Alma Mater Studiorum · Università di Bologna

Scuola di Scienze
Dipartimento di Fisica e Astronomia "Augusto Righi"
Corso di Laurea Magistrale in Astrofisica e Cosmologia

A new emulator code for Cosmology: an application of machine learning algorithms to speed up cosmological analyses

Relatore:
Prof. Federico Marulli

Correlatrice:
Dott.ssa Sofia Contarini

Presentata da:
Pierpaolo Nicolosi

Anno Accademico 2021/2022

Abstract

In recent years, artificial intelligence has become a part of our daily life. Machine learning algorithms are now used in many sectors, from telephony to entertainment platforms.

In the same way, the field of Cosmology has benefited from the enormous advantages offered by this technology. More and more scientific researchers apply nowadays machine learning algorithms to process and analyze the huge amount of data provided by wide-field surveys and cosmological simulations. The number of publications related to the usage of these techniques in the astrophysical sector has indeed grown exponentially in recent decades, driven by the enormous successes that have been achieved.

Our Thesis work fits perfectly into this context. What we present is a machine learning-based code aimed at drastically speeding up cosmological analyses. It is in fact an emulator, i.e. an algorithm able to reproduce a given theoretical model in a very short time and with great accuracy.

We implemented the code into the public libraries `CosmoBolognaLib` (Marulli, Veropalumbo & Moresco, 2016), providing the users with a powerful tool to emulate the cosmological functions already present in these libraries. The main limitation of these functions is their run time in the context of Bayesian analyses. Indeed, having to calculate these models millions of times, this type of statistical analysis becomes extremely slow.

Exploiting the machine learning algorithms provided by the numerical library `CosmoPower` (Spurio Mancini et al., 2022), we built a neural network aimed at imitating the output of the theoretical model of the two-point correlation function, a type of statistic widely used in Cosmology. As a first implementation example, we focused on emulating the model by varying four cosmological parameters: the total matter density parameter, Ω_{m}, the energy density parameter, Ω_{de}, the amplitude of the primordial power

spectrum, A_s and the redshift, z. The training process of the network was carried out using the large computational resources provided by the Department of Physics and Astronomy of the University of Bologna. We then validated the emulator by comparing its output to that of the original function, verifying its accuracy for the full range of input parameter combinations.

Finally, we applied our code to the analysis of cosmological simulations. In particular, we measured the two-point correlation function of dark matter particles at three different redshifts of the DUSTGRAIN-*pathfinder* (Giocoli, Baldi & Moscardini, 2018; Hagstotz et al., 2019). We performed a Bayesian analysis to derive the posterior probability distribution of the parameters Ω_m, Ω_{de} and A_s, using both the original function of the `CosmoBolognaLib` and its emulated version. The results obtained with our emulator show almost perfect correspondence with that of the original model, with small differences that completely fall within the statistical fluctuations of the method. The really innovative part, however, lies in the timing of the analysis. With our emulator, the code becomes thousands of times faster, bringing the total execution time from several tens of hours to a few seconds.

Our code will also be improved by extending the emulation to other cosmological functions and expanding the emulator's range of validity to cover a wider parameter space. The potential applications of this methodology in the future are numerous, and its use will soon become a key element for future cosmological analyses conducted within the `CosmoBolognaLib`.

Sommario

In questi ultimi anni, l'intelligenza artificiale è diventata parte della nostra vita quotidiana. Gli algoritmi di *machine learning* vengono oggi impiegati in moltissimi settori, da quello della telefonia alle piattaforme di intrattenimento.

Anche la Cosmologia ha usufruito degli enormi vantaggi che offre questa tecnologia. Sempre più ricerche infatti utilizzano oggi gli algoritmi di *machine learning* per processare e analizzare l'enorme quantità di dati che ci viene fornita dalle survey a grande campo e dalle simulazioni cosmologiche. Il numero di pubblicazioni riguardanti l'applicazione di queste tecniche nel settore astrofisico e cosmologico è infatti cresciuto ad un ritmo esponenziale negli ultimi decenni, guidato dagli enormi successi che sono stati ottenuti.

Il nostro lavoro di Tesi si colloca perfettamente in questo contesto. Quello che presentiamo è un codice basato sul *machine learning*, volto a velocizzare drasticamente le analisi cosmologiche. Si tratta infatti di un emulatore, ovvero un algoritmo in grado di riprodurre un dato modello teorico in tempi brevissimi e con una grande accuratezza.

Il codice che abbiamo implementato è inserito nelle librerie *free software* `CosmoBolognaLib` (Marulli, Veropalumbo & Moresco, 2016) e offre un potente strumento per emulare le funzioni cosmologiche già presenti in queste librerie. Il limite principale di queste funzioni è dato dal loro tempo di esecuzione nel contesto delle analisi Bayesiane. Infatti, dovendo calcolare questi modelli milioni di volte, questo tipo di analisi statistica risulta estremamente lento.

Sfruttando gli algoritmi di *machine learning* forniti dalla libreria numerica `CosmoPower` (Spurio Mancini et al., 2022), abbiamo costruito una rete neurale volta ad imitare l'output del modello teorico della funzione di correlazione a due punti, un tipo di statistica enormemente utilizzato in Cosmologia. Come primo esempio di implementazione, ci siamo concen-

trati sull'emulazione del modello al variare di quattro parametri cosmologici: il parametro di densità totale della materia, $\Omega_{\rm m}$, il parametro di densità di energia, $\Omega_{\rm de}$, l'ampiezza dello spettro di potenza primordiale, $A_{\rm s}$ e il redshift, z. Il processo di allenamento della rete è stato svolto sfruttando le grandi risorse computazionali fornite dal Dipartimento di Fisica e Astronomia dell'Università di Bologna. Abbiamo poi validato l'emulatore comparandone l'output con quello della funzione originale, e verificandone l'accuratezza per tutte le combinazioni di parametri in input.

Abbiamo infine applicato il nostro codice all'analisi di simulazioni cosmologiche. In particolare, abbiamo misurato la funzione di correlazione a due punti delle particelle di materia oscura a tre diversi redshift delle DUSTGRAIN-*pathfinder* (Giocoli, Baldi & Moscardini, 2018; Hagstotz et al., 2019). Abbiamo poi condotto un'analisi Bayesiana per derivare la distribuzione della probabilità a posteriori dei parametri $\Omega_{\rm m}$, $\Omega_{\rm de}$ e $A_{\rm s}$, utilizzando come modello sia la funzione originale delle `CosmoBolognaLib`, che la sua versione emulata. I risultati ottenuti con il nostro emulatore mostrano una corrispondenza quasi perfetta con quella del modello originale, con piccole differenze che rientrano nelle fluttuazioni statistiche del metodo.

La parte veramente innovativa, però, sta nelle tempistiche dell'analisi. Con il nostro emulatore il codice diventa migliaia di volte più veloce, portando il tempo complessivo di esecuzione da diverse decine di ore a qualche secondo.

