a nice companion of all the adventures. To Federico, my experienced colleague always ready for the next challenge; we've done lots of great things together and it's always been a lot of fun. To Alberto, the eclectic engineer who is always ready to have fun and enjoy some good time. To Mohammed, who has taught me the secrets of Morocco and how to enjoy your time. To Cesare, who reminds me how beautiful Genova is and how beuatiful is to have fun with friends. To Sanket, who is a great flatmate and has taught me a different perspective on things. To all the other friends who make my days happier and worth living.

Last but not least, a special thanks to Professor Alessandro Verri, who met me the first time I came to Genova. He has been a true idol from the beginning, and has convinced me to choose the modest, cruise-shaped Dibris instead of the marvelous University of Lugano. In hindsight, it has been the best choice!

Contents

List of Figures

List of Tables

Dedicated to my family

Chapter 1

Introduction

1.1 Artificial Intelligence

Artificial intelligence (AI) refers to intelligence demonstrated by machines, as opposed to natural intelligence displayed by animals including humans.

AI has had a rapid evolution in the last decade. Today it is a thriving field with many practical applications and active research topics. Its applications include: advanced web search engines (e.g., Google), recommendation systems (used by YouTube, Amazon and Netflix), understanding human speech (such as Siri and Alexa), self-driving cars (e.g., Tesla), strategic game playing (such as chess and Go), predicting 3D proteing structures, to name a few.

Machine Learning (ML) is the main discipline boosting what has been called the AI revolution. It is largely perceived as one of the main disruptive technologies of our ages. A popular definition of machine learning, due to Tom Mitchell (Mitchell, 1997) is as follows:

> 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 experience of ML algorithms consists of learning from data. In order to do so, almost all machine learning algorithms are built by combining at least four fundamental building blocks: a dataset, an objective function, a model and an optimisation method. More details on these basic ingredients will be provided in Chapter 2.

Deep learning is a specific type of machine learning, which in the last few years has caused many breakthroughs in the field of AI. Since the publication of the paper entitled "Deep Learning" in 2015, the importance and success of this new technology has allowed the authors to receive the Turing Prize for the year 2018[1]. Deep learning is specialized in extracting complex patterns from raw data. For instance, there are specialized models which can recognize objects in images (e.g, Krizhevsky, Sutskever, and Hinton, 2017) or recognize words in speech (e.g., Abdel-Hamid et al., 2014).

Three main factors have powered the AI revolution through deep learning: (a) the large availability of data, (b) the progress in GPU technology, (c) the development of more complex algorithms. The availability of data is what sometimes is considered the most impactful factor. An article from The Economist dated 2017 has been indeed titled "the world's most valuable resource is no longer oil, but data" [2]. The more the quantity of available data the more accurate information the machine learning models can learn from them.

[1] https://www.nytimes.com/2019/03/27/technology/turing-award-ai.html

[2] https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-\no-longer-oil-but-data

In today's deep learning-centric research paradigm, advances in artificial intelligence are primarily achieved through sheer scale: bigger datasets, larger models, more compute. For this reason, in the last years problems of efficiency and sustainability of ML algorithms have risen. A group of researchers estimated that training a single deep learning model can generate up to 626,155 pounds of CO2 emissions - roughly equal to the total lifetime carbon footprint of five cars (Strubell, Ganesh, and McCallum, 2019). Since then, a website has been created to compute the footprint of machine learning models[3].

For instance, consider the model Generative Pre-trained Transformer 3 (GPT-3)[4], a language model that uses deep learning to produce human-like text created by OpenAI[5], a San Francisco-based artificial intelligence research laboratory. GPT-3 is the largest machine learning model ever creted. Its full version has a capacity of 175 billion parameters, requiring an enormous computational power for training.

In this work, there will be a focus on machine learning algorithms having efficiency as their core design principle. They are used for the main experiments and compared to more traditional approaches, showing a great advantage in terms of time and computational requirements.

1.2 The search for new physics

The field of particle physics is based upon theories and discoveries of thousands of physicists since the 1930s. They have resulted in a remarkable insight into the fundamental structure of matter: everything in the universe is found to be made from a few basic building blocks called fundamental particles, governed by four fundamental forces.

Our best understanding of how these particles and three of the forces are related to each other is encapsulated in the Standard Model of particle physics. Developed in the early 1970s, it has successfully explained almost all experimental results and precisely predicted a wide variety of phenomena. Over time and through many experiments, the Standard Model has become established as a well-tested physics theory (Particle Data Group et al., 2020).

An overview of the Standard Model is depicted in Figure 1.1. It is composed of two groups of particles, quarks and leptons, and three fundamental forces resulting from the exchange of force-carrier particles, called bosons. At the center of the picture there is the Higgs boson which was discovered in 2012 (ATLAS, 2012).

In astrophysics, the ΛCDM (Lambda Cold Dark Matter) is a model that satisfactorily reproduces the observations of Big Bang cosmology, explaining in particular the observations of the cosmic background radiation, the large-scale structure of the universe and supernovae that indicate an accelerating expanding universe. Being the simplest model in agreement with observations, it is referred to as the current standard model of cosmology (Perivolaropoulos and Skara, 2021).

Although these models are extremely successful from a theoretical and experimental point of view, future experiments are expected to reveal phenomena which have never observed before, or to measure known phenomena with unprecedented accuracy. In fact, physicists are convinced that new physics (i.e., physical laws that

[3] https://mlco2.github.io/impact/
[4] https://github.com/openai/gpt-3
[5] https://openai.com/

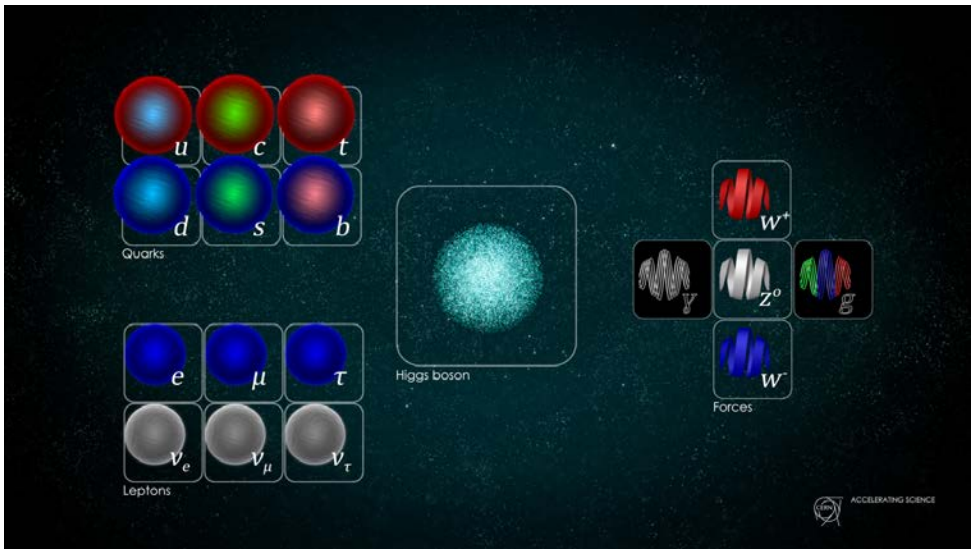


FIGURE 1.1: An overview of the Standard Model (SM). It is composed of three generations of quarks and three generations of leptons (on the left). The force-carrier particles (on the right) are the "gluon", the "photon" and the "W and Z bosons". Credits: https://home.cern/science/physics/standard-model

are not yet established) exists, because of the open problems of the reference models. For instance, the Standard Model does not currently include a microscopic description of gravitation and does not explain dark matter and the matter-antimatter asymmetry observed in the universe. For this reason, searching for new physics is considered the absolute priority of the field.

One of the main experimental tools that physicists have at their disposal to search for new physics are particle accelerators. A particle accelerator is a machine that uses electromagnetic fields to propel charged particles to very high speeds and energies, and to contain them in well-defined beams.

The Large Hadron Collider (LHC) located at CERN, Geneva, is the world's largest particle accelerator, consisting of a 26.7 km ring. The LHC accelerates protons to nearly the speed of light and then collides them at four points around its ring.

There are eight experiments at the LHC using detectors to analyse the outcome of the collisions. CMS and ATLAS are two independent general purpose detectors to explore a wide range of physical phenomena. The experiments LHCb, ALICE, TOTEM, LHCf, MoEDAL and FASER focus on specific investigations[6].

In the current thesis the focus will be on LHC-like data produced by software simulation. Each datum describes a particle collision and is called event. Each event carries some particle measurements under a specific coordinate system, e.g., the CMS coordinate system shown in Figure 1.5. In such case, the beam direction is parallel to the z-axis, and collisions occur at approximately $x = y = 0$. Typical examples of measured quantities are:

- The azimuthal angle ϕ, measured in the x-y plane
- The pseudorapidity $\eta = -\ln(\tanh(\theta/2))$, with θ being the polar angle in the z-y plane

[6]For more information see https://home.cern/science/experiments

FIGURE 1.2: An aereal view of the area where the LHC ring is located. Credits: https://cds.cern.ch/record/1295244

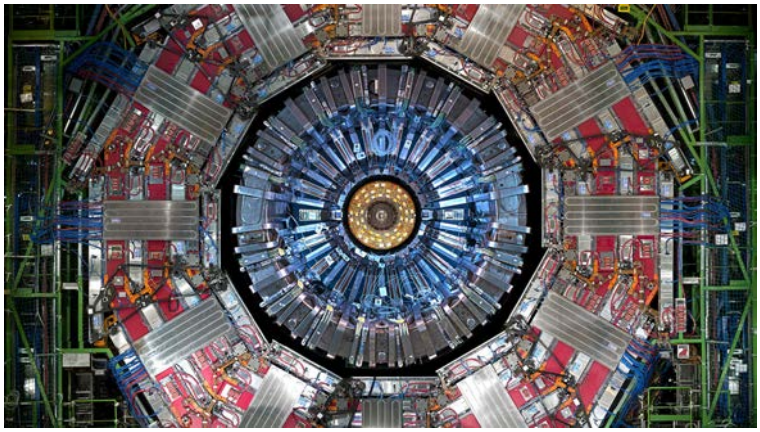


FIGURE 1.3: A picture of the CMS detector. Credits: https://cms.cern/detector

- The transverse momentum p_T, which is the component of the momentum p in the x-y plane

1.3 Machine Learning for High Energy Physics

In parallel to the rise of machine learning techniques in industrial applications, scientists have increasingly become interested in the potential of ML for fundamental research, and physics is no exception. ML has been used for data analysis in physics since the 1990s (Albertsson et al., 2018) and the recent advances in deep learning approaches have had a significant impact in the high energy physics research community. Indeed, the large size and the variety of data produced at accelerators like the LHC makes machine learning the ideal tool for trying to answer many physical questions. For a thorough overview see Carleo et al., 2019, Albertsson et al., 2018. In what follows a few examples are reported.

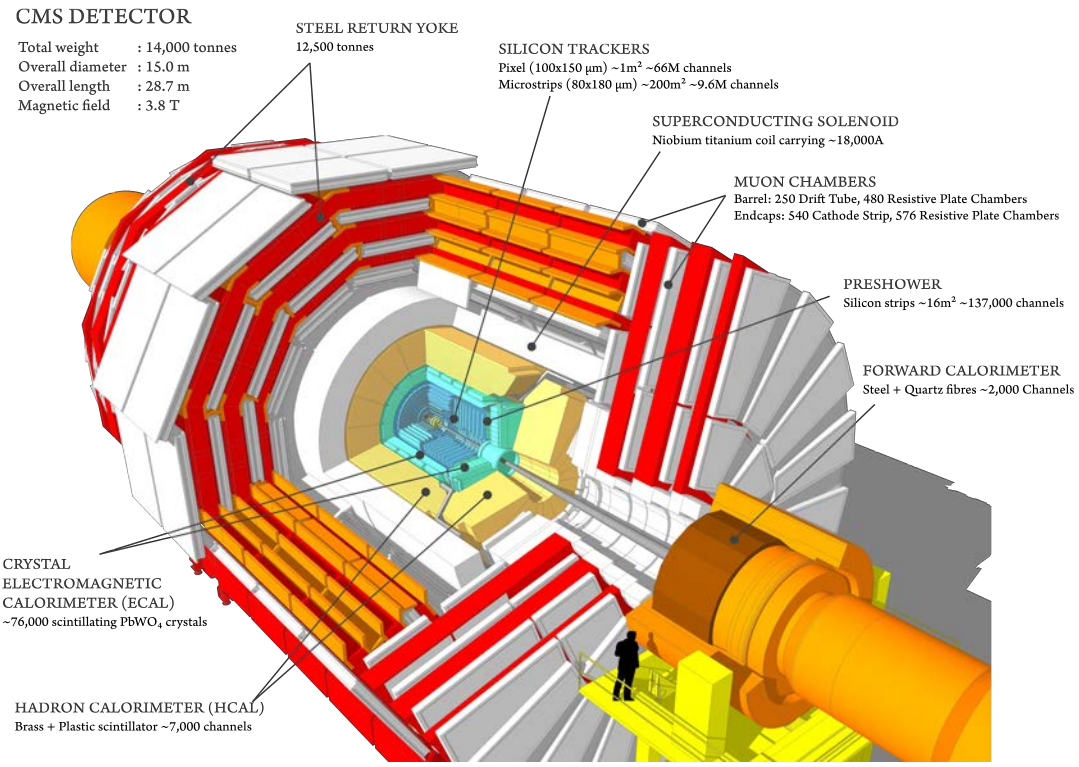


FIGURE 1.4: An overview of the layers composing the CMS detector.
Credits: https://cms.cern/detector

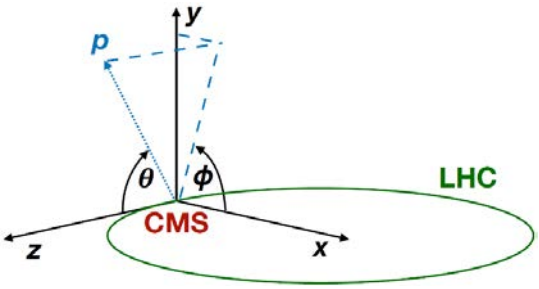


FIGURE 1.5: The coordinate system of the CMS detector

- *Simulation.* Particle discovery relies on the ability to accurately compare the observed detector response data with expectations based on the hypotheses of the Standard Model or other models of new physics. Computing the detector response analytically is intractable, thus Monte Carlo simulation softwares are employed. However, such simulations are highly computationally expensive. Fast generative models based on Machine Learning, such as Generative Adversarial Networks and Variational Auto Encoders have been proposed as a promising alternative for simulation (Paganini, Oliveira, and Nachman, 2018, Rodríguez et al., 2018).

- *Real-time analysis and Triggering.* The traditional approach to data analysis in particle physics assumes that the interesting events recorded by a detector can be selected in real-time with a reasonable efficiency (the process known as *triggering*). To exploit the full information the LHC delivers, it will increasingly be necessary to perform more of the data analysis in real-time. Machine learning methods offer the possibility to perform the real-time reconstruction that

enables real-time analysis in the first place. For example, the CMS experiment uses boosted decision trees in the Level 1 trigger to approximate muon momenta (Gligorov and Williams, 2013). Furthermore, machine learning models such as deep neural networks have recently been proposed to facilitate the early identification of potential physics signals, as well as improve the current data reconstruction pipeline in the Level 1 trigger system (Golubovic et al., 2020, Losapio, 2021).

- *Object Reconstruction, Identifiation, and Calibration.* The physical processes of interest in high energy physics experiments occur on time scales too short to be observed directly by particle detectors. For instance, a Higgs boson produced at the LHC will decay within approximately 10^{-22} seconds and thus decays essentially at the point of production. However, the decay products of the initial particle, which are observed in the detector, can be used to infer its properties. Better knowledge of the properties (e.g. type, energy, direction) of the decay products permits more accurate reconstruction of the initial physical process. Experiments have trained Machine Learning algorithms on the features from combined reconstruction algorithms to perform particle identification for decades. In the past decade Boosted Decision Trees have been one of the most popular techniques in this domain. More recently, experiments have focused on extracting better performance with deep neural networks (Baldi, Sadowski, and Whiteson, 2014, Baldi et al., 2016, Choma et al., 2018).

- *Offline data analysis.* New physics may manifest itself as unusual or rare events. One approach is to accurately identify the Standard Model processes and search for anomalies. Machine Learning algorithms can be used to identify Standard Model events. Once an event is classied as likely to be from a known physics process, it can be filtered out, and remaining events can be further analyzed for new physics clues. Techniques such as supervied/unsupervised classifiers, anomaly detections and density estimation have recently been used for this purpose (D'Agnolo and Wulzer, 2019, Chakravarti et al., 2021, Farina, Nakai, and Shih, 2020).

Different facts testify the emerging collaboration between the machine learning and high-energy physics communities. First of all, the creation in 2017 of the annual workshop "Machine Learning and the Physical Sciences"[7] during the conference Neural Information Processing Systems (NeurIPS), one of the most influential conference in the field of Machine Learning.

Other two important events have been The LHC Olympics (LHCO)[8] and the Dark Machine[9] data challenge hosted during 2020. They were international competition in which participants were provided a set of black-box datasets composed mostly of Standard Model (SM) background events and were charged with identifying any anomalous events that would be a sign of new physics. The main results have been presented during two conferences of the field in Spring and Summer 2020.

Finally, it is worth also pointing out that one of the last issue of the "CERN Courier", the official magazine of CERN, was dedicated to Artificial Intelligence with "New trends in AI for particle physics" being its introductory article[10].

[7] https://ml4physicalsciences.github.io/2021/
[8] https://lhco2020.github.io/homepage/
[9] https://darkmachines.org/
[10] https://home.cern/resources/courier/physics/cern-courier-sepoct-2021

The work of the current thesis is focused on new physics searches. The main contributions as well as the related works are presented in the following paragraphs.

1.4 Related works

Strategies to search for traces of new physics in experimental data can be either model-dependent or model-independent. In the first case, data analysis is targeted towards specific signatures determined, for example, by a candidate new theory. The second type of analysis should instead be sensitive to any generic departure from a given reference model. Recently, there has been a strong push towards developing solutions based on machine learning for (partial or full) model-independent searches.

A recent proposal by Chakravarti et al., 2021 shares some similarities with the current work. They developed a model-independent strategy based on classifiers to perform hypothesis testing on Standard Model samples and experimental measurements. They implement a hypothesis test procedure based on a traditional train-test split which causes a reduction in overall available data for new physics detection.

The work Farina, Nakai, and Shih, 2020 employs a different class of machine learning models, namely deep autoencoders, for model-independent search for new physics. Autoencoders are explored in both weakly supervised and unsupervised forms. Differently from the current work the datasets are made up of images (produced by particles called jets) and the core algorithm are convolutional neural networks.

The work Cerri et al., 2019 uses unsupervised variational autoencoders trained on known physics processes for model-independent new physics searches. However, differently from the current work the target level of analysis is the trigger system of the LHC experiments, whereas the target of this thesis is an offline analysis.

The paper De Simone and Jacques, 2019 proposes a new scientific application of unsupervised learning techniques to boost the ability to search for new phenomena in data, by measuring the degree of compatibility between two data samples. In particular, a statistical hypothesis is built test upon a test statistic which measures deviations between two datasets, relying on a Nearest-Neighbors technique to estimate the ratio of the local densities of points in the samples. However, differently from the current work, the model is not tested on real, high-dimensional dataset.

The proposal of Collins, Howe, and Nachman, 2018 is that a classifier can be trained to discern the auxiliary characteristics of the signal (if present) directly from data, without reference to any specific signal model hypothesis. The output of this classifier can then be used to select signal-like events and reject background events, producing a new distribution in the resonant variable that remains smooth in the case that no signal was present, but that may enhance the significance of the bump if a real signal is present. In the event that a signal is discovered, the output of the classifier can then be studied to infer the signal characteristics. Despite being similar in its conceptual foundation, this work is different from the current thesis because a nested cross validation with a train-test split is applied for the reconstruction of the test statistics and for inference, whereas in the current work no split is done in order to strictly preserve model-independence.

In the past years the searches for new physics signals in the high-dimensional experimental data have been primarily conducted using model-dependent data analysis methods. These searches are generally structured as likelihood ratio tests based on a model assumption for the specific new signal (Cowan et al., 2011, The ATLAS

Collaboration and The CMS Collaboration, 2011). Due to the high-dimensionality of the data, tests based on classifiers that are optimized to detect a particular hypothesized signal have been preferred over density estimation or mixture model approaches. Specifically, tests based on supervised, multivariate classification algorithms such as neural networks and boosted decision trees have proved beneficial. The classifier output is then used to extract a signal enriched sample which is used to perform likelihood ratio tests for the detection of the signal (ATLAS, 2012), Chatrchyan et al., 2012). The main drawback of such approaches is that an algorithm trained on a wrong signal model might completely miss the actual signal. This is the main reason why model-independent approach have ultimately been preferred over model-dependent ones.

1.5 Main contributions

The objective of this work is to propose an efficient ML approach to model-independent new physics searches powered by kernel methods and based upon a recent work by D'Agnolo and Wulzer, 2019. No prior assumptions are made on the nature of the new physics signals. Therefore, the proposed algorithm is potentially able to detect any type of deviations from the reference model.

The basic strategy is to train a classifier on experimental measurements and on Standard Model data (simulated or from a control region) to build a hypothesis testing procedure based on the likelihood ratio test (see Chapter 3). The goal is then to assess the compatibility of the data with the Standard Model and detect the presence of potential new physics clues. The core algorithm of the proposed approach is powered by recent large scale implementations of kernel methods, namely Falkon and LogFalkon (Meanti et al., 2020), which are non-parametric models that can approximate any continuous function given enough data.

The final goal is to provide physicists a versatile tool which can be employed to analyse experimental data and look for any potential discrepancy from the reference models.

The main contributions are the followings:

- the theoretical framework introduced in D'Agnolo and Wulzer, 2019 describing a model-independent approach to new physics discovery based on neural networks has been generalized to different loss functions and machine learning models.
- starting from the algorithm proposed in D'Agnolo et al., 2021, a new, efficient algorithm based on kernel methods has been introduced along with a different hyperparameter tuning procedure (see Chapter 4). The main limitations of the original algorithm have been overcome by delivering comparable performances with orders of magnitude gain in training times and considerably less computational resources.
- a Python module named `ml4hep` has been created (described in Appendix B), incorporating all the features to train neural networks and kernel methods on local machines or on a farm of computers

The current work has been presented during the "Machine Learning for the Physical Sciences" workshop at the 35th Conference on Neural Information Processing

Systems (NeurIPS 2021) and at the 20th International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT 2021).[11]

The thesis is organized as follows. Chapter 2 contains an overview of Machine Learning, ranging from the definition of empirical risk minimization, to a focus on classification, anomaly detection, with a section on neural networks and kernel methods. Chapter 3 provides a description of the problem of new physics searches in a statistical framework, with an overview of hypothesis testing and likelihood ratio tests. Chapter 4 describes the details of the algorithms used to search for new physics, with a general formulation of the problem and specific applications of the two machine learning models considered. Chapter 5 presents the experiments and the results, with a focus on the results obtained with the proposed method plus additional experiments. Chapter 6 concludes the thesis with a discussion of the results and an outlook on possible future work.

[11] Link to ACAT paper: https://indi.to/d6qmF
Link to NeurIPS paper: https://ml4physicalsciences.github.io/2021/files/NeurIPS_ML4PS_2021_146.pdf

Chapter 2

Machine Learning

In the following, basic concepts about machine learning will be introduced along with a description of the two models used throghout this work, neural networks and kernel methods. For more details see Murphy, 2012; Friedman, Hastie, Tibshirani, et al., 2001; Bishop, 2006.

2.1 Background

Machine learning is the study of computer algorithms that allow computer programs to automatically improve through experience (Mitchell, 1997). There are two main approaches to machine learning: supervised and unsupervised. In the following the focus will be on supervised machine learning.

Many practical problems can be formulated as requiring a computer to perform a mapping $f : X \to Y$, where X is an input space and Y is an output space (see fig. 2.1). Depending on the space Y, it is possible to distinguish between two type of problems:

- regression if $Y \in \mathbb{R}^p$ (with $p \geq 1$)
- classification if Y is a finite set, $|Y| = k$ with $k \geq 2$ number of classes

For instance, in the context of the current work, X could be the space of particle measurements and Y could be the interval $[0, 1]$ indicating the probability of observing a physical process of interest. Such a function f can be for instance specified through rule-based approaches. On the other hand, supervised machine learning offers an alternative approach that takes advantage of the possibility of collecting examples $x \in X \times Y$ of the desired mapping, e.g., via software simulations.

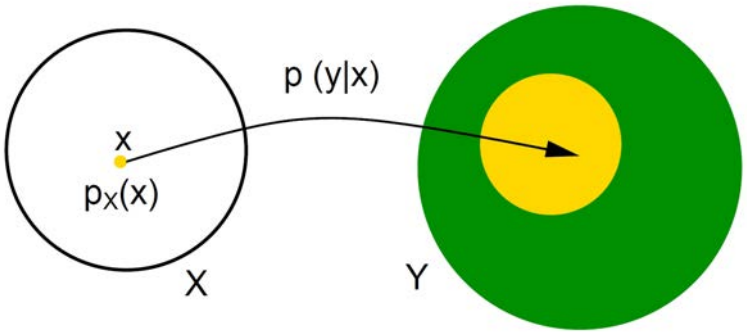


FIGURE 2.1: Intuitive visualization of a mapping between an input space and an output space with the corresponding data distributions. Credits: Rosasco, 2017

The approach of supervised machine learning is based on the fact that the machine is programmed to "learn" f by solving an optimization problem

$$\min_{f \in \mathcal{H}} \mathbb{E}_{(x,y)\sim p}\Big[\ell\big(f(x), y\big)\Big] \tag{2.1}$$

with

$$\mathbb{E}_{(x,y)\sim p}\Big[\ell\big(y, f(x)\big)\Big] = \int_{X\times Y} dx\, dy\, p(x,y)\, \ell\left(f(x), y\right) \tag{2.2}$$

given:

- a fixed but unknown probability distribution P jointly describing the input space X and the output space Y, assumed to be factorizable as $P = p(x)\, p(y|x)$. $p(y|x)$ models a non-deterministic relation between input data $x \in X$ and output data $y \in Y$, whereas $p(x)$ models the uncertainty in the input data generation from the space X.
- a space of candidate functions $\mathcal{H}$ where to look for $f : X \to Y$. Typically, $\mathcal{H}$ is a parametrized space, with Θ being the space of parameters. Each candidate function $f \in \mathcal{H}$ is described as

  $$f : (X; \Theta) \to Y$$

- a loss function

  $$\ell : Y \times Y \to [0, \infty)$$
  $$(f(x), y) \rightarrowtail \ell\left(f(x), y\right)$$

  which measures the error incurred in approximating y with $f(x)$.

A *learning algorithm* A_γ returns a function $\hat{f}\gamma$ aimed to be an approximate solution to the problem (2.1). More formally, A_γ can be defined as an exploration procedure of a hypothesis space made of measurable functions $\mathcal{H}$, generally depending on some (hyper)parameters γ (as clarified later). The learning algorithm A_γ takes as input a *training set*, that is a collection of examples $\mathcal{S} = \{(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)\}$ sampled from P and returns a solution $\hat{f}_\gamma \in \mathcal{H}$:

$$A_\gamma : (X \times Y; \mathcal{H}, \theta) \to \mathcal{H}\,, \quad A_\gamma(\mathcal{S}) = \hat{f}_\gamma \tag{2.3}$$

The field of statistical learning theory offers the tools to understand when $\hat{f}_\gamma$ is a good approximation of the ideal solution to the problem (2.1) and to quantify the probability of how close they are to each other.

Note that the quantity $\mathbb{E}_{(x,y)\sim P}\Big[\ell\big(y, f(x)\big)\Big] = L(f)$ is called the *expected risk* and cannot be computed since P is unknown. Different strategies to define learning algorithms are present in the literature, such as the one described in the next paragraph.

2.2 Empirical risk minimization

A strategy in machine learning used to design learning algorithms is *empirical risk minimization* (ERM), which is based on replacing the expected value in (2.1) with an

empirical mean on the dataset $\mathcal{S}$:

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell\big(y_i, f(x_i)\big) \tag{2.4}$$

Note that $\frac{1}{n} \sum_{i=1}^{n} \ell\big(y_i, f(x_i)\big) = \hat{L}_{\hat{\mathcal{S}}}(f)$ is called the empirical risk giving the name to the method.

Making a step forward, very often the target problem for a learning algorithm A_γ is a slightly different version of (2.4), called penalized ERM:

$$A_\gamma(\mathcal{S}) = \hat{f}_\theta = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ell\big(y_i, f_\theta(x_i)\big) + \lambda \Phi(f_\theta) \tag{2.5}$$

where the function f depends on some (hyper)parameters θ and also the minimization problem is modified by the presence of a novel term $\lambda\Phi(f_\theta)$ called regularizer. Thanks to the regularizer, the exploration procedure is penalized in order to favor certain types of solutions.

For instance, consider the regularized least squares algorithm (RLS or ridge regression):

$$\mathrm{RLS}_\gamma(\mathcal{S}) = \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left(y_i - w^\top x_i\right)^2 + \lambda \left\|w\right\|^2 \tag{2.6}$$

where:

- the loss function is the squared loss $\ell\left(f(x), y\right) = \left(f(x) - y\right)^2$ (further discussed in section 2.3.1)
- the exploration procedure happens in the space of linear functions

$$\mathcal{H} = \{f : \mathbb{R}^d \to \mathbb{R},\ \exists\, w \in \mathbb{R}^d \ \text{ s.t. } \ f(x) = w^\top x \ \ \forall x \in \mathbb{R}^d\}$$

- the regularizer is the squared L2 norm of the parameters w:

$$\lambda\Phi(f_w) = \lambda \left\|w\right\|^2$$

  where $\lambda \in \mathbb{R}$ is called the regularization parameter and controls how much the regularizer affects the final solution.

- the parametrization of the solution $A_\gamma = \hat{f}_\gamma$ is here explictly defined as $\gamma = \{\lambda, \theta\}$.

It is worth pointing out that regularization methods (when and how to define regularizers) are important to make the error small not only on the training set but also on future (unknown) data, not present in the dataset $\hat{\mathcal{S}}$ at hand. This concept is implicitly contained in the original problem formulation (2.1) but not in the approximate solution (2.4) and hence it is one of the most relevant problem in machine learning. In the literature, such a problem is referred to as generalization problem and it is formally characterized through statistical estimator properties applied to the solution, for instance bias-variance decomposition (further discussed in the paragraph 2.3.2).

2.3 Machine learning pipeline

The general steps used to solve a supervised learning problem are:

1. Model selection: select the best learning algorithm A_γ as well as the best (hyper)parameters γ for the problem at hand;

2. Model assessment (or error estimation): estimate the generalization error, i.e. the error that the selected model will exhibit on future (previously unseen) data by relying only on the available data $\mathcal{S}$.

Typically, three different datasets are employed for different purposes to solve a supervised learning problem instead of the single dataset $\mathcal{S}$:

- a training set $\mathcal{S}_{\text{train}}$ and a validation set $\mathcal{S}_{\text{val}}$ are used in the model selection phase. In particular, the training set $\mathcal{S}_{\text{train}}$ is the only dataset employed as input of the learning algorithm to learn a candidate solution, namely $\hat{f}_\gamma = A_\gamma(\mathcal{S}_{\text{train}})$. On the other hand, the validation set $\mathcal{S}_{\text{val}}$ is used to choose the best set $\hat{\gamma}$ of (hyper)parameters for the algorithm at hand;

- a test set $\mathcal{S}_{\text{test}}$ is used in model assessment phase, for instance to estimate the global performances of the trained learning algorithm according to appropriate metrics.

Usually, the original dataset $\mathcal{S}$ is split into three parts to obtain these different sets (once or multiple times depending on the model selection/assessment strategies, where the size of each single dataset matters, as described in the next paragraphs):

$$\mathcal{S} = \mathcal{S}_{\text{train}} \cup \mathcal{S}_{\text{val}} \cup \mathcal{S}_{\text{test}} \tag{2.7}$$

A reasonable split is, for example, $60\% - 20\% - 20\%$.

It is worth noting that the overall procedure (sometimes called machine learning pipeline) is intended to offer a statistically sound estimation of the generalization error. A lot of care should be put on the design of the whole pipeline since it can be very computational demanding. Moreover, the effectiveness of this procedure is only guaranteed in a limited scenario where:

- a large amount of data is available (e.g. thousands or hundreds thousands)

- the samples are identically and independently distributed (the so called *i.i.d. hypothesis*).

Dealing with time series or with dataset where the number of features is much larger than the number of samples is, for example, challenging since no guarantess can be offered by this pipeline.

2.3.1 Model selection and Cross Validation

Model selection is the problem of selecting the best learning algorithm A_γ for the problem at hand (recall that here the concept of a good algorithm is defined in terms of statistical estimator properties, e.g. generalization, and a problem is intended to be defined by a dataset $\mathcal{S}$ and a corresponding probability distribution over the data space $(X \times Y, P)$). In practice, model selection consists of the following steps:

1. Choose A_γ among different learning algorithms (e.g., SVMs, random forests, neural networks, kernel ridge regression);

2. Choose the structure of the learning algorithm A_γ (e.g., loss function, type of regularizer, family of kernels)

3. Set the hyperparameters γ of A_γ (if any, e.g., regularization parameters, layers and neurons of a neural network, number of trees in a forest, number of centroids in k-nearest neighbors ...)

The first two steps are essentially based on the nature of the problem at hand and the choices are guided by the known pros and cons of the algorithms as suggested by the literature or by the experience. The last step is called parameter tuning (or model tuning) and is, instead, often purely based on a trial and error procedure called *cross validation*.

Cross validation is a procedure that, given a learning algorithm A_γ and a set of (hyper)parameters $\{\gamma_j\}_{j=1}^T$, returns the best parameter $\hat{\gamma}$ for the problem at hand, chosen as the one which minimizes the empirical risk on the validation set $\mathcal{S}_{\text{val}}$:

$$\hat{\gamma} = \arg\min_{j=1,\dots,T} \hat{L}_{\mathcal{S}_{\text{val}}}(\hat{f}_{\gamma_j}) = \arg\min_{j=1,\dots,T} \frac{1}{|\mathcal{S}_{\text{val}}|} \sum_{(x,y)\in\mathcal{S}_{\text{val}}} \ell\big(y, \hat{f}_{\gamma_j}(x)\big) \tag{2.8}$$

where $\hat{f}_{\gamma_j}$ has been previously learned on the training set, namely $\hat{f}_{\gamma_j} = A_{\gamma_j}(\mathcal{S}_{\text{train}})$, e.g. according to the algorithm used to solve (2.5).

Evaluating the choice of $\hat{\gamma}$ on a different dataset with respect to the one used to learn $\hat{f}_{\gamma_j}$ is a statistically sound procedure because the empirical risk computed on the validation set is an unbiased estimator of the expected risk, namely

$$\mathbb{E}_{\mathcal{S}_{\text{val}}\sim P}\Big[\hat{L}_{\mathcal{S}_{\text{val}}}\big(\hat{f}_\gamma\big)\Big] \approx L\big(\hat{f}_\gamma\big) = \mathbb{E}_{(x,y)\sim P}\Big[\ell\big(y, \hat{f}_\gamma(x)\big)\Big] \tag{2.9}$$

where:

- $\hat{f}_\gamma$ has been learned on the training set $\mathcal{S}_{\text{train}}$, namely $\hat{f}_\gamma = A_\gamma(\mathcal{S}_{\text{train}})$
- $\hat{L}_{\mathcal{S}_{\text{val}}}\big(\hat{f}_\gamma\big)$ is the empirical risk of $\hat{f}_\gamma$ evaluated on the validation set $\mathcal{S}_{\text{val}}$
- $L\big(\hat{f}_\gamma\big)$ is the expected risk of $\hat{f}_\gamma$
- the first expected value is intended on the sampling of the validation set $\mathcal{S}_{\text{val}}$ from P, while the second is more generally intended on the whole data space $X \times Y$.