Il nostro codice verrà inoltre migliorato estendendo l'emulazione ad altre funzioni cosmologiche e ampliando il range di validità dell'emulatore per coprire un più ampio spazio dei parametri. Le applicazioni future di questa metodologia sono innumerevoli e il suo utilizzo costituirà presto un elemento chiave per le future analisi cosmologiche condotte nell'ambito delle `CosmoBolognaLib`.

Contents

Introduction

Massive breakthroughs have involved the field of Cosmology in the last decades. Deep and wide cosmological surveys have precisely mapped the distribution of visible matter in the sky, enhancing our knowledge of the Universe properties and their evolution over time.

The cosmological model commonly assumed to describe these large-scale observations is the Λ-cold dark matter (ΛCDM). The ΛCDM model is based on the validity of the cosmological principle, i.e. the hypothesis of homogeneity and isotropy at the Universe large scales, as well as on the validity of the Einstein's General Relativity. According to this framework, the Universe is about 13.8 billion years old and is constituted by $\sim 5\%$ of baryons (i.e. the visible matter made of protons, neutrons and electrons) and by $\sim 25\%$ of a mysterious component named dark matter. This model assumes also the presence of the cosmological constant, Λ, an unknown component representing the $\sim 70\%$ of the total energy density of the today Universe. The cosmological constant can be also interpreted as the so-called dark energy and is assumed to be responsible for the current accelerated expansion of the Universe.

However, the ΛCDM model has often been questioned, mainly because of the lack of a physical explanation about the nature of its components, as well as about the exact mechanisms regulating the Universe accelerated expansion. In addition, in recent years, statistical tensions have emerged on the estimate of a few cosmological parameters from different probes (see Abdalla et al., 2022, for a review), puzzling the scientific community.

For these reasons, many missions aimed at investigating these problems have already been launched and are planned for the coming years. Cosmological simulations, in the same way, have become larger and more precise, reproducing a huge number of different realizations of the universe and covering a variety of cosmological scenarios. The amount of data at our disposal has therefore become increasingly larger, and thus even more difficult to manage and analyze.

In this context, machine-learning algorithms have taken on a dominant role, allowing us to push the limits of cosmological investigations even further. They enable us to quickly process catalogues of cosmic objects and carry on precise cosmological analyses in very short times, as well as to derive cosmological constraints directly from the comparison between simulations and real data.

Drawing on the numerous studies published in recent years on this topic, and motivated by the great success of the application of neural networks in Cosmology, we present in this Thesis work a new machine-learning based code aimed at improving the efficiency of cosmological analyses. The code we developed is an emulator, i.e. an algorithm able to reproduce with great accuracy a given model, for wide range of input parameters. It is integrated in the framework of the *free software* numerical libraries `CosmoBolognaLib` (Marulli, Veropalumbo & Moresco, 2016) and allows us to emulate any function implemented in these libraries.

We are particularly interested in the cosmological models available in the `CosmoBolognaLib`, since they are employed for extensive statistical analyses. As an illustrative example, we prepared our emulator to reproduce the basic theoretical model for the two-point correlation function, a statistic widely used in Cosmology. We employed machine-learning algorithms provided by the library `CosmoPower` (Spurio Mancini et al., 2022) to train the neural network, and subsequently we validated the accuracy of the emulated model.

Then, we tested the efficiency of our emulator in the context of a Bayesian analysis. We modelled the two-point correlation function measured in cosmological simulations, sampling the posterior distribution of the model cosmological parameters by using the original `CosmoBolognaLib` function and the emulated version we implemented. Besides confirming the high accuracy of our emulator, this test revealed a speed-up factor between $\mathcal{O}(10^3)$ and $\mathcal{O}(10^4)$. In practise, our emulator reduced the computational time required for the analysis from tens of hours to a few seconds. This result proves the high potentiality of the method, which will be extended in the near future to a multiplicity of cosmological applications and will dramatically improve the efficiency of the statistical analyses performed with the `CosmoBolognaLib`.

We briefly summarize the structure of this Thesis work in the following:

- in Chapter 1, we provide a brief overview of the theoretical background of modern Cosmology, starting with an exploration of the essential components of General Relativity and concluding with an examination of the standard cosmological model;

- in Chapter 2, we delve into the structure formation and clustering of the Universe at large scales. We outline the linear perturbation theory, which is essential to the introduction of the two-point correlation function, i.e. the statistic that we aim at emulating in this work. Then we briefly present the nonlinear perturbation theory, which is essential to introduce the N-body simulations;

- in Chapter 3, we discuss the usage of machine learning and Bayesian inference in the field of Cosmology. We focus on supervised learning methods and the concept of deep neural network models, and how they can be implemented through the `CosmoPower` libraries to compute the desired emulated models. We finally provide an introduction to the basics of Bayesian statistics, as well as to the Markov Chain Monte Carlo technique that we need to derive the test cosmological constraints;

- in Chapter 4, we present the C++/Python set of libraries `CosmoBolognaLib`, which represent the numerical environment in which our code is implemented and can be exploited for cosmological analyses. Then, we introduce our code by presenting the new class we implemented. We also describe the training and the validation procedure we performed to prepare our emulator. In the final part of the chapter, we provide two examples of the usage of the new class for emulators, one to appreciate the accuracy of the model and the other to appreciate its computational speed.;

- in Chapter 5, we present an application of our emulator to a Bayesian analysis. We measure the two-point correlation function in the DUSTGRAIN-*pathfinder* simulations (Giocoli, Baldi & Moscardini, 2018; Hagstotz et al., 2019) and we extract test constraints on the cosmological parameters of our interest. Finally, we compare results obtained with the original `CosmoBolognaLib` model function and those derived with its emulated version, both in terms of accuracy and efficiency;

- in Chapter 6 we sum up the achieved results, providing also an insight into the future developments of our code.

Chapter 1

Cosmological framework

In this chapter we briefly introduce the fundamental theoretical topics that underline the scientific analyses presented in this Thesis. This is done with no aim of completeness, but only with a clarity purpose. For a depth dissertation of the these concepts, we invite the reader to rely on the two fundamental textbooks, Ray & Vickers (2022) and Coles & Lucchin (2002), which were used as reference for writing the following sections.

1.1 Elements of General Relativity

Our starting point for the building of a general theory for cosmology is the postulation that gravitational forces are the primary interactions in our Universe on the largest scales. We rely in particular on the widely accepted Einstein's theory of General Relativity (GR), which provides an accurate description of local gravitational interactions, linking the geometrical features of the space-time with the the presence of mass and energy within it.

We assume that the whole set of all events, seen by an observer, is a **differentiable manifold**. In the GR framework an event is described by a **four-vector** $x^\mu = (x^0, x^1, x^2, x^3)$, where the first element represents the time coordinate, while the other ones the space coordinates. Let us consider two observers, each with their own reference frame, observing a pair of events. A distance operator between the events can be defined over a differentiable manifold by the **metric tensor**:

$$g_{\mu\nu} = \eta_{\alpha\beta} \frac{d\xi^\alpha}{dx^\mu} \frac{d\xi^\beta}{dx^\nu} \ , \tag{1.1}$$

where $\eta_{\alpha\beta}$ is the metric tensor used by a second observer (historically an inertial one) and the two events are characterized by the two four-vectors, ξ^α and ξ^β, respectively.