The equality (2.9) holds only if data are i.i.d. with respect to the probability P defining the problem and $\hat{f}_\gamma$ has been previously learned on a different dataset. This result tells you that by averaging the error computed on more and more samples $\mathcal{S}_{\text{val}}$ you will obtain a result which is closer and closer to the expected risk (the true generalization error).

Moreover, using Hoeffding's inequality it is possible to prove that the distance between the expected risk and the empirical risk goes to zero with high probability nearly as fast as $1/\sqrt{v}$ where v is the size of the validation set $\mathcal{S}_{\text{val}}$, namely

$$\mathbb{P}\left(\left|\hat{L}_{\mathcal{S}_{\text{val}}}\big(\hat{f}_\gamma\big) - L\big(\hat{f}_\gamma\big)\right| \le \frac{1}{\sqrt{2v}}\sqrt{\log\left(\frac{2T}{\delta}\right)}\right) \ge 1-\delta \tag{2.10}$$

where:

- T is the number of the (hyper)parameters $\gamma_j, j = 1,\dots,T$ to be validated
- $\delta \in (0,1)$ is the quantity to be fixed to achieve the above concept "with high probability"

The inequality (2.10) means that the larger the validation set $\mathcal{S}_{\text{val}}$ the more the confidence in saying that the error computed on the validation set is close to the true generalization error.

Here is a list of algorithms used for cross validations:

- hold-out cross validation is the one mentioned before. The original dataset is split into three independent sets, as described by eq. 2.7. The training set is used for the training procedure, the validation set is used for model selection, the test set is used for model assessment
- k-fold cross validation consists of splitting the dataset in k folds/sets, after removing a test set. The model is trained, in turn, on $k-1$ sets and validated on the remaining set to compute a performance measure. An intuitive visualization is shown in figure 2.2 for $k = 5$. The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop. This approach can be computationally expensive, but does not waste too much data (as is the case when fixing an arbitrary validation set), which is a major advantage when the number of samples is small.

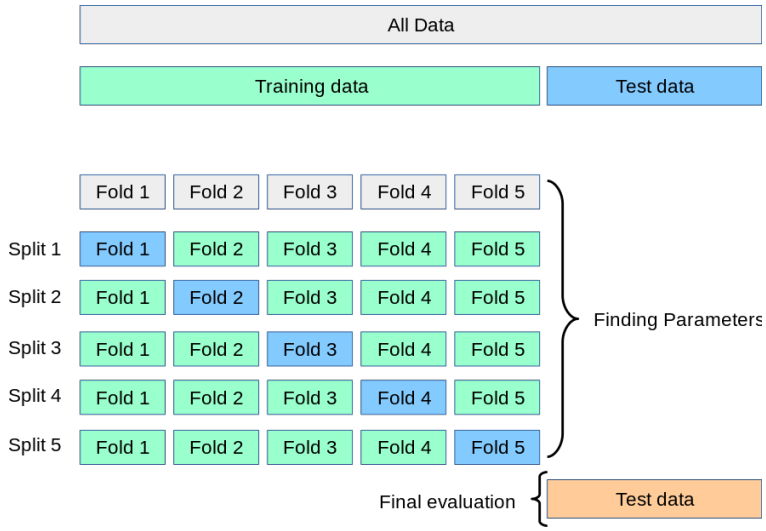


FIGURE 2.2: A visualization of the data splitting used in k-fold cross validation when $k = 5$. Credits: https://scikit-learn.org/stable/modules/cross_validation.html

- leave-p-out cross-validation involves using p observations as the validation set and the remaining observations as the training set (after removing a test set). A typical choice is $p = 1$. This approach can be computationally expensive, but preserves lots of data for the training procedure, which is a major advantage when the number of samples is really small.
- nested cross validation involves repeating the procedure k-fold cross validation twice in a nested fashion. The pseudocode is reported in figure 2.3. Also this approach is computationally demanding since there are two nested loops but can be useful when data is scarce.

Loss functions and Target functions

In supervised learning, the loss function (or cost function) is used to quantify the error incurred in using the learned function $f(x)$ to estimate the true output y. As

Algorithm *Nested cross-validation*

1. Split the set S of N available samples into k disjunct sets S_i, $i = 1, \ldots, k$, of size N/k, cf. fig. 2.14. /* outer cross-validation */
2. **for** i=1 to k
3. $\tilde{S} := S \backslash S_i$, $\tilde{N} := |\tilde{S}|$ (number of samples in $\tilde{S}$)
4. **for** each parameter set p /* parameter selection */
5. Split the set $\tilde{S}$ of $\tilde{N}$ available samples into $\tilde{k}$ disjunct sets $\tilde{S}_j$, $j = 1, \ldots, \tilde{k}$, of size $\tilde{N}/\tilde{k}$. /* inner cross-validation */
6. **for** j=1 to $\tilde{k}$
7. Train the classifier on the training set $\tilde{S}_t = \tilde{S} \backslash \tilde{S}_j$ (white in fig. 2.14).
8. Calculate test error $\tilde{e}_j$ on the parameter test set $\tilde{S}_j$ (hatched in fig. 2.14).
9. compute inner CV test error $\tilde{e}_{cv} = \frac{1}{\tilde{k}} \sum_{j=1}^{\tilde{k}} \tilde{e}_j$
10. Select parameter set p with minimum $\tilde{e}_{cv}$
11. Train classifier with selected parameter set on $S_t = \tilde{S} = S \backslash S_i$.
12. Calculate test error e_i on the test set S_i (colored in fig. 2.14).
13. Calculate outer CV test error $e_{cv} = \frac{1}{k} \sum_{i=1}^{k} e_i$.

FIGURE 2.3: Pseudocode for the nested cross validation. Credits: Petersohn, 2010

described before, it is useful to introuce a notion of loss function as any arbitrary function mapping the cartesian product of the output space into the positive real axis

$$\begin{aligned} \ell : Y \times Y &\to [0, \infty) \\ (f(x), y) &\rightarrowtail \ell\left(f(x), y\right) \end{aligned} \tag{2.11}$$

The loss function is one of the two main quantities of the learning problem since it defines the expected risk (2.1). For this reason, the loss function is a critical choice of the learning problem during the model selection phase.

The fundamental role of the loss function can be appreciated by considering the concept of *target function*, which can be introduced by further manipulating the expected risk as follows:

$$L(f) = \int_{X \times Y} dxdy\, P(x,y) \ell\left(f(x), y\right) = \int_X dx p(x) \underbrace{\int_Y dy p(y|x) \ell\left(f(x), y\right)}_{L_X(f)} \tag{2.12}$$

where the new quantity $L_X(f) : \mathbb{R} \to [0, \infty)$ is called the inner expected risk and defines the ideal error on the output data. It is possible to show that

$$\min_{f \in \mathcal{H}} L(f) = \int_X dx p(x) \min_{f(x) \in \mathbb{R}} L_X(f(x)) \tag{2.13}$$

Moreover, it can be shown that the minimizers of the expected risk can be derived "pointwise" considering the inner risk. The above result significantly simplifies the problem of characterizing the target function associated to a given loss function, since the inner expected risk is a real valued function.

$$f^*(x) = \arg\min_{a \in \mathbb{R}} L_X(a) = \arg\min_{a \in \mathbb{R}} \int_Y dy p(y|x) \ell\left(a, y\right) \tag{2.14}$$

is called the *target function* and gives the point-wise minimizer satisfying eq. (2.13).

Different loss functions define different goals via the corresponding target function.

Despite the general notion of arbitrary function, in practice convex loss functions are considered in order to accomplish the requirements of the optimization problem solver (further discussed in the paragraph 2.4).

Here is a list of examples of a few standard functions for classification and their corresponding target function:

- the 0-1 (or misclassification) loss function is the most simple example. It returns 1 if the prediction $f(x)$ and the actual label y are different, and 0 otherwise. It can be written as

$$\ell\big(f(x), y\big) = \Theta(yf(x)) \tag{2.15}$$

  where Θ is the Heaviside step function $\Theta(a) = 1$, if $a = 0$ and 0 otherwise (with classification labels $Y = \{0, 1\}$). The 0-1 loss function is a non-convex function. Its corresponding target function is the so called Bayes classification rule given by $f_P(x) = \max(p(0|x), p(1|x))$.

- the hinge loss is used in Support Vector Machines

$$\ell\big(f(x), y\big) = |1 - yf(x)|_+ = \max(0, 1 - yf(x)) \tag{2.16}$$

  Its corresponding target function is the Bayes classification rule defined before.

- the square loss

$$\ell\big(f(x), y\big) = (y - f(x))^2 \tag{2.17}$$

  is a popular loss for regression but can be also employed for classification. With classification labels $Y = \{-1, 1\}$ it can be rewritten as $\ell\big(f(x), y\big) = (1 - yf(x))^2$ and its target function is $p(1|x) - p(-1|x)$ so that its sign is exactly the Bayes rule.

- the logistic loss

$$\ell\big(f(x), y\big) = \log(1 + e^{yf(x)}) \tag{2.18}$$

  is widely used for binary classification (with labels $y = \pm 1$). Its target function is $f_P(x) = \left(\dfrac{p(1|x)}{1 - p(1|x)}\right)$. When the labels are $y = \{0, 1\}$ it is known as cross entropy loss and reads as

$$\ell\big(f(x), y\big) = y \log(1 + e^{-yf(x)}) + (1 - y) \log(1 + e^{yf(x)}) \tag{2.19}$$

  For the derivation of the target function see section 2.5.4.

The loss functions introduced so far are represented in figure 2.4.

2.3.2 Model assessment

Model assessment is the problem of estimating the error that the selected model $\hat{f}_{\hat{\gamma}}$ will exhibit on future data by relying only on the data at hand. Analogously to the previous case, the empirical risk computed on the test set is an unbiased estimator of the expected risk:

$$\mathbb{E}_{\mathcal{S}_{\text{test}} \sim P}\Big[\hat{L}_{\mathcal{S}_{\text{test}}}\big(\hat{f}_{\hat{\gamma}}\big)\Big] = L\big(\hat{f}_{\hat{\gamma}}\big) = \mathbb{E}_{(x,y)\sim P}\Big[\ell\big(y, \hat{f}_{\hat{\gamma}}(x)\big)\Big] \tag{2.20}$$

where:

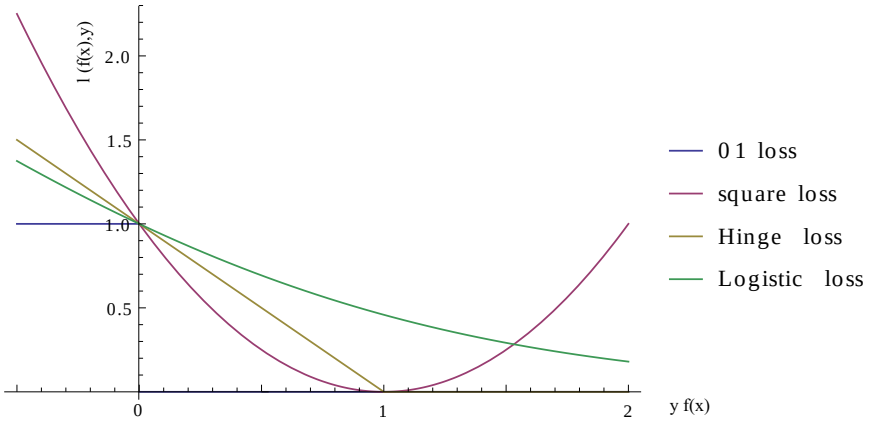


FIGURE 2.4: Examples of loss functions. Credits Rosasco, 2017

- $\hat{f}_{\hat{\gamma}}$ has been learned on the training set $\mathcal{S}_{\text{train}}$, namely $\hat{f}_{\hat{\gamma}} = A_{\hat{\gamma}}(\mathcal{S}_{\text{train}})$ with $\hat{\gamma}$ chosen on the validation set $\mathcal{S}_{\text{val}}$
- $\hat{L}_{\mathcal{S}_{\text{test}}}(\hat{f}_{\hat{\gamma}})$ is the empirical risk of $\hat{f}_{\hat{\gamma}}$ evaluated on the test set $\mathcal{S}_{\text{test}}$
- $L(\hat{f}_{\hat{\gamma}})$ is the expected risk of $\hat{f}_{\hat{\gamma}}$
- the first expected value is intended on the sampling of the test set $\mathcal{S}_{\text{test}}$ from P, while the second is more generally intended on the whole data space $X \times Y$.

The equality (2.20) holds only if data are i.i.d. with respect to the probability P defining the problem and $\hat{f}$ and its hyperparameters $\hat{\gamma}$ have been previously learned on different datasets. This result tells you that by averaging the error computed on more and more samples $\mathcal{S}_{\text{test}}$ you will obtain a result which is closer and closer to the expected risk (the true generalization error).

Underfitting, overfitting and generalization

The solution to the problem (2.1) can be intended as an estimator $\hat{\theta}$, i.e., a procedure applied to a dataset $\mathcal{S}$ which returns an estimand, e.g., a vector of parameters $\hat{\theta} = \hat{\theta}(\mathcal{S})$. So far the approach of frequentist statistics has been followed, by treating the data as a random variable, drawn from some true but unknown distribution, P. This induces a distribution over the estimand, $p^*(\mathcal{S})$, known as the sampling distribution.

The quality of an estimator can be measured through two different quantities, namely bias and variance, which measure how close the estimand is to the true value (real but unknown). An intuitive representation of bias and variance is depicted in figure 2.5(A).

The bias of an estimator is defined as

$$\text{bias}(\hat{\theta}) = \mathbb{E}\left[\hat{\theta}(\mathcal{S})\right] - \theta^* \tag{2.21}$$

where θ^* is the true parameter value, and the expectation is with respect to "nature's distribution" $p(\mathcal{S}|\theta^*)$. If the bias is zero, the estimator is called unbiased.

It seems intuitively reasonable that we want our estimator to be unbiased. However, being unbiased is not enough. The variance of an estimator is also important, defined as follows:

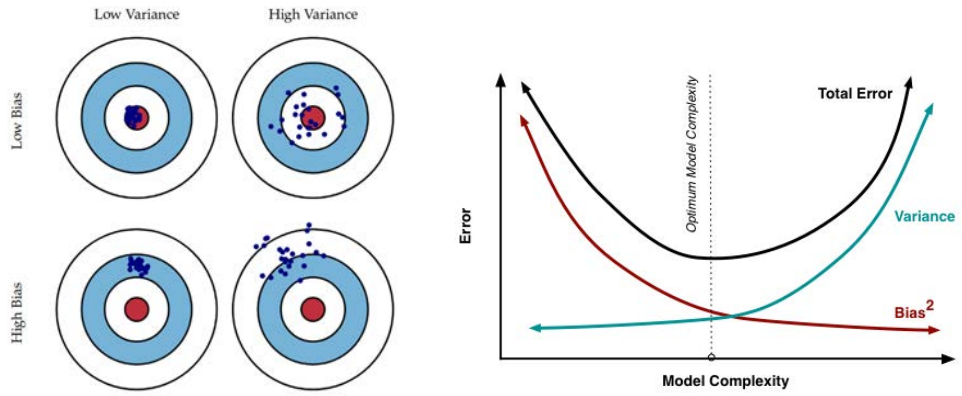
$$\text{var}(\hat{\theta}) = \mathbb{E}\left[\hat{\theta}^2\right] - \left(\mathbb{E}\left[\hat{\theta}\right]\right)^2 \tag{2.22}$$

where the expectation is taken again with respect to "nature's distribution" $p(\mathcal{S}|\theta^*)$. This measures how much the estimand will change as the data changes.

The mean square error (MSE) provides a good estimate of the overall approximation error because it takes into account at the same time the bias and the variance of the estimator. It can be computed as:

$$\begin{aligned} \text{MSE}\left(\hat{\theta}\right) &= \mathbb{E}\left[\left(\hat{\theta} - \theta^*\right)^2\right] \\ &= \text{var}\left(\hat{\theta}\right) + \left(\text{bias}\left(\hat{\theta}\right)\right)^2 \end{aligned} \tag{2.23}$$

A trade-off between the bias and the variance is required to achieve a good approximation error, the so called *bias-variance* trade-off. Figure 2.5(B) shows the different effects of the bias and variance on the approximation error. On the x-axis the model complexity is reported, a general theoretical notion which can be intended as the ability of the estimator to represent functions (determined by the size of the space of candidate functions $\mathcal{H}$ chosen through the algorithm A_γ).



(A) (B)

FIGURE 2.5: (A) Intuitive visualization of bias and variance, (B) Effect of the bias and the variance on the total error with varying model complexity. Credits https://scott.fortmann-roe.com/docs/BiasVariance.html

The notion of bias and variance is strictly related to other three fundamental concepts in machine learning, *underfitting*, *overfitting* and *generalization* .

A model that perfectly fits the training data, but fails to predict on new data (e.g., the test set) is said to suffer from overfitting. Likely, the model is too complex in this case, i.e., the estimator has low bias but high variance (right part of figure 2.5(B)). On the contrary, a model that loosely fits the training data and the new data is said to suffer from underfitting. In this case, the model is likely to show a poor complexity, i.e., the estimator has high bias but low variance (left part of figure 2.5(B)). For this reason, the bias-variance trade-off is also an underfitting-overfitting trade-off.

In its original formulation, the problem of supervised machine learning (2.1):

$$\min_{f \in \mathcal{H}} \underbrace{\mathbb{E}_{(x,y)\sim P}\left[\ell\left(f(x), y\right)\right]}_{L(f)} \tag{2.24}$$

does not contain underfitting/overfitting issues since the expectation is taken over the true underlying data distribution (real but unknown). The quantity $L(f)$ is

indeed called generalization error (or expected risk or Bayes estimator) and represents the error committed on the "entire universe" of data.

When considering the empirical minimzation, the generalization error is approximated by an average on a finite set of training data $\mathcal{S} = (x_i, y_i)_{i=1}^n$:

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell\big(y_i, f(x_i)\big) \tag{2.25}$$

As a consequence, the notion of generalization is no longer implicitly there but has to be enforced by other conditions (e.g., the use of regularization, validation and test sets, as explained before).

Figure (2.6) shows an example of underfitting and overfitting scenarios with polynomial models with different degree D. It is possible to see that the training error goes to 0 as the model becomes more complex. However, the test error has a characteristic U-shaped curve: on the left, where $D = 2$, the model is underfitting; as D grows the model tends to overfit; with $D = 14$ the model complexity seems "just right".

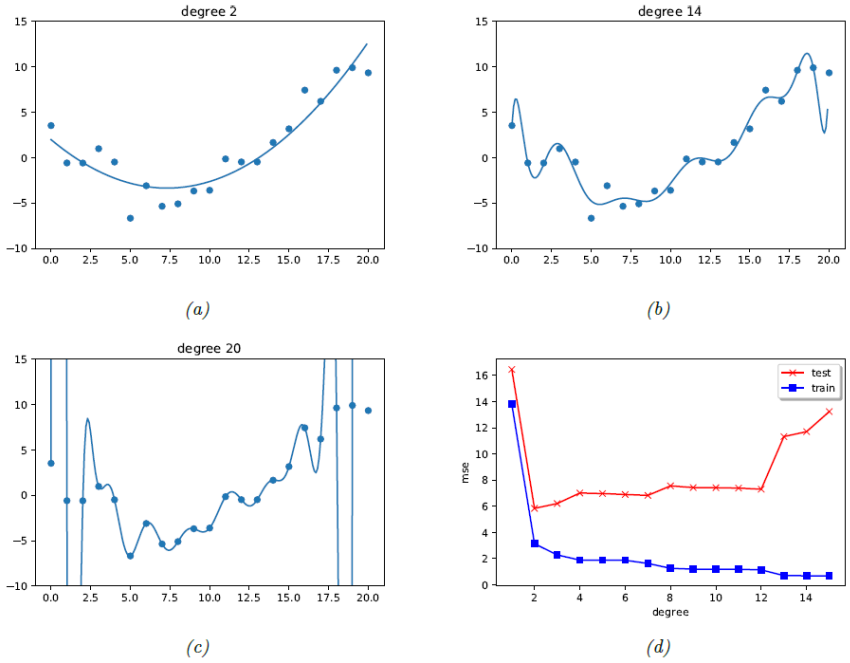


FIGURE 2.6: (a-c) Polynomials of degrees 2, 14 and 20 fit to 21 datapoints. (d) MSE vs degree. Credits: Murphy, 2022.

2.4 Gradient descent

In order to solve the optimization problem posed by empirical risk minimization (2.4), iterative methods are often used based on the compuation of the gradient. The method known as gradient descent is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. Given a real valued function of f with m parameters $\theta \in \mathbb{R}^m$, the partial derivative $\frac{\partial}{\partial \theta_i} f(x; \theta)$ measures the rate of change of f given a small variation of the parameter θ. The gradient of f in a point

$(x; \theta_1, \theta_2, \ldots, \theta_m)$ is defined as (if f is differentiable):

$$\nabla f(\theta) = \left[\frac{\partial}{\partial \theta_1} f(x;\theta), \frac{\partial}{\partial \theta_2} f(x;\theta), \ldots, \frac{\partial}{\partial \theta_m} f(x;\theta) \right]^\top \tag{2.26}$$

and provides the direction of the maximum local variation of the function. It follows that if

$$\theta_{t+1} = \theta_t - \eta \nabla f(x; \theta_t) \tag{2.27}$$

for a small enough η, then $f(x;\theta_{t+1}) \leq f(x;\theta_t)$. By repeating such step, it is possible to construct a monotonic sequence of points converging to the local minimum of f. When the gradient is computed over the whole set of training data $\mathcal{S}_{\text{train}}$, the step (2.27) is called *batch gradient descent* and represents a possible update of the parameters θ of the learned function towards the objective (2.4). In practice, variations to such an update rule are applied in order to avoid the problem of getting stuck into a local minimum. In the following, some variations will be briefly presented.

Steepest descent This method is effective for linear systems of the form

$$A\theta = b \tag{2.28}$$

where θ is an unknown vector, b is a known vector, and A is a known, square, symmetric, positive-definite matrix (i.e., for every non-zero vector x, $x^\top Ax > 0$). It is possible to show that the solution to the system (2.28) minimizes the quadratic form

$$f(\theta) = \frac{1}{2}\theta^\top A\theta - b^\top \theta + c \tag{2.29}$$

where c is a scalar constant. Because A is positive-definite, the surface defined by $f(\theta)$ is shaped like a paraboloid bowl in 3D. An example is shown in Figure 2.7. In the method of Steepest Descent, we start at an arbitrary point θ_0 and slide down to the bottom of the paraboloid. We take a series of steps $\theta_1, \theta_2, \ldots$ until we are satisfied that we are close enough to the solution θ^*. When we take a step, we choose the direction in which f which is the direction opposite $f'(\theta)$, i.e.,

$$-f'(\theta) = b - A\theta \tag{2.30}$$

Now let us introduce a few definitions. The error $e_i = \theta_i - \theta^*$ is a vector that indicates how far we are from the solution. The residual $r_i = b - A\theta_i$ indicates how far we are from the correct value of b. The residual is also equal to the direction of the steepest descent $r_i = -f'(\theta_i)$. The step can be now written as:

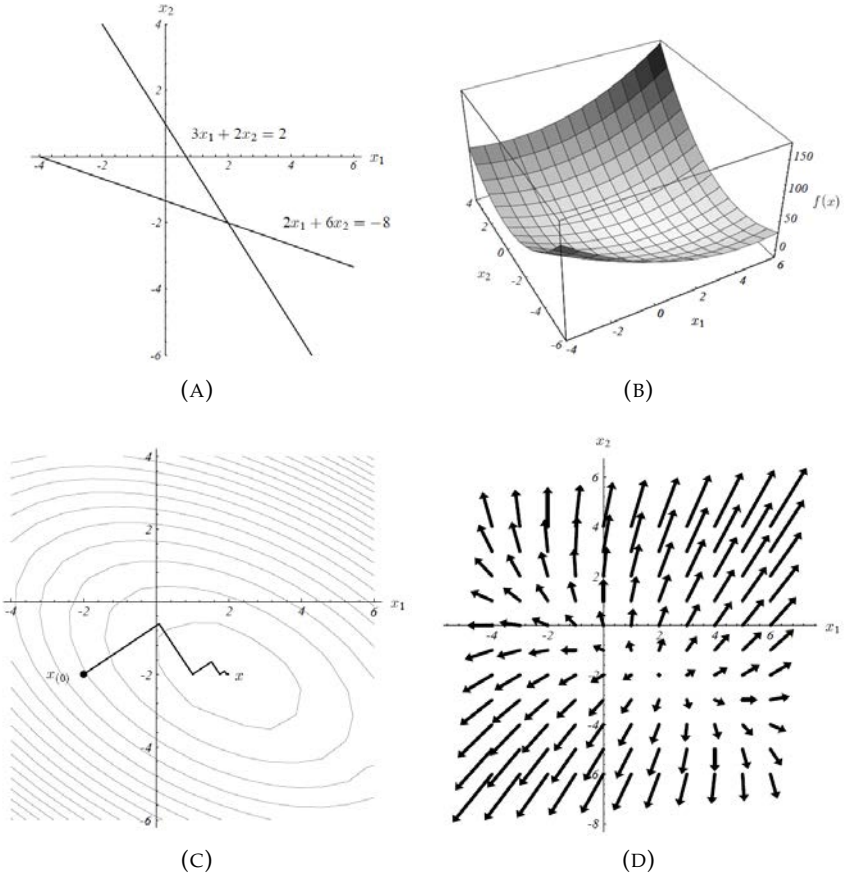
$$\theta_{t+1} = \theta_t + \eta r_i \tag{2.31}$$

A line search is a procedure that chooses η to minimize f along a line (e.g., in 3D the line at the intersection between the paraboloid f and the hyperplane along the steepest descent direction). From basic calculus, α minimizes f when the directional derivarive $\frac{\partial}{\partial \alpha} f(\theta_{i+1})$ is equal to zero. By the chain rule

$$\frac{\partial}{\partial \alpha} f(\theta_{i+1}) = f'(\theta_{i+1})^\top \frac{\partial}{\partial \alpha} \theta_{i+1} = f'(\theta_{i+1})^\top r_i \tag{2.32}$$

Setting this expression to zero, we find that α should be chosen so that $f'(\theta_{i+1})$ is orthogonal to r_i. The corresponding α is given by:

$$\alpha = \frac{r_i^\top r_i}{r_i^\top A r_i} \tag{2.33}$$



(A) (B)

(C) (D)

FIGURE 2.7: (A) Sample two-dimensional linear system. The solution lies at the intersection of the lines. (B) Graph of a quadratic form $f(x)$. The minimum point of this surface is the solution to $Ax = b$. (C) Contours of the quadratic form with the steps of the method. Each ellipsoidal curve has constant $f(x)$. Notice the zigzag path, which appears because each gradient is orthogonal to the previous gradient. (D) Gradient $f'(x)$ of the quadratic form. For every x, the gradient points in the direction of steepest increase of $f(x)$, and is orthogonal to the contour lines.

Conjugate gradient Steepest Descent often finds itself taking steps in the same direction as earlier steps (see Figure 2.7(C)). A better idea is to pick a set of conjugate directions $d_0, d_1, \ldots, d_{n-1}$. Two vectors d_i, d_j are said to be conjugate if

$$d_i^\top A d_j = 0 \tag{2.34}$$

Conjugate Gradient is a method of Conjugate Directions where the search directions are constructed by conjugation of the residuals $r_i = -f'(\theta_i)$. The steps are the following:

$$x_{i+1} = x_i + \alpha_i d_i$$
$$\alpha_i = \frac{r_i^\top r_i}{d_i^\top A d_i}$$

with the initial direction being

$$d_0 = r_0 = b - A\theta_0$$

and the next conjugate directions computed as

$$d_{i+1} = r_{i+1} + \beta_{i+1} d_i$$
$$r_{i+1} = r_i - \alpha_i A d_i$$
$$\beta_{i+1} = \frac{r_{i+1}^\top r_{i+1}}{r_i^\top r_i}$$

It is possible to show that Conjugate Gradients converges in a finite number n of iterations, where n is the number of conjugate gradients which are used as directions (see Figure 2.8)

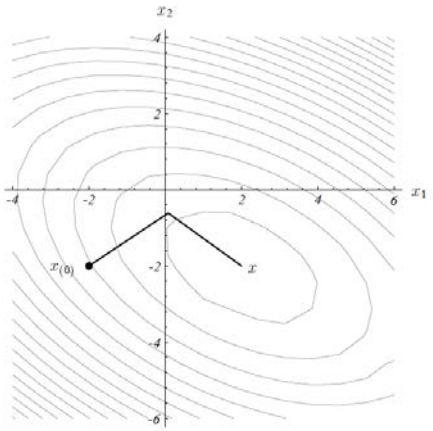


FIGURE 2.8: The steps computed by the method of Conjugate Gradients

In most cases, preconditioning is necessary to ensure fast convergence of the conjugate gradient method. Preconditioning is a technique for improving the condition number of a matrix. Suppose that M is a symmetric, positive-definite matrix that approximates A, but is easier to invert. We can solve $A\theta = B$ indirectly by solving

$$M^{-1}A\theta = M^{-1}b \tag{2.35}$$

If $\kappa(M^{-1}A) \ll \kappa(A)$ (where κ denotes the condition number), then we can iteratively solve Equation 2.35 more quickly than the original problem. Now consider that for every symmetric, positive-definite M there is a (not necessarily unique) matrix E that has the property that $EE^\top = M$ (such an E can be obtained, for instance, by

Cholesky factorization.). The system 2.35 can be transformed into the problem

$$E^{-1}AE^{-\top}\hat{\theta} = E^{-1}b \quad \hat{\theta} = E^{\top}x \tag{2.36}$$

Because $E^{-1}AE^{-\top}$ is symmetric and positive-definite $\hat{\theta}$ can be found by Steepest Descent or Conjugate Gradient.

The effectiveness of a preconditioner M is determined by the condition number of $M^{-1}A$, and occasionally by its clustering of eigenvalues. The problem remains of finding a preconditioner that approximates A well enough to improve convergence enough to make up for the cost of computing the product $M^{-1}r_i$ once per iteration.

For instance, one possibility is incomplete Cholesky preconditioning. Cholesky factorization is a technique for factoring a matrix A into the form $LL^{\top}$, where L is a lower triangular matrix. A is approximated by the product $\hat{L}\hat{L}^{\top}$ where $\hat{L}$ might be restricted to have the same pattern of nonzero elements as A; other elements of L are thrown away. To use $\hat{L}\hat{L}^{\top}$ as a preconditioner the solution to $\hat{L}\hat{L}^{\top}w = x$ is computed by backsubstitution (the inverse of $\hat{L}\hat{L}^{\top}$ is never explicitly computed). Conjugate gradient with Choleski preconditioner is used in Falkon/LogFalkon (section 4.5).

Stochastic gradient descent Stochastic gradient descent (often abbreviated SGD) is a stochastic approximation of gradient descent optimization, since it replaces the actual gradient calculated on the entire data set with an estimate calculated on a randomly selected subset of the data, called *mini-batch*. Especially in high-dimensional optimization problems this reduces the computational burden, achieving faster iterations in trade for a lower convergence rate (see Bottou, 2010). When used to minimize the above function $f(x;\theta)$, the (mini-batch) stochastic gradient descent method would perform the following updates:

$$\theta_{t+1} = \theta_t - \frac{\eta}{|\mathcal{B}|}\sum_{x\in\mathcal{B}}\nabla f(x;\theta_t), \tag{2.37}$$

The method works as follows:

1. Choose an initial vector of parameters θ_0, a learning rate η and a size b of the mini-batch
2. Repeat until an approximate minimum is obtained:
   - Randomly shuffle examples in the training set and assign them to different mini-batches $\mathcal{B}_1, \mathcal{B}_2, \ldots \mathcal{B}_g$
   - For each minibatch $\mathcal{B}_1, \mathcal{B}_2, \ldots \mathcal{B}_g$ compute the update (2.37)

Note that, according to the size b of the mini-batch, the parameters θ will be updated b times in the inner loop and at the end of the inner loop the training set of data will be used entirely. A full pass over the entire training set is called *epoch* and typically a maximum number of epochs is chosen as a stop criterion for the step 2 to reach an approximate solution. A popular variation of SGD introduces a term called momentum, which adds a fraction of the update vector of the past time step to the current update vector:

$$v_{t+1} = \gamma v_t + \eta\nabla_\theta f(\theta; x)$$
$$\theta_{t+1} = \theta_t - v_t$$

Essentially, when using momentum, you push a ball down a hill. The ball accumulates momentum as it rolls downhill, becoming faster and faster on the way (until it reaches its terminal velocity if there is air resistance). The same thing happens to parameter updates: the momentum term increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions. As a result, faster convergence with reduced oscillation is gained.

Adam Adaptive Moment Estimation (Adam, Kingma and Ba, 2014) is another method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients, Adam also keeps an exponentially decaying average of past gradients, similar to momentum. Whereas momentum can be seen as a ball running down a slope, Adam behaves like a heavy ball with friction, which thus prefers flat minima in the error surface.

The update is as follows:

$$
\begin{aligned}
m_t &= \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta f(\theta; x) \\
v_t &= \beta_2 v_{t-1} + (1-\beta_2)\left(\nabla_\theta f(\theta; x)\right)^2 \\
\hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\
\hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\
\theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t
\end{aligned}
$$

where m_t and v_t are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients respectively, whereas $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected version of the first and second moment estimates towards zero. The authors propose default values of $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. They show empirically that Adam works well in practice and compares favorably to other adaptive learning-method algorithms.

Adam is the method used to train neural networks in this work.

2.5 Classification

The problem of classification in machine learning can be described as follows. Suppose to be given a training set $\mathcal{S}_{\text{train}} = (x_i, y_i)_{i=1}^n$ with corresponding class labels. It is possible to represent the inputs with a *design matrix* X and the ouputs with a column vector Y:

$$
X = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,d} \\ x_{2,1} & x_{2,2} & \dots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \dots & x_{n,d} \end{bmatrix}, \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}
$$

where:

- each row represents a single sample $x_i \in \mathbb{R}^d$ with the associated label $y_i \in L$. In the binary case, $L = \{-1, 1\}$ (or $L = \{0, 1\}$ based on convenience), whereas the more general multiclass setting of T possible classes is obtained by letting $L = \{0, 1, ..., T-1\}$;

- each column is said to be a *feature*, corresponding to a specific dimension of the input space $\mathbb{R}^d$.

More specifically, binary classification can be formalized as the problem of finding a function $c : \mathbb{R}^d \to \{-1, 1\}$ (or $c : \mathbb{R}^d \to \{0, 1\}$, depending on the labels) that will correctly classify unseen data points given the training set $\mathcal{S}_{\text{train}}$. The classifier c is usually constructed by thresholding a real valued function $f : \mathbb{R}^d \to \mathbb{R}$ such that $f(x) > t_h$ is considered a positive label and $f(x) < t_h$ a negative one, with t_h being a suitable threshold. A common terminology is to call f a classifier and distinguish between its value and thresholded classification as the "soft" and "hard" classification, respectively. Classification thus focuses on finding a real valued function that will lead to useful predictions when thresholded in a simple manner. Like any machine learning algorithm, a good classifier should ensure at the same time fitting on the training data and generalization on new data.

In order to evaluate the accuracy of the classifier c, the following definitions are considered:

- the true error rate of a classifier c is

$$L(c) = \mathbb{P}\left(\{c(x) \neq y\}\right) \tag{2.38}$$

  which measures the probability of a mistaken classification

- the empirical error rate or training error rate is

$$\hat{L}(c) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(c(x_i) \neq y_i) \tag{2.39}$$

  which coincides with the 0-1 (or misclassification) loss described in section 2.3.1

2.5.1 Design of discriminative binary classifiers

The design of a classifier can be considered from a probabilistic perspective. Consider $x \in \mathbb{R}^d$ as a random variable being emitted by one of two sources C_0 and C_1 (or, equivalently, belonging to one of two different classes C_0, C_1).

Each source can be associated to an independent data generating process characterized by a probability distribution $p(x|C_0)$, $p(x|C_1)$ and a prior distribution $p(C_0)$, $p(C_1)$, i.e. prior knowledge about the classes (intuitively, the emitting probability of the two sources). Of interest is classifying x as being emitted from C_0 or C_1 (or, equivalently, belonging to class C_0 or C_1).

The most popular choice is to build a so-called *discriminative classifier*, which consists of fitting a model on the class posterior. In case of binary classification, it is possible to consider either $p(C_1|x)$ or $p(C_0|x)$ since they must sum up to 1.

Appying the Bayes theorem it is possible to easily derive a convenient expression for the soft classifier f (recall that f is the function to be thresholded):

$$p(C_1|x) = \frac{p(x|C_1)p(C_1)}{p(x|C_1)p(C_1) + p(x|C_0)p(C_0)} \tag{2.40}$$

$$= \frac{1}{1 + \frac{p(C_0)}{p(C_1)} \frac{p(x|C_0)}{p(x|C_1)}} \tag{2.41}$$

$$= \frac{1}{1 + \frac{p(C_0)}{p(C_1)} e^{-\log\left(\frac{p(x|C_1)}{p(x|C_0)}\right)}} \tag{2.42}$$

$$= \frac{1}{1 + \frac{p(C_0)}{p(C_1)} e^{-f(x)}} \tag{2.43}$$

where

$$f(x) = \log\left(\frac{p(x|C_1)}{p(x|C_0)}\right) \tag{2.44}$$

It is worth noting that introducing the exponential and the logarithm in the step (2.42) is useful both analytically (e.g., to directly work with the exponent of Gaussian distributions) and numerically (e.g., turning products into sums makes the computation less prone to overflow or underflow).

The Bayes classification rule c^* is

$$c^*(x) = \begin{cases} C_1 & \text{if } p(C_1|x) > p(C_0|x) \\ C_0 & \text{otherwise} \end{cases} \tag{2.45}$$

The Bayes rule is optimal, that is, if c is any other classification rule then the true error rate of the Bayes classifier c^* is smaller or equal to that of c, i.e., $L(c^*) \leq L(c)$.

Now if we consider the optimal Bayes decision rule in this case we get

$$p(C_1|x) > p(C_0|x) \tag{2.46}$$

$$\frac{1}{1 + \frac{p(C_0)}{p(C_1)} e^{-f(x)}} > 1 - \frac{1}{1 + \frac{p(C_0)}{p(C_1)} e^{-f(x)}}$$

$$\frac{p(C_0)}{p(C_1)} e^{-f(x)} < 1$$

$$f(x) > \log\left(\frac{p(C_0)}{p(C_1)}\right) \tag{2.47}$$

and then we obtain the following classifier $c : \mathbb{R}^d \to \{C_0, C_1\}$:

$$c(x) = \begin{cases} C_1 & \text{if } f(x) > \log\left(\frac{p(C_0)}{p(C_1)}\right) \\ C_0 & \text{otherwise} \end{cases} \tag{2.48}$$

If the probability distributions are known it is possible to compute:

- the decision boundaries Ψ, i.e. surfaces in the feature space $\mathbb{R}^d$ where ties occur among classifications, namely $\Psi = \{x : p(C_1|x) = p(C_0|x)\}$:

$$\Psi = \left\{ x : f(x) = \log\left(\frac{p(C_0)}{p(C_1)}\right) \right\} \tag{2.49}$$

- the decision regions $\mathcal{R}_0, \mathcal{R}_1$, i.e. regions of the feature space $\mathbb{R}^d$ partitioned by the decision boundary, corresponding to a specific classification:

$$\mathcal{R}_0 = \left\{ x : f(x) < \log\left(\frac{p(C_0)}{p(C_1)}\right) \right\} \quad \text{corresponding to classification } C_0 \tag{2.50}$$

$$\mathcal{R}_1 = \left\{ x : f(x) > \log\left(\frac{p(C_0)}{p(C_1)}\right) \right\} \quad \text{corresponding to classification } C_1 \tag{2.51}$$

An intuitive representation is provided in figure 2.9 for 1D, 2D and 3D Gaussians.

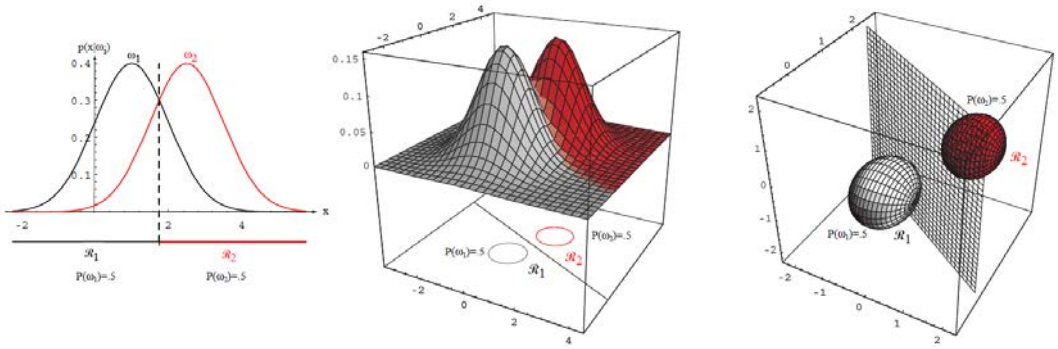


FIGURE 2.9: Decision boundaries and decision regions for 1D, 2D and 3D Gaussians. Credits: Hart, Stork, and Duda, 2000.

It is worth noting that the prior distributions provide a specific threshold for our decision rule. By considering the empirical class frequencies as the priors:

$$p(C_0) = \frac{n_0}{n_0 + n_1} \tag{2.52}$$

$$p(C_1) = \frac{n_1}{n_0 + n_1} \tag{2.53}$$

the decision rule (2.47) now reads

$$f(x) > \log\left(\frac{n_0}{n_1}\right) \tag{2.54}$$

Clearly, as the priors are changed, the decision boundary Ψ shifts. For instance, by increasing the assumed discrepancy $n_0 \gg n_1$ (i.e., data imbalances with class C_0 more frequent than class C_1), the decision region $\mathcal{R}_0$ corresponding to class C_0 increseas in size, while $\mathcal{R}_1$ conversely decreases its dimension.

The optimal decision rule is modified accordingly: the more one expects the next sample comes from class C_0 the more one is biased towards predicting C_0. On the

other hand, one may makes the "risky choice" of C_1 only when the confidence associated to the decision overcomes the biased expectation. See fig. 2.10 for a visual intuition.

The design of discriminative binary classifiers described so far can be easily generalized to the broader topic of "Bayes decision theory". In the simplest case, the main difference is that the optimal decision rule (2.46) should be formally expressed in terms of risks instead of conditional probabilities:

$$R(C_1|x) > R(C_0|x) \tag{2.55}$$

where $R(C_0|x), R(C_1|x)$ are now the risks (costs) associated to the decision C_0 and C_1, respectively. Usually, the following table 2.1 is considered at this point

TABLE 2.1: Risks matrix

	$\hat{y}=1$	$\hat{y}=0$
$y=1$	0	L_{FN}
$y=0$	L_{FP}	0

where L_{FP} is the arbitrary cost of a false positive (a.k.a. type-II error, i.e. misclassifying class C_1), L_{FN} is the arbitrary cost of a false negative (a.k.a. type-I error, i.e. misclassifying class C_0). According to this notation, the decision rule (2.55 becomes:

$$L_{FP}\ p(C_1|x) > L_{FN}\ p(C_0|x) \tag{2.56}$$

and the decision boundary Ψ (2.49) is shifted accordingly:

$$\Psi: \quad f(x) = \log\left(\frac{p(C_0)}{p(C_1)}\right) + \log\left(\frac{L_{FN}}{L_{FP}}\right) \tag{2.57}$$

$$= \log\left(\frac{n_0}{n_1}\right) + \log\left(\frac{L_{FN}}{L_{FP}}\right) \tag{2.58}$$

Note that increasing the cost $L_{FN} \gg L_{FP}$ has the same effect as increasing the discrepancy $n_0 \gg n_1$. The higher the cost the more confidence is required to take the risky decision.

Typically, in machine learning problems the probability distributions $p(x|C_1), p(x|C_0)$ are unknown, therefore it is not possible to follow the procedure described so far. However, sticking to a probabilistic perspective, it is possible to look at the maximum likelihood principle and settle an optimization problem to learn the soft classifier $f(x)$. This is the topic of the following section.

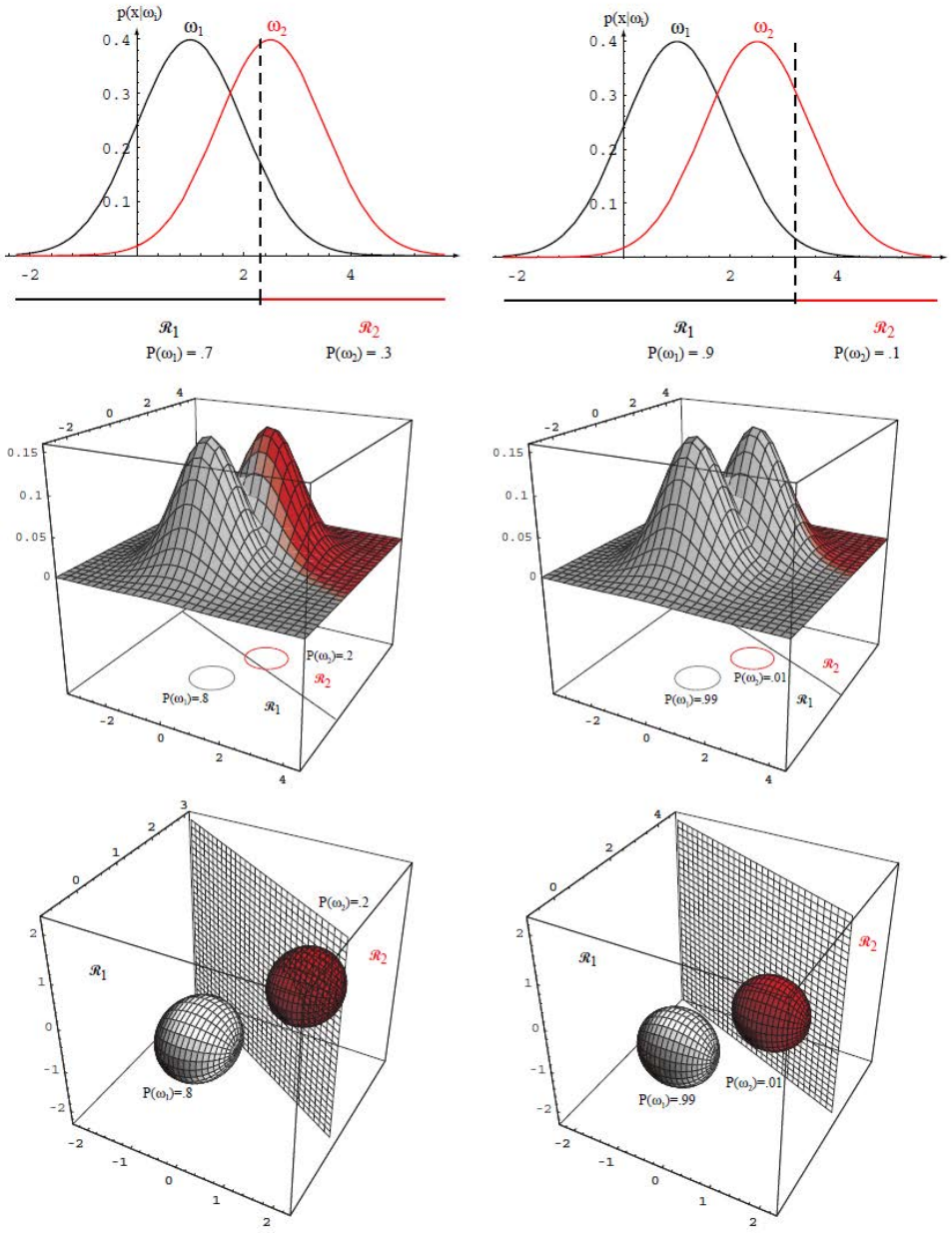


FIGURE 2.10: How priors affect the decision boundary in case of 1D, 2D and 3D Gaussians. Credits: Hart, Stork, and Duda, 2000.

2.5.2 Cost-sensitive learning

In a binary classification setting an alternative decomposition of eq. (2.2) may give further insights:

$$\mathbb{E}_{x,y}\left[\ell\left(y, f(x)\right)\right] = \int dx\, dy\; p(y)p(x|y)\,\ell\left(y, f(x)\right) \tag{2.59}$$

$$= p(C_0)\underbrace{\int dx\, p(x|C_0)\ell_{C_0}(f(x))}_{\mathbb{E}\left[\ell_{C_0}\right]} + p(C_1)\underbrace{\int dx\, p(x|C_1)\ell_{C_1}(f(x))}_{\mathbb{E}\left[\ell_{C_1}\right]} \tag{2.60}$$

where:

- ℓ_{C_0} is the cost (or risk) associated to class C_0 or type-I error, i.e. the cost of misclassifying points belonging to C_0 according to the loss function ℓ:

$$\ell_{C_0}(f(x)) = \ell(0, f(x)) \tag{2.61}$$

- ℓ_{C_1} is the cost (or risk) associated to class C_1 or type-II error, i.e. the cost of misclassifying points belonging to C_1 according to the loss function ℓ:

$$\ell_{C_1}(f(x)) = \ell(1, f(x)) \tag{2.62}$$

- $p(C_0), p(C_1)$ are the prior probabilties associated to the class, which sum up to 1.

Now it is easy to see that the objective function can be interpreted as a weighted sum of the expected costs associated to the two different classes, with the weights being the prior probabilities of the two classes. For this reason, it is possible to interpret empirical risk minimization as a cost-sensitive learning procedure.

2.5.3 Handling data imbalances

Real world data is inevitably noisy and unlikely to be perfectly separable by a simple rule. Adding to the difficulties of noisy data is that the training data may be imbalanced. Imbalances occur when the relative proportions of some parts of the training data are different than what we wish to train on. There is indeed a variety of real cases which require careful considerations since final users are mainly (or uniquely) interested in models and performances which are sensitive to the rare events. Broadly speaking, data imbalance occurs when:

- the actual distribution of real data is misrepresented by the training set because of implicit or explicit faults in the sampling procedure;
- the cost of mislabeling a class is higher with respect to other classes.
- the multiclass classification problem is transformed into a series of binary tasks: each "one vs. all" classifier obtains its positive examples from a single class and labels the rest of the training data as negative. As such, the imbalance between positive and negative samples increases with the number of classes.

Several methods have been proposed in the literature to handle data imbalances (see Branco, Torgo, and Ribeiro, 2016 for a thorough overview). In the following, the most popular strategies will be briefly presented.

Let n_0 and n_1 be the number of samples of class C_0 and C_1, respectively, so that $n = n_0 + n_1$. Without loss of generality, assume that a data imbalance exists because there are too few training examples of class 1:

$$n_1 \ll n_0 \tag{2.63}$$

There are a number of modifications one might make to the objective (2.5) to try to correct this imbalance:

- Oversample (sample with replacement) the minority points of class C_1 to obtain the desired quantity αn_1 of them.

- Add different weights to the cost of each class ("reweighting")

$$\beta(y) = \begin{cases} \beta_0 & \text{if} \quad y = 0 \\ \beta_1 & \text{if} \quad y = 1 \end{cases} \tag{2.64}$$

  with the explicit result being

$$\beta(y)\, \ell\left(y, f(x)\right) = \beta_0\, \ell_{C_0}\left(f(x)\right) + \beta_1\, \ell_{C_1}\left(f(x)\right) \tag{2.65}$$

- Use a smaller regularization term on coefficients that correspond to training samples of class C_1.
- Increase the magnitude of the positive labels to some $\delta > 1$.

In the following sections, some intuitions will be provided of why in general the first two methods can be considered useful and almost equivalent.

It is worth noting that by restricting to the problem of regularized least squares (RLS, see eq. (2.6)), it turns out that the first three alternatives can be perfectly equivalent (Paskov, 2010). On the contrary, the fourth alternative may have no meaning if other loss functions are considered.

In this thesis, there will be a focus on the problem of binary classification with reweighting.

2.5.4 Maximum likelihood principle

The maximum likelihood principle is an alternative principle to empirical risk minimization towards building a classifier.

Consider the following probabilistic model based on the Bernoulli distribution:

$$p(y|x) = \begin{cases} p & \text{if} \quad y = 1 \\ 1 - p & \text{if} \quad y = 0 \end{cases} \tag{2.66}$$

Given a dataset $\mathcal{S} = (x_i, y_i)_{i=1}^n$ of i.i.d. samples, and a parametric model for p, say p_w, the maximum likelihood principle provide the best parameters w according to the data:

$$\arg\max_w \prod_{i=1}^{n} (p_w)^{y_i} \left(1 - p_w\right)^{1-y_i} \tag{2.67}$$

In order to efficiently settle an optimization problem, the negative log-likelihood is minimized:

$$\begin{aligned} \mathrm{NLL}(\mathcal{D}; w) &= -\log \prod_{i=1}^{n} (p_w)^{y_i} \left(1 - p_w\right)^{1-y_i} \\ &= -\sum_{i=1}^{n} y_i \log p_w + (1 - y_i) \log(1 - p_w) \end{aligned} \tag{2.68}$$

According to the previous general notation, recall that p is now equivalent to $p(C_1|x)$. Thus, it is possible to resort to the same probabilistic model of (2.43)

$$p = \frac{1}{1 + e^{-f_w(x)}} \tag{2.69}$$

where $f_w(x)$ is an arbitrary function with parameters w (e.g., a linear model or a neural network) defining at the same time the soft classifier to be learned and the probabilistic model. More specifically, $f_w(x)$ corresponds now to the ratio $\frac{p(C_1|x)}{p(C_0|x)}$, i.e. there is no control over the priors.

Plugging (2.69) into (2.68) one gets the cross entropy loss (presented in section 2.3.1):

$$\begin{aligned}\mathrm{NLL}(\mathcal{D};w) &= -\sum_{i=1}^{n} y_i \log\left(\frac{1}{1+e^{-f_w(x_i)}}\right) + (1-y_i) + \log\left(1-\frac{1}{1+e^{-f_w(x_i)}}\right)\Bigg] \\ &= \sum_{i=1}^{n} y_i \underbrace{\log\left(1+e^{-f_w(x_i)}\right)}_{\ell_{C_1}} + (1-y_i)\underbrace{\log\left(1+e^{f_w(x_i)}\right)}_{\ell_{C_0}} \end{aligned} \tag{2.70}$$

Now consider the optimization problem

$$\arg\min_{w} \mathrm{NLL}(w) \tag{2.71}$$

which is - according to the Bayes decision theory - an approximate solution to the expected risk (Bayes estimator, see eq. (2.1)) through a cost-sensitive procedure where the loss function can be seen as a surrogate function of the 0-1 loss (see sect. 2.3.1). Note that the use of surrogate loss functions is necessary to learn a parametric classifier through a gradient descent procedure (see sect. 2.4).

Given the loss function it is possible to equivalently derive back the underlying probabilistic model by deriving the target function. With slight abuse of notation let $f = f(x)$ be a real number for every $x \in \mathbb{R}^d$):

$$\arg\min_{f\in\mathbb{R}} \quad \int dp(y|x)\ \left[y\log\left(1+e^{-f}\right) + (1-y)\log\left(1+e^{f}\right)\right]$$

$$\arg\min_{f\in\mathbb{R}} \quad \underbrace{p\ \log\left(1+e^{-f}\right) + (1-p)\log\left(1+e^{f}\right)}_{L}$$

Now consider $\frac{dL}{df} = 0$:

$$\begin{aligned} p\,\frac{1}{1+e^{-f}}\left(-e^{-f}\right) + (1-p)\,\frac{1}{1+e^{f}}\,e^{f} &= 0 \\ \log\left(p\,e^{-f}\right) &= \log\left(1-p\right) \end{aligned} \tag{2.72}$$

The target function of the cross entropy loss is then:

$$f = \log\left(\frac{p}{1-p}\right) \tag{2.73}$$

with the corresponding probabilistic model

$$p = \frac{1}{1 + e^{-f}} \tag{2.74}$$

Consider now the method of adding a weight to the cost of each class (reweighting to handle data imbalances)

$$\beta(y) = \begin{cases} \beta_0 & \text{if} \quad y = 0 \\ \beta_1 & \text{if} \quad y = 1 \end{cases} \tag{2.75}$$

the loss function is now

$$\ell(\mathcal{D}; w) = \sum_{i=1}^{n} y_i \underbrace{\beta_1 \ \log\left(1 + e^{-f_w(x_i)}\right)}_{\ell_{C_1}} + \ (1 - y_i) \underbrace{\beta_0 \ \log\left(1 + e^{f_w(x_i)}\right)}_{\ell_{C_0}} \tag{2.76}$$

and the corresponding target function can be obtained as

$$\arg\min_{f \in \mathbb{R}} \quad \int dp(y|x)\, \beta(y) \ \left[y \log\left(1 + e^{-f(x;w)}\right) + (1 - y) \log\left(1 + e^{f(x;w)}\right) \right]$$

$$\arg\min_{f \in \mathbb{R}} \quad \beta_1\, p \ \log\left(1 + e^{-f}\right) + \beta_0\, (1 - p) \log\left(1 + e^{f}\right)$$

with the final result being

$$\begin{aligned} & -\beta_1\, p\, e^{-f} + \beta_0\, (1 - p) = 0 \\ & \log\left(\beta_1\, p\, e^{-f}\right) = \log\left(\beta_0 (1 - p)\right) \\ & f = \log\left(\frac{\beta_1\, p}{\beta_0\, (1 - p)}\right) \quad : \text{target function} \qquad (2.77) \\ & f = \log\left(\frac{p}{(1 - p)}\right) + \log\left(\frac{\beta_1}{\beta_0}\right) \\ & p = \frac{1}{1 + \dfrac{\beta_1}{\beta_0}\, e^{-f}} \quad : \text{probabilistic model} \qquad (2.78) \end{aligned}$$

Differently from the previous case, the new target function (2.77) shows that the role of the added weights corresponds to the role of the priors on the underlying probabilistic model (2.43). As a consequence, by letting $\beta_1 = n_0$ and $\beta_0 = n_1$ it is possible to introduce the empirical priors over the two classes in the cost-sensitive learning procedure:

$$\frac{p(C_0)}{p(C_1)} = \frac{n_0}{n_1} = \frac{\beta_1}{\beta_0} \tag{2.79}$$

This explicit change in the probabilstic model is confirmed by the fact that now the decision rule is:

$$f > 0 \implies \log\left(\frac{p}{(1-p)}\right) > \log\left(\frac{\beta_1}{\beta_0}\right) \tag{2.80}$$

Oversampling can be seen as a reweighting procedure, thereby achieving the same results. Just assume to oversample each data point of the minority class with the same quantity $n_0/n_1 = \beta_1/\beta_0$. Clearly, this corresponds to eq. (2.76) where the value of the loss function is just scaled accordingly:

$$\ell(\mathcal{D}) = \sum_{i=1}^{n} y_i \underbrace{\frac{\beta_1}{\beta_0} \log\left(1 + e^{-f_w(x_i)}\right)}_{\ell_{C_1}} + (1 - y_i) \underbrace{\log\left(1 + e^{f_w(x_i)}\right)}_{\ell_{C_0}} \tag{2.81}$$

leading to an equivalent optimization problem but with different weights in the loss function.

2.6 Anomaly detection

Anomaly detection refers to the problem of finding patterns in data that do not conform to standard behavior. Most commonly, anomaly detection is synonymous with outlier detection and the identification of Out-of-Distribution (OoD) examples.

The detection of anomalous data is a fundamental problem in machine learning with applications to many domains, such as fraud detection for credit cards, insurance, or health care, intrusion detection for cyber-security, fault detection in safety critical systems, and military surveillance for enemy activities.

Three broad categories of anomaly detection techniques exist (Chandola, Banerjee, and Kumar, 2009):

- Supervised Anomaly Detection. Techniques trained in supervised mode assume the availability of a training data set that has labeled instances for normal as well as anomaly classes. A typical approach in such cases is to build a predictive model for normal vs. anomaly classes. Any unseen data instance is compared against the model to determine which class it belongs to. There are two major issues that arise in supervised anomaly detection. First, the anomalous instances are far fewer compared to the normal instances in the training data. Issues that arise due to imbalanced class distributions have been discussed in section 2.5.3. Second, obtaining accurate and representative labels, especially for the anomaly class is usually challenging. Other than these two issues, the supervised anomaly detection problem is similar to building predictive models.

- Semisupervised Anomaly Detection. Techniques that operate in a semisupervised mode assume that the training data has labeled instances only for the normal class. Since they do not require labels for the anomaly class, they are more widely applicable than supervised techniques. The typical approach used in such techniques is to build a model for the class corresponding to normal behavior, and use the model to identify anomalies in the test data.

- Unsupervised Anomaly Detection. Techniques that operate in unsupervised mode do not require training data, and thus are most widely applicable. The techniques in this category make the implicit assumption that normal instances are far more frequent than anomalies in the test data. Many semisupervised

techniques can be adapted to operate in an unsupervised mode by using a sample of the unlabeled data set as training data. Such adaptation assumes that the test data contains very few anomalies and the model learned during training is robust to these few anomalies.

The approach discussed in this thesis is similar to a supervised anomaly detection case, as will be clarified later in chapter 3.

2.7 Learning non-linear functions

Two different machine learning models capable of approximating non-linear functions have been used throughout the experiments: neural networks and kernel methods. They will be presented in the following sections.

2.7.1 Neural networks

Neural networks are among the most popular machine learning models. They belong to the family of parametric models, i.e., models with a fixed parametric form for the prediction function.

Neural networks can be viewed as a set of layers, each composed by a number of computational units called neurons. Each layer applies a function to the output of the previous layer. As a consequence, the functions are nested to form a chain of representations - called *hidden layers* - up to the last layer - called *output layer*

$$f(x) = o^{(d)}\underbrace{(\dots(h^{(3)}(h^{(2)}(h^{(1)}(x)))))}_{\text{hidden layers}} \tag{2.82}$$

where:

- d is the depth of the network, i.e., the number of layers
- $h^{(1)}$ is the first hidden layer, $h^{(2)}$ is the second hidden layer and so on. Each of them applies a non linear transformation of the input. More precisely, each hidden layers computes an affine transformation of the input, then applies a non linear function called activation function

$$\begin{aligned}
&\text{Layer 1:} \quad h^{(1)} = g(W^{(1)}x + b^{(1)}) \\
&\text{Layer 2:} \quad h^{(2)} = g(W^{(2)}h^{(1)} + b^{(2)}) \\
&\text{Layer 3:} \quad h^{(3)} = g(W^{(3)}h^{(2)} + b^{(3)}) \\
&\qquad\vdots
\end{aligned}$$

  where W is a matrix of weights and b is an array of biases to be learned (with different sizes per layer, depending on the number of neurons per layer), g is the non linear activation function.

- $o^{(d)}$ is the output layer, which applies an affine transformation similar to the hidden layers plus eventually another function depending on the task addressed by the neural network.

Once the architecture of the network has been chosen (i.e., the number of layers, the number of neurons per layer, the activation function and the output layer), an iterative optimizer like the ones presented in section 2.4 are used to adjust all the weights

and biases of the network in order to find a solution to the minimization problem (2.4). The computation of the gradients and the contribution of each weight to the gradients are easily computed through a popular algorithm called *Back Propagation* (Rumelhart, Hinton, and Williams, 1986). Popular choices for the activation functions are the ReLU $g(z) = \max(0, z)$, the sigmoid function $g(z) = 1/(1 + e^{-z})$ or the hyperbolic tangent $g(z) = \tanh(z)$.

Neural networks are loosely inspired by the computations happening in the human brain. Each hidden layer can be viewed as composed by many single computational units which resemble neurons. They apply a weighted sum of the inputs and outputs a signal depending on the magnitude of the weighted sum. This is in accordance to a very simple model of neural computation, shown in figure 2.11.

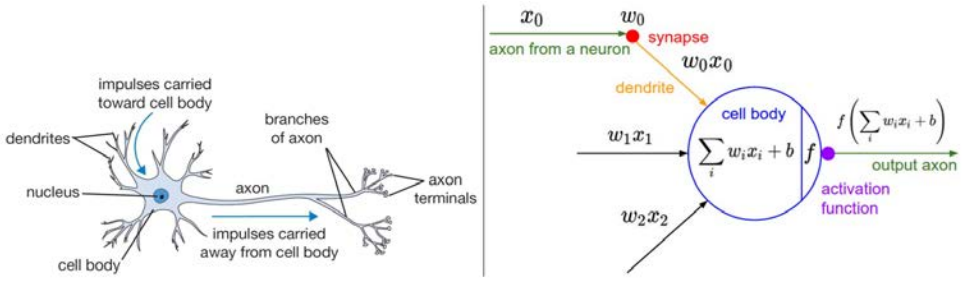


FIGURE 2.11: A cartoon drawing of a biological neuron (left) and its computational model inside a neural network. Credits: https://cs231n.github.io/neural-networks-1/

Instead of thinking to neural networks as models of the brain, it is more appropriate to view them as universal function approximators. It has been shown that a neural network with a linear output layer and at least one hidden layer that has an activation function among those listed above can approximate a continuous function on a closed and limited set of $\mathbb{R}^n$. The error may be arbitrary if a network with high depth is used (Cybenko, 1989). This theorem provides a very general theoretical justification for the functioning of neural networks, but does not specify any details concerning the architecture to be used in order to obtain some guarantees in the approximation. The choices concerning the number of layers, the number of weights for each layer, the type of connectivity between the neurons, activation functions, output layer, etc. always derive from experimental rather than theoretical results. For this reason, hyperparameter tuning is an important challenge to be taken into account when using neural networks.

Typically, when the number of layers is large (e.g., $d > 5$), the network is called deep networks and the corresponding learning procedure is called deep learning. One of the most powerful aspect of deep learning compared to other machine learning approaches is the so called *representation learning*. Deep enough neural networks can be fed with raw input data and learn an efficient representation of the input towards their objective. On the contrary, other machine learning methods (kernel methods included, as will be explained in the next section) require a preliminary feature extraction from the raw input data in order to address the final problem in an efficient way.

2.7.2 Kernel methods

Kernel methods are nonparametric models, i.e., do not assume a fixed parametric form for the prediction function, but instead try to estimate the function itself (rather

than the parameters) directly from data. The key idea is that the function value is observed at a fixed set of N points, namely $y_i = f(x_i)$ for $i = 1 : n$, where f is the unknown function, so to predict the function value at a new point, say x', one just has to compare how "similar" x' is to each of the n training points, and then can predict that $f(x')$ is some weighted combination of the $\{f(x_i)\}$ values. Thus there is need to "remember" the entire training set, $\mathcal{S} = \{(x_i; y_i)\}$, in order to make predictions at test time — it is not possible to "compress" $\mathcal{S}$ into a fixed-sized parameter vector. The weights that are used for prediction are determined by the similarity between x' and each x_i, which is computed using a special kind of function known as *kernel function*, $k(x, x')$.

A (positive definite) kernel is a real-valued function of two arguments $k(x, x') \in \mathbb{R}$, typically symmetric ($k(x, x') = k(x', x)$) and non-negative ($k(x, x') \geq 0$) which can be interpreted as a measure of similarity between two input points. The basic idea of using kernels is the following: if $k(x, x')$ is large, meaning the inputs are similar, then the output of the function is expected to be similar as well, so $f(x) \approx f(x')$.

The prediction function of kernel methods is typically of the form:

$$f(x; w, \mathcal{S}) = \sum_{i=1}^{n} w_i k(x, x_i) \tag{2.83}$$

which is a linear combination of the kernel function computed on the training data points $\{x_i\}$. The weights $\{w_i\}$ are adjusted according to a learning procedure.

Provided with enough data kernel based models are universal, in the sense that they can recover any continuous function (Micchelli, Xu, and Zhang, 2006).

The most widely used kernel for real-valued inputs is the squared exponential kernel (SE kernel), also called the exponentiated quadratic, Gaussian kernel or RBF kernel. It is defined by

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right) \tag{2.84}$$

where σ is called the bandwidth parameter and corresponds to the length scale of the kernel, i.e., the distance over which we expect differences to matter. The RBF kernel measures similarity between two vectors in $\mathbb{R}^d$ using (scaled) Euclidean distance.

Another interpretation of kernel methods may arise when considering the regularized least squares problem (eq. 2.6):

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left(y_i - w^\top x_i\right)^2 + \lambda \|w\|^2$$

with solution

$$w = \left(X^\top X + \lambda n I\right)^{-1} X^\top Y \tag{2.85}$$

where X is the $n \times d$ input matrix, Y is the $n \times 1$ output vector, λ is the L2 regularization parameter, I is the $d \times d$ identity matrix.

According to the representer theorem, the solution of the problem depends on the input points only through inner products:

$$f(x) = \sum_{i=1}^{n} c_i x_i^\top x \tag{2.86}$$

with the coefficients $\{c_i\}$ computed as $\left(XX^\top + \lambda n I\right)^{-1} Y$. Kernel methods can be

seen as replacing the inner product with a more general function $k(x, x')$. In this case, the representer theorem becomes

$$f(x) = \sum_{i=1}^{n} c_i k(x, x_i) \tag{2.87}$$

and the computation of the coefficients $\{c_i\}$ becomes $(K_{X,X} + \lambda n I)^{-1} Y$ where $K_{X,X}$ is the so called kernel matrix (or Gram matrix) and contains the kernel function computed on all the input points.

$$K_{X,X} = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \dots & k(x_1, x_n) \\ k(x_2, x_1) & k(x_2, x_2) & \dots & k(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n, x_1) & k(x_n, x_2) & \dots & k(x_n, x_n) \end{bmatrix} \tag{2.88}$$

The corresponding problem is called kernel ridge regression (KRR):

$$\min_{c \in \mathbb{R}^n} \frac{1}{n} \| K_{X,X}\, c - Y \| + \lambda c^\top K_{X,X}\, c \tag{2.89}$$

where c is the vector of n coefficients $[c_1, c_2, \dots, c_n]^\top$.

Chapter 3

The problem of new physics searches

The paradigm at the basis of the experimental High Energy Physics programme can be summarized as the process of looking for discrepancies between two sets of data: the experimental data collected from an experimental apparatus (e.g., the Large Hadron Collider and its detectors, described in section 1.2) and the reference data (e.g., generated by a software simulator according to a given physical model of interest).

We will refer to the two sets of data as:

- the reference sample $\mathcal{R} = \{x_i\}_{i=1}^{\mathcal{N}_\mathcal{R}}$, produced by the software simulator
- the data sample $\mathcal{D} = \{x_i\}_{i=1}^{\mathcal{N}_\mathcal{D}}$, produced by the experiment

Each of the two process has its own corresponding statistical distribution. The statistical distribution of the reference sample can be predicted on the basis of the physical laws that constitute the reference model (the Standard Model, briefly presented in section 1.2). On the other side, the true statistical distribution of the data sample is unknown and it includes potential new physics effect which are not predicted by the reference model.

The goal is to test the reference model distribution against the actual data with a statistical hypothesis test (described in the next section 3.1) to determine if the experimental dataset follows the reference model or if it contains small departures intepretable as new physics clues.

Broadly speaking, strategies to search for traces of new physics in experimental data can be either model-dependent or model-independent.

- Model-dependent strategies are targeted towards specific signatures determined, for example, by a candidate new theory. This is a traditional approach which has been very successful in cases where strong theoretical priors were available, such as in the discovery of the Higgs boson at the Large Hadron Collider (ATLAS, 2012). While very powerful to exclude particular hypotheses, this approach is however largely insensitive to other types of departures beside the one into consideration. While it is certainly possible to perform many analyses exploring different alternatives, there is a limitation from a theoretical perspective, given by the ability of formulating new physical laws and, from a phenomenological point of view, by the sheer number of possible discrepancies.
- Model-independent strategies should ideally be sensitive to any generic departure from the reference model. In practice, this is a really challenging strategy

given the physical constraints stated before (i.e., the "similarity" of new physics to reference model).

The work developed in this thesis can be classified as a model independent strategy.

The main challenge of new physics searches stems from the fact that the experimental data distribution including new physics effects will be extremely "similar" to the reference one. This is expected because of existing constraints on new physics. Notice that "similar" does not mean that the effect of new physics cannot be large. However, if it is large it will be localized in a low-probability region of the space of observations where only a a small fraction of the events is present. Alternatively, the effect can be spread in a large region of the data space, but in this case it will be a small modification of the reference distribution.