For our purposes, it is necessary to introduce the **Rienmann tensor**, which can be written as follows:

$$R^\mu_{\alpha\beta\gamma} = \frac{d\Gamma^\mu_{\alpha\gamma}}{dx^\beta} - \frac{d\Gamma^\mu_{\alpha\beta}}{dx^\gamma} + \Gamma^\mu_{\sigma\beta}\Gamma^\sigma_{\gamma\alpha} - \Gamma^\mu_{\sigma\gamma}\Gamma^\sigma_{\beta\alpha} \ , \tag{1.2}$$

where Γ is the **affine connection**, that is a geometric object used to describe the relationship between two points on a curved surface. We also introduce the **Ricci tensor** and the **Ricci scalar**, by the contraction, respectively, of the Riemann tensor with the Kronecker tensor, ϵ^γ_μ, and than with $g^{\mu\nu}$:

$$R_{\alpha\beta} \equiv R^\mu_{\alpha\beta\gamma}\epsilon^\gamma_\mu = R^\mu_{\alpha\beta\mu} \qquad R \equiv g^{\mu\nu}R_{\mu\nu} \ . \tag{1.3}$$

After the definition of all the terms we need to describe the geometrical proprieties of the space-time, we have to introduce the term that is responsible for such features in the GR framework: the **energy-momentum** tensor, $T_{\mu\nu}$, which represents the content of energy and matter in the Universe. Now, assuming the **minimal gravitational coupling** and the **correspondence principles**[1], it is possible to link the energy-momentum to the metric tensor, the Ricci tensor and the Ricci scalar, defining the so-called **Einstein field equation** (EFE):

$$R_{\mu\nu} - \frac{1}{2}g_{\mu\nu}R = \frac{8\pi G}{c^4}T_{\mu\nu} \ , \tag{1.4}$$

where G is the gravitational constant, while c is the speed of light. According to the EFE, the energy-momentum tensor is the source of the space-time curvature, which then determines the motion of physical objects. We will see in Sect. 1.4 that only non-static solution satisfy the EFE.

1.2 The Friedmann-Lemâitre-Robertson-Walker metric

To characterize the geometry of the space-time, we have first to figure out the terms of the metric tensor. For this purpose, we start from the definition of the distance between two

[1]The first states that the amount of terms that describe a gravitational interaction between two or more objects must be minimized, while the second that the GR and Newton's theory of gravity must be consistent.

events, i.e. $ds^2 = g_{\mu\nu}dx^\mu dx^\nu$, which can be expressed as:

$$ds^2 = g_{00}dt^2 + 2g_{0i}dtdx^i + g_{ij}dx^i dx^j \ , \tag{1.5}$$

with $i, j = 1, 2, 3$. Here x^i are the three spatial coordinates and t is the coordinate time. The **Cosmological Principle** (CP) asserts that on scales of hundreds of megaparsecs or larger, the Universe can be considered to be homogeneous and isotropic, meaning that no direction or event is favored over others. As a result, the mixed term in Eq. (1.5) can be disregarded.

In order to get the value of g_{00}, we can consider the motion of a photon, for which $ds^2 = 0$ by definition. Denoting with dl^2 the spatial term, Eq. (1.5) yields:

$$\begin{aligned} 0 &= g_{00}dt^2 - dl^2 \\ g_{00} &= \frac{dl^2}{dt^2} \\ g_{00} &= c^2 \ , \end{aligned}$$

where the negative sign is due to the metric signature assumed, i.e. (1,3)[2].

We can express analytically the spatial distance dl^2 using polar coordinates (r, θ, ϕ):

$$dl^2 = a^2(t)\left[\frac{dr^2}{1-kr^2} + r^2d\theta^2 + \sin\theta^2 d\phi^2\right] \ , \tag{1.6}$$

where (r, θ, ϕ) are the **comoving polar coordinates** and t is the **proper time** (also called as **cosmic time**), both defined in a reference system at rest with the Universe expansion, $a(t)$ has the dimensions of a length and it is the so-called **cosmic scale factor**, while k is the **curvature parameter**. The parameter k can have three values only:

- -1, if the geometry of the universe is hyperbolic, i.e. a **open universe**;
- 0, if it is Euclidean, i.e. a **flat universe**;
- 1, if it is hyperspherical, i.e. a **closed universe**.

[2]The metric signature is a pair of numbers associated with a metric tensor, indicating the number of positive and negative eigenvalues of the tensor.

Taking into account all these assumptions, the metric in Eq. (1.5) can be written as follows:

$$ds^2 = c^2dt^2 - a^2(t)\left[\frac{dr^2}{1-kr^2} + r^2d\theta^2 + \sin\theta^2 d\phi^2\right] , \tag{1.7}$$

This is well-known **Friedmann-Lemâitre-Robertson-Walker** (FRLW) metric, which is the most general one - up to coordinate transformations - fulfilling the CP.

1.3 The Hubble's Law and the redshift

Let us consider two points, P_0 and P not causally connected, such that $dt = 0$ and also $d\theta = d\phi = 0$ (for the sake of simplicity, but without lost generality thanks to the CP). If we move P is at a distance r from P_0, we can define the **proper distance** as:

$$D_{\rm pr} = a(t)\int_0^r \frac{dr'}{\sqrt{1-kr'^2}} = a(t)F(r,k) , \tag{1.8}$$

where $F(r,k)$ depends only on r and k.

To evaluate the previous equation at the present time, $t = t_0$, we define the **comoving distance** as:

$$D_{\rm C} \equiv a(t_0)F(r,k) = D_{\rm pr}(t_0) = \frac{a(t_0)}{a(t)}D_{\rm pr}(t) , \tag{1.9}$$

so that we obtain:

$$D_{\rm pr}(t) = \frac{a(t)}{a_0}D_{\rm C} , \tag{1.10}$$

where the subscript 0 denotes that the corresponding quantity is computed at the present time. Equation (1.9) tells us how the proper distances are related to the comoving ones.

Considering the time derivative of the proper distance, we can compute the radial velocity of any two points in the universe:

$$\frac{d}{dt}D_{\rm pr} = \frac{\dot{a}(t)}{a_0}D_C = \frac{\dot{a}(t)}{a(t)}D_{\rm pr} \equiv H(t)D_{\rm pr} \tag{1.11}$$

This equation is the so-called **Hubble's law** and $H(t)$ is called the **Hubble's parameter**. The latter describes the isotropic expansion rate of the Universe and at any given cosmic time t, and it has the same value across all Universe. Its value at present day is denoted as H_0 and is called the **Hubble constant**. The global motion of cosmological objects in the Universe with respect to each other is known as **Hubble flow**.