Thus, the problem is that the prior knowledge indicates that the vast majority of the collected events will agree with the reference model but at the same time this prior knowledge is insufficient to know where to look for discrepancies.

3.1 Hypothesis testing

Design of experiments is a cornerstone of the practice of statistics, with applications in virtually all areas of research. The goal is to design an experiment in order to confirm or reject a hypothesis.

Whenever you see references to statistical significance it is typically in the context of the classical statistical inference "pipeline" (see Figure 3.1). Hypothesis tests, also called significance tests, are ubiquitous in the traditional statistical analysis of published research. Their purpose is to help you learn whether random chance might be responsible for an observed effect. The need for an hypothesis is a way to protect researchers from being fooled by random chance.

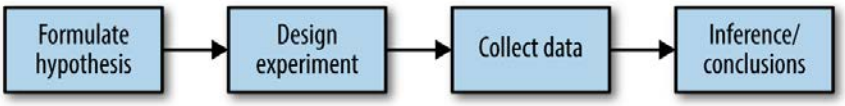


FIGURE 3.1: Classical statistical inference pipeline. Credits: Bruce, Bruce, and Gedeck, 2020

Hypothesis tests use the following logic: "Given the human tendency to react to unusual but random behavior and interpret it as something meaningful and real, in the experiments will be required a proof that the difference between groups is more extreme than what chance might reasonably produce." (Bruce, Bruce, and Gedeck, 2020). This involves a baseline assumption that the two possible alternatives (hypothesis true/false) are equivalent, and any difference between the alternatives is due to chance. This baseline assumption is termed the *null hypothesis*. The objective of a statistical hypothesis test is to quantify the compatibility of the data with the null hypothesis. A well defined hypothesis test requires an alternative hypothesis, which is accepted when rejecting the null hypothesis.

The hypothesis test is phrased in terms of a statistical model $\mathfrak{F}$, which is a set (family) of distributions (or densities or regression functions). A parametric model is a set $\mathfrak{F}$ that can be parameterized by a finite number of parameters

$$\mathfrak{F} = \left\{ f(x;\theta),\, \theta \in \Theta \right\} \tag{3.1}$$

where θ is an unknown parameter (or vector of parameters) that can take values in the parameter space Θ.

For example, if we assume that the data come from a Normal distribution, then the model is:

$$\mathfrak{F} = \left\{ f(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right), \quad \mu \in \mathbb{R},\, \sigma > 0 \right\} \tag{3.2}$$

where now $\theta = \{\mu, \sigma\}$ and $\Theta = \mathbb{R} \times \mathbb{R}^+$. $\mathfrak{F}$ represents the family of all the possible (univariate) Normal distributions.

Now suppose to partition the parameter space Θ into two disjoint sets Θ_0 and Θ_1 and to test

$$H_0 : \theta \in \Theta_0 \quad \text{versus} \quad H_1 : \theta \in \Theta_1 \tag{3.3}$$

where H_0 is the null hypothesis, whereas H_1 is the alternative hypothesis. Let x be a random variable and let $\mathcal{X}$ be the range of x. We test a hypothesis by finding an appropriate subset of outcomes $R \subset \mathcal{X}$ called the rejection region. If $x \in R$ the null hypothesis is rejected, otherwise, the null hypothesis is retained:

$$\begin{aligned} x \in R &\implies \text{reject } H_0 \\ x \notin R &\implies \text{retain } H_0 \end{aligned}$$

Usually, the rejection region R is of the form

$$R = \left\{ x : T(x) > c \right\} \tag{3.4}$$

where T is a test statistic and c is a critical value. The problem in hypothesis testing is to to find an appropriate test statistic T and an appropriate critical value c.

There are two types of errors we can make. Rejecting H_0 when H_0 is true is called a type I error. Retaining H_0 when H_1 is true is called a type II error.

The power function of a test with rejection region R is defined by

$$\beta(\theta) = \mathbb{P}_\theta(x \in R) \tag{3.5}$$

The size of a test is defined to be

$$\alpha = \sup_{\theta \in \Theta_0} \beta(\theta) \tag{3.6}$$

A test is said to have level α if its size is less than or equal to α.

A hypothesis of the form $\theta = \theta_0$ is called a simple hypothesis. A hypothesis of the form $\theta > \theta_0$ or $\theta < \theta_0$ is called a composite hypothesis. It would be desirable to find the test with highest power under H_1, among all size α tests. Such a test, if it exists, is called most powerful.

Reporting "reject H_0" or "retain H_0" is not very informative. Instead, we could ask, for every α, whether the test rejects at that level. Generally, if the test rejects at level α it will also reject at level $\alpha' > \alpha$. Hence, there is a smallest α at which the test rejects and we call this number the p-value.

Suppose that for every $\alpha \in (0,1)$ there is a size α test with rejection region R_α. Then, the p-value is defined as

$$p_{\text{val}} = \inf\left\{\alpha : T(X^n) \in R_\alpha\right\} \tag{3.7}$$

That is, the the p-value is the smallest level at which we can reject H_0.

More informally, the p-value is the probability, given a chance model, of observing a value of the test statistic the same as or more extreme than what was actually observed.

Consider an observed test-statistic t from unknown distribution T. Then the p-value p is the probability of observing a test-statistic value at least as "extreme" as t if null hypothesis were true. That is:

- $p_{\text{val}} = \mathbb{P}(T \geq t|H_0)$ for a one-sided right-tail test,
- $p_{\text{val}} = \mathbb{P}(T \leq t|H_0)$ for a one-sided left-tail test,
- $p_{\text{val}} = 2\min\{\mathbb{P}(T \geq t|H_0), \mathbb{P}(T \leq t|H_0)\}$ for a two-sided test.

If the p-value is very small, then either the null hypothesis is false or something unlikely has occurred. In a formal significance test, the null hypothesis H_0 is rejected if the p-value is less than a pre-defined threshold value α, which is referred to as the alpha level or significance level. Table 3.1 shows the evidence scale used by researchers.

TABLE 3.1: Typical p-values

p-value	evidence
< .01	very strong evidence against H_0
.01 - .05	strong evidence against H_0
.05 - .10	weak evidence against H_0
> .1	little or no evidence against H_0

High-energy physics requires even lower p-values to announce evidence or discoveries. The threshold for "evidence of a particle," corresponds to $p_{\text{val}} = 3 \times 10^{-3}$, and the standard for "discovery" is 3×10^{-7}. The Higgs boson discovery was indeed announced with a significance of five sigma. In short, five-sigma corresponds to a p-value, or probability, of 3×10^{-7}, or about 1 in 3.5 million (as later explained). This is not the probability that the Higgs boson does or doesn't exist; rather, it is the probability that if the particle does not exist, the data that CERN scientists collected at the LHC would be at least as extreme as what they observed. The reason for such stringent standards is that several events with lower p-values have later turned out to be statistical anomalies, and physicists are loath to declare discovery and later find out that the result was just a blip (Lamb, 2012).

The p-value is a function of the chosen test statistic T and is therefore a random variable. If the distribution of the test statistic under the null hypothesis is known, then p-value can be computed as:

$$p_{\text{val}} = \int_{t_{\text{obs}}}^{\infty} p(t|H_0)dt = 1 - \Gamma(t_{\text{obs}}) \tag{3.8}$$

where $\Gamma(\cdot)$ is used to denote the cumulative density function of $p(t|H_0)$.

Typically in practice, the statistical significance is expressed in terms of a z-score given by a number of sigmas

$$z = \Phi^{-1}(1 - p_{\text{val}}) = \Phi^{-1}(\Gamma(t_{\text{obs}})) \tag{3.9}$$

where $\Phi(\cdot)$ represents the cumulative density function of the standard Gaussian distribution. Figure 3.2 provides a visual interpretation of the p-value.

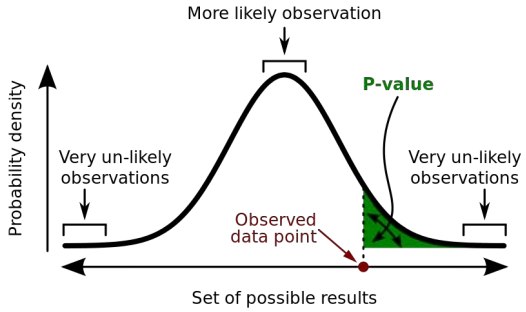


FIGURE 3.2: Visualization of the p-value. Credits: https://en.wikipedia.org/wiki/P-value

3.1.1 The importance of the Likelihood Ratio

Let X be a discrete random variable with probability mass function p depending on a parameter θ. Then the function

$$\mathcal{L}(\theta|x) = p(x|\theta) \tag{3.10}$$

considered as a function of θ is the likelihood function. The likelihood is equal to the probability that a particular outcome x is observed when the true value of the parameter is θ.

The likelihood function is the central object that summarizes the information from an experiment needed for inference of model parameters. It is key to many areas of science that report the results of classical hypothesis tests or confidence intervals using the (generalized) likelihood ratio as a test statistic (Cranmer, Pavez, and Louppe, 2015).

Let X be a random vector with values $x \in \mathbb{R}^d$ and let $p(x|\theta)$ denote the density probability of X at value x under the parametrization θ. Let also assume i.i.d. observed data $\mathcal{S} = \{x_1, \ldots, x_n\}$. In the setting where one is interested in simple hypothesis testing between a null $\theta = \theta_0$ against an alternative $\theta = \theta_1$, the Neyman-Pearson lemma states that the likelihood ratio

$$\gamma(\mathcal{L}, \theta_0, \theta_1) = \prod_{x \in \mathcal{S}} \frac{p(x|\theta_0)}{p(x|\theta_1)} \tag{3.11}$$

is the most powerful test statistic. This means that for a fixed probability of rejecting H_0 when it is true, this has the highest probability of rejecting H_0 when H_1 is true.

In order to evaluate γ, one must be able to evaluate the probability densities $p(x|\theta_0)$ and $p(x|\theta_1)$ at any value x. However, it is increasingly common in science that one has a complex simulation that can act as generative model for $p(x|\theta)$, but one cannot evaluate the density directly. For instance, this is the case in high energy

physics where the simulation of particle detectors can only be done in the forward mode.

3.1.2 Goodness of fit

The goodness of fit of a statistical model describes how well it fits a set of observations. Measures of goodness of fit typically summarize the discrepancy between observed values and the values expected under the model in question. Such measures can be used in statistical hypothesis testing, e.g. to test for normality of residuals, to test whether two samples are drawn from identical distributions (see Kolmogorov–Smirnov test), or whether outcome frequencies follow a specified distribution (see Pearson's chi-square test).

Pearson's chi square test Pearson's chi-square test verifies a null hypothesis stating that the frequency distribution of certain events observed in a sample is consistent with a particular theoretical distribution. It is named after Karl Pearson.

Let O be a set of N observations divided into b bins over an interval r. We would like to test if X is sampled from a distribution $p_c(x)$. The null and the alternative hypothesis are:

$$H_0 : O = \{x \,|\, x \sim p_c(x)\}$$
$$H_1 : O = \{x \,|\, x \nsim p_c(x)\}$$

The value of the test-statistic is computed as:

$$\chi^2_\nu = \sum_{i=1}^{b} \frac{(O_i - E_i)^2}{E_i} \tag{3.12}$$

where

- χ^2_ν is the Pearson's cumulative test statistic, which asymptotically approaches a χ^2 distribution.
- O_i is the number of observations in the bin i.
- E_i is an expected count for bin i, asserted by the null hypothesis, i.e.,

$$E_i \;=\; \Big(F(Y_u) \;-\; F(Y_l)\Big)\, N \tag{3.13}$$

  with F being the cumulative distribution function for the probability distribution p_c being tested, Y_u the upper limit for bin i, Y_i the lower limit for bin i.

The resulting value can be compared with a χ^2 distribution to determine the goodness of fit. The chi-square distribution has $(b - c)$ degrees of freedom, where b is the number of non-empty bins and c is the number of estimated parameters (including location and scale parameters and shape parameters) for the distribution p_c plus one. A corresponding p-value can be computed:

$$p_{\text{val}} = \int_{\chi^2_\nu}^{+\infty} \chi^2_{p-c}(t)dt \tag{3.14}$$

and the null hypothesis H_0 can be accepted/rejected based on a specific threshold (e.g., see Table 3.1).

Kolmogorov-Smirnov test In statistics, the Kolmogorov–Smirnov test (K-S test) is a nonparametric test that can be used to compare a sample with a reference probability distribution (one sample K-S test), or to compare two samples (two samples K-S test). In essence, the test answers the question "What is the probability that this collection of samples could have been drawn from that probability distribution?" or, in the second case, "What is the probability that these two sets of samples were drawn from the same (but unknown) probability distribution?". It is named after Andrey Kolmogorov and Nikolai Smirnov. In the following, the one sample K-S test will be briefly presented.

Let $O = \{x_i\}_{i=1}^{N}$ be a set of N observations and let $\hat{F}(x)$ be the empirical cdf computed as

$$\hat{F}(x) = \frac{\text{number of elements in the sample} \leq x}{n} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{O(i)\leq x} \tag{3.15}$$

where $\mathbb{1}$ denotes the indicator function.

In the one sample K-S test we would like to test if O is sampled from a cdf $G(x)$. The null and the alternative hypothesis are:

$$H_0 : O = \{x \,|\, x \sim G(x)\}$$
$$H_1 : O = \{x \,|\, x \nsim G(x)\}$$

The test statistic is the maximum absolute difference between the empirical cdf $\hat{F}(x)$ and the hypothesized cdf $G(x)$.

$$D^* = \max_x \left(|\hat{F}(x) - G(x)|\right) \tag{3.16}$$

For 'large' values of D^* the null hypothesis is rejected, whereas the null hypothesis is accepted for 'small' values of D^*. A corresponding p-value can be computed:

$$p_{\text{val}} = \int_{D^*}^{+\infty} K(t)dt \tag{3.17}$$

where $K(t)$ is the Kolmogorov distribution. The null hypothesis H_0 can be accepted/rejected based on a specific threshold (e.g., see Table 3.1).

3.2 New physics searches as a hypothesis test

The problem of searching new physics can be phrased in terms of a statistical hypothesis test. Recall that there are two sets of data:

- the reference sample $\mathcal{R} = \{x_i\}_{i=1}^{\mathcal{N}_\mathcal{R}}$, composed by simulated data (following the Standard Model)
- the data sample $\mathcal{D} = \{x_i\}_{i=1}^{\mathcal{N}_\mathcal{D}}$, composed by experimental data (potentially containing new physics)

The null hypothesis H_0 is that the data sample $\mathcal{D}$ is drawn from the same data distribution of the reference sample $\mathcal{R}$. The alternative hypothesis H_1 is that the data

sample $\mathcal{D}$ does not follow the same distribution of the reference sample $\mathcal{R}$, but contains a significant discrepancy which can be identified as new physics. To sum up:

- $H_0 : \mathcal{D} = \{x \,|\, x \sim p(x|R)\}$
- $H_1 : \mathcal{D} = \{x \,|\, x \sim p(x|NP)\}$

where $p(x|R)$ denotes the reference data distribution (given by the Standard Model), whereas $p(x|NP)$ is the data distribution of the new physics.

Here and in what follows $n(x)$ will be denoted as differential distribution, i.e., the probability density function (p.d.f.) of x normalized to the total number of expected events in the experiment, namely

$$n(x) = Np(x), \quad N = \int dx n(x) \tag{3.18}$$

This is an experimental quantity which can be predicted by theoretical models. If an experiment is run for a given amount of time then the reference model will predict a specific number of events to happen. Namely, it can be calculated from the theoretical *cross-section* of a certain process of interest - which measures the probability of occurrence - and the *luminosity* of the experiment - which measures the number of collisions produced per units of area and time (Hagiwara et al., 2002). The actual number of collected events can then be modeled as a Poisson random variable with mean N.

In the experiments of the present thesis, the size of the reference sample $\mathcal{N}_\mathcal{R}$ is given along with the number of expected background events in the data sample, namely $N(R)$. Expected background events are the events which are expected to be observed in the hypothesis that the data does not contain any new physics. Furthermore, some experiments requires the injection of new physics signal in the data sample, and a corresponding number of expected signal events denoted by $N(S)$ is provided. It is worth noting that the reference sample $\mathcal{R}$ is assumed to be quite large, for example $\mathcal{N}_\mathcal{R} = 100\, N(R)$, in order to reduce the effect of statistical fluctuations on the experiments and have a more accurate representation of the distribution predicted by the reference model.

The alternative hypothesis H_1 is composite, labeled by a number of free parameters w and denoted by its differential distribution $n(x|w)$. The distribution of the data sample - even in the case when specifying new physics - is "similar" to the reference one, hence it is convenient to parametrize $n(x|w)$ in terms of the reference differential distribution $n(x|R)$. Taking also into account that $n(x|w)$ is necessarily positive and that the log-likelihood ratio will be used for hypothesis testing, $n(x|w)$ is expressed as:

$$n(x|w) = n(x|R)e^{f(x;w)} \tag{3.19}$$

in terms of a set of real functions $\mathcal{F} = \{f(x;w) \forall w\}$. The corresponding density ratio is

$$e^{f(x;w)} = \frac{n(x|w)}{n(x|R)} \tag{3.20}$$

Once the set of alternative hypotheses is specified in this parametrized form, the optimal statistical test for the reference model is defined by the Neyman–Pearson construction (Neyman and Pearson, 1933), based on the maximum likelihood principle. The idea is to compare the reference with the best-fit distribution $n(x|\hat{w})$, obtained at the point $w = \hat{w}$ that maximizes the likelihood. More precisely, to exploit knowledge of the expected number of events, extended likelihoods are considered

as in Barlow, 1990. Compared to the usual likelihood, the extended likelihood has a Poisson factor that takes into account the fluctuations in the number of collected events $\mathcal{N}_{\mathcal{D}}$.The extended likelihood of the data sample $\mathcal{D}$ corresponding to the reference distribution is given by

$$\begin{aligned}\mathcal{L}(\mathcal{D}, R) &= \frac{e^{-N(R)} N(R)^{\mathcal{N}_{\mathcal{D}}}}{\mathcal{N}_{\mathcal{D}}!} \prod_{x \in \mathcal{D}} p(x|R) \\ &= \frac{e^{-N(R)}}{\mathcal{N}_{\mathcal{D}}!} \prod_{x \in \mathcal{D}} n(x|R) \end{aligned} \tag{3.21}$$

The extended likelihood of the data sample $\mathcal{D}$ corresponding to the alternative parametrized distribution is

$$\begin{aligned}\mathcal{L}(\mathcal{D}, \hat{w}) &= \frac{e^{-N(\hat{w})} N(\hat{w})^{\mathcal{N}_{\mathcal{D}}}}{\mathcal{N}_{\mathcal{D}}!} \prod_{x \in \mathcal{D}} p(x|\hat{w}) \\ &= \frac{e^{-N(\hat{w})}}{\mathcal{N}_{\mathcal{D}}!} \prod_{x \in \mathcal{D}} n(x|\hat{w}) \end{aligned} \tag{3.22}$$

This leads to the test statistic

$$t(\mathcal{D}) = 2 \log \left[\frac{\mathcal{L}(\mathcal{D}, \hat{w})}{\mathcal{L}(\mathcal{D}, R)} \right] \tag{3.23}$$

$$t(\mathcal{D}) = 2 \log \left[\frac{e^{-N(\hat{w})}}{e^{-N(\mathrm{R})}} \prod_{x \in \mathcal{D}} \frac{n(x|\hat{w})}{n(x|\mathrm{R})} \right] \tag{3.24}$$

$$= 2 \left[N(R) - N(\hat{w}) + \sum_{x \in \mathcal{D}} \log \left(\frac{n(x|\hat{w})}{n(x|\mathrm{R})} \right) \right]$$

$$= 2 \left[N(R) - N(\hat{w}) + \sum_{x \in \mathcal{D}} f(x; w) \right] \tag{3.25}$$

where $N(R)$ is the expected number of events in the reference model and $N(\hat{w})$ is the expected number of events in the alternative hypothesis

Since the two distributions are unknown, it is not possible to directly compute the test statistic $t(\mathcal{D})$. A machine learning approach used to learn the test statistic from data will be introduced in the Chapter 4.

3.2.1 Out-of-distribution and In-distribution anomaly detection

The problem of searching new physics is a hybrid between a supervised anomaly detection and a semisupervised anomaly detection. The similarity to supervised problems is the design of a predictive model to classify reference data versus experimental data. On the other hand, the similarity to semisupervised problem is that while the reference class represents, by construction, standard behavior, the class of experimental data is not labelled as anomalous.

Model-independent new physics searches can be considered as an instance of *out-of-distribution* anomaly detection, i.e. new physics signals can manifest as local overdensities in low probability density area of the data. However, in order to be sensitive to any new physics clue one has to consider also *in-distribution* anomalies

Symbol	Description
	Datasets
$\mathcal{R}$	Reference sample composed by simulated data
$\mathcal{D}$	Data sample composed by experimental data
	Events
$N(R)$	Expected reference events in the data sample $\mathcal{D}$
$N(S)$	Expected signal events in the data sample $\mathcal{D}$
$N(\hat{w})$	Predicted number of events in the data sample $\mathcal{D}$ by the machine learning model
$\mathcal{N}_{\mathcal{R}}$	Size of the reference sample $\mathcal{R}$
$\mathcal{N}_{\mathcal{D}}$	Size of the data sample $\mathcal{D}$
	Likelihoods
$\mathcal{L}(\mathcal{D}, R)$	Extended likelihood of the data sample $\mathcal{D}$ corresponding to the reference distribution R
$\mathcal{L}(\mathcal{D}, \hat{w})$	Extended likelihood of the data sample $\mathcal{D}$ corresponding to the alternative parametrized distribution
	Distributions
$p(x\|R)$	Probability distribution of x in the reference model R
$n(x\|R)$	Differential distribution of x in the reference model R
$p(x\|NP)$	Probability distribution of x in the new physics model NP
$n(x\|NP)$	Differential distribution of x in the new physics model NP
$n(x\|\hat{w})$	Differential distribution of x estimated by the machine learning model
	Test statistic
$t(\mathcal{D})$	Test statistic computed by the machine learning model on the data sample $\mathcal{D}$
$t_{\text{id}}(\mathcal{D})$	Ideal test statistic computed on the data sample $\mathcal{D}$
$p(t\|R)$	Probability distribution of the test statistic t in the reference model R
$p(t\|NP)$	Probability distribution of the test statistic t in the new physics model NP

TABLE 3.2: Main symbols used throughout the thesis

- loosely defined as a small local overdensities of samples that can reside in areas of the data with high probability density.

In such a regime it is no longer desirable to only determine if the probability density of a sample is high or low, instead one wish to compare the density of a sample to that of its neighbours along some conditional dimension of interest.

The only assumption about the anomalous data is that they are far fewer compared to the normal instances in the training data. Furthermore, the experimental data containing anomalies are small in size compared to the reference data generated by software simulations (by design, as discussed in the previous section), causing data imbalances. Issues that consequently arise due to imbalanced class distributions are handled with a custom loss function or with modifications to standard loss functions, as described in section 4.2.

To summarize, model-independent new physics searches as formulated in the context of the present work can be considered as a mix of out-of-distribution and in-distribution anomaly detection with a partial supervision on the training data.

Chapter 4

Designing a classifier for new physics discoveries

In this chapter, we reprise the main ideas introduced previously to tackle the problem of discovering new physics signals with tools from statistics and machine learning.

4.1 From hypothesis testing to binary classification

The basic strategy followed in D'Agnolo and Wulzer, 2019 and in this thesis work is to train a classifier on experimental measurements and on reference data to build a hypothesis testing procedure based on the likelihood ratio test.

By selecting the appropriate loss function, a classifier can be trained learn the likelihood ratio

$$f(x) = \log\left(\frac{p(1|x)}{p(0|x)}\right) \tag{4.1}$$

as shown in section 2.19 for the binary cross-entropy loss. This is sometimes called "the likelihood ratio trick" (Valsecchi, 2020, Kasieczka, Nachman, and Shih, 2020). The (learned) likelihood ratio can then be evaluated on the data of interest

$$\gamma(\mathcal{S}) = \prod_{x\in\mathcal{S}} e^{f(x)} \tag{4.2}$$

In order to compute a p-value, one needs a value of the test statistics for the data of interest and the distribution of the test statistics under the null hypothesis (as explained in D'Agnolo and Wulzer, 2019). The former is obtained by training the model to separate the reference data from the experimental data. The latter can be estimated by training the model multiple times on reference data only. The p-value can then be computed as follows:

$$p_{\text{val}} = \int_{t_{\text{obs}}}^{\infty} p(t|H_0)dt \tag{4.3}$$

It is worth noting that in practice building such a classifier for new physics discoveries is a challenging task given:

- the complexity of the experimental data in modern experiments
- the fact that the new physics signal is expected to be "small" and/or located in a region of the input features which is already populated by events predicted by the reference model.

The design choices presented in the following sections are intended to overcome the above issues.

4.2 The choice of the loss function

Custom loss We first review the loss function proposed in the paper D'Agnolo and Wulzer, 2019 which is directly based on the likelihood ratio test statistcs in Eq. 3.25. The authors also chose to parametrize the alterntive hypothesis using neural networks (see section 2.7.1 for a review).

To reconstruct the test statistic in Eq.(3.25), the number of expected events in the alternative hypothesis $N(w)$ needs to be computed. Using the density ratio in Eq.(3.20), we have that

$$N(w) = \int dx\, n(x|w) = \int dx\, n(x|R) e^{f(x;w)} \tag{4.4}$$

with

$$e^{f(x;w)} = \frac{n(x|w)}{n(x|R)} \tag{4.5}$$

Since the reference distribution $n(x|R)$ is not known analytically, the above expression is estimated by using a first-order Monte Carlo approximation

$$N(w) = \int dx\, N(R) p(x|R) e^{f(x;w)} = \frac{N(R)}{\mathcal{N}_{\mathcal{R}}} \sum_{x \in \mathcal{R}} e^{f(x;w)} \tag{4.6}$$

Eq. (3.25) thus becomes

$$\begin{aligned} t(\mathcal{S}) &= -2 \min_w \left[\frac{N(R)}{\mathcal{N}_{\mathcal{R}}} \sum_{x \in \mathcal{R}} e^{f(x;w)} - N(R) - \sum_{x \in \mathcal{D}} f(x;w) \right] \\ &= -2 \min_w \left[\frac{N(R)}{\mathcal{N}_{\mathcal{R}}} \sum_{x \in \mathcal{R}} (e^{f(x;w)} - 1) - \sum_{x \in \mathcal{D}} f(x;w) \right] \end{aligned} \tag{4.7}$$

The above expression can be rewritten as a single sum over events by introducing a target variable y which is set to $y = 0$ for the events in the reference sample $\mathcal{R}$ and to $y = 1$ for those in the data sample $\mathcal{D}$. Explicitly, the custom loss function is defined as follows:

$$\ell_{\text{npl}}((f(x), y)) = (1 - y) \frac{N(R)}{\mathcal{N}_{\mathcal{R}}} \sum_{x \in \mathcal{R}} (e^{f(x;w)} - 1) - y f(x) \tag{4.8}$$

The minimization of the loss ℓ_{npl} with respect to the neural networks parameters w can thus be carried out as a standard supervised training procedure. The test statistic is immediately computable from the loss function itself at the end of the training procedure:

$$t(\mathcal{D}) = -2 \sum_{x,y} \ell_{\text{npl}}(f(x;w), y) \tag{4.9}$$

The trained machine learning model $f(x;w)$ is in principle the maximum likelihood fit to the log-ratio of the data and reference distributions. The target function can indeed be shown to be:

$$f(x, \hat{w}) \approx \log \left[\frac{n(x|NP)}{n(x|R)} \right] \tag{4.10}$$

Instead of learning the likelihood ratio, the target function is a modified version of the log-likelihood ratio, where differential distributions appear in place of the normal likelihoods.

Notice that training unavoidably requires some sort of regularization because the loss function ℓ_{npl} (Eq. 4.8) is unbounded from below, namely it approaches negative infinity if f diverges at some value of x belonging to the $\mathcal{D}$ (i.e., $y = 1$) class. A possible solution to overcome this issue is discussed in section 4.4.

The result obtained with the custom loss function ℓ_{npl} and neural networks can be in principle generalized to other machine learning models with different loss functions, provided that the target function is the same as Eq. (4.10). This is one of the main arguments of the thesis and all the experiments are based upon such assumptions. In particular, two options are considered in this work: (a) a reweighted cross entropy loss, (b) a reweighted square loss. Both losses have been introduced in section 2.3.1. Kernel methods (presented in section 2.7.2) are used in place of neural networks.

It is worth noting that in both the proposed options, reweighting is a strategy which is useful at the same time to obtain the correct target function of Eq. (4.10) and to cure the problem of data imbalance (as discussed in section 2.5.3).

In the following, the alternative loss functions used in this work are presented and their corresponding target function is computed.

Weighted cross entropy loss Recall that the reweighted cross entropy loss reads

$$\ell_{\text{wbce}}(f(x), y) = \sum_{x,y} y_i \, \beta_1 \, \log\left(1 + e^{-f(x_i)}\right) + \, (1 - y_i) \, \beta_0 \, \log\left(1 + e^{f(x_i)}\right) \tag{4.11}$$

and its corresponding target function is

$$f = \log \frac{\beta_1 \, p(x|1)}{\beta_0 \, p(x|0)} \tag{4.12}$$

Now consider:

$$\begin{cases} \beta_1 = \dfrac{\mathcal{N}_{\mathcal{R}}}{N(R)} & \text{if } y = 1 \quad (y \in \mathcal{D}) \\ \beta_0 = 1 & \text{if } y = 0 \quad (y \in \mathcal{R}) \end{cases}$$

The resulting loss is:

$$\ell(y, f(x; w)) = \sum_{i=1}^{n} \frac{\mathcal{N}_{\mathcal{R}}}{N(R)} y_i \log\left(1 + e^{-f(x;w)}\right) + (1 - y_i) \log\left(1 + e^{f(x;w)}\right) \tag{4.13}$$

with the target function being

$$f = \log\left(\frac{\mathcal{N}_{\mathcal{R}}\, p}{N(R)\,(1-p)}\right)$$

$$= \log\left(\frac{p}{(1-p)}\right) + \log\left(\frac{\mathcal{N}_{\mathcal{R}}}{N(R)}\right)$$

Now recall that

$$p = p(y=1|x) = p(\mathcal{D}|x) = p(\mathcal{D})p(x|\mathcal{D})$$

Explicitly, the target function of the weighted cross entropy is:

$$f = \log \frac{\mathcal{N}_{\mathcal{R}}\, p\,(\mathcal{D}|x)}{N(R)\, p\,(\mathcal{R}|x)}$$

$$= \log \frac{\mathcal{N}_{\mathcal{R}}}{N(R)}\, \frac{p(\mathcal{D})\, p\,(x|\mathcal{D})}{p(\mathcal{R})\, p\,(x|\mathcal{R})} \qquad \text{(Bayes theorem)}$$

$$\simeq \log \frac{\mathcal{N}_{\mathcal{R}}}{N(R)}\, \frac{\mathcal{N}_{\mathcal{D}}\, p\,(x|\mathcal{D})}{\mathcal{N}_{\mathcal{R}}\, p\,(x|\mathcal{R})} \qquad \text{(empirical priors)}$$

$$\simeq \log \frac{N(\hat{w})\, p\,(x|w)}{N(\mathrm{R})\, p\,(x|\mathrm{R})} \qquad \text{(Poisson fluctuations } \mathcal{N}_{\mathcal{D}} \sim \text{Pois}(N(\hat{w}))\text{)}$$

$$= \log \frac{n\,(x|\hat{w})}{n\,(x|\mathrm{R})}$$

Thus,

$$f \approx \log \frac{n\,(x|\hat{w})}{n\,(x|\mathrm{R})}$$

The target function of the weighted cross entropy loss is the same as the target function of the loss ℓ_{npl}. The test statistic can be explicitly computed using eq. (3.25).

Weighted square loss The target function of the square loss is computed as:

$$\arg\min_{f(x)\in\mathbb{R}} \int dp(y|x)\ (y - f(x))^2 \tag{4.14}$$

For binary classification with $y = \pm 1$:

$$\arg\min_{f(x)\in\mathbb{R}} \int dp(y|x)\ (1 - yf(x))^2 = p(1|x) - p(-1|x) \tag{4.15}$$

Now consider to add a weight $a(y) > 0$ for each class:

$$a(y) = \begin{cases} a & \text{if } y = 1 \\ b & \text{if } y = -1 \end{cases} \tag{4.16}$$

The target function becomes

$$\arg\min_{f\in\mathbb{R}} \int dp(y|x)\ (1-yf)^2\, a(y) \tag{4.17}$$

where the notation $f(x)$ has been simplified to f for brevity. Solving for f:

$$\int dp(y|x)\, a(y)\, \frac{d}{df}\,(1-yf)^2 = 0$$

The target function reads as:

$$f(x) = \frac{a\, p(1|x) - b\, p(-1|x)}{a\, p(1|x) + b\, p(-1|x)} \tag{4.18}$$

The corresponding probability $p = p(1|x)$ is

$$p = \frac{b(f+1)}{a+b+f(b-a)} \tag{4.19}$$

The likelihood ratio is given by

$$\frac{a}{b}\,\frac{p}{(1-p)} = \frac{f+1}{1-f} \tag{4.20}$$

Now, going back to the problem of new physics discovery, if $a = \frac{\mathcal{N}_\mathcal{R}}{N(R)}$ and $b = 1$, the learned function f can be manipulated to compute the target likelihood ratio:

$$\frac{f+1}{1-f} = \frac{\mathcal{N}_\mathcal{R}}{N(R)}\,\frac{p(\mathcal{D}|x)}{p(\mathcal{R}|x)} \tag{4.21}$$

which leads to the same steps seen before with the weighted binary cross entropy, resulting in

$$\frac{f+1}{1-f} \approx \frac{n\,(x|\hat{w})}{n\,(x|\mathrm{R})} \tag{4.22}$$

Thus, using a reweighted version of the square loss does not lead directly to the same target function of the loss ℓ_{npl} and ℓ_{wbce} but the target likelihood ratio can be computed by applying an algebraic transformation to the learned function. In this case the test statistic can be computed as

$$t(\mathcal{D}) = 2\left[N(R) - N(\hat{w}) + \sum_{x\in\mathcal{D}} \log\left(\frac{f(x;w)+1}{1-f(x;w)}\right)\right] \tag{4.23}$$

4.3 A Machine Learning algorithm for searching new physics

To sum up, the problem of searching new physics can be phrased in terms of a statistical hypothesis test whose ingredients are the following:

- Two sets of data: a reference sample $\mathcal{R}$ of simulated events and a data sample $\mathcal{D}$ of experimental events
- Null hypothesis H_0: experimental data come from the reference distribution

$$n(x|\mathrm{R}) \tag{4.24}$$

- Alternative hypothesis H_1: experimental data come from a different, new physics distribution which can be parametrized with respect to the reference distribution

$$n(x|w) = n(x|\mathrm{R})\, e^{f(x;w)} \tag{4.25}$$

- Test statistic:

$$t(\mathcal{D}) = 2\left[N(R) - N(\hat{w}) + \sum_{x\in\mathcal{D}} \log\left(\frac{n(x|\hat{w})}{n(x|\mathrm{R})}\right)\right] \tag{4.26}$$

where $N(R)$ is the number of expected background events, $N(\hat{w})$ is the predicted number of events in the alternative hypothesis (see Eq. 4.7), $\hat{w}$ is the parameter provided with a maximum likelihood estimation

$$\hat{w} = \arg\max_{w}\ n(x|w)$$

Machine learning algorithms can be used to compute the test statistic $t(\mathcal{D})$ by learning the density ratio

$$\log\left(\frac{n(x|\hat{w})}{n(x|\mathrm{R})}\right) \tag{4.27}$$

through a binary classification problem in which the model is taught to distinguish the reference sample from the data sample. A schematic representation of the algorithm is shown in Figure 4.1.