Let us now consider two photons, one emitted at time $t_{\rm e}$ and the other at time $t_{\rm e}+\delta t_{\rm e}$. The two signals will arrive to an observer at time $t_{\rm o}$ and $t_{\rm o}+\delta t_{\rm o}$, respectively. Since $ds^2 = 0$ for any light path, from Eq. (1.7) (assuming again $d\theta = d\phi = 0$) we have:

$$\int_{t_e}^{t_o} \frac{cdt}{a(t)} = F(r,k) = \int_{t_e+\delta t_e}^{t_o+\delta t_o} \frac{cdt}{a(t)} \ . \tag{1.12}$$

If $\delta t_{\rm e}$ and $\delta t_{\rm o}$ are small, solving the two integrals yields to:

$$\frac{\delta t_e}{a(t_e)} = \frac{\delta t_o}{a(t_o)} \ . \tag{1.13}$$

Taking into account that $\delta t = 1/\nu$, $\lambda = c/\nu$ and subtracting -1 at both side of the equation, it turns out:

$$z \equiv \frac{\lambda_o - \lambda_e}{\lambda_e} = \frac{a(t_o)}{a(t_e)} - 1 \ , \tag{1.14}$$

where z is the shifting of the electromagnetic radiation, which is called **redshift**. If we now consider a photon emitted at time t and observed at the present-time, we can re-write Eq. (1.14) as:

$$1 + z = \frac{a_0}{a(t)} \ . \tag{1.15}$$

Equation (1.15) tells us that the redshift of a photon beam is inherently linked to the cosmic scale factor and therefore to the distance between two objects in the Universe. Due to the development of spectroscopy and photometry techniques, the redshift measure can be applied to infer the distance of extragalactic sources. Moreover, once the distance of an extragalactic object is established, it is possible to estimate the amount of time since the emission of the light signal.

1.4 Cosmological constant and Friedmann models

In order to solve the EFE as stated in Eq. (1.4), an energy-momentum tensor must be specified. The simplest assumption is to considering the Universe like as a perfect fluid so that the energy-momentum tensor becomes:

$$T_{\mu\nu} = -pg_{\mu\nu} + (p + \rho c^2)u_\mu u_\nu \ , \tag{1.16}$$

where p and ρ are the pressure and the density of all components of the perfect fluid, respectively, with an implicit time dependency, while u_μ is the four-velocity of the fluid particle.

Solving the EFE with respect to a, we obtain two fundamental equations, the so-called **First** and **Second Friedmann Equations**:

$$\ddot{a} = -\frac{4}{3}\pi G\Big(\rho + \frac{3p}{c^2}\Big)a \ , \tag{1.17}$$

$$\dot{a}^2 + kc^2 = \frac{8}{3}\pi G\rho a^2 \ , \tag{1.18}$$

which describe the dynamic evolution of the Universe, since solving this two equations one can obtain the time evolution of the scale factor.

At the beginning of the 20th century, around the time the Friedmann equations were developed, it was widely believed among the scientific community that the universe was static. This requirement translates into imposing $\ddot{a} = \dot{a} = 0$ in Eq. (1.17) and Eq. (1.18), in order to obtain a static solution. From Eq. (1.17) it turns out:

$$\rho = -\frac{3p}{c^2} \ . \tag{1.19}$$

However this solution does not follow the correspondence principle, one of the GR principles (see Sect. 1.1), because in Newtonian physics both the density and the pressure must be positive quantities. To solve this problem while preserving the principle of minimal gravitational coupling, so minimizing the number of corrective terms, Einstein introduced an additional term such that EFE became:

$$R_{\mu\nu} - \frac{1}{2}g_{\mu\nu}R - \Lambda g_{\mu\nu} = \frac{8\pi G}{c^4}T_{\mu\nu} \ , \tag{1.20}$$

where the constant Λ take the name of **cosmological constant** and in modern cosmology it is interpreted as an unknown Universe component generally denoted as **dark energy**

(DE).

Let us now put $-\Lambda g_{\mu\nu}$ into the right hand side of Eq. (1.20) and rewrite the energy momentum tensor as $\tilde{T}_{\mu\nu} = T_{\mu\nu} + \frac{\Lambda c^4}{8\pi G} g_{\mu\nu}$. Equations (1.17) and (1.18) become:

$$\ddot{a} = -\frac{4}{3}\pi G\Big(\tilde{\rho} + \frac{3\tilde{p}}{c^2}\Big)a \ , \tag{1.21}$$

$$\dot{a}^2 + kc^2 = \frac{8}{3}\pi G\tilde{\rho}a^2 \ , \tag{1.22}$$

where $\tilde{\rho} = \rho + \frac{\Lambda c^2}{8\pi G}$ and $\tilde{p} = p - \frac{\Lambda c^4}{8\pi G}$ are the new values for the density and the pressure, respectively. Substituting $\tilde{\rho}$ and $\tilde{p}$ into Eq. (1.19) yields:

$$\rho = \frac{\Lambda c^2}{4\pi G} - \frac{3p}{c^2} \ , \tag{1.23}$$

satisfying the correspondence principle. In modern cosmology, the cosmological constant is no longer utilized to derive stationary solutions; instead, it is related to DE, which is thought to be the element of the Universe responsible for its accelerated expansion.

1.4.1 Curvature density parameter

Let us now focus on the second Friedmann equation to link the density ρ with the curvature parameter k. Dividing each term of Eq. (1.18) by a^2, we obtain:

$$H^2\Big(1 - \frac{\rho}{\rho_{\text{crit}}}\Big) = -\frac{kc^2}{a^2} \ , \tag{1.24}$$

where ρ_{crit}, the **critical density**, is equal to:

$$\rho_{\text{crit}} \equiv \frac{3H^2}{8\pi G} \ , \tag{1.25}$$

therefore for $\rho = \rho_{\text{crit}}$ we have $k = 0$ (flat universe), while for $\rho > \rho_{\text{crit}}$ we have $k < 0$ (closed universe) and for $\rho < \rho_{\text{crit}}$ we have $k > 0$ (open universe). We will analyze in more details these three cases in Sect. 1.4.2.

Until now, we have implicitly denoted with ρ the total density ρ_{tot} of the perfect fluid. However ρ can be re-written as the sum of the different components, i, constituting the

Universe:

$$\rho_{\rm tot} \equiv \sum_i \rho_{w_i} \ . \tag{1.26}$$

Here w_i are constants arising from the **equation of state** (EoS):

$$p = w\rho c^2 \ , \tag{1.27}$$

where we have $w = 0$ for the non-relativistic elements of the fluid (i.e. the matter component), $w = 1/3$ for the relativistic ones (i.e. the radiation component) and $w = -1$ for the DE.