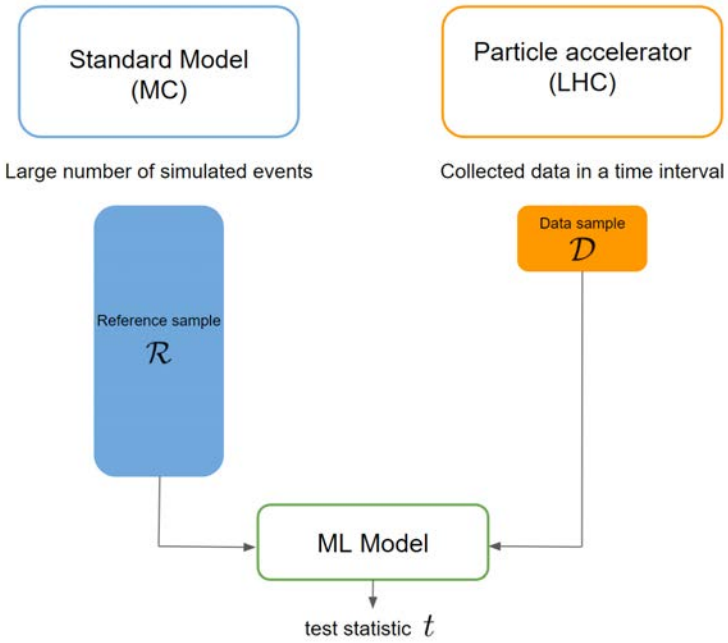


FIGURE 4.1: Schematic representation of the algorithm for searching new physics

In the original formulation of the problem (D'Agnolo and Wulzer, 2019), a custom loss function (eq. 4.8) is introduced to train a deep neural network model. In the present work, the problem is generalized to different loss functions and extended to other machine learning models, namely kernel methods. The general algorithm can be summarized as follows (see also the pseudocode Algorithm 1).

1. Simulate a large number of events to generate a reference sample $\mathcal{R}$. Collect experimental data to form a data sample $\mathcal{D}$.

2. Train a ML model to distinguish samples coming from the reference sample $\mathcal{R}$ vs the data sample $\mathcal{D}$. Use the custom loss function introduced in D'Agnolo and Wulzer, 2019 or one of its alternatives (see section 4.2).

3. Use the learned test statistic t_{obs} to compute the p-value

$$p_{\text{val}} = \int_{t_{\text{obs}}}^{\infty} p(t|R)dt \tag{4.28}$$

and its corresponding z-score (3.9)

$$z = \Phi^{-1}(1 - p_{\text{val}}) \tag{4.29}$$

4. If z is sufficiently high to signal a tension with the reference hypothesis, use the learned log-ratio $\log[n(x|\hat{w})/n(x|R)]$ to discover the nature of the discrepancy (new physics).

Algorithm 1: Hypothesis test for new physics discovery

Result: Determine if there is a meaningful discrepancy in the experimental data (new physics)

Input
- $\mathcal{D}$ = data sample
- $\mathcal{R}$ = reference sample
- M_l = machine learning model
- ℓ = loss function

Output
- z-score, log-ratio $\log[n(x|\hat{w})/n(x|R)]$

Pseudo-code

$\hat{M} \leftarrow$ train the machine learning model to classify $\mathcal{D}$ vs $\mathcal{R}$ with the loss function ℓ

$t_{\text{obs}} \leftarrow$ compute the observed test statistic

$p_{\text{val}} \leftarrow$ compute the p-value and its corresponding z-score using the χ^2 approximation

Post-processing

If z is sufficiently high to signal a tension with the reference hypothesis, use the log-ratio to analyze the nature of the discrepancy

The main problem of the above algorithm is that the quantity $p(t|R)$ needed to compute the p-value - which is the distribution of the test statistic under the null hypothesis - is unknown. The paper D'Agnolo et al., 2021 proposes to use the classical results by Wilks and Wald (Cowan et al., 2011) according to which the maximum log-likelihood ratio test statistics is distributed in the asymptotic limit as a χ^2 with a number of degrees of freedom equal to the number of free parameters in the alternative probability model. This means that $P(t|R)$ approaches in the Asymptotic Limit a χ^2 with a number of degrees of freedom given by the number of parameters of the machine learning model.

The problem is then to determine the number of degrees of freedom of the χ^2 distribution. In the paper D'Agnolo et al., 2021 the number of parameters of the neural network is used to define a target χ^2 distribution. Given that kernel methods are non-parametric, a different approach to hyper-parameter tuning is outlined in this thesis (described in section 4.4).

It is worth noting that despite its formulation is that of a binary classification, the problem at hand is not a standard machine learning problem. The main reason is that the final goal of the Algorithm 1 is a statistical hypothesis test, and machine learning models play the role of computing the test statistic. There is no interest in measuring classification accuracy. The emphasis is instead on properly estimating the density ratio given by Eq. (4.10) and implicity estimating the data generating densities.

The main objective is to build a model-independent strategy of new physics searches. This means that the machine learning model has to be sensitive to any generic departure from a given reference model. For instance, tuning parameters on a validation set containing a certain new physics signal (e.g., artificially injected) is not appropriate because the model will be biased towards the specificities of the injected signal. As a consequence, the machine learning pipeline presented in Section 2.3 is not applicable in its standard form. A possible solution is to tune the ML model on data coming from the reference model only. The target of the hyperparameter tuning procedure will be to match the distribution of the test statistic under the null hypothesis $p(t|R)$ to a specific χ^2 distribution, as discussed above.

4.4 Learning new physics with neural networks

The paper D'Agnolo and Wulzer, 2019 proposes to parametrize the alternative hypothesis (3.19) with neural networks (described in section 2.7.1). In particular, $f(x;w)$ is taken to be a fully connected neural networks, with free parameters w that correspond to the weights and biases of the network. The network has the following structure:

$$(i, h_1, h_2, \ldots, h_L, o) \tag{4.30}$$

with i size of the input layer, $h_1, h_2, \ldots, h_L$ size of the L hidden layers, o size of the output layer. The activation function is a sigmoid in the hidden layers, whereas the output is linear (a design choice but not a requirement).

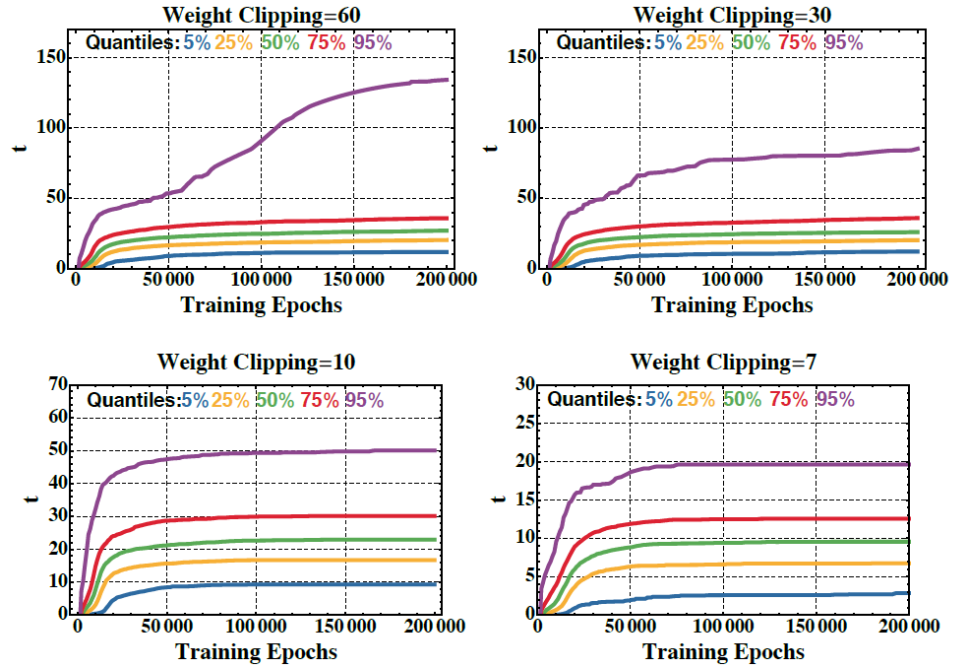
The custom loss (4.8) is employed as loss function in order to directly estimate the test statistic (3.25). In order to cope with the problem of the loss function being unbounded, the proposed solution is to enforce an upper bound (set by the so-called "weight clipping" parameter W_{clip}) on the absolute value of each weight. This forbids the neural network to diverge and to produce sharp features on a scale $\Delta x \lessapprox 1/W_{\text{clip}}$.

The hyperparameters of the neural network are tuned so that the distribution of the test statistic under the null hypothesis matches a target χ^2 distribution, with a number of degrees of freedom given by the number of parameters of the neural network. The choice of the architecture, the weight clipping parameter W_{clip} and the number of epochs used for training are tuned according to the χ^2 compatibility.

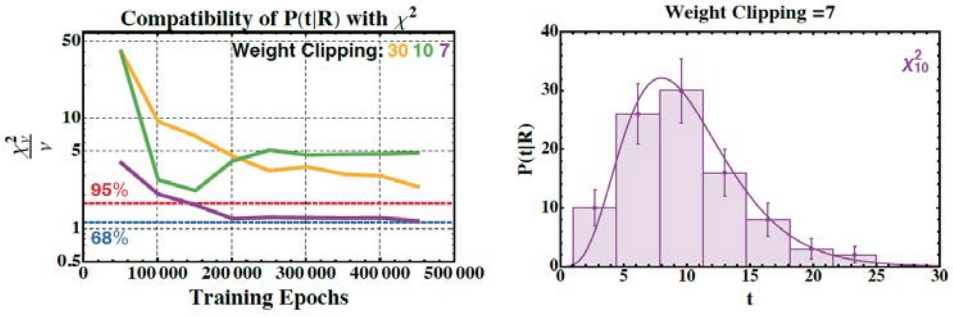
In particular, for each combination of the hyperparamaeters a number $N_{\text{toy}} \approx 100 - 300$ of classification problem is solved to learn the distribution of the test statistic under the null hypothesis and to see if it matches the target χ^2 distribution through a Pearson's χ^2-test. Notice that in each classification problem the data

sample $\mathcal{D}$ is composed of reference events only since the goal is to learn the distribution of the test statistic under the null hypothesis. The pseudocode of the procedure is presented in Algorithm 2. The main steps are the following. Given a neural network architecture, different values of the weight clipping parameter W_{clip} are tested and then:

1. The largest W_{clip} is selected for which the test statistic achieves a plateau over the training time (as shown in Figure 4.2(A))
2. The largest W_{clip} is selected which ensures the compatibility with the target χ^2 distribution (as shown in Figure 4.2(B))
3. The minimum number of epochs needed to achieve the plateau of the Pearon's χ^2 test statistic is chosen to stop the training procedure



(A)



(B)

FIGURE 4.2: Visualization of the hyperparameter tuning procedure proposed by D'Agnolo et al., 2021. (A) Value of the test statistic over the training time for different values of the weight clipping parameter W_{clip} (B - Left) Pearson's χ^2 test over the training time for different values of the weight clipping parameter W_{clip} (B - Right) Visualization of the χ^2 compatibility of $p(t|R)$ with $W_{\text{clip}} = 7$

Algorithm 2: χ^2 compatibility with max flexibility of the neural network model

Result: Select a set of hyperparameters which offer both flexibility of the model and a good Pearson's χ^2 test statistic on the $p(t|R)$ distribution obtained with N_{toy} experiments

Input

- $\mathcal{N}_R$ = size of the reference sample
- $N(R)$ = expected number of background events in the data sample
- ℓ = loss function
- $\beta = [\beta_1, \dots, \beta_r]$ set of r hyperparams to be tested
- N_{toy} = number of experiments

Output

- $T = N_{\text{toy}} \times m \times r$ matrix in which $T[\,i, j, k\,]$ is the test statistic $t_{\text{obs}} \sim p(t|R)$ computed on the toy data sample $\mathcal{D}_i$ at iteration j with hyperparams β_k
- $\frac{\chi_\nu}{\nu} = r \times m$ matrix in which $\frac{\chi_\nu}{\nu}[\,k, j\,]$ is the Pearson's χ^2 test statistic computed on the $p(t|R)$ distribution at iteration j with hyperapams β_k

Pseudo-code

$\mathcal{R} \leftarrow$ create reference sample $(\mathcal{N}_R)$
$[\mathcal{D}_1, \dots \mathcal{D}_{N_{\text{toy}}}] \leftarrow$ create N_{toy} data samples $(N(R))$

init T, $\frac{\chi_\nu}{\nu}$
for $k = 1 : r$ **do**
  for $i = 1 : N_{toy}$ **do**
    $X_i, Y_i \leftarrow$ create training set $(\mathcal{R}, \mathcal{D}_i)$
    net_i , $\text{hist}_i \leftarrow$ train network$(X_i, Y_i, \ell, \beta_k)$
    $T[\,i, :, k\,] \leftarrow$ compute the i-th row of t_{obs} test statistics $(\text{hist}_i, \dots)$
  end
  $\frac{\chi_\nu}{\nu}[\,k, :\,] \leftarrow$ compute k-th row of Pearson's χ^2 test statistics (on each column of $T[\,:, :, k\,]$)
end

Post-processing

(1) Plot the 95% quantiles of each column of T vs the number of epochs. Choose the hyperparameters $\beta^* \subseteq \beta$ for which the 95% quantile reaches a plateau

(2) Plot $\frac{\chi_\nu}{\nu}$ vs the number of epochs. Select the hyperparams $\beta_\chi \in \beta^*$ for which the Pearson's χ^2 test statistic achieves a plateau between 95% and 68% of significance

(3) The best number of iterations is the minimum value such that the plateau of the Pearson's χ^2 test statistic is reached for β_χ

The main drawback of the procedure is that it is really expensive in terms of computational time. For each hyperparamer to be tested, $N_{\text{toy}} \approx 100 - 300$ neural networks have to be trained. For instance, testing 5 different hyperparameters requires $5 \times 100 = 500$ trainings. Using a single machine is not feasible, since it will take too much time. Having a farm of computers and running each training in parallel is the only feasible solution.

4.5 Learning new physics with kernel methods

Overcoming the limitation of the approach presented in section 4.4 is the main objective of this thesis. Two different kernel methods - namely Falkon and LogFalkon - will be presented in this section as a viable alternative. They have the advantage of being order of magnitude faster in training time compared to neural networks. This means that hyperparameter tuning procedures for Falkon and LogFalkon can be deployed on single local machines, differently from neural networks.

Falkon Despite excellent theoretical properties, kernel methods presented in section 2.7.2 have limited applications in large scale learning because of time and memory requirements, typically at least quadratic in the number of data points.

Falkon is an algorithm proposed by Rudi, Carratino, and Rosasco, 2017 which provide an efficient approach to apply kernel methods on millions of points, and have been tested on a variety of large scale problems outperforming previously proposed methods while utilizing only a fraction of computational resources. At the same time it provides optimal theoretical guarantees.

The basic idea is to solve the kernel ridge regression problem (Eq. 2.89) by using the so called Nystrom approximation twice. First of all, the computation of the $n \times n$ kernel matrix $K_{X,X}$ is avoided. Instead, random projections are used, i.e., only a random subset of training data are considered for the final solution:

$$f(x) = \sum_{i=1}^{M} c_i k(x, x_i) \tag{4.31}$$

where $\{x_1, x_2, \ldots, x_M\} \subseteq \{x_1, x_2, \ldots, x_n\}$ are the subset of input points called Nystrom centers.

Second, in order to find the coefficients $\{c_i\}$ other approximations are used, namely preconditioning and conjugate gradient (described in section 2.4), resulting in an approximate Newton step. The preconditioner plays the role of approximate Hessian. Given the set of Nystrom centers selected uniformly at random from the input points, the approximate Hessian $\tilde{H}$ is computed in terms of its Choleski decomposition

$$\tilde{H}^{-1} = PP^\top \tag{4.32}$$

where

$$P = T^{-1}A^{-1} \tag{4.33}$$

with

- T being the Choleski decomposition of the kernel matrix of the Nystrom centers $K_{m,m}$

$$T = \text{chol}(K_{m,m}) \tag{4.34}$$

- A being the Choleski decomposition of the approximate Hessian

$$A = \text{chol}\left(\frac{1}{m} TT^\top + \lambda I\right) \tag{4.35}$$

  with λ the L2 regularization parameter and I the $m \times m$ identity matrix.

Then, conjugate gradient is applied to solve the preconditioned problem by solving the system

$$P^\top \left(K_{n,m}^\top K_{n,m} + \lambda I\right) P\beta = P^\top K_{n,m}^\top \tag{4.36}$$

with β being the unknown.

It has been shown that roughly $\mathcal{O}(n\sqrt{n})$ computations and $\mathcal{O}(n)$ memory are sufficient for optimal accuracy (with n number of the training points).

Figure 4.3 shows a pseudocode of Falkon.

Algorithm 1 Pseudocode for the Falkon algorithm.

function FALKON($X \in \mathbb{R}^{n\times d}, \boldsymbol{y} \in \mathbb{R}^n, \lambda, m, t$)
  $X_m \leftarrow$ RANDOMSUBSAMPLE(X, m)
  $T, A \leftarrow$ PRECONDITIONER(X_m, λ)
  function LINOP(β)
    $\boldsymbol{v} \leftarrow A^{-1}\beta$
    $\boldsymbol{c} \leftarrow k(X_m, X)k(X, X_m)T^{-1}\boldsymbol{v}$
    return $A^{-\top}T^{-\top}\boldsymbol{c} + \lambda n\boldsymbol{v}$
  end function
  $R \leftarrow A^{-\top}T^{-\top}k(X, X_m)\boldsymbol{y}$
  $\beta \leftarrow$ CONJUGATEGRADIENT(LINOP, R, t)
  return $T^{-1}A^{-1}\beta$
end function

function PRECONDITIONER($X_m \in \mathbb{R}^{m\times d}, \lambda$)
  $K_{mm} \leftarrow k(X_m, X_m)$
  $T \leftarrow \text{chol}(K_{mm})$
  $K_{mm} \leftarrow {}^1\!/_m TT^\top + \lambda \boldsymbol{I}$
  $A \leftarrow \text{chol}(K_{mm})$
  return T, A
end function

Note: LinOp performs the multiplication $\tilde{P}^\top H\tilde{P}\beta$ as in Eq. (8), via matrix-vector products.

FIGURE 4.3: Pseudocode of the Falkon algorithm. Credits: Meanti et al., 2020

LogFalkon LogFalkon is a variation of Falkon in which the square loss is replaced by the logistic loss (see eq. 2.18). Falkon can indeed be extended more generally to self-concordant loss functions, including the softmax loss (Meanti et al., 2020). In this case, iterative solvers are the default option since there is no closed form solution. Nyström method can be used a first time to reduce the size of the problem, and then a second time to derive an approximate Newton step. More precisely, at every step preconditioned conjugate gradient is run for a limited number of iterations with a decreasing sequence of regularization parameters $\{\lambda_k\}$, down to the desired regularization level. In practice, this requires running Algorithm 1 multiple times with small number of iterations and with decreasing λ.

Some computations are modified with respect to Falkon:

- the Choleski decomposition of the approximate Hessian is

$$A = \text{chol}\left(\frac{1}{m}TD_kT^\top + \lambda_k I\right) \tag{4.37}$$

  where D_k is the $m \times m$ diagonal matrix whose i-th diagonal element is the second derivative of the loss function computed on the i-th Nystrom center (i.e., $(D_k)_{i,i} = \ell''(f(x_i), y_i), i = 1, \ldots, M$), λ_k is the k-th regularization parameter

- The system corresponding to the preconditioned problem is

$$P^\top\left(K_{n,m}^\top D_k K_{n,m} + \lambda I\right)P\beta = P^\top K_{n,m}^\top g_k \tag{4.38}$$

  where g_k is the $n \times 1$ array containing the first derivatives of the loss function computed on the training data points, i.e. $(g_k)_i = \ell'(f(x_i), y_i)$.

Making these ideas practical requires efficiently implementing and deploying Algoritm 1, making full use of the available computational architectures.

To this purpose, a public library has been released with an efficient implementation of Falkon and LogFalkon algorithm[1] with the objective of increasing the density of operations per byte, and reduce as much as possible the required memory

[1] https://github.com/FalkonML/falkon

use transfers. Towards this end, a preconditioned gradient solver for kernel methods has been designed exploiting both GPU acceleration and parallelization with multiple GPUs, implementing out-of-core variants of common linear algebra operations to guarantee optimal hardware utilization. Further, the numerical precision of different operations has been optimized and the efficiency of matrix-vector multiplications has been maximized. As a result, dramatic speedups on datasets with billions of points have been shown experimentally, while still guaranteeing state of the art performance (Meanti et al., 2020).

A challenge in both Falkon and LogFalkon is the hyperparameter tuning procedure. The number of Nystrom centers M, the regularization level λ_k, the kernel function $k(x, x')$ and its parameters need to be tuned with respect to the data at hand. A common procedure is evaluating a grid search of the parameters over a train-validation-test split of the dataset (as described in section 2.3.2). Reasonable values for M are $M > \sqrt{n}$ (with n number of training points).

As pointed out in section 4.3, it is not possible to apply the above grid search procedure for the problem at hand. A custom hyperparameter tuning procedure has been introduced, described in the following paragraph.

Five types of kernel functions are implemented in the library but we use only the Gaussian kernel described above since it is the default choice and it is well-suited to our data:

$$k(x, x') = \exp\left(\frac{\|x - x'\|}{2\sigma^2}\right) \tag{4.39}$$

Hyperparameter tuning There are three main hyperparameters in Falkon/LogFalkon:

- the number of Nystrom centers M
- the regularization parameter λ
- the bandwidth of the Gaussian kernel σ (the default kernel choice, used also in this work)

There is no evident way of predicting the degrees of freedom of the target χ^2 distribution given the above hyperparameters. However, it is possible to show that as the model complexity grows the degrees of freedom of the target χ^2 distribution grows as well. As a consequence, the proposed tuning procedure takes into account the impact that the three hyperparameters have at the same time on the χ^2 compatibility.

The larger the number of Nystrom centers the more accurate the model and the more the time needed for training, as shown in Figure 4.4(C). The procedure has been tested on all the datasets presented in this thesis but here we explictly refer to examples coming from a simple univariate problem (described in section 5.1.1). Large values of λ or σ produces a simpler model, whereas small values tend to fit the data. The steps of the proposed procedure are the following:

1. Fix σ corresponding to the median or the 90th percentile of the pairwise Euclidean distances of the training data (see Figure 4.4(A)). We do this heuristic approach to capture the most relevant scales of the problem, while avoiding an extreme overfit of the noise.

2. Explore different pairs (M, λ) and launch $\sim$20 experiments for each of them. Consider a number of centers at least as large as $M = \sqrt{N}$, with N number of samples in the training set (optimal statistical bounds can be achieved with

such quantity). Plot the median test statistic versus the number of Nystrom centers M for each value of λ. See Figure 4.4(B) for an example. Choose M reaching a plateau.

3. With fixed M explore different pairs (σ, λ) and launch $\sim$20 experiments for each of them. Plot the median test statistic versus σ for each λ. See Figure 4.4(C) for an example. Choose the smallest λ ensuring stability and a good training time (the highest curve). Choose the smallest σ reaching a plateau.

4. With the best configurations (M, λ, σ) obtained so far launch at least 100 experiments to check the χ^2 compatibility with a Kolmogorov-Smirnov test. The best hyperparameters are the ones providing the best compatibility with the highest degrees of freedom. Figure 4.4(D) shows an example of χ^2 compatibility.

Although based on heuristics, the procedure provides a flexible solution to find the χ^2 compatibility and to ensure a good training time.

It is worth noting that for low values of λ the algorithm may become slow or unstable, producing an observed test statistic equals to 'nan'. Such values of λ should be detected and discarded during steps 2,3.

4.6 Performance assessment

As already stated in section 4.3, although the machine learning model is trained to solve a classification problem there is no interest in measuring classification accuracy. The emphasis is on properly estimating the likelihood ratio given by Eq. (4.10).

The ideal test statistic can be employed as a figure of merit to assess the performances of the method. It is defined as the value of the test statistic obtained by using the exact (ideal) likelihood ratio test statistics

$$t_{\mathrm{ID}}(\mathcal{D}) = 2 \log \left(\frac{e^{-N(NP)}}{e^{-N(R)}} \prod_{x \in \mathcal{D}} \frac{n(x|NP)}{n(x|R)} \right) \tag{4.40}$$

where $n(x|NP)$ is the differential distribution of a specific new physics process and $N(NP)$ its corresponding number of expected events. According to the so-called Neyman–Pearson lemma (Cowan et al., 2011), t_{ID} is the optimal discriminant between the reference and the new physics hypotheses. It is the one that produces the smallest median p-value if NP is the true distribution of the data sample. This test statistic is denoted as "ideal" because it is the one which is most suited to discover data departures from the reference model, but it can be used only when the true data distribution is known a priori.

In the experiments, the true data distribution is known only in a simple univariate problem which is described in section 5.1.1. More generally, t_{ID} is estimated using simulated data and model-dependent analyses that leverage what is known about the type of new physics in the data. For every experiment the procedure for the computation of t_{ID} will be reported.

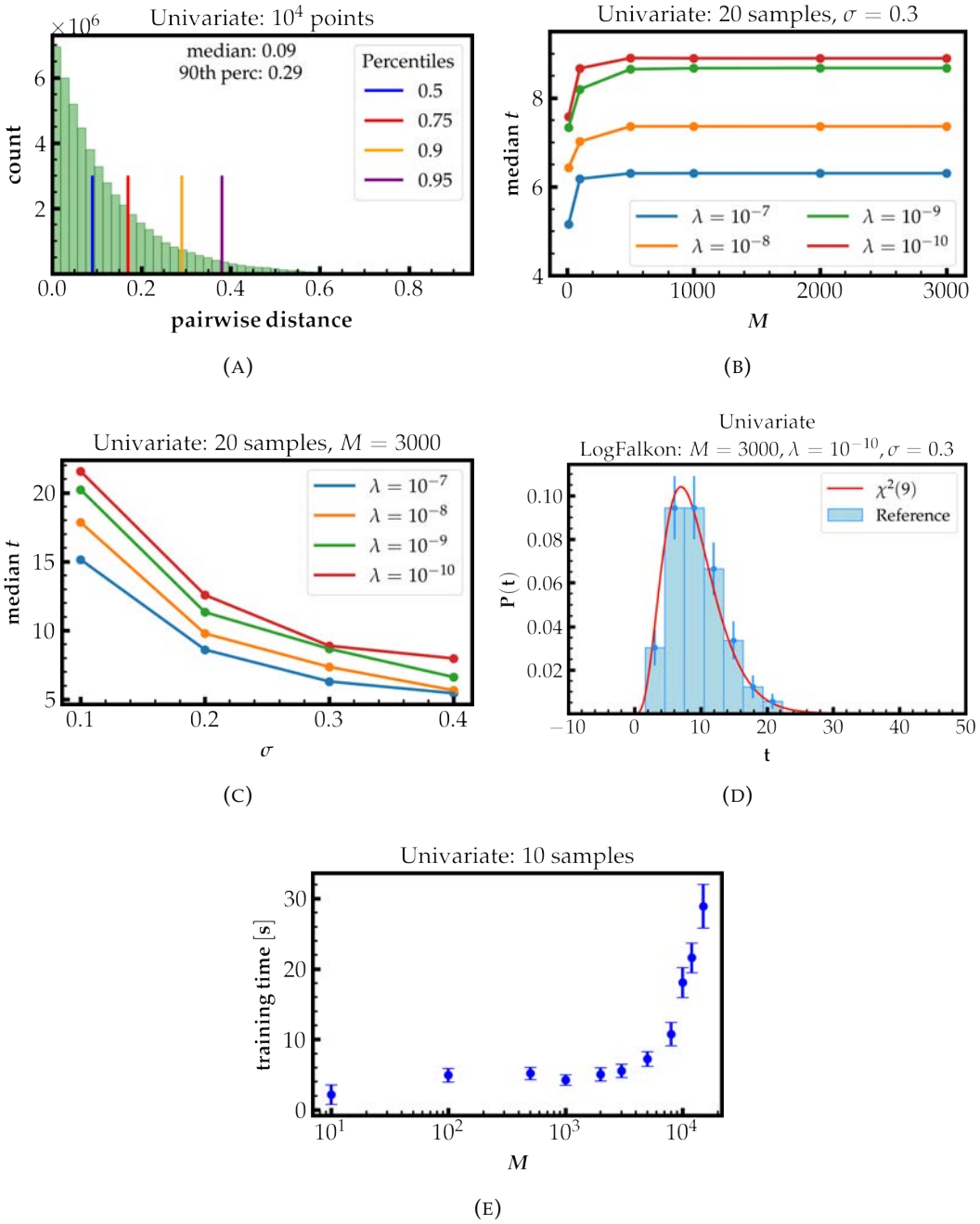


FIGURE 4.4: Visualization of the hyperparameter tuning procedure proposed in this work in the case of a simplified univariate example (A) Euclidean pairwise distance (B) Median test statistic as a function of the number of Nystrom centers (C) Median test statistic as a function of the kernel bandwidth (D) χ^2 compatibility (D) Average training time as a function of the number of Nystrom centers.

Chapter 5

Experiments and results

In this section, the problem of New Physics searches as described in Chapter 4 is applied to different datasets with the two machine learning models presented in sections 4.4 and 4.5. After a description of the datasets, the main results are presented and discussed.

5.1 Datasets

The first dataset used comes from a simple univariate "toy" model which has been used to test the entire pipeline. Apart from this simple "toy" problem, realistic simulated high energy physics datasets are employed with an increasing number of dimensions. Each dataset is made of two classes: a reference class, containing events following the Standard Model, and a data class, made of reference events with the injection of a new physics signal. Each case includes a set of features given by kinematical variables as measured by the particle detectors (plus additional quantities when available, such as reconstructed missing momenta and b-tagging information) that are called low-level feature features. From the knowledge of the intermediate physics processes, one can compute additional high-level features that are functions of low-level ones and posses a higher discriminative power. The different features are used to test the flexibility of the models.

The features of the different datasets are visualized in Appendix A.

5.1.1 Univariate data

The dataset is taken from D'Agnolo and Wulzer, 2019. In this case, both the reference distribution and the new physics distribution are simple univariate distribution with $x \in [0, 1]$ and they are known explicitly.

The reference distribution is a steeply-falling exponential

$$p(x|R) = \lambda\, e^{-\lambda x} \tag{5.1}$$

with $\lambda = 8$. The number of events in the reference sample is $\mathcal{N}_{\mathcal{R}} = 2 \times 10^5$, whereas the expected number of background events in the data sample is $N(R) = 2 \times 10^3$.

There are three types of signals, two of them localized in a specific region of the input feature (resonant) and one spread across the whole range of x (non-resonant). They are as follows:

- A Gaussian distribution in the tail of the exponential background, $\mathcal{N}(\mu, \sigma)$ with $\mu = 0.8$, $\sigma = 0.02$ and a number of expected events $N(S_1) = 10$ (Figure 5.1(A))

- A Gaussian distribution in the bulk of the exponential background, $\mathcal{N}(\mu, \sigma)$ with $\mu = 0.2$, $\sigma = 0.02$ and a number of expected events $N(S_2) = 90$ (Figure 5.1(B))

- A non-resonant signal given by

$$p(x|S_3) = \beta x^2 e^{-\lambda x} \tag{5.2}$$

with $\beta = 256$, $\lambda = 8$ and a number of expected events $N(S_3) = 90$ (Figure 5.1)(C).

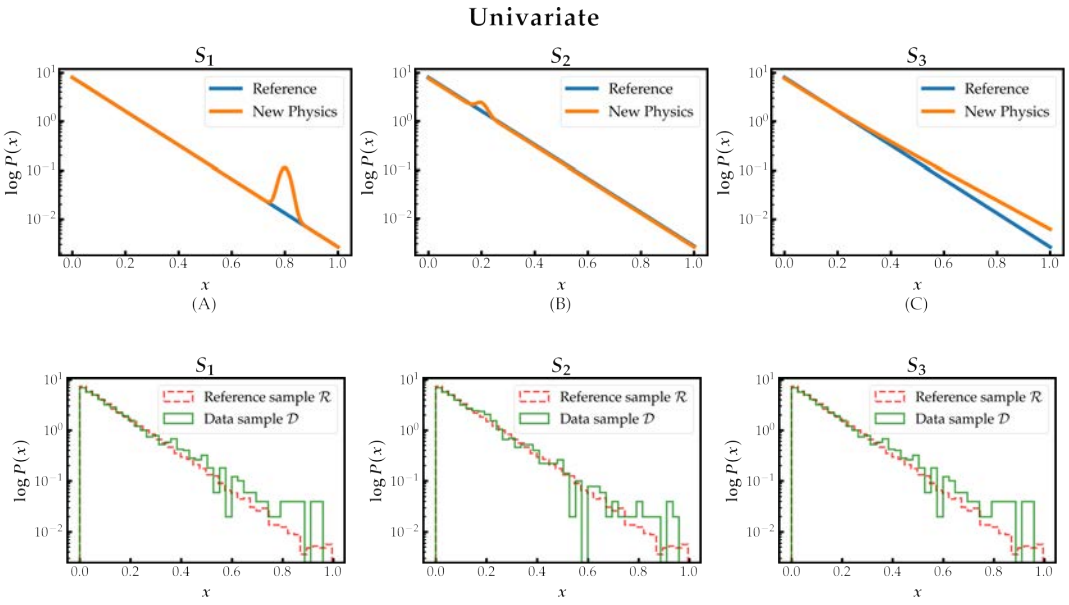


FIGURE 5.1: (Top row) Visualization of the univariate distributions: (A) Gaussian signal in the tail (B) Gaussian signal in the bulk (C) Non resonant signal (Bottom row) Visualization of the distribution of the reference sample versus the data sample for each type of signals.

The New Physics distribution for x is obtained combining the reference distribution with the signal:

$$\begin{aligned} p(x|\text{NP}) &= p(x|\text{R}+\text{S}) \\ &= \frac{1}{N(R)+N(S)}\Big(n(x|R)+n(x|S)\Big) \\ &= \frac{1}{1+N(S)/N(R)}\Big[p(x|R)+\frac{N(S)}{N(R)}p(x|S)\Big] \end{aligned}$$

where $N(S)/N(R)$ is known as signal to background ratio. The total expected number of events is

$$N(\text{NP}) = N(S) + N(R)$$

and the corresponding new phisics differential distribution is:

$$n(x|\text{NP}) = \frac{N(S)+N(R)}{1+N(S)/N(R)}\left[p(x|R)+\frac{N(S)}{N(R)}p(x|S)\right]$$

In this case it is possible to compute the ideal test statistic exactly (Eq. 4.40) for performance assessment since the distributions are known.

5.1.2 Multivariate data

Three realistic simulated high energy physics datasets are employed with an increasing number of dimensions.

DiMuon This is a five dimensional simulated dataset that was introduced in D'Agnolo et al., 2021 and it is composed of LHC-like dimuon processes $pp \to \mu^+\mu^-$, at a center-of-mass energy of 13 TeV. The low-level features are the transverse momenta and pseudorapidities of the two muons and their relative azimuthal angle, i.e., $x = [p_{T1}, p_{T2}, \eta_1, \eta_2, \Delta\phi]$. The high-level feature is given by the dimuon invariant mass $m_{\ell\ell}$. We consider two types of new physics contributions: the first one is a new vector boson Z' for which we consider different mass values ($m_{Z'} = 200, 300$ and 600 GeV); the second type of signal is instead a non-resonant signal given by a contact interaction, with three values for the coupling constant ($c_W = 1, 1.2$ and 1.5 TeV^{-2}). The distribution of the features is shown in Figure A.1.

The size of the reference sample is $N_0 = 10^5$, the expected number of background events in the data sample is $N(R) = 2 \times 10^4$. The signal component is in the range $[6, 80]$.

The ideal significance is estimated via a cut-and-count strategy in the invariant mass $m_{\ell\ell}$ space around $m_{Z'}$ for the resonant signals, while a likelihood ratio test on the binned $m_{\ell\ell}$ distribution is used for the non-resonant case.

SUSY The SUSY dataset Baldi, Sadowski, and Whiteson, 2014 is composed of simulated events in which the final state is made of two charged leptons $\ell\ell$ and missing momentum of invisible particles. The latter are given, in the Standard Model, by two neutrinos $\nu\nu$ coming from the decay of two W bosons. The new physics scenario also includes the decay of a pair of electrically charged supersymmetric particles $\chi^\pm$ into two neutral supersymmetric particles $\chi^0\chi^0$ and two W bosons. The dataset has 8 raw features which are shown in Figure A.2 and 10 high-level feautures which are shown in Figure A.3.

The experimental parameters are $N(R) = 10^5$ and $\mathcal{N}_\mathcal{R} = 5 \times 10^5$, the signal component $N(S)$ is selected in the range $[200, 650]$.

The ideal significance is estimated by training a supervised classifier to discriminate between background and signal with a total of 2M examples, following the approach in Baldi, Sadowski, and Whiteson, 2014. The significance is then estimated by a cut-and-count analysis on the classifier output.

HIGGS The HIGGS dataset Baldi, Sadowski, and Whiteson, 2014 is made of simulated events in which the NP signal is given by the production of heavy Higgs bosons H. The final state is given by a pair of vector bosons $W^\pm W^\mp$ and two bottom quarks $b\bar{b}$ for both the reference and the signal components. The dataset has 21 raw feaures and 7 high-level feautures. The low level features are shown in Figure A.4, whereas the high level features are shown in Figure A.6.

The experimental parameters are $N(R) = 10^5$, $\mathcal{N}_\mathcal{R} = 5 \times 10^5$ and the signal component $N(S)$ is in the range $[1000, 2500]$.

The ideal significance is estimated as in the previous case by using the output of a supervised classifier trained to separate signal from background.

5.2 Data preparation

Dataset come in the form of single or different files. In particular, the DiMuon datasets have 66 files containing reference events, 22 files containing non-resonant signal events for each of the three types of non resonant signal, and a single file for each of the three types of the resonant signal. Multiple files are used for a more efficient memory management (i.e., to avoid loading data all at once when only part of them are used for each experiment).

For the generation of the reference sample $\mathcal{R}$ the events are randomly sampled from the files containing reference events. The order of files is shuffled beforehand.

For the generation of the data sample $\mathcal{D}$ the following steps are executed:

1. the number of reference events in the data sample is extracted from a Poisson distribution with mean $N(R)$
2. if signal is to be injected in the data sample:
   (a) the number of signal events in the data sample is extracted from a Poisson distribution with mean $N(S)$
   (b) if the signal type is resonant, then the reference events are sampled from the files containing reference events and the signal events are extracted from the files containing signal events. Afterwards, the samples are merged together.
   (c) if the signal type is non-resonant, then both the reference and signal events are sampled from the same set of files containing the non-resonant signal.

After the generation of the reference sample and the data samples the features allowing negative values, as η, are normalized with a "z-score normalization" (subtracting their mean and dividing by their standard deviation). Strictly positive features, such as p_T , are simply divided by their mean.

The HIGGS and the SUSY datasets comes into a single file where background events are labeled as $y = 0$, whereas signal events are labeled as $y = 1$. A set of different files is generated for each class in order to reconduct the sampling procedure to the one utilized for the DiMuon dataset. The HIGGS and SUSY dataset are already normalized and no further preprocessing is needed.

5.3 Experiments with LogFalkon

The experiments with LogFalkon and Falkon have been run on a single GPU machine, equipped with an AMD EPYC 7301 16-Core Processor and a Nvidia Quadro RTX 6000 with 22 GB of dedicated video memory.

Univariate data The tuning curves for the univariate case are shown in Figure 4.4. The selected configuration is $M = 3000, \sigma = 0.3, \lambda = 10^{-10}$. Figure 5.2 shows the results on the different types of signal. The observed z-score are in order 2.55, 2.94, 3.15. The corresponding ideal z-score are 4.7, 4.4, 4.1. A strong correlation is registered between the observed z-score and the ideal z-scores for all the types of signals, especially for the non-resonant signal with a difference of less than 1σ. In all the experiments that follow, the compatibility of the distribution of the test statistic under the reference model with a χ^2 distribution is determined following Algorithm 3, whereas testing the sensitivity to the New Physics signal is done in accordance to

Algorithm 3: Find a compatible χ^2 distribution

Input
- $\mathcal{N}_\mathcal{R}$: size of the reference sample
- $N(R)$: number of expected background events in the data sample
- M_L : machine learning model with tuned hyperparameters

Parameters
- N_{toy} : number of toy data samples used to estimate $p(t|\,\text{R})$

Output
- $p(t|R)$: distribution of the test statistic under the reference model
- dof : degrees of freedom of the χ^2 distribution approximating $p(t|R)$

Pseudo-code

1. Estimate $p(t|R)$
for *i = 1 : N_{toy}* **do**
  $\mathcal{R} \leftarrow$ create reference sample ($\mathcal{N}_\mathcal{R}$)
  $\mathcal{D} \leftarrow$ create data sample containing only reference events ($N(R)$)
  $\hat{M}_L \leftarrow$ train the machine learning model to distinguish between $\mathcal{R}, \mathcal{D}$
  $t^i_{\text{obs}} \leftarrow$ compute the observed test statistic
end
2. dof $\leftarrow$ perform a Kolmogorov-Smirnov test to find the most compatible χ^2 distribution to the set of test statistics $\{t^i_{\text{obs}}\}_{i=1}^{N_{\text{toy}}}$

Algorithm 4. In this case, a total of 300 experiments have been run to reconstruct the test statistic distribution in the reference hypothesis, whereas 100 experiements have been run to contruct the distribution of the test statistic in the New Physics case. The average training time is approximately 7 seconds per experiment.

In all the figures the reported z-score z_{obs} is computed following the χ^2 approximation. The error bars are computed as $\sqrt{c_i / n\, w_i}$ with c_i count of the i-th bin, n total number of samples, w_i width of the i-th bin.

Multivariate data The steps of the hyperparameter tuning procedure have been applied to each dataset, analogously to the univariate case. The first and the last tuning steps are reported for each case, along with a test of the sensitivity to the New Physics signal.

Figure 5.3 shows the tuning steps on the DiMuon dataset. The selected configuration is $M = 2 \times 10^4, \sigma = 3, \lambda = 10^{-6}$. Figure 5.4 shows the tuning steps on the HIGGS dataset. The chosen configuration is $M = 10^4, \sigma = 7, \lambda = 10^{-6}$. Figure 5.5 shows the tuning steps on the SUSY dataset. The selected configuration is $M = 10^4, \sigma = 4.5, \lambda = 10^{-6}$.

With the hyperparameters selected as above, the LogFalkon model has been applied to study the sensitivity of the model to the presence of new physics signals in the data. The distribution of the test statistics is reconstructed with 300 toy samples for the reference and 100 for the data. The degrees of freedoms of the corresponding χ^2 is determined using a Kolmogorov-Smirnov test (described in section 3.1.2).

Figure 5.6 shows the median observed significance against the estimated ideal significance with the model trained on low-level features only. These experiments were performed by varying the signal fraction $N(S)/N(R)$ (at fixed luminosity, i.e., at fixed $N(R)$ and $\mathcal{N}_\mathcal{R} = 5N(R)$) and varying the type of signal in the DiMuon case,

Algorithm 4: Test sensitivity to New Physics signal

Input
- $\mathcal{N}_{\mathcal{R}}$: size of the reference sample
- $N(R)$: number of expected background events in the data sample
- $N(S)$: number of expected signal events in the data sample
- M_L : machine learning model with tuned hyperparameters
- dof : degrees of freedom of the χ^2 distribution approximating $p(t|R)$

Parameters
- N_{toy} : number of toy data samples used to estimate $p(t|\,\text{NP})$

Output
- $p(t|NP)$: distribution of the test statistic under the New Physics model
- z_{obs} : observed z-score

Pseudo-code

1. Estimate $p(t|NP)$

for *i = 1 :* N_{toy} **do**
  $\mathcal{R} \leftarrow$ create reference sample ($\mathcal{N}_{\mathcal{R}}$)
  $\mathcal{D} \leftarrow$ create data sample injecting signal events ($N(R), N(S)$)
  $\hat{M}_L \leftarrow$ train the machine learning model to distinguish between $\mathcal{R}, \mathcal{D}$
  $t^i_{\text{obs}} \leftarrow$ compute the observed test statistic
end

2. Compute $p(t|NP)$ by normalizing to one the histogram of the test statistics $\{t^i_{\text{obs}}\}_{i=1}^{N_{\text{toy}}}$

3. Compute the median observed z-score as

$$z_{\text{obs}} = \text{median}\left(\int_{t^i_{\text{obs}}}^{\infty} \chi^2_{\text{dof}}(t)\, dt\right) \quad i = 1, \ldots, N_{\text{toy}}$$

in which different resonant or non-resonant signal types are available. The error bars represent the 68% confidence interval. In all cases it is possible to notice a good correlation between the observed and the ideal significance. In the DiMuon case, the observed significance seems to depend weakly on the type of new physics signal, and an observed z-score $z_{\text{obs}} \sim 3\sigma$ is obtained for an ideal significance $\hat{z}_{\text{id}} > 8$. In the HIGGS case an observed z-score $z_{\text{obs}} \sim 3\sigma$ is obtained for $\hat{z}_{\text{id}} > 14$, whereas in the SUSY case for $\hat{z}_{\text{id}} > 10$. This means that a smaller sensitivity is registered in higher dimensions (the SUSY and the HIGGS dataset) as well as in the case of the non-resonant signal of the DiMuon dataset (denoted by $c_w = 1.0, 1.2, 1.5$ TeV^{-2} in the figure). It is likely that model-dependent approach can produce a larger observed significance for the same values of the ideal significance. However, the model-independent approach experimented here is capable of detecting a significant tension between the reference data and the experimental data in all the cases.

Figure 5.7 shows explicitly for the Z' new physics with $m_{Z'} = 300$ Gev the estimated probabilities to find a discrepancy of at least α for a given value of $\hat{Z}_{id}$. Similar results are obtained with the other types of signal.

To test the ability of the model to extract useful information from data, Figure 5.8 shows that including the high-level features in the training data does not significantly improve the results, especially in higher dimensions. This means that high

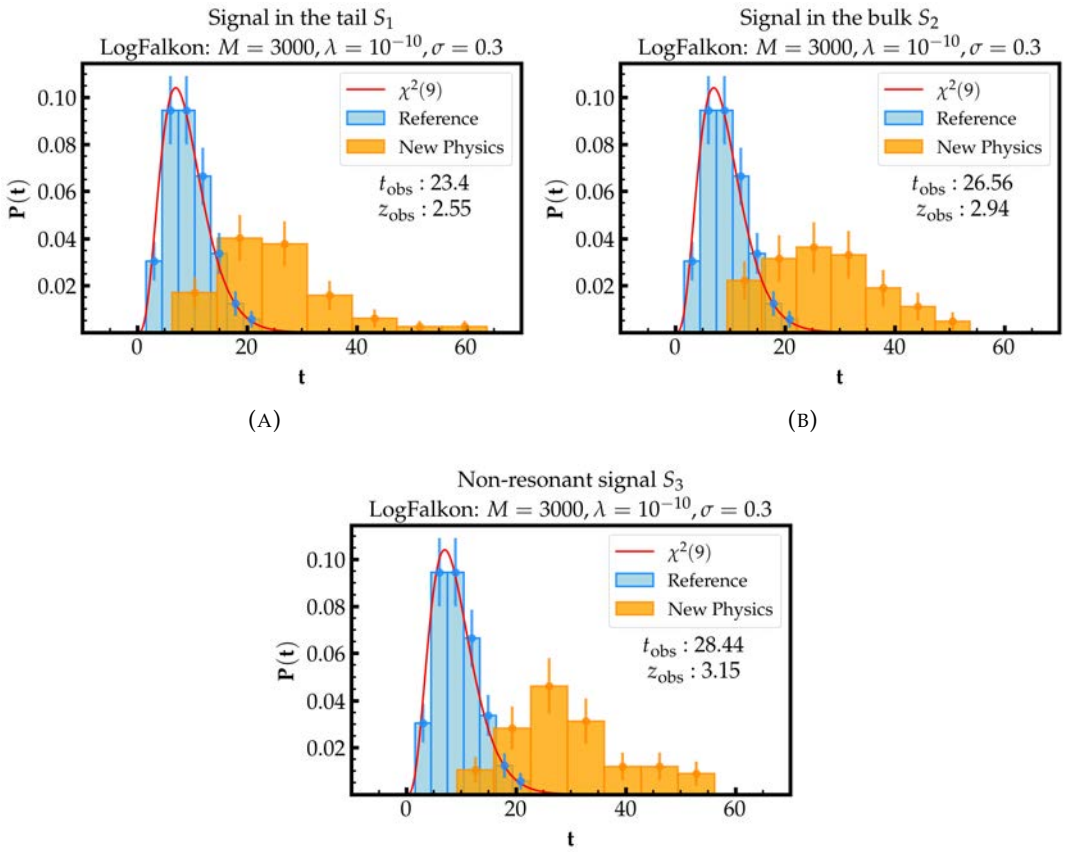


FIGURE 5.2: Distribution of the test statistic obtained with LogFalkon in the univariate case

level feature does not play an important role in detecting New Physics and can be omitted during the training phase.

As discussed in Section 4.2, the function approximated by the model is the density ratio given in Eq.(4.10). The latter can be directly inspected to characterize the nature of the "anomalies" in the experimental data, if a significant tension with respect to the reference sample has been manifested by the hypothesis test. Figure 5.9 reports some examples of the reconstructed density ratios as functions of certain high-level features together with estimates of the true ratios and extrapolations from the data used for training.

Notice that the high level feature x_{hl} chosen for constructing the density ratio comes from an augmented version of the reference sample denoted by $\hat{\mathcal{R}}$ or an augmented version of the data sample denoted by $\hat{\mathcal{D}}$, in which the high level feature x_{hl} has been considered along with the low level features. Note that x_{hl} is not necessarily given as input to the training algorithm (i.e., $\hat{\mathcal{R}}, \hat{\mathcal{D}}$ do not necessarily coincide with $\mathcal{R}, \mathcal{D}$).

To compute the likelihood ratios four different distributions are defined:

- the "true" differential distribution according to the reference model

$$n_R = \frac{N(R)}{\mathcal{N}_{\mathcal{R}}} h_R \tag{5.3}$$

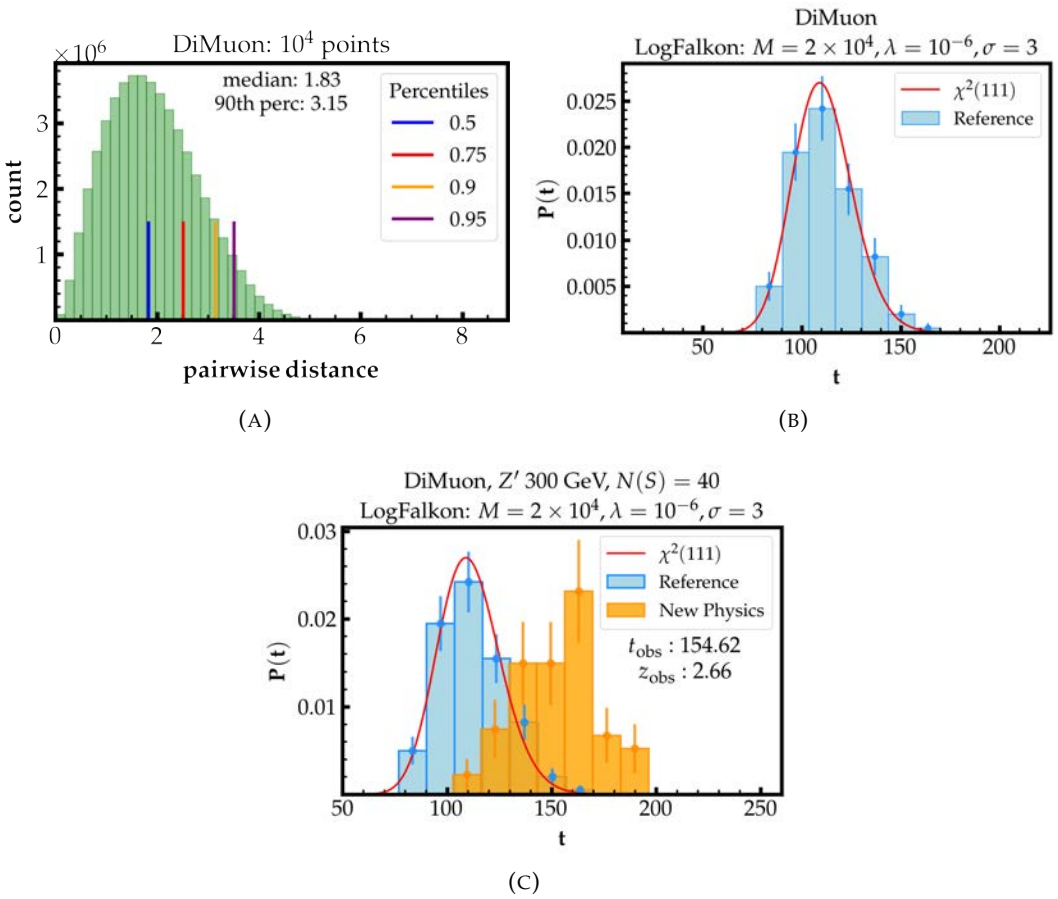


FIGURE 5.3: (A)-(B) Tuning LogFalkon on the DiMuon dataset, (C) Results on the resonant signal Z' 300 GeV

where h_R is the histogram of the high level feature x_{hl} (i.e., $h_R = [h_1, h_2, \ldots, h_b]$ is the vector containing the counts for each bin, with b number of bins)

- the "true" differential distribution according to the New Physics model is obtained by creating a new large dataset $\mathcal{I}$ containing one million events (preserving signal over background ratio of the data sample $\mathcal{D}$). The histogram of the high level feature $x_{hl} \in \mathcal{I}$ is normalized to the sum of the expected background and signal events $N(R) + N(S)$ in the data sample $\mathcal{D}$:

$$n_{NP} = \frac{N(R) + N(S)}{|\mathcal{I}|} h_{\mathcal{I}} \tag{5.4}$$

where $h_{\mathcal{I}}$ is the histogram of the high level feature in the new dataset $\mathcal{I}$

- the empirical differential distribution reconstructed from the data sample $\mathcal{D}$ at hand is obtained by normalizing the histogram of the high level feature in the data sample to the sum of the expected background and signal events $N(R) + N(S)$ in the data sample $\mathcal{D}$

$$n_D = \frac{N(R) + N(S)}{\mathcal{N}_{\mathcal{D}}} h_D \tag{5.5}$$

where h_D is the histogram of the high level feature in the data sample $\hat{\mathcal{D}}$

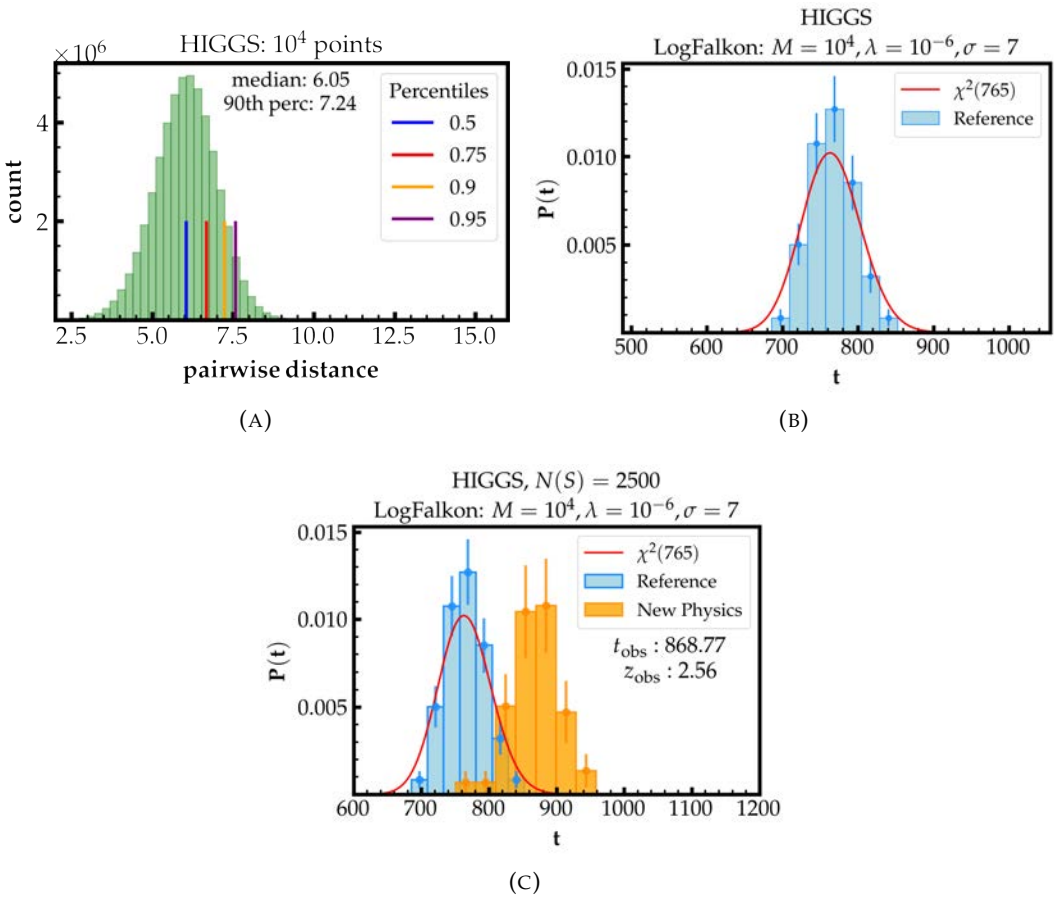


FIGURE 5.4: (A)-(B) Tuning LogFalkon on the HIGGS dataset, (C) Results with injected signal

- the predicted differential distribution is obtained according to the parametrized alternative hypothesis (3.19) by re-weighting the relevant high-level feature x_{hl}

$$n(x_{hl}|\hat{w}) = n(x_{hl}|R)e^{f(x,\hat{w})} \tag{5.6}$$

where x_{hl} is taken from the augmented version of the reference sample $\hat{\mathcal{R}}$, whereas x represents the corresponding low level features and f the learned function. Notice that in practice the quantity $n(x_{hl}|R)$ is computed as the histogram of the high level feature x_{hl}, and $e^{f(x,\hat{w})}$ is used as the weight of each point towards the count. The obtained histogram is denoted by $n_{\hat{w}}$.

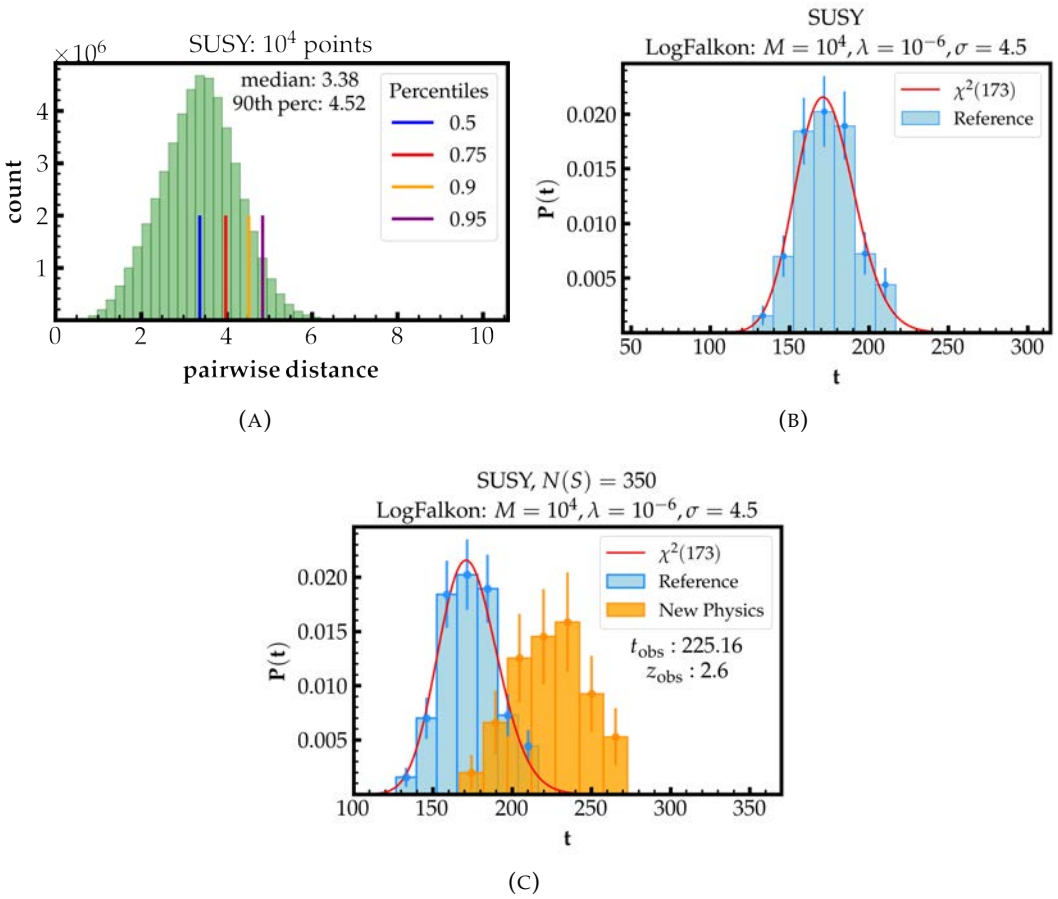
Given the above quantities, it is possible to compute:

- the "true" density ratio considered as ground truth, estimated from the ideal dataset $\mathcal{I}$ (the yellow curve in Figure 5.9)

$$\frac{n_{NP}}{n_R} \tag{5.7}$$

- the empirical density ratio estimated from the data sample $\mathcal{D}$ at hand (the red curve in Figure 5.9)

$$\frac{n_D}{n_R} \tag{5.8}$$



(A) (B)

(C)

FIGURE 5.5: (A)-(B) Tuning LogFalkon on the SUSY dataset, (C) Results with injected signal

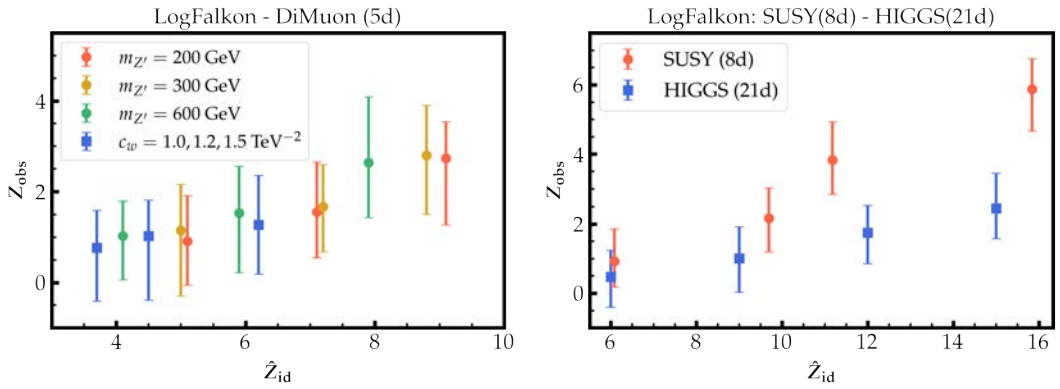
- the predicted density ratio obtained using predictions from the machine learning model after training (the blue curve in Figure 5.9)

$$\frac{n_{\hat{w}}}{n_R} \tag{5.9}$$

In Figure 5.9 it is possible to notice a peak in the ideal and in the toy density ratio which is well approximated by the learned density ratio. This means that, provided a sufficiently large observed signficance, it is possible to inspect the learned density ratio to localize the discrepancy and eventually apply further model-dependent analysis to correctly investigate the nature of the discrepancy.

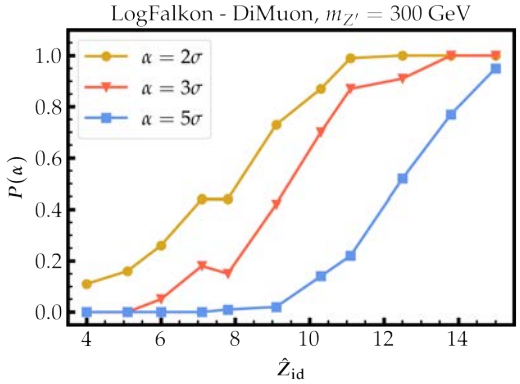
5.4 Additional experiments

In this section, additional experiements are presented. In particular, experiments with Falkon show the possibility of using the mean square loss as an alternative to the binary cross entropy loss. Experiments with neural networks are aimed to reproduce some of the work by D'Agnolo et al., 2021 and to extend their results to the use of the cross entropy loss.



(A) (B)

FIGURE 5.6: Observed significance against estimated ideal significance with low-level input features.



(A)

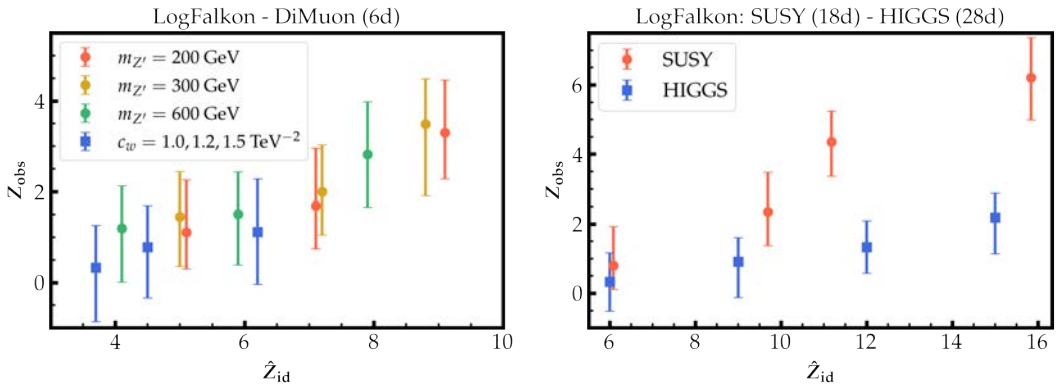
FIGURE 5.7: Probability of finding a evidence $\alpha = 2\sigma, 3\sigma, 5\sigma$ for new physics as a function of the ideal significance.

5.4.1 Experiments with Falkon

Falkon has been tuned for all the datasets analogously to what have been done with LogFalkon. The results are very close to those obtained with LogFalkon. Some examples are shown in Figure 5.10.

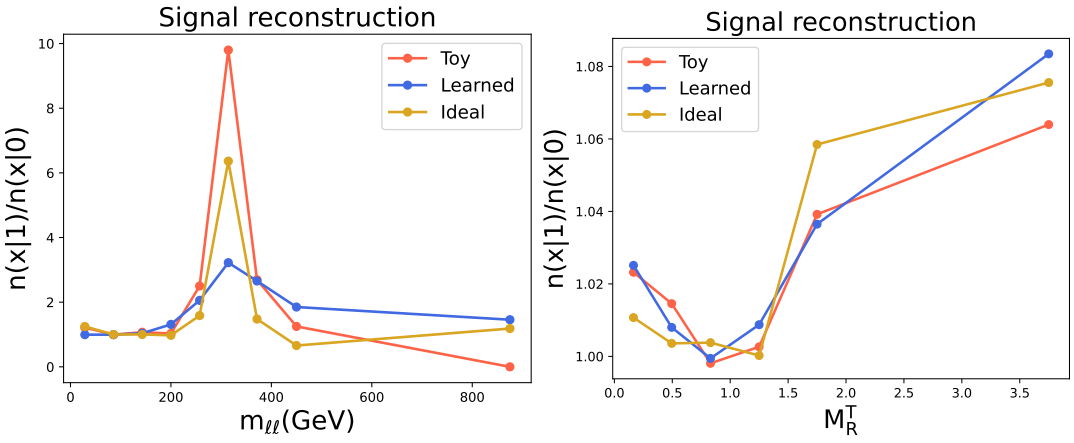
An experiment has been conducted in order to evaluate the impact of the random selection of Nystrom centers on the global performance. Two observations are made: (a) the sampling of Nystrom centers is uniform with respect to the distribution of the input points, (b) the dataset is dominated by background events. The following hypothesis is drawn: if the New Physics signal is located in the tail (low probability region) there will be a few Nystrom centres sampled in that region and, therefore, a loss of performance with all other parameters being equal (i.e., we expect a lower median z-score compared to the case in which the signal is in a high probability region).

The reference distribution is the same exponential distribution of the univariate data (Eq. 5.1). The size of the reference sample has been fixed to $\mathcal{N}_{\mathcal{R}} = 2 \times 10^5$, the number of background events to $N(R) = 2 \times 10^3$. Five different signal components have been considered: each of them is sampled from a Normal distribution $\mathcal{N}(\mu, \sigma)$ with $\sigma = 0.02$ and $\mu = [0.2, 0.4, 0.9, 0.95, 0.99]$. The Falkon parameters are $\sigma =$



(A) (B)

FIGURE 5.8: Observed significance against estimated ideal significance with all input features (including high-level features).



(A) (B)

FIGURE 5.9: Examples of reconstructed density ratios as a functions of high-level features (not given as inputs) for the DIMUON (left) and SUSY (right) datasets with new physics components in the data. Note that the SUSY dataset is normalized.

$0.3, \lambda = 10^{-9}, M = 500$.

The number of signal events $N(S)$ for each type of signal has been set in order to obtain an ideal median z-score equals to 5 sigmas. In order to keep a fixed z_{id} fixed we have to decrease S as μ increases. This makes finding anomalies more difficult for the model as $N(S)/N(R)$ becomes very small and therefore a model independent approach is disadvantaged with respect to model dependent approaches.

The pseudocode used to compute the distribution of the ideal test statistic is reported in Algorithm 5, whereas the pseudocode used to compute the number of signal events corresponding to a given z-score is reported in Algorithm 6.

The distribution of the ideal test statistic in the reference hypothesis is calculated with 10 million reference data samples, whereas in the new physics hypothesis 300 data samples have been used.

Figure 5.11 shows the distribution of the test statistic at the end of the training procedures. The initial hypothesis turns out to be false. Indeed, moving the signal towards a low probability region does not cause any loss of performance. On the

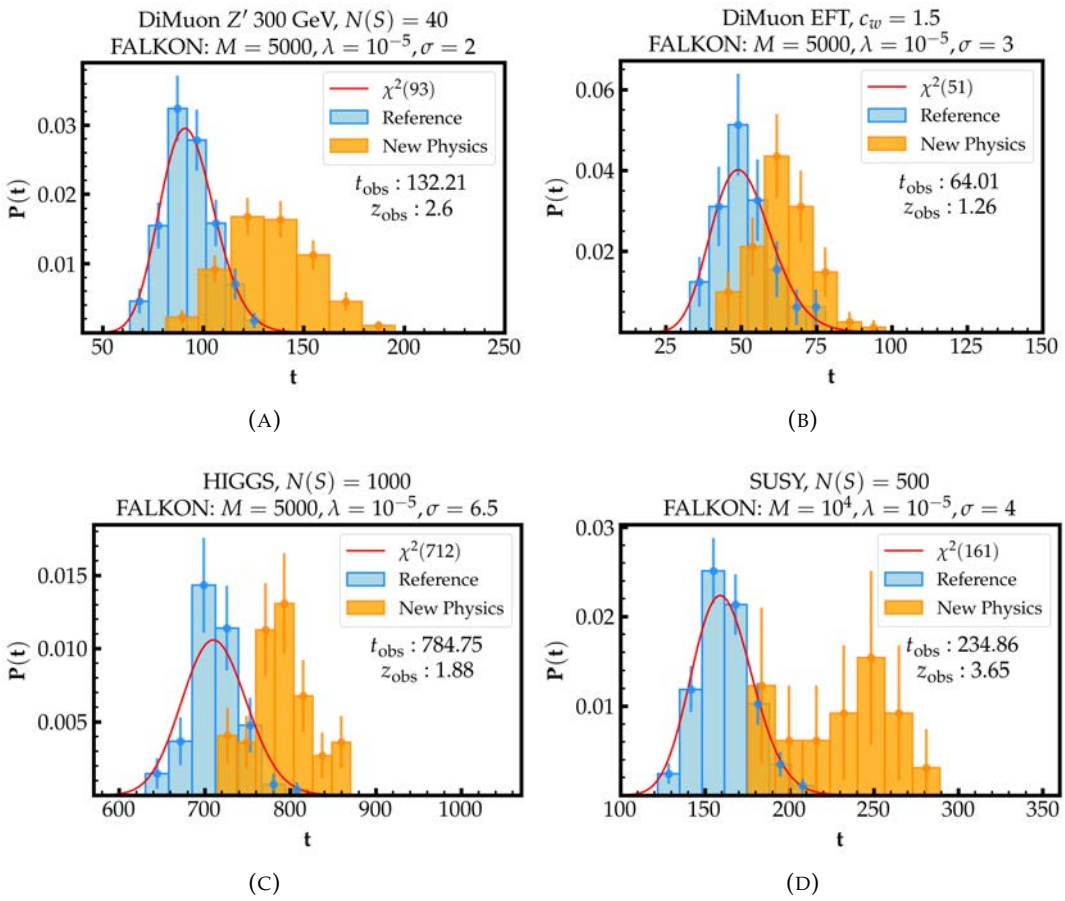


FIGURE 5.10: Results obtained with Falkon on the different datasets.

contrary, moving the mean of the signal from $\mu = 0.2$ to $\mu = 0.9$ produces an increase in the observed median z-score (from 0.45 to about 1.5).