By imposing the adiabatic condition $\mathrm{d}\mathcal{U} = -p\mathrm{d}V$, which in our case can be written as $\mathrm{d}\big(\rho c^2 a^3\big) = -p\mathrm{d}a^3$, and by taking into account the EoS, we can derive at the following expression:

$$\rho_w = \rho_{w,0}\left(\frac{a}{a_0}\right)^{-3(1+w)} \propto (1+z)^{3(1+w)} \ . \tag{1.28}$$

This result demonstrates that each component of the universe evolves in density at a different rate over time. Furthermore, the equation indicates that these components dominate over each other at different times in the history of the universe, the so-called **cosmic epochs**, with one prevailing over the others, as shown in Fig. 1.1.

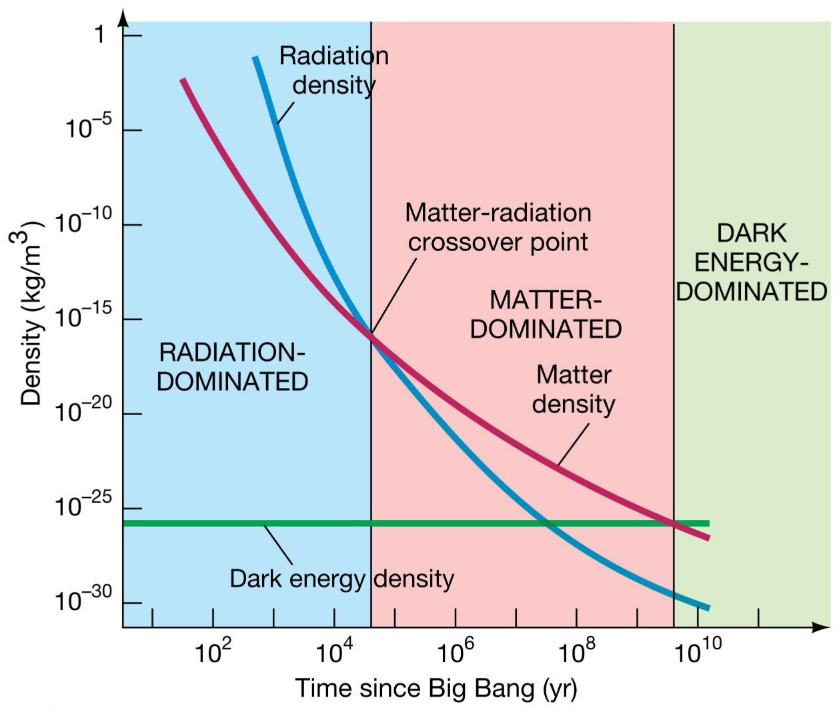

Figure 1.1: Density evolution of the cosmic components: radiation, matter, DE. Through the cosmic time, different components start to predominate over the others. Credits to: https://pages.uoregon.edu/jimbrau/astr123/Notes/Chapter27.html.

Let us define the **density parameter** as the ratio of the density ρ of a generic component and the critical density: $\Omega \equiv \rho/\rho_{\rm crit}$, where we kept implicit the time dependency of the densities. With this definition we can now express the Eq. (1.18) as:

$$1 - \Omega = -\frac{kc^2}{a^2 H^2} \ . \tag{1.29}$$

We notice that since the right hand side of the equation cannot change its sign during the expansion, so neither can the left hand side. This mean that a universe ruled by Friedmann equations cannot change its geometry over time.

By combining Eqs. (1.11), (1.15), (1.27) and (1.29) we obtain:

$$H^2(z) = {H_0}^2(1+z)^2\left[\Omega_{0,k} + \sum_i \Omega_{0,w_i}(1+z)^{1+3w_i}\right] \ , \tag{1.30}$$

where $\Omega_{0,k} \equiv 1 - \sum_i \Omega_{0,w_i}$ is the so-called **curvature density parameter**. At very high redshifts, the curvature density parameter becomes virtually null, allowing us to consider the Friedmann universes approximately flat in the initial epochs.

1.4.2 Flat, open and closed universe

Let us now consider a universe with a single component. By inserting Eq. (1.27) into Eq. (1.17) we obtain:

$$\ddot{a} = -\frac{4}{3}\pi G\rho(1+3w)a \ , \tag{1.31}$$

which is a good approximation of the first Friedmann equation at the early universe. As shown in Fig. 1.1, at each cosmic epoch a single universe component results dominant. In particular, the DE component, characterized by $w < -1/3$, becomes relevant near the present time. This leads to $\ddot{a} < 0$, meaning that the scalar factor grows monotonically. This implies that, at some point in the past, a was equal to 0: this event is known as **Big Bang** and represents the singularity of Friedmann models.

Having stated that a Friedmann universe cannot change its geometry over time, we can conclude that if we consider a universe made up of a single component, then there are only three possible scenarios:

- $\Omega_0 < 1$ open universes, the scale factor a grows indefinitely with time;
- $\Omega_0 = 1$ flat universes, the scale factor a grows indefinitely with time but with a slower rate, tending to $\dot{a} = 0$ at infinity;
- $\Omega_0 > 1$ closed universes, the scale factor a grows up to a maximum, then it decreases until it becomes null again.

These scenarios are represented schematically in Fig. 1.2, which shows the three temporal trends of the scale factor.

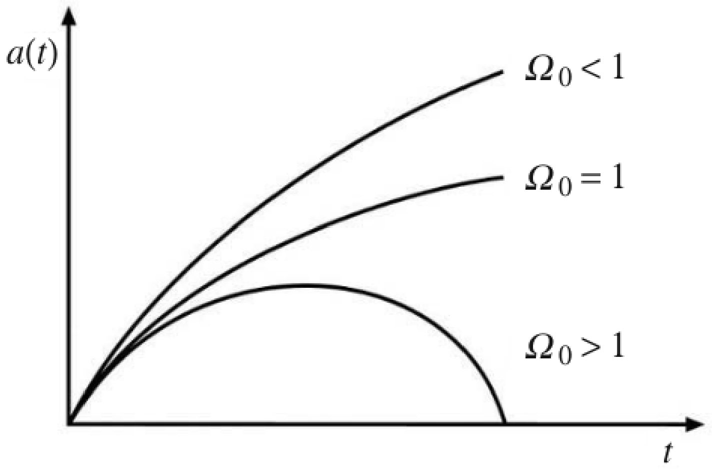

Figure 1.2: The scale factor trend in the three different scenarios: open ($\Omega_0 < 1$), flat ($\Omega_0 = 1$) and closed ($\Omega_0 > 1$) universes (Coles & Lucchin, 2002).

1.5 The standard cosmological scenario

Since the start of the 21st century, the most widely accepted model of the Universe has been the ΛCDM model. This model is backed up by a number of observational evidences and gives us the basis to understand the structure formation. It describes a nearly flat Universe, which is consistent with by the CP and whose development is directed by the Friedmann equations, and thus by GR (see Sect. 1.2).