It is difficult to draw conclusions for the cases in which the signal is at the limit of the domain ($\mu = 0.95$, $\mu = 0.99$) since about 50% of the observed t are 'nan' values, meaning that the algorithm becomes unstable. The presence of 'nan' values is also registered in all the other experiments but less significantly (about 5-15 'nan' out of 300 values). We observe that Falkon is typically more unstable than LogFalkon. This could be due to how the test statistics is calculated in Eq. 4.23, in which the numerator and the denominator of the density ratio is explicitly calculated and thus more prone to numerical errors. One possible way to alleviate this problem may be to take a larger λ (more regularized model).

For each type of signal, the experiments has been repeated with $M = 2000$ but no interesting increase in performance is observed with the increase of M.

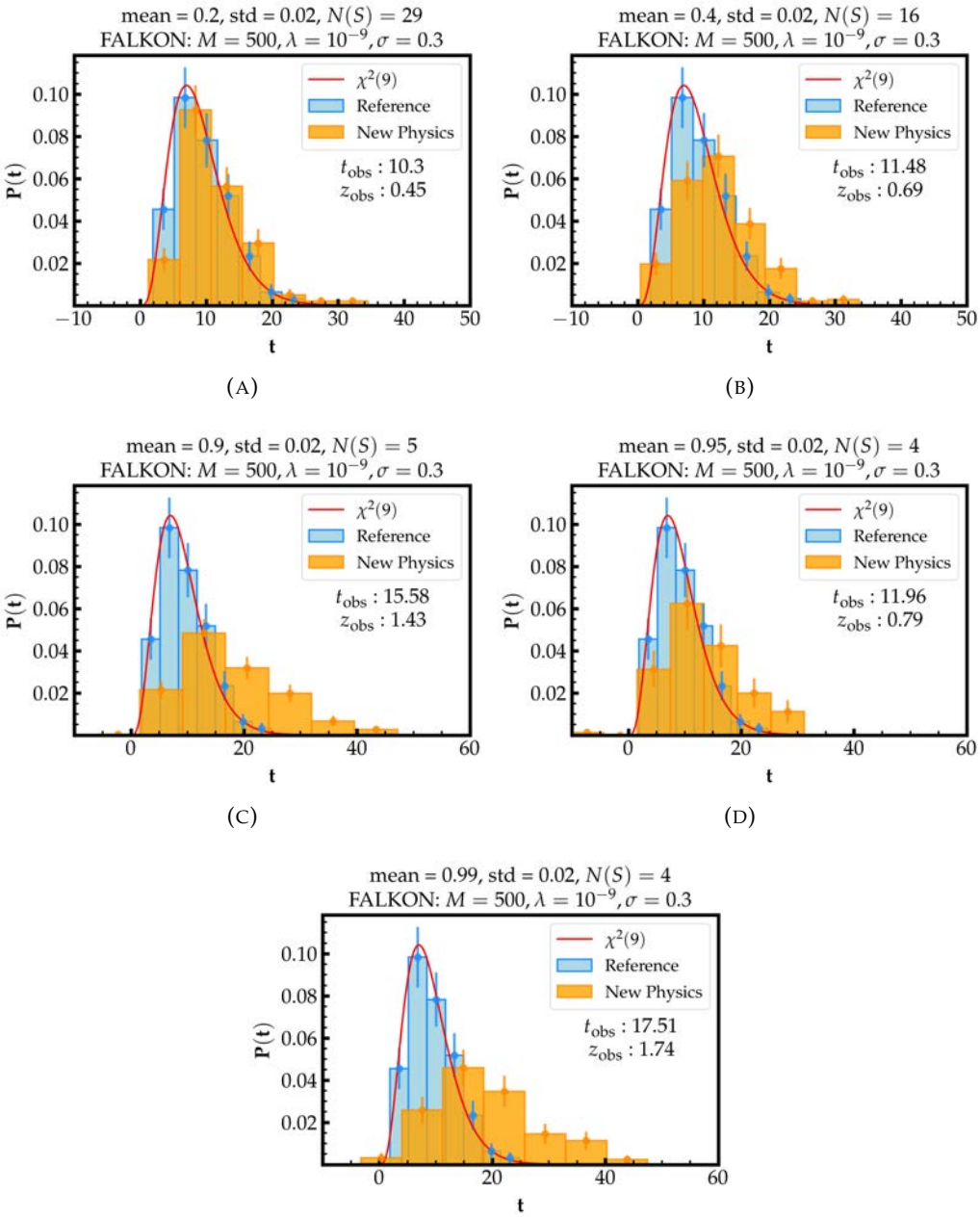


FIGURE 5.11: Distribution of the test statistic at varying mean of the signal with $N(S)$ corresponding to $z_{id} = 5$.

5.4.2 Experiments with neural networks

Neural networks have been implemented in Tensorflow (Abadi et al., 2015). The following notation will be used to describe a neural architecture:

$$(i, h_1, h_2, \ldots, h_L, o) \tag{5.10}$$

Algorithm 5: Compute $p(t_{\text{id}}|\,\text{R})$, $p(t_{\text{id}}|\,\text{NP})$, z_{id}

Input
- $p(x|R)$: reference distribution
- $p(x|S)$: signal distribution
- B : number of background events in the data sample
- S : number of signal events in the data sample

Parameters
- n_{REF} : number of toy data samples used to estimate $p(t_{\text{id}}|\,\text{R})$
- n_{NP} : number of toy data samples used to estimate $p(t_{\text{id}}|\,\text{NP})$
- n_{bins} : number of bins used in the hisograms of $p(t_{\text{id}}|\,\text{R})$, $p(t_{\text{id}}|\,\text{NP})$

Output
- $p(t_{\text{id}}|\,\text{R})$
- $p(t_{\text{id}}|\,\text{NP})$
- median z_{id}

Pseudo-code

1. Create n_{REF} data samples $\mathcal{D}_i^{\text{R}} = \left\{ (x_j)_{j=1}^{B} \sim p(x|R) \right\}, i = 1, \ldots, n_{\text{REF}}$
2. Create n_{NP} data samples
   $\mathcal{D}_i^{\text{NP}} = \left\{ (x_j)_{j=1}^{B} \sim p(x|R) \right\} \bigcup \left\{ (x_k)_{k=1}^{S} \sim p(x|S) \right\}, i = 1, \ldots, n_{\text{NP}}$
3. $p(t_{\text{id}}|\,\text{R}) \xleftarrow{\text{normalized histogram with } n_{\text{bins}}} 2\left[\sum_{x \in \mathcal{D}_i^{\text{R}}} \log\left(1 + \frac{S\,p(x|S)}{B\,P(x|R)}\right) - S \right], i = 1, \ldots, n_{\text{REF}}$
4. $p(t_{\text{id}}|\,\text{NP}) \xleftarrow{\text{normalized histogram with } n_{\text{bins}}}$
   $2\left[\sum_{x \in \mathcal{D}_i^{\text{NP}}} \log\left(1 + \frac{S\,p(x|S)}{B\,P(x|R)}\right) - S \right], i = 1, \ldots, n_{\text{NP}}$
5. Compute median z_{id}

with i size of the input layer, $h_1, h_2, \ldots, h_L$ size of the L hidden layers, o size of the output layer. The total number of parameters in the network is

$$N_{\text{par}} = \sum_{i=1}^{L+1} a_i(a_{i-1} + 1) \tag{5.11}$$

with a_i being a generic layer. This quantity is used to determine the degrees of freedom of the target χ^2 distribution used during hyperparameter tuning (as described in section 4.5).

All the experiments with neural networks have been run on a farm of parallel CPUs equipped with dual Intel 64 bit processors. It is worth noting that parallelization is essential in order to conduct experiments in a reasonable amount of time. Using a single machine is not a feasible choice (see Table 5.1).

Univariate data A (1,4,1) neural network has been trained on univariate data to reproduce the results in D'Agnolo and Wulzer, 2019 with the loss function ℓ_{npl} (Eq. 4.8) and different values of the weight clipping parameter W_{clip}. The compatibility with χ^2_{13} has been tested following the hyperparameter tuning procedure described

Algorithm 6: Compute S given z_{id}

Input
- $p(x|R)$: reference distribution
- $p(x|S)$: signal distribution
- B : number of background events in the data sample
- z^*_{id} : target ideal z-score

Parameters
- S_{list} : a list of S values to be tested
- n_{REF} : number of data samples used to estimate $p(t_{\text{id}}|\,\text{R})$
- n_{NP} : number of data samples used to estimate $p(t_{\text{id}}|\,\text{NP})$

Output
- S

Pseudo-code

1. Compute t^*_{id} corresponding to z^*_{id}:

$$t^*_{\text{id}} = \Gamma^{-1}\big(\Phi(z^*_{\text{id}})\big)$$

where $\Gamma^{-1}(\cdot)$ is the inverse of the c.d.f. of $p(t|\,\text{R})$, $\Phi(\cdot)$ is the c.d.f. of the standard Normal distribution.

2. Find the minimum S such that the estimated median z_{id} is greater than or equals to the target z^*_{id}:

$$\arg\min_{S\in S_{\text{list}}} \Big\{ \text{median}\, z_{\text{id}}\Big(p(x|R), p(x|S), B, S, n_{\text{REF}}, n_{\text{NP}}\Big) \;\Big|\; \text{median}\, z_{\text{id}}(S) \geq z^*_{\text{id}} \Big\}$$

where median z_{id} is computed according to the Algorithm 5 and $\Big(p(x|R), p(x|S), B, S, n_{\text{REF}}, n_{\text{NP}}\Big)$ are its arguments.

in Algorithm 2 (section 4.4). The results are shown in Figure 5.12. A total of $N_{\text{toy}} = 100$ experiments for each configuration of the weight clipping parameter W_{clip} have been performed. The 95th percentile of the test statistic reaches a plateau for $W_{\text{clip}} = 35, 40, 100$. The best χ^2 compatibility is provided by $W_{\text{clip}} = 100$.

After the choice of the weight clipping has been made, the model has been tested with the three different types of univariate signals. A number of experiments $N_{\text{toy}} = 300$ has been chosen for each signal type. The size of the reference sample is $\mathcal{N}_{\mathcal{R}} = 2 \times 10^5$, the number of expected background events is $N(R) = 2 \times 10^3$, the number of expected signal events is $N(S) = 10$ for the signal S_1, $N(S) = 35$ for the signal S_2, $N(S) = 90$ for the signal S_3. The results are shown in Figure 5.13 (A)-(B)-(C).

The reference distribution approximates a χ^2_{13} distribution. The first signal and the third signals are well distinguished from the reference with a z-score around three sigmas. The signal in the bulk is not detected since the z-score is lower than one. In this case, more signal events are to be considered in order to get a larger significance (e.g., $N(S) = 90$). An example of the learned likelihood ratio is shown in Figure 5.14. It is possible to notice a bump in the learned ratio in correspondence of the zone of the signal, which is centered around 0.8.

The same experiment has been performed by changing the loss function to the weighted binary cross entropy loss ℓ_{wbce} (Eq. 4.11). The results are shown in the bottom row of Figure 5.13. They are very similar to the ones obtained with the use

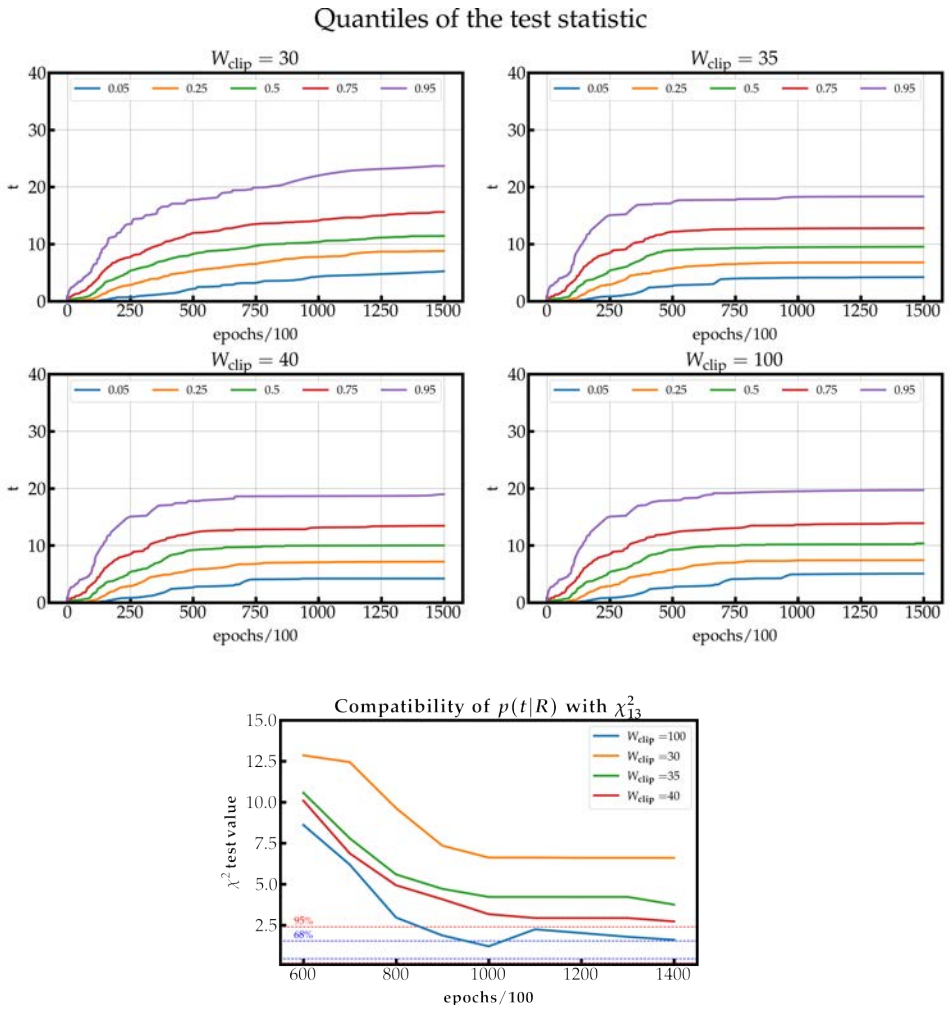


FIGURE 5.12: Tuning the weight clipping parameter on the univariate data: (Top row) Quantiles of the test statistic over training epochs (Bottom row) χ^2 compatibility check using Pearson's χ^2 test

of the loss ℓ_{npl}. The difference is that no hyperparameters have been tuned with the weighted binary cross entropy loss. This means that using the weighted binary cross entropy loss in the univariate case has the advantage of saving the computational times needed for the hyperparameter tuning procedure and at the same time preserving the significance obtained with the custom loss ℓ_{npl}.

The average training time is 4 hours per single training (150k epochs).

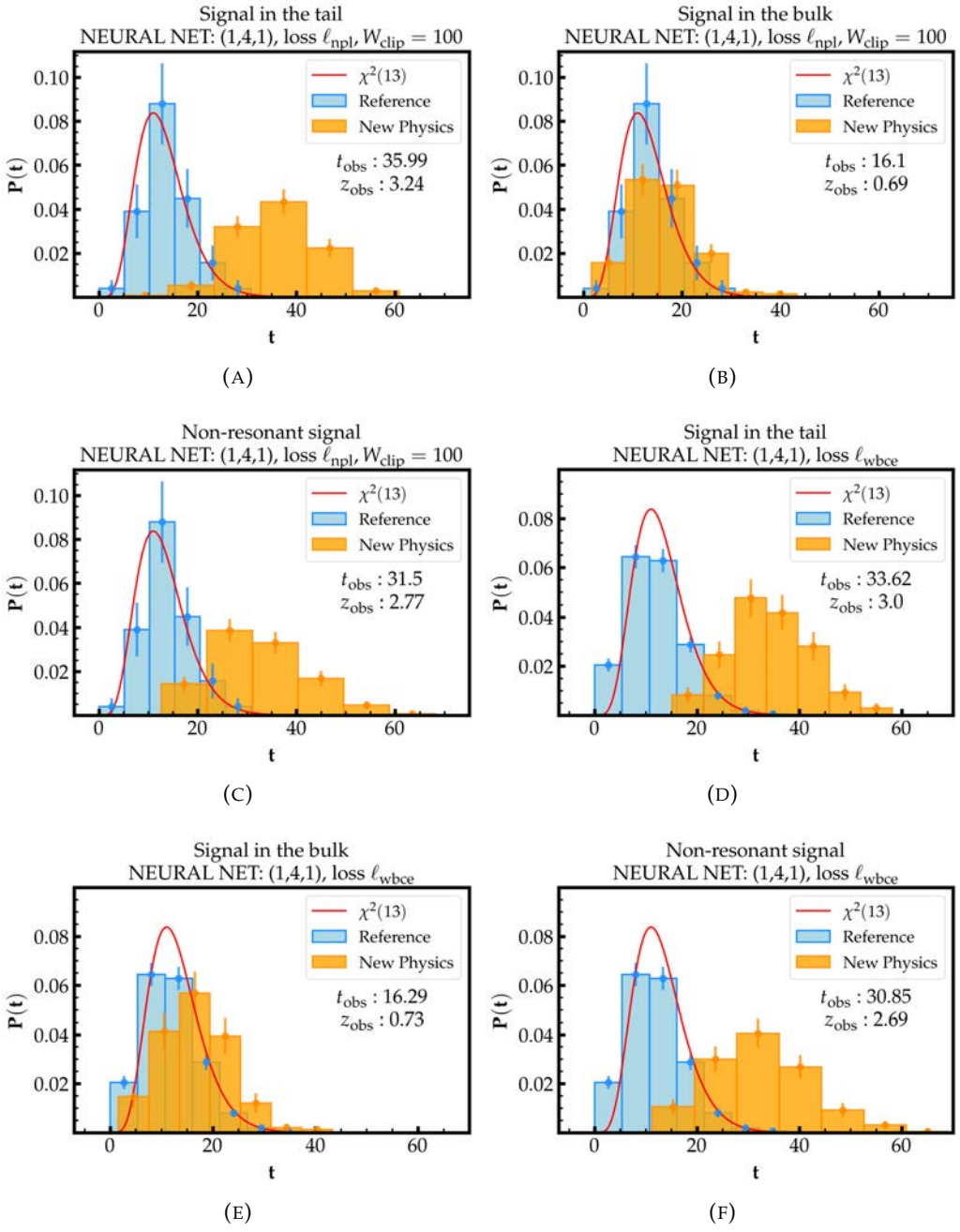


FIGURE 5.13: Distribution of the test statistic of the reference model vs the new physics model on the univariate data. (A)-(B)-(C) Results obtained with the loss ℓ_{npl} (D)-(E)-(F) Results obtained with the weighted binary cross entropy loss ℓ_{wbce}

Multivariate data A (5,5,5,5,1) architecture has been trained on the DiMuon dataset with the signal Z' 300 GeV. A number of $N_{\text{toy}} = 100$ experiments have been performed to check the χ^2 compatibility and the value of the weight clipping parameter $W_{\text{clip}} = 2.15$ suggested in the paper D'Agnolo et al., 2021.

The size of the reference sample has been set to $\mathcal{N}_{\mathcal{R}} = 10^5$, the number of expected background events to $N(R) = 2 \times 10^4$, the signal events to $N(S) = 40$. Training is stopped after 3×10^5 epochs. The result is shown in Figure 5.15. The reference

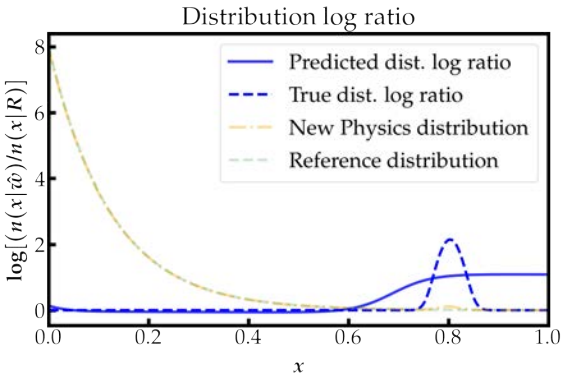


FIGURE 5.14: Example of learned likelihood ratio in the case of signal in the bulk

distribution is compatible with the χ^2 distribution with 96 degrees of freedom, as predicted by the number of parameters of the neural networks. A good sensitivity to the signal is registered, amounting to a z-score of almost 3 sigmas.

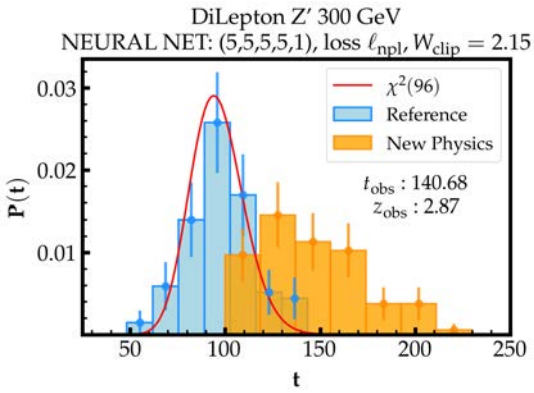


FIGURE 5.15: Distribution of the test statistic for the DiMuon dataset with the signal Z' 300 GeV obtained with the loss ℓ_{npl}

The same neural architecture has been trained with the weighted cross entropy loss ℓ_{wbce} (Eq. 4.11). The same experimental parameters have been used. This time, differently from the univariate case, an L2 regularization parameter has been also considered. The same tuning strategy used for the weight clipping has been applied to the L2 regularization parameter (i.e., the quantiles of the median test statistic over the training epochs are considered). The results are shown in Figure 5.16. The selected value is $\lambda = 0.11$ as it produces a plateau of the 95th percentile of the test statistic. The degrees of freedom of the compatible χ^2 distribution are determined with a Kolmogorov-Smirnov test (described in section 3.1.2).

Figure 5.17 shows the distributions of the test statistic. The reference distribution is compatible with a χ^2 distribution with 60 degrees of freedom. The observed significance is comparable to the one obtained with the loss ℓ_{npl}, amounting to more than 2 sigmas. Further improvements may be considered to obtain a better compatibility with the χ^2 distribution. In fact, some points of the histograms are located around zero, and this is due to an instability of the learning algorithm.

Neural networks have also been applied to the HIGGS and the SUSY datasets. The considered architecture has 2 hidden layers with 10 neurons per layer and a weight clipping $W_{clip} = 0.87$ for SUSY, whereas it has 5 layers with 6 neurons per

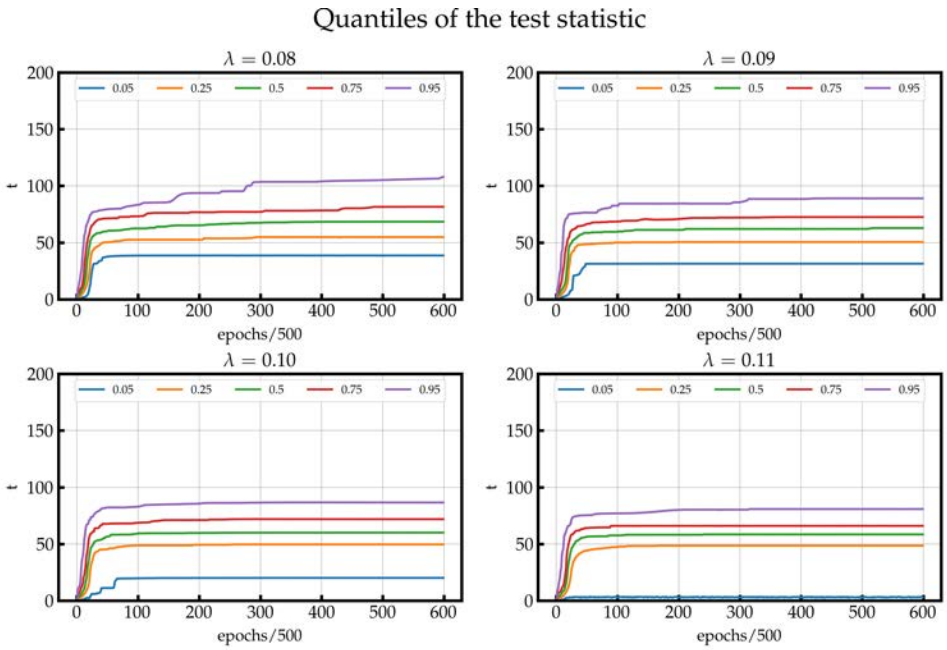


FIGURE 5.16: Tuning the L2 regularization parameter on the DiMuon dataset: quantiles of the test statistic over training epochs

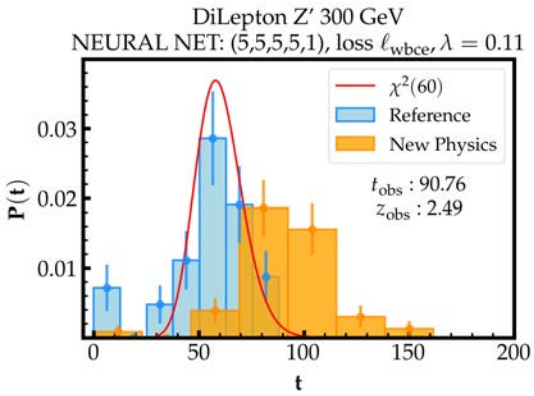


FIGURE 5.17: Distribution of the test statistic for the DiMuon dataset with the signal Z' 300 GeV obtained with the loss ℓ_{wbce}

layer and $W_{\text{clip}} = 0.65$ for HIGGS. Training is stopped after 3×10^5 epochs. The results are visualized in Figure 5.18(B) as the median observed significance against the estimated ideal significance with the model trained on low-level features only.

In all cases it is possible to notice a good correlation between the observed and the ideal significance. In the SUSY case an observed z-score $z_{\text{obs}} \sim 3\sigma$ is obtained for an ideal significance $\hat{z}_{\text{id}} > 10$, in the HIGGS case for $\hat{z}_{\text{id}} > 14$. This means that a smaller sensitivity is registered in higher dimensions.

5.5 Comparison of the machine learning models

In this section, a comparison of the two machine learning models used to search new physics - Falkon/LogFalkon and neural networks - is addressed.

Formally, neural networks are parametric models, i.e. models with a fixed parametric form for the prediction function which can be trained to learn the density

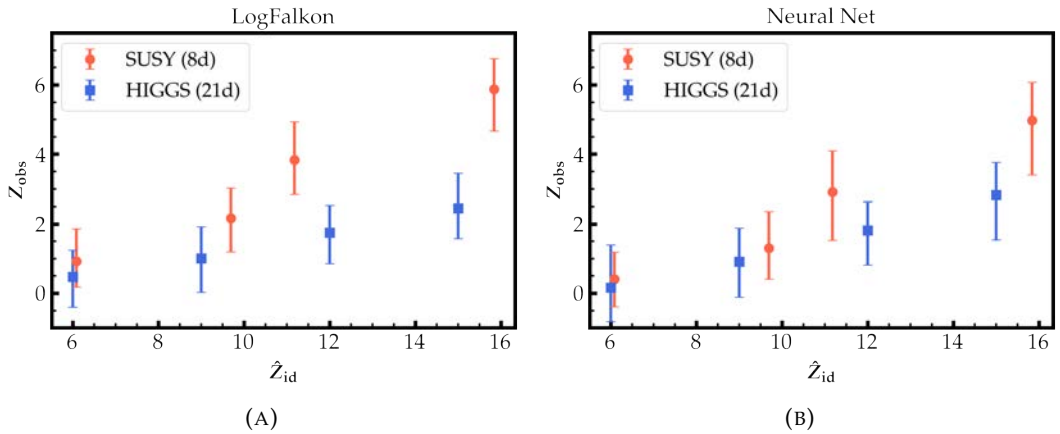


FIGURE 5.18: Observed significance against estimated ideal significance with low-level input features on the HIGGS and the SUSY dataset: (A) LogFalkon versus (B) Neural networks

ratio of Eq. 4.27 with either the custom loss ℓ_{npl} (Eq. 4.8) or the weighted cross entropy loss ℓ_{wbce} (Eq. 4.11). The test statistic can be directly computed if the loss ℓ_{npl} is used, whereas the Eq. 4.26 can be used if the loss ℓ_{wbce} is employed.

Falkon/LogFalkon is a scalable non-parametric approach which try to estimate the target function directly from data. With either the weighted cross entropy loss (LogFalkon) or the square loss (Falkon) it is possible to learn the same density ratio as the neural networks and to compute the test statistic with Eq. 4.26 (if the weighted cross entropy loss is used) or Eq. 4.23 (if the square loss is used).

Concerning the hyperparameter tuning procedure:

- neural networks have a specific procedure for hyperparameter tuning (see Algorithm 2 of section 4.4) in which a target χ^2 distribution - determined by the number of parameters of the neural network - is used to guide the choice of the neural architecture, the value of the weight clipping or the L2 regularization parameter and the number of epochs used to train the network.

- Falkon/LogFalkon has a different hyperparameter tuning procedure (see section 4.5) based on empirical curves that show the dependence of the median test statistic on the different hyperparameters of the method, namely the number of Nystrom centers, the regularization parameter and the bandwidth of the kernel function. An experiment has shown a weak dependence of the sensitivity of the method on the random uniform Nystrom centers sampling.

Concerning the performance, overall the two machine learning models give similar results. On the univariate case LogFalkon is slightly less sensitive to resonant new physics signal with observed z-scores on the two types of signals being 2.5, 2.9 against 3.2, 3.1, of the neural networks. On the other hand, LogFalkon shows a larger significance on the non-resonant signal with a z-score of 3.1 against 2.6 of the neural networks (D'Agnolo and Wulzer, 2019).

Figure 5.19 shows a comparison of the observed significance on the DiMuon dataset. On the left side, the results with neural networks from the paper D'Agnolo et al., 2021 are reported, whereas on the right the results obtained with LogFalkon are reproduced from Figure 5.6. It is possible to observe that the kernel approach is slightly less sensitive, especially in some cases. Still, a similar correlation is observed for the majority of points.

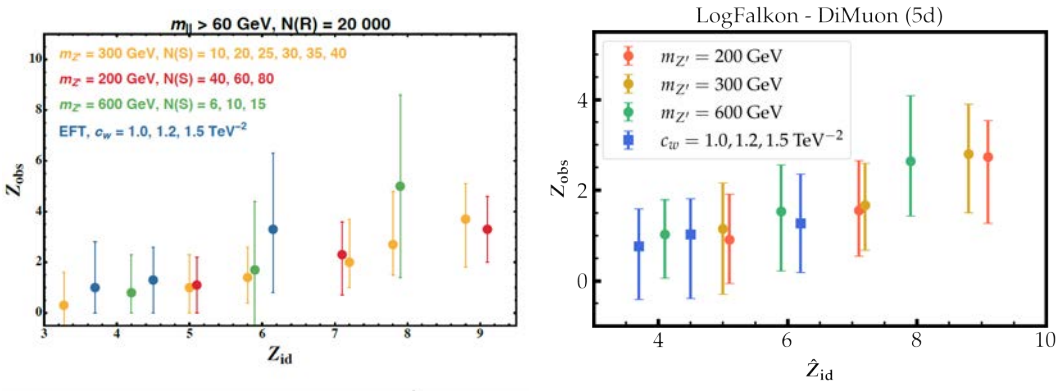
Model	DiMuon	SUSY	HIGGS
LogFalkon	88.7 ± 1.9 s	44.8 ± 1.5 s	89.7 ± 2.2 s
Neural networks	4.23 ± 0.73 h	73.1 ± 10 h	112 ± 9 h

TABLE 5.1: Average training times per single run with standard deviations (datasets with low level features only).

Figure 5.20 shows the probability of finding a discrepancy of at least α for a given value of $\hat{Z}_{id}$ for the dataset DiMuon with signal Z' 300 GeV. On the left there are the results reported from paper D'Agnolo et al., 2021 obtained with neural networks, whereas on the right there are the results obtained with LogFalkon. Also in this case it is possible to observe very similar results, since the curves are almost the same.

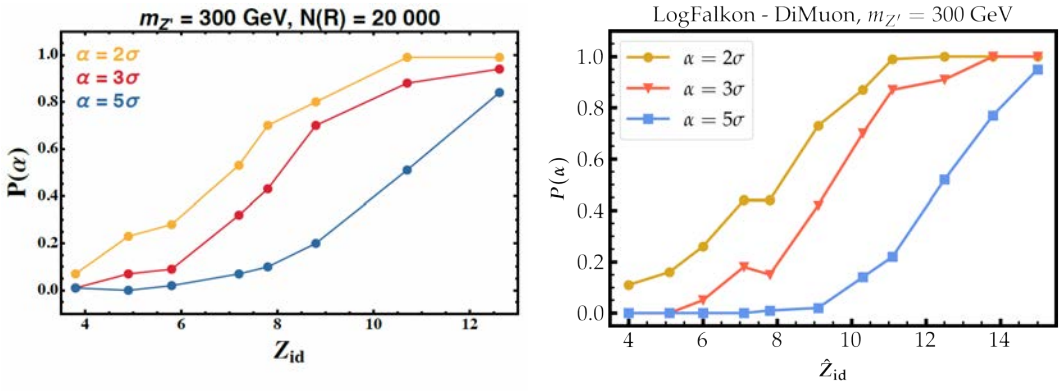
Figure 5.18 shows a comparison of the observed significance on the HIGGS and the SUSY datasets. It is possible to observe that in the HIGGS case the kernel approach gives a higher observed significance. On the other hand, when applied to the SUSY dataset the two models give almost identical results.

Concerning the average training times, the Table 5.1 demonstrates that Falkon/LogFalkon is orders of magnitude faster than neural networks. This allows to train the model efficiently on single GPU machines and ensures high scalability for multi-GPU systems (as shown in Meanti et al., 2020). On the other hand, neural networks require a farm of computers to run parallel experiments, whereas implementation on single machines is not feasible. Therefore, kernel methods presented in this thesis constitute an efficient approach to model-independent new physics searches, making the procedure easily deployable in real-case scenarios. After a straightforward tuning procedure, they allow to inspect experimental data with the same sensitivity as the neural networks with much less computational time and resources.



(A) (B)

FIGURE 5.19: Observed significance against estimated ideal significance with low-level input features on the DiMuon dataset. (A) Results from the paper D'Agnolo et al., 2021 (with neural networks) versus (B) LogFalkon



(A) (B)

FIGURE 5.20: Probability of finding a discrepancy of at least α for a given value of $\hat{Z}_{id}$ for the dataset DiMuon with signal Z' 300 GeV: (A) Results from the paper D'Agnolo et al., 2021 (with neural networks) versus (B) LogFalkon

Chapter 6

Conclusion

In this thesis a novel machine learning method for the discovery of new physics has been proposed, based on the work by D'Agnolo and Wulzer, 2019. A model-independent approach has been considered, i.e., no prior assumptions have been made on the nature of the new physics signals. The resulting algorithm represents a versatile tool for physicists which can be employed to analyse experimental data and look for any potential discrepancy from the reference models.

The main focus of this work is on computational efficiency with the core algorithm being powered by recent large scale implementations of kernel methods, namely Falkon and LogFalkon (Meanti et al., 2020), which are non-parametric models that can approximate any continuous function given enough data.