According to this cosmological scenario, the Universe composition is dominated by a form of DE, resulting from the cosmological constant Λ (introduced in Eq. 1.20) and secondarily by a matter component called **cold dark matter** (CDM). Here the term "dark" refers to a non-baryonic form of matter which does not emit light. The dark matter (DM) is in fact supposed to interact through the gravitational force only. Despite many researches have been carried out to investigate the nature of this mysterious component, a direct detection of DM particles has not been achieved up to date. Moreover, the DM is assumed to be "cold", i.e. non-relativistic at the time of its decoupling.

The ΛCDM model explains that our Universe has experienced a thermal history and its evolution is strongly correlated to its temperature. The present temperature of the electromagnetic radiation which pervades the Universe is 2.7255 ± 0.0006 K (Fixsen et al., 1996; Planck Collaboration et al., 2020). This radiation is called **cosmic microwave background** (CMB) and was produced during the period of the recombination, i.e. the moment in which electrons and protons first bonded to form hydrogen atoms, and the Universe became electrically neutral ($z \simeq 1100$). After this, the Universe was dominated by matter and gravitationally bound objects began to form and this led to the emergence of the large-scale structures we see today.

The ΛCDM model describes a present-day universe composed by $\sim 70\%$ of DE, linked to the cosmological constant, by $\sim 25\%$ of a cold non-ordinary matter component, by $\sim 5\%$ of ordinary observable matter (i.e. baryons) and by a negligible fraction of radiation ($\sim 0.001\%$). These energy densities are consistent with the flatness condition $\Omega_{0,\mathrm{tot}} \sim 1$.

In Sect. 1.4.2, we saw that a universe not dominated by the DE component, which is what we expect in a early epoch, has a decelerated expansion. Having now established that the Universe is undergoing an accelerated expansion, with $\ddot{a} > 0$, we must include a flex in the scale factor function. It is easy to demonstrate that, given the current values of the densities, this inversion in the expansion rate occurs at $z_\mathrm{f} \sim 0.7$. Moreover, mathematical derivations can be used to find the moment of the matter-cosmological constant equivalence, i.e. $\Omega_\mathrm{m}(z_{\mathrm{eq},\Lambda}) = \Omega_\Lambda(z_{\mathrm{eq},\Lambda})$. This takes place at $z_{\mathrm{eq},\Lambda} \sim 0.33$, which is very close to the present time. This implies that DE contribution became significant in recent times.

In more detail, a full characterisation of the ΛCDM scenario requires the definition of six essential parameters:

- Ω_m: total matter density parameter,
- Ω_b: baryonic matter density parameter,
- H_0: Hubble constant,
- A_s: primordial power spectrum amplitude,
- n_s: spectral index of the primordial power spectrum,
- τ : reionisation optical depth,

where Ω_m and Ω_b are usually expressed with their present-day values, respectively. The strongest constraints on this set of parameters arise from the analysis of the CMB power spectrum combined with lensing measurements. The values for these fundamental parameters, as reported in Planck Collaboration et al. (2020) are: $\Omega_{\mathrm{m}}h^2 = 0.143 \pm 0.001$, $\Omega_{\mathrm{b}}h^2 = 0.0224 \pm 0.0001$, $\ln(10^{10}A_{\mathrm{s}}) = 3.04 \pm 0.01$, $H_0 = 67.4 \pm 0.5$ km s^{-1} Mpc^{-1}, $n_{\mathrm{s}} = 0.965 \pm 0.004$, and $\tau = 0.054 \pm 0.007$.

Employing Eq. (1.21), it is straightforward to demonstrate that a multi-component universe containing the cosmological constant and matter yields:

$$\ddot{a} = -aH^2(t)\frac{\Omega_{\mathrm{m}}}{2} + aH^2(t)\Omega_\Lambda \text{ , with } \Omega_\Lambda \equiv \frac{\Lambda c^2}{3H^2(t)} \ . \tag{1.32}$$

In order to achieve an accelerated expansion it must be $\Omega_\Lambda > \Omega_{\mathrm{m}}/2$, which is fulfilled by the present-day density values.

The ΛCDM model is currently the most widely accepted cosmological model due to the high accuracy of its predictions with the majority of present-day cosmological observations and its simplicity. However, there are still theoretical aspects of it that are not fully understood. For instance, we have yet to provide a physical description for the main matter component, the DM, and justify the presence of an ever more enigmatic component, i.e. the DE, which currently cannot be associated with any known form of energy.

Chapter 2

Structures formation and clustering

Structure formation and clustering refer to the processes by which the large-scale structure of the universe, including galaxies, clusters of galaxies, and larger structures, have formed and evolved over time. The study of structure formation involves modeling the evolution of the distribution of matter in the universe over time, using computer simulations and theoretical models. One of the key assumption in this area of research is the existence of DM, which is thought to make up the majority of the matter in the universe and provides the gravitational force that enables the formation of large structures. Clustering, on the other hand, refers to the tendency of galaxies and other structures to be grouped together forming the so-called **cosmic web**. Clustering is measured using statistical methods, such as correlation functions and power spectra, which quantify the degree to which the distribution of matter is more or less clustered with respect to a random distribution.

In this chapter, we provide a brief overview of the fundamental structure formation and clustering concepts.

2.1 Linear theory

The formulation of a comprehensive cosmological theory is essential to understand how galaxies, galaxy clusters, and other **cosmic structures** have formed and how quickly have evolved. Since the recombination ($z \simeq 1100$), thanks to gravity, the density fluctuations of baryonic matter have grown in amplitude, giving rise to the luminous structures we can observe in today's Universe.

At the time of recombination, the baryonic matter distribution was highly homogeneous. This is demonstrated by the fact that the amplitude of the perturbations in the density field must have been of the same order as the temperature fluctuations observed in the CMB:

$$\frac{\delta\rho}{\rho} \propto \frac{\delta T}{T} \sim 10^{-5} \ . \tag{2.1}$$

At the beginning of the 20th century, James Jeans, in an attempt to understand the formation of stars and planets, demonstrated the existence of a fundamental instability in evolving clouds of gas. This instability, now known as **Jeans instability** or gravitational instability, is related to the evolution of small fluctuations into collapsed structures. The theory that describes this phenomenon is called **Jeans Theory**.

2.1.1 Jeans theory

The Jeans theory describes how initial density fluctuations in the early Universe developed over time to produce the observed inhomogeneities at the present day. This model can be applied to non-relativistic matter, assuming small perturbations (i.e. $\delta \ll 1$), and describes the evolution of density perturbations on spatial scales not exceeding the **cosmological horizon**. The latter defines the volume of the Universe that is in causal contact with the observer and is expressed as:

$$R_{\mathrm{h}} \equiv a(t) \int_{t_{\mathrm{BB}}}^{t} \frac{c}{a(t')} dt' \ , \tag{2.2}$$

where $t_{\mathrm{BB}} = 0$ indicates the time singularity of all Friedmann model, i.e. the Big Bang. From Eq. (2.2) we can define two separate scales of the Universe:

- scales smaller than the cosmological horizon, $r < R_{\mathrm{h}}$, where the microphysical processes become more important and each cosmic component of the system follow a different density evolution;
- scales bigger than the cosmological horizon, $r > R_{\mathrm{h}}$, where gravity is the only force in action and the growth of perturbations has to be treated with the relativistic theory.

Since we are interested in small scales in a non-relativistic framework, we will use a Newtonian approximation for the rest of the section.

Let us assume a fluid to be homogeneous and satisfying the following equations:

- **Continuity equation**:
$$\frac{\partial \rho}{\partial t} + \nabla(\rho \vec{v}) = 0 \ , \tag{2.3}$$
where $\vec{v}$ is the velocity of the fluid element;

- **Euler equation**
$$\frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \nabla) \cdot \vec{v} = -\frac{1}{\rho}\nabla p - \nabla \Phi \ , \tag{2.4}$$
where Φ is the Newtonian gravitational potential generated by the fluid element;

- **Poisson equation**
$$\nabla^2 \Phi = 4\pi G \rho \ . \tag{2.5}$$

The equation of state of this fluid is $p = p(S, \rho)$, where S denotes the entropy. Nevertheless, for our purposes, we can consider **adiabatic perturbations** which means $\mathrm{d}S/\mathrm{d}t = 0$. It follows that the pressure p is assumed to be a function of the density ρ only.

Let us assume to know a static solution of the set of Eqs. (2.3) to (2.5):

$$\begin{cases} \rho = \rho_{\mathrm{b}} = const \\ \Phi = \Phi_{\mathrm{b}} = const \\ p = p_{\mathrm{b}} = const \\ \vec{v} = 0 \end{cases} , \tag{2.6}$$

where the subscript b denotes the background. The set of Eq. (2.6) implies a static universe with no perturbations of any kind. This cannot be a real solution because from Eq. (2.5) we have that $\Phi = const$ if and only if $\rho = 0$, but we are interested in the density perturbations case.

Let us now add small perturbations to our static solution, in order to have an almost negligible change compared to the background values of the four variables:

$$\begin{cases} \rho = \rho_{\rm b} + \delta\rho \\ \Phi = \Phi_{\rm b} + \delta\Phi \\ p = p_{\rm b} + \delta p \\ \vec{v} = \delta\vec{v} \end{cases} . \tag{2.7}$$

If we impose that the values of the variables expressed in Eq. (2.7) are solutions of Eqs. (2.3, 2.4, 2.5), neglecting all the second- and higher-order terms of the perturbations, we obtain the following set of equations:

$$\begin{cases} \frac{\partial\delta\rho}{\partial t} + \rho_b \nabla\delta\vec{v} = 0 \\ \frac{\partial\delta\vec{v}}{\partial t} = -\frac{v_{\rm s}^2}{\rho_b}\nabla\delta\rho - \nabla\delta\Phi \\ \nabla^2\delta\Phi = 4\pi G\delta\rho\delta p \end{cases} , \tag{2.8}$$

where $v_{\rm s}$ is the sound speed in the fluid we are considering.

Let us assume that every perturbation has a wave-like analytical expression:

$$\delta\rho(r,t) = \delta\rho_k e^{ikr+i\omega t} , \tag{2.9}$$

where $k = 2\pi/\lambda$ and $\omega = 2\pi/\nu$ is the wavenumber and the frequency of the fluid element, respectively. This leads to a new system of equations in whose the determinant is null if:

$$\omega^2 = v_{\rm s}^2 k^2 - 4\pi G\rho_{\rm b} . \tag{2.10}$$

This condition is satisfied at the scale:

$$k = \frac{2}{v_{\rm s}}\sqrt{\frac{\pi}{G\rho_{\rm b}}} . \tag{2.11}$$

Let us now define the **Jeans length** as:

$$\lambda_{\rm J} \equiv \frac{2\pi}{k_{\rm J}} = v_{\rm s}\sqrt{\frac{\pi}{G\rho_{\rm b}}} . \tag{2.12}$$

We will see in the next section how this scale determines different behaviours in the evolution of the density perturbations according to their size.

2.1.2 Jeans instability in an expanding universe

Let us assume a homogeneous and isotropic fluid having a constant matter density and enclosed in an expanding Universe. We can look for a solution similar to Eq. (2.7), including this time the background expansion. Therefore $\vec{v}$ is:

$$\vec{v} = \frac{\mathrm{d}\vec{r}}{\mathrm{d}t} = \frac{\mathrm{d}(a\vec{x})}{\mathrm{d}t} = \frac{\mathrm{d}a}{\mathrm{d}t}\vec{x} + a\frac{\mathrm{d}\vec{x}}{\mathrm{d}t} = H\vec{r} + \vec{v_{\mathrm{p}}} \ , \tag{2.13}$$

where $\vec{r}$ is the proper position of the fluid element, while $\vec{x}$ is the comoving ones. The previous equations show us that each fluid element moves with a velocity that is the sum of two contributions: the speed of the Hubble flow, $H\vec{r}$, and the peculiar speed with respect the flow, $\vec{v_{\mathrm{p}}}$.

Using the definition of the **density contrast** $\delta = \delta\rho/\rho_{\mathrm{b}}$, assuming $\delta\vec{v} = \vec{v_{\mathrm{p}}}$, imposing that Eq. (2.7) are solutions of Eqs. (2.3) to (2.5) and looking for wave-like solutions, in a linear regime ($\delta \ll 1$), we can derive at the so-called **dispersion relation** for the wave-like perturbation:

$$\ddot{\delta_k} + 2H\dot{\delta_k} + \left[\frac{k^2}{a^2}v_s^2 - 4\pi G\rho_b\right]\delta_k = 0 \ . \tag{2.14}$$

This has different solutions, which depend on the size of the perturbation, i.e. if it is smaller or larger than the Jeans length. If $\lambda < \lambda_{\mathrm{J}}$ the solution of the differential equation is a wave-like function whose amplitude does not increase over time. Otherwise, if $\lambda > \lambda_{\mathrm{J}}$ the dispersion relation has growing and decaying mode solutions, i.e. a linear combination of two solutions: the increasing one, δ_+, and the decreasing one, δ_-.

The solution that leads the formation of collapsed structures is the increasing one, which for a generic cosmological model is:

$$\delta_+(z) = -H(z)\int_0^z \frac{1+z'}{a_0^2 H^3(z')}dz' \ . \tag{2.15}$$

Instead of Eq. (2.15), which has no analytical solutions, it is generally used a parametric equation defining the **growing factor** f:

$$f \equiv \frac{\mathrm{d}\ln\delta_+}{\mathrm{d}\ln a} \approx \Omega_m^\gamma + \frac{\Omega_\Lambda}{70}\left(1 + \frac{\Omega_\mathrm{m}}{2}\right) , \tag{2.16}$$

where the value of γ depends on which gravity model is assumed. For example, for GR $\gamma \approx 0.545$ (Linder & Jenkins, 2003). From Eq. (2.16) we see how the formation of cosmic structures depends strongly on the density fraction of matter and to a lesser extent on the cosmological constant.