A classifier has been trained on simulated Standard Model or new physics datasets of increasing dimensionality to build a hypothesis testing procedure based on the likelihood ratio test. The compatibility of the data with the Standard Model has been assessed in order to detect the effective presence of new physics clues. Despite being formulated as a classification problem, there is no interest in measuring classification accuracy because the final goal is to fit the likelihood ratio and perform a hypothesis test. For this reason, performance assessment is done by comparing the ideal statistical significance with the observed significance.

An hyperparameter tuning procedure has been defined to balance the complexity of the model and to match the distribution of the test statistic under the null hypothesis with a χ^2 distribution. The parameters of the model have been tuned on all the datasets.

Experiments with LogFalkon have shown the ability of the model to detect the presence of new physics with a meaningful statistical significance. Furthermore, a good correlation between the observed and the ideal significance has been noticed in all the cases. Including high level features, i.e., features which can be extrapolated from the other ones, no particular boost of performance is observed, confirming the ability of the model to detect new phsyics clues with low-level features only.

Experiments with Falkon have shown the possibility of successfully using the mean square loss as an alternative loss to solve the same problem. In addition, an experiment has demonstrated the weak dependence of the random sampling of Nystrom centers, i.e., the data points on which the approximation function is constructed, on the performance of the algorithm.

Experiments with neural networks based on the work by D'Agnolo et al., 2021 have shown the possibility of using two different loss functions to solve the problem. Comparable performance with respect to Falkon/LogFalkon is registered. However, concerning the average training times, Falkon/LogFalkon is orders of magnitude faster than neural networks. Falkon/LogFalkon can be efficiently trained on single GPU machines while possessing high scalability for multi-GPU systems (Meanti et al., 2020). In contrast, the neural network implementation crucially relies on per

experiment parallelization, hence the need for large scale resources such as CPU clusters. For this reason, our proposed method with Falkon/LogFalkon represents an extremely efficient alternative to neural networks which can be easily used by physicists in real-case scenarios.

Future work can include:

- the creation of a complete pipeline for data analysis. Although our method is sensitive to any new physics signals, model-dependent strategies are likely to perform better if precisely tuned on the specific signal at hand. Therefore, a complete pipeline for data analysis should include the use of our method as a preliminary analysis, followed by a model dependent strategy which should be focused on the discrepancy detected by our model.

- a different hyperparameter tuning strategy. Although working in practice, the current procedure is based on heuristics and a more theory-grounded procedure should be considered to make the results more robust. For instance, it is possible to investigate how to predict a target χ^2 distribution given the Falkon/LogFalkon hyperparameters. Another possibility would be to find a more principled way to relate Falkon/LogFalkon hyperparameters to physical quantities. This could also allow the introduction of explicit quantities to be optimized, opening to the possibility of applying modern optimization techniques for the selection of the hyperparameters.

- a more refined strategy for Nystrom centers selection. For instance, it is possible to consider the strategy proposed in the work Rudi et al., 2018. It would be interesting to evaluate the impact of a more careful selection procedure on the performance of the method.

- treatment of systematic uncertainties in the prediction of the reference model. This aspect was successfully addressed on a recent work (d'Agnolo et al., 2021) in the context of the neural network implementation.

- parallelization on multiple GPU machines to further speed up the training procedure.

- apply this analysis strategy to other applications, beyond the search for new physics, and to other domains.

Appendix A

Datasets visualization

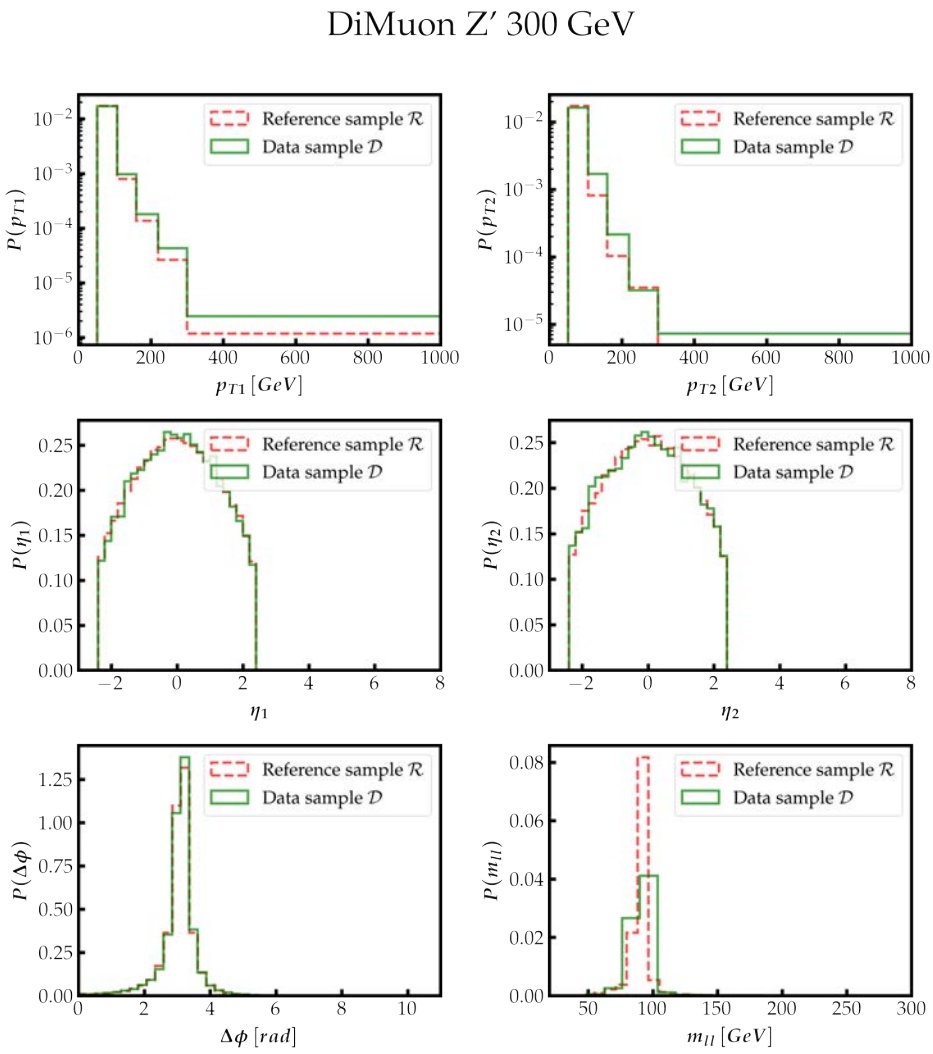


FIGURE A.1: Visualization of the DiMuon dataset

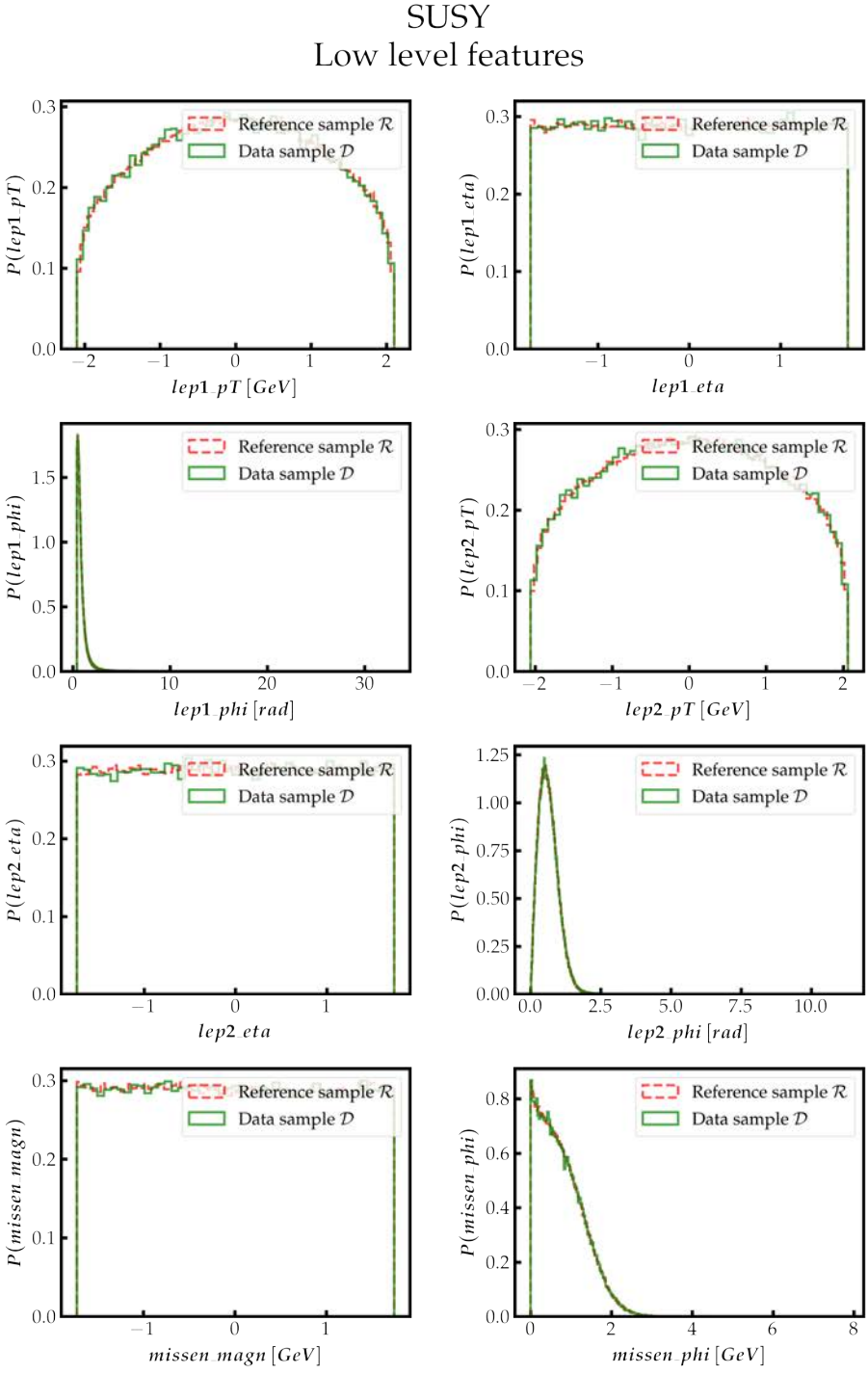


FIGURE A.2: Visualization of the low level features of the SUSY dataset

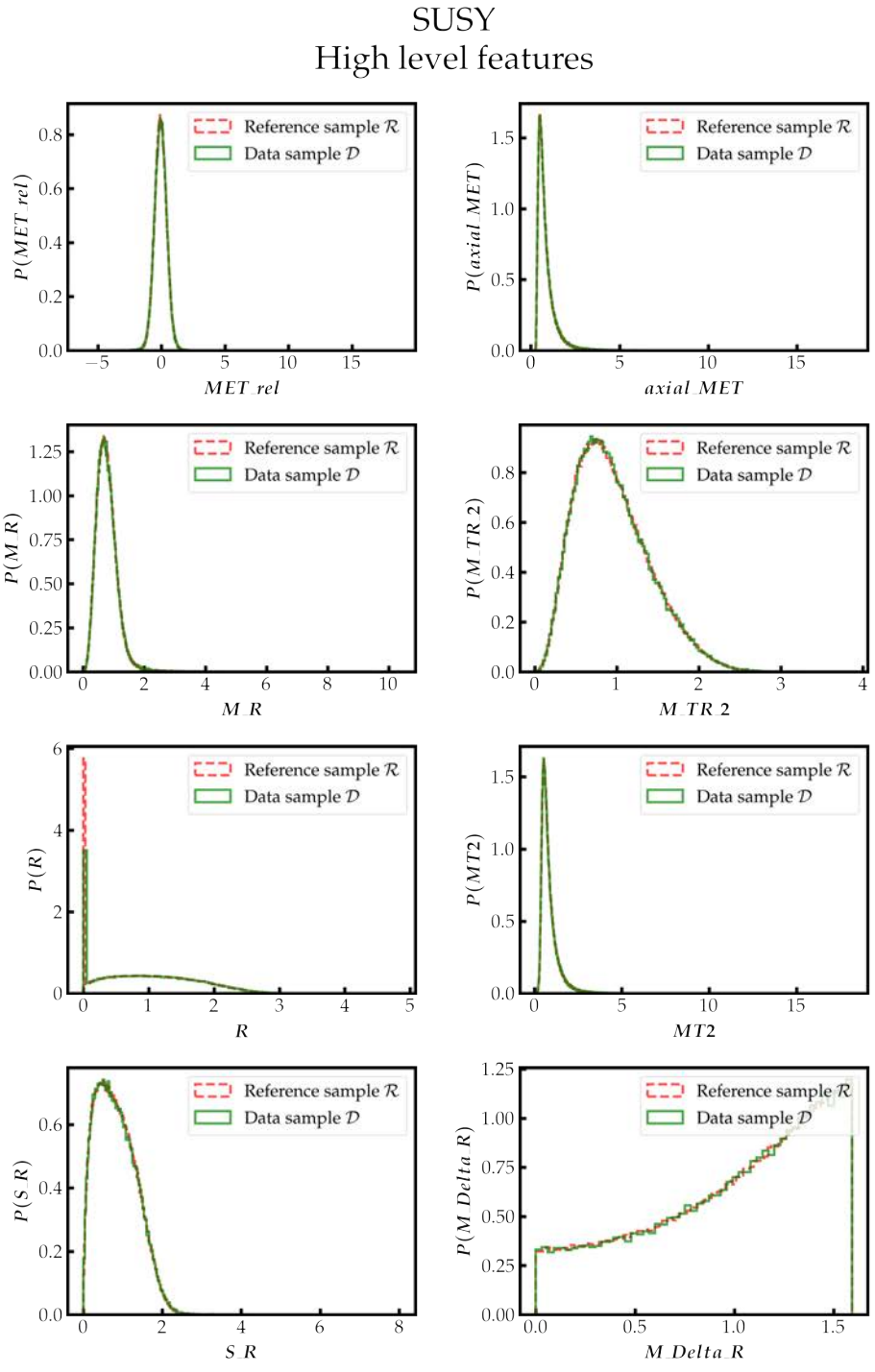


FIGURE A.3: Visualization of the high level features of the SUSY dataset

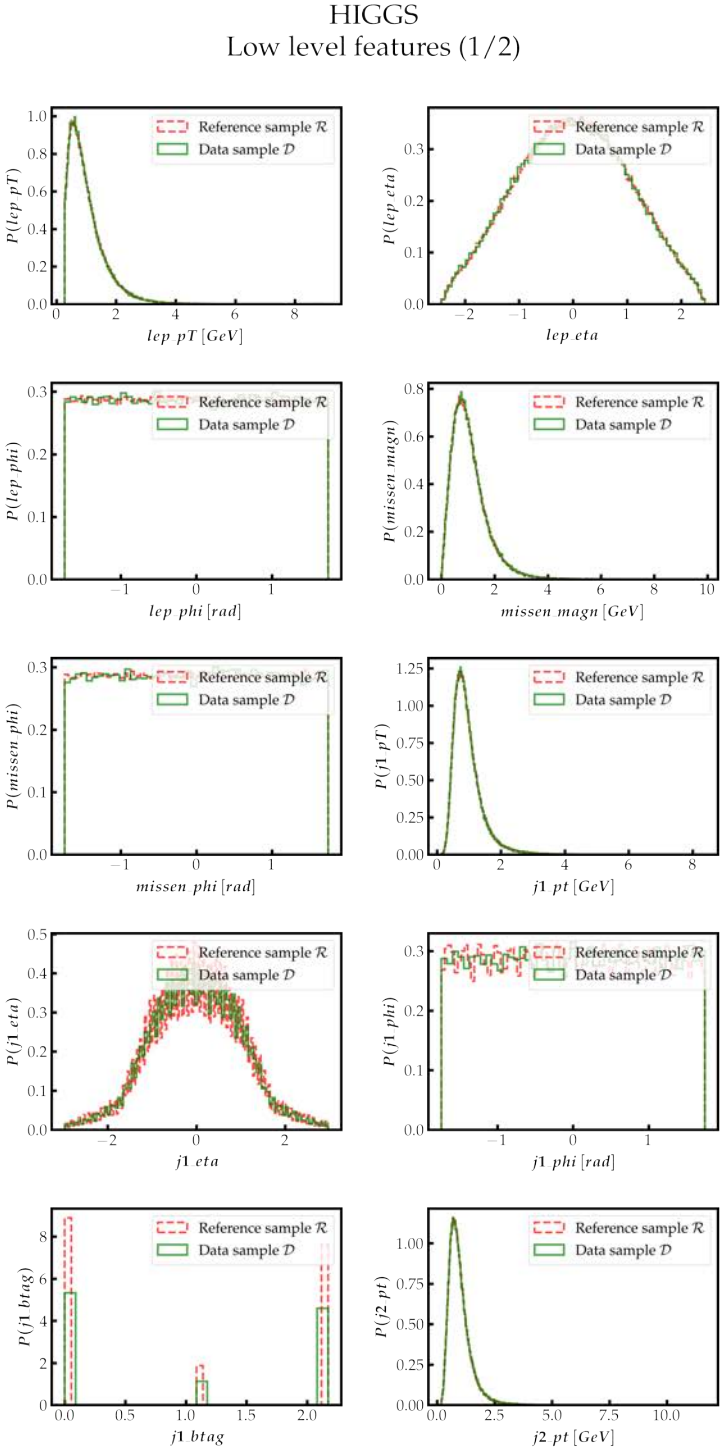


FIGURE A.4: Visualization of the first set of the low level features of the HIGGS dataset

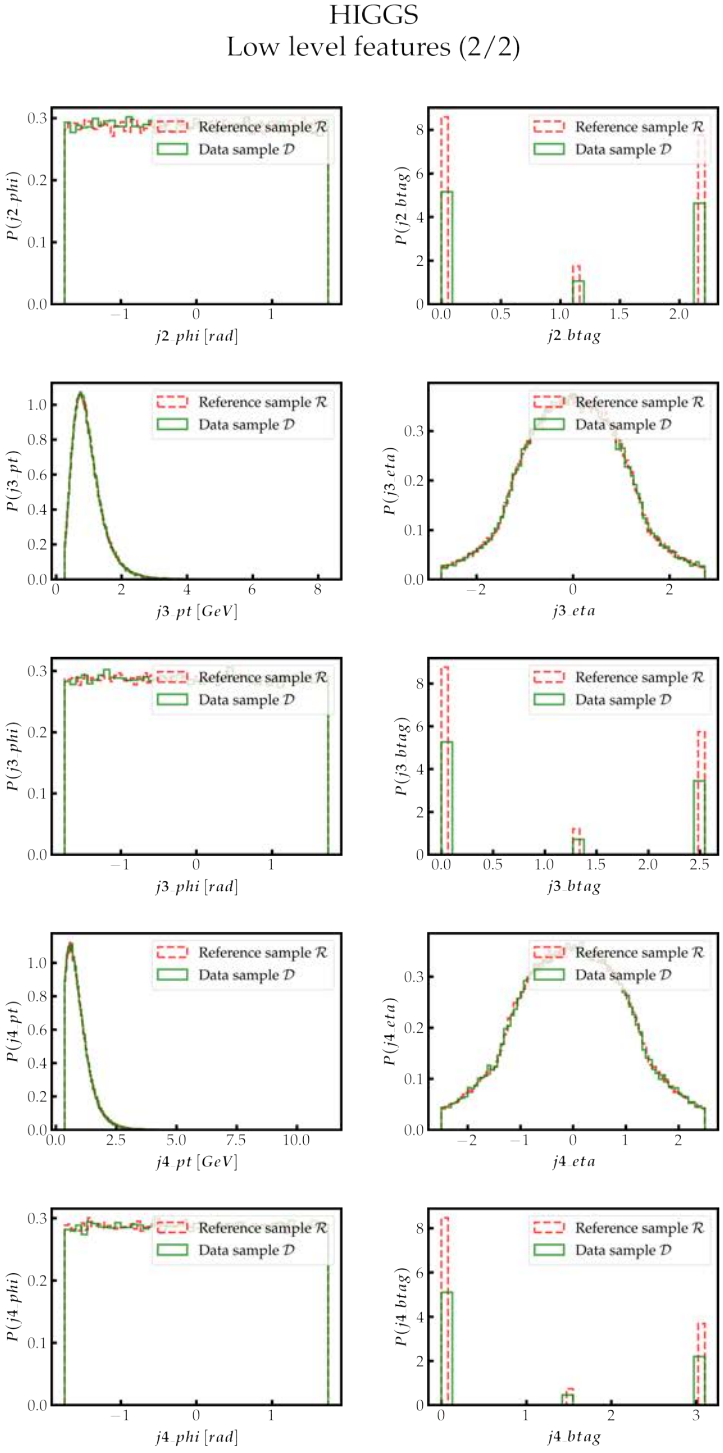


FIGURE A.5: Visualization of the second set of the low level features of the HIGGS dataset

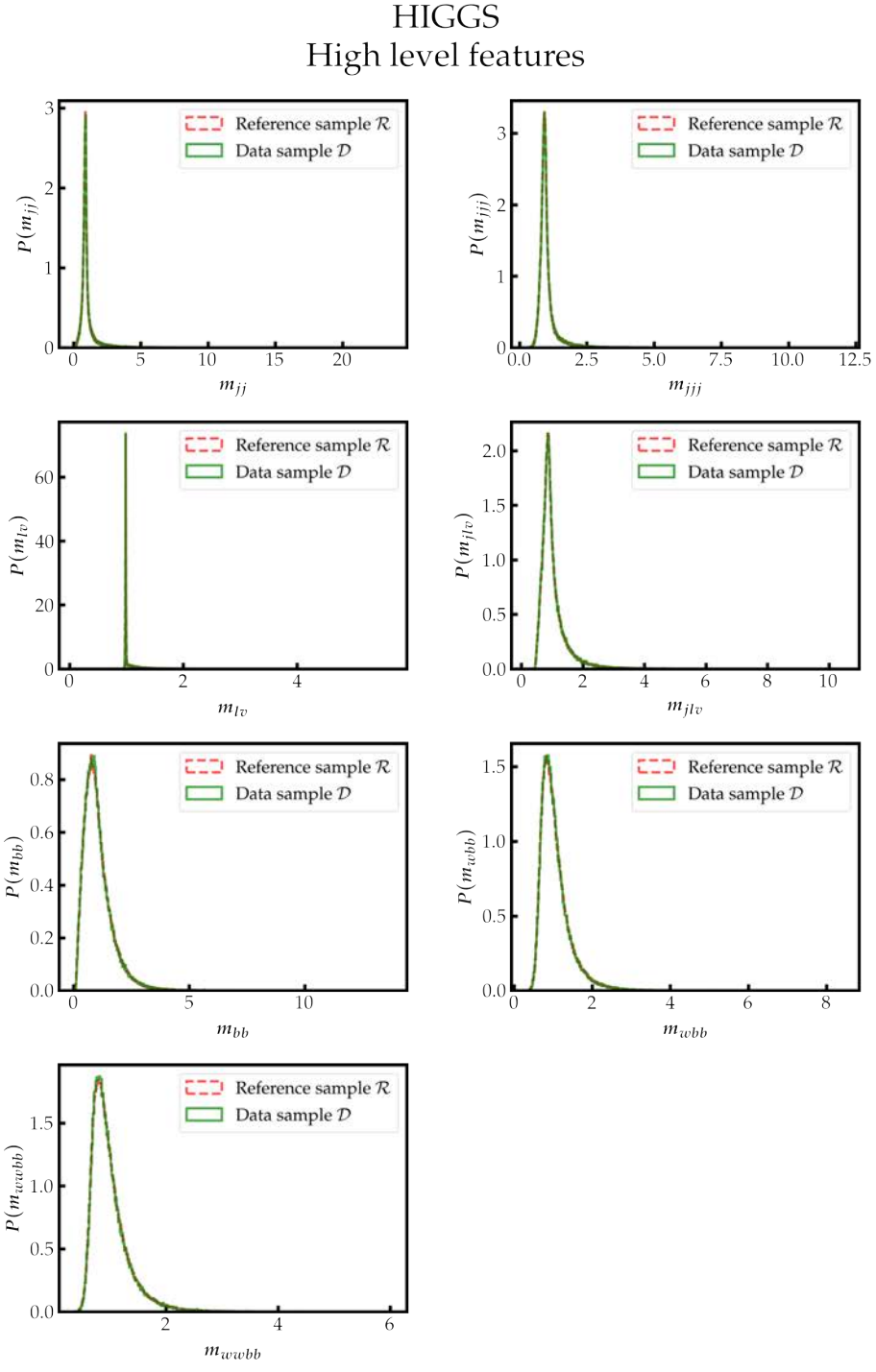


FIGURE A.6: Visualization of the high level features of the HIGGS dataset

The Figures have been obtained with the following parameters:

- DiMuon: $\mathcal{N}_{\mathcal{R}} = 10^5, N(R) = 2 \times 10^4, N(S) = 40$
- SUSY: $\mathcal{N}_{\mathcal{R}} = 5 \times 10^5, N(R) = 10^5, N(S) = 650$
- HIGGS: $\mathcal{N}_{\mathcal{R}} = 5 \times 10^5, N(R) = 10^5, N(S) = 2500$

Appendix B

Description of the code

The package `ml4hep` is available at the link https://github.com/gvlos/ml4hep. It can be used to reproduce the experiments of this thesis. It is possible to train Falkon/LogFalkon as well as neural networks on local machines or on a farm of computers.

The repository is organized as follows:

- the folder `notebook` contains a few Jupyter Notebook and a couple of python scriptx:
  - `data.ipynb` contains the code used for data analysis and to generate the pictures of the various datasets
  - `data_generation.py` contains the function which can be used to generate multiple h5 files from a csv file. It can be used to convert the HIGGS and the SUSY dataset in a format compatible with the library.
  - `falkon_tuning.ipynb` contains some steps of the hyperparameter tuning procedure with Falkon
  - `fig.mplstyle` is a Matplotlib style file used to define some common properties of the figures
  - `fig_utils.py` contains some utilities for the generation of the figures
  - `test_figures.py` contains the code used to generate the histograms of the test statistics
- the folder `unified_code` contains used to run experiment:
  - `config.ini` is used to specify the parameters of the experiment
  - `main.py` is the main used to launch a single experiment
  - `hyperparam_tuning.py` is used to launch multiple experiments
  - `requirements.txt` is the file containing the Python libraries required to run the code

The instructions to run the code are reported in the repository. Basically, there are three steps:

1. Editing the `config.ini` files with the desired experimental parameters
2. Activating the virtual environment
3. Launching a single experiment with `main.py` or multiple experiments with `hyperparam_tuning.py`

There are four execution modes:

- on the local machine with sequential execution

- on the local machine with parallel execution
- on the Condor system of CERN[1] (an account is required)
- on the LSF system of the INFN farm of Genova[2] (an account is required)

The output is generated in the specified output folder and consists of multiple files as descripted in the Github documentation.

The code has been tested on Ubuntu 18.04.

[1] https://batchdocs.web.cern.ch/index.html

[2] https://web.ge.infn.it/calcolo/joomla/8-calcolo/5-farm-di-calcolo

Bibliography

Abadi, Martín et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

Abdel-Hamid, Ossama et al. (2014). "Convolutional neural networks for speech recognition". In: *IEEE/ACM Transactions on audio, speech, and language processing* 22.10, pp. 1533–1545.

Albertsson, Kim et al. (2018). "Machine learning in high energy physics community white paper". In: *Journal of Physics: Conference Series*. Vol. 1085. 2. IOP Publishing, p. 022008.

ATLAS, Collaboration (2012). "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC". In:

Baldi, Pierre, Peter Sadowski, and Daniel Whiteson (2014). "Searching for exotic particles in high-energy physics with deep learning". In: *Nature communications* 5.1, pp. 1–9.

Baldi, Pierre et al. (2016). "Parameterized neural networks for high-energy physics". In: *The European Physical Journal C* 76.5, pp. 1–7.

Barlow, Roger (1990). "Extended maximum likelihood". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 297.3, pp. 496–506.

Bishop, Christopher M (2006). "Pattern recognition". In: *Machine learning* 128.9.

Bottou, Léon (2010). "Large-scale machine learning with stochastic gradient descent". In: *Proceedings of COMPSTAT'2010*. Springer, pp. 177–186.

Branco, Paula, Luís Torgo, and Rita P Ribeiro (2016). "A survey of predictive modeling on imbalanced domains". In: *ACM Computing Surveys (CSUR)* 49.2, pp. 1–50.

Bruce, Peter, Andrew Bruce, and Peter Gedeck (2020). *Practical statistics for data scientists: 50+ essential concepts using R and Python*. O'Reilly Media.

Carleo, Giuseppe et al. (2019). "Machine learning and the physical sciences". In: *Reviews of Modern Physics* 91.4, p. 045002.

Cerri, Olmo et al. (2019). "Variational autoencoders for new physics mining at the large hadron collider". In: *Journal of High Energy Physics* 2019.5, pp. 1–29.

Chakravarti, Purvasha et al. (2021). *Model-Independent Detection of New Physics Signals Using Interpretable Semi-Supervised Classifier Tests*. arXiv: 2102.07679 [stat.AP].

Chandola, Varun, Arindam Banerjee, and Vipin Kumar (2009). "Anomaly Detection: A Survey". In: *ACM Comput. Surv.* 41.3. ISSN: 0360-0300. DOI: 10.1145/1541880.1541882. URL: https://doi.org/10.1145/1541880.1541882.

Chatrchyan, Serguei et al. (2012). "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC". In: *Physics Letters B* 716.1, pp. 30–61.

Choma, Nicholas et al. (2018). *Graph Neural Networks for IceCube Signal Classification*. arXiv: 1809.06166 [cs.LG].

Collins, Jack, Kiel Howe, and Benjamin Nachman (2018). "Anomaly detection for resonant new physics with machine learning". In: *Physical review letters* 121.24, p. 241803.

Cowan, Glen et al. (2011). "Asymptotic formulae for likelihood-based tests of new physics". In: *The European Physical Journal C* 71.2, pp. 1–19.

Cranmer, Kyle, Juan Pavez, and Gilles Louppe (2015). "Approximating likelihood ratios with calibrated discriminative classifiers". In: *arXiv preprint arXiv:1506.02169*.

Cybenko, George (1989). "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4, pp. 303–314.

D'Agnolo, Raffaele Tito and Andrea Wulzer (2019). "Learning new physics from a machine". In: *Physical Review D* 99.1, p. 015014.

D'Agnolo, Raffaele Tito et al. (2021). "Learning multivariate new physics". In: *The European Physical Journal C* 81.1, pp. 1–21.

d'Agnolo, Raffaele Tito et al. (2021). "Learning New Physics from an Imperfect Machine". In: *arXiv preprint arXiv:2111.13633*.

De Simone, Andrea and Thomas Jacques (2019). "Guiding new physics searches with unsupervised learning". In: *The European Physical Journal C* 79.4, pp. 1–15.

Farina, Marco, Yuichiro Nakai, and David Shih (2020). "Searching for new physics with deep autoencoders". In: *Phys. Rev. D* 101 (7), p. 075021. DOI: 10.1103/PhysRevD.101.075021. URL: https://link.aps.org/doi/10.1103/PhysRevD.101.075021.

Friedman, Jerome, Trevor Hastie, Robert Tibshirani, et al. (2001). *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York.

Gligorov, V V and M Williams (2013). "Efficient, reliable and fast high-level triggering using a bonsai boosted decision tree". In: *Journal of Instrumentation* 8.02, P02013–P02013. ISSN: 1748-0221. DOI: 10.1088/1748-0221/8/02/p02013. URL: http://dx.doi.org/10.1088/1748-0221/8/02/P02013.

Golubovic, Dejan et al. (Apr. 2020). *CTD2020: 40 MHz Scouting with Deep Learning in CMS*. DOI: 10.5281/zenodo.4034400. URL: https://doi.org/10.5281/zenodo.4034400.

Hagiwara, Kaoru et al. (2002). "Review of particle physics: Particle data group". In: *Physical Review D* 66.1 I, pp. 100011–10001958.

Hart, Peter E, David G Stork, and Richard O Duda (2000). *Pattern classification*. Wiley Hoboken.

Jumper, John et al. (2021). "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873, pp. 583–589.

Kasieczka, Gregor, Benjamin Nachman, and David Shih (2020). *LHC Olympics 2020 Summary Talk*. URL: https://indico.cern.ch/event/809820/contributions/3708303/attachments/1971116/3347225/SummaryTalk.pdf.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2017). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Commun. ACM* 60.6, 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386. URL: https://doi.org/10.1145/3065386.

Lamb, Evelyn (2012). *5 sigma what's that?* URL: https://blogs.scientificamerican.com/observations/five-sigmawhats-that/.

Losapio, Gianvito (Sept. 2021). *Machine Learning for 40 MHZ Scouting at CMS*. DOI: 10.5281/zenodo.5536345. URL: https://doi.org/10.5281/zenodo.5536345.

Meanti, Giacomo et al. (2020). "Kernel methods through the roof: handling billions of points efficiently". In: *arXiv preprint arXiv:2006.10350*.

Micchelli, Charles A, Yuesheng Xu, and Haizhang Zhang (2006). "Universal Kernels." In: *Journal of Machine Learning Research* 7.12.

Mitchell, Tom (1997). *Machine learning*. McGraw hill New York.

Murphy, Kevin P (2012). *Machine learning: a probabilistic perspective*. MIT press.

Murphy, Kevin P. (2022). *Probabilistic Machine Learning: An introduction*. MIT Press. URL: probml.ai.

Neyman, J. and E. S. Pearson (1933). "On the Problem of the Most Efficient Tests of Statistical Hypotheses". In: *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 231, pp. 289–337. ISSN: 02643952. URL: http://www.jstor.org/stable/91247.

Paganini, Michela, Luke de Oliveira, and Benjamin Nachman (2018). "CaloGAN: Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks". In: *Physical Review D* 97.1. ISSN: 2470-0029. DOI: 10.1103/physrevd.97.014021. URL: http://dx.doi.org/10.1103/PhysRevD.97.014021.

Particle Data Group et al. (2020). "Review of particle physics". In: *Progress of Theoretical and Experimental Physics* 2020.8, p. 083C01.

Paskov, Hristo Spassimirov (2010). "A regularization framework for active learning from imbalanced data". PhD thesis. Massachusetts Institute of Technology.

Perivolaropoulos, Leandros and Foteini Skara (2021). "Challenges for ΛCDM: An update". In: *arXiv preprint arXiv:2105.05208*.

Petersohn, Christian (2010). *Temporal video segmentation*. Jörg Vogt Verlag.

Rodríguez, Andres C. et al. (2018). "Fast cosmic web simulations with generative adversarial networks". In: *Computational Astrophysics and Cosmology* 5.1. ISSN: 2197-7909. DOI: 10.1186/s40668-018-0026-4. URL: http://dx.doi.org/10.1186/s40668-018-0026-4.

Rosasco, Lorenzo (2017). *Introductory Machine Learning Notes*. URL: http://lcsl.mit.edu/courses/ml/1718/MLNotes.pdf.

Rudi, Alessandro, Luigi Carratino, and Lorenzo Rosasco (2017). "Falkon: An optimal large scale kernel method". In: *arXiv preprint arXiv:1705.10958*.

Rudi, Alessandro et al. (2018). "On Fast Leverage Score Sampling and Optimal Learning". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Montréal, Canada: Curran Associates Inc., 5677–5687.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors". In: *nature* 323.6088, pp. 533–536.

Silver, David et al. (2017). "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm". In: *CoRR* abs/1712.01815. arXiv: 1712.01815. URL: http://arxiv.org/abs/1712.01815.

Strubell, Emma, Ananya Ganesh, and Andrew McCallum (2019). "Energy and Policy Considerations for Deep Learning in NLP". In: *CoRR* abs/1906.02243. arXiv: 1906.02243. URL: http://arxiv.org/abs/1906.02243.

The ATLAS Collaboration and The CMS Collaboration (2011). *Procedure for the LHC Higgs boson search combination in Summer 2011*. Tech. rep. Geneva: CERN. URL: https://cds.cern.ch/record/1379837.

Valsecchi, Davide (2020). *Students seminars from the Prefit20 School*. URL: https://indico.cern.ch/event/912027/contributions/3835656/.