2.1.3 Two-point correlation function and power spectrum

Inflation is thought to have caused the initial density fluctuations in the Universe, changing its structure and metric. From a Newtonian perspective, these fluctuations can be interpreted as perturbations in the gravitational potential Φ. In particular, the Poisson equation Eq. (2.5) links Φ to the density fluctuation δ.

Despite the Universe being a unique entity, a frequentist approach can be employed to model the properties and evolution of large-scale structures. This is possible because, in accordance with the ergodic hypothesis, different volumes of the Universe can be considered as an ensemble of independent universes as long as they are not superimposed and are sufficiently large to be considered independent (Ellis, Maartens & MacCallum, 2012).

Primordial density perturbations can be described by imaginary numbers, with random phase, with a Gaussian distribution. A statistical analysis of the fluctuations can be carried on in both the real and Fourier space. We will consider $\delta(\vec{x})$ in the first case and $\delta(\vec{k})$, or simply δ_k, in the second case. The two quantities are related by the following equation:

$$\delta(\vec{x}) = \frac{1}{(2\pi)^3}\int \delta(\vec{k})e^{-i\vec{k}\cdot\vec{x}}d\vec{k} \ . \tag{2.17}$$

The **two-point correlation function** (2PCF) is a fundamental statistic, which quantifies the spatial clustering of the matter in the Universe, and is defined as:

$$\xi(r) \equiv \langle\delta(\vec{x})\delta(\vec{x}+\vec{r})\rangle \ , \tag{2.18}$$

where the angle brackets indicate the average value between all points that are at a co-moving distance $|\vec{r}|$ from $\vec{x}$. Thanks to the isotropy assumption (see Sect. 1.2), the 2PCF does not depend on the direction of $\vec{r}$.

It is possible to express the 2PCF in Fourier space by combining Eqs. (2.17) and (2.18):

$$\xi(r) = \frac{1}{(2\pi)^3} \int P(k) e^{-i\vec{k}\cdot\vec{r}} \mathrm{d}^3 k \ , \tag{2.19}$$

where we have defined the **power spectrum** as:

$$P(k)\delta_{\mathrm{D}}^{(3)}(\vec{k}+\vec{k'}) = \frac{\langle \delta(\vec{k})\delta(\vec{k'})\rangle}{(2\pi)^3} \ , \tag{2.20}$$

in which $\delta_{\mathrm{D}}^{(3)}$ is the three-dimensional Dirac Delta.

Eq. (2.19) shows us that the 2PCF and the power spectrum are related by a Fourier transform. Additionally, Eq. (2.20) reveals that the power spectrum is proportional to $\langle |\delta(k)|^2 \rangle$, which is the average of the squared magnitude of all the perturbations that are identified by a wavenumber k, regardless of their position $\vec{x}$.

The inflation theory states that primordial density perturbations arise from stochastic quantum fluctuations of a scalar field, the **inflaton**, during a period of exponential expansion (Guth & Pi, 1982). Therefore the distribution of the perturbation amplitudes is accurately described by a Gaussian. Under the assumption that during the creation of the perturbations there was no preferential scale, the power spectrum can be expected to be scale invariant, therefore the **primordial power spectrum** follows a power law given by:

$$P(k) = Ak^n \ . \tag{2.21}$$

Here n is the spectral index whose value is generally assumed to be close to the unity (Zeldovich, 1972), while A is the scalar amplitude and it has to be constrained through observations.

Since standard inflationary models predict that primordial density fluctuations have approximately a Gaussian distribution, we can relate the variance of this distribution to the power spectrum. Defining the variance as follows:

$$\sigma^2 = \langle \delta^2(\vec{x}) \rangle = \frac{1}{V} \int \langle |\delta(\vec{x})|^2 \rangle \mathrm{d}^3x \ , \tag{2.22}$$

where V is the total volume of the Universe and the average is taken from a sample of different realizations of V. Now, considering Eq. (2.22) and the Parseval's relation[1], it turns out that:

$$\sigma^2 = \frac{1}{2\pi^2} \int_0^{\infty} P(k) k^2 \mathrm{d}k \tag{2.23}$$

Unfortunately, there is no way to access to the value of the density contrast in each point of the Universe, and so measure $\delta(\vec{x})$ directly. We can however assume the cosmic structures as indicators of the evolution of density fluctuations, counting the number of galaxies inside a specific volume V to estimate $\delta(\vec{x})$. If galaxies are used as tracers of the density perturbation, then the density contrast in the i-th volume can be written as follows:

$$\delta_{\mathrm{gal}}(V_i) = \frac{N_{\mathrm{gal}}(V_i) - \overline{N_{\mathrm{gal}}}(V)}{\overline{N_{\mathrm{gal}}}(V)} \ , \tag{2.24}$$

where $N_{\mathrm{gal}}(V_i)$, is the measured number of galaxies in the volume V_i , and $\overline{N_{\mathrm{gal}}}(V)$ is the expectation value of the number of galaxies in V.

We assume that the density contrast of the number of galaxies is linearly related to their mass, such that:

$$b = \frac{\delta_{\mathrm{gal}}}{\delta_{\mathrm{M}}}, \tag{2.25}$$

where b is a measure of the discrepancy between the observed distribution of galaxies and the total mass distribution, the so-called **linear bias factor**. While we can define the density contrast of the total mass density fluctuations, δ_{M}, in a given volume V_i in a similar way of Eq. (2.24):

$$\delta_{\mathrm{M}}(V_i) = \frac{M(V_i) - \overline{M}(V)}{\overline{M}(V)} \ , \tag{2.26}$$

where the expected and measured number of galaxies have been substituted with the total mass in the volume V and V_i, respectively.

The discrete matter density fluctuation is the result of the convolution between the

[1] Let $F(k)$ and $G(k)$ be the Fourier transform of $f(x)$ and $g(x)$, respectively. Then $\int_{-\infty}^{+\infty} f(x)\overline{g(x)}dx = \int_{-\infty}^{+\infty} F(k)\overline{G(k)}dk$, where $\overline{z}$ denotes the complex conjugate.

continuous density fluctuation, $\delta(\vec{x})$, and a **window function**, $W(\vec{x}, R)$. We obtain in this way the following filtered function:

$$\delta_{\rm M}(\vec{x}) = \delta(\vec{x}) \otimes W(\vec{x}, R) \ , \tag{2.27}$$

where R is the radius of the sphere inside which we want to estimate $\delta_{\rm M}$.

Combining now Eqs. (2.23) and (2.27), we derive the definition of **mass variance**:

$$\sigma^2_{\rm M} \equiv \langle \delta^2_{\rm M} \rangle = \frac{1}{(2\pi)^3} \int P(k) \hat{W}^2(\vec{k}, R) {\rm d}^3 k \ , \tag{2.28}$$

where $\hat{W}(\vec{k}, R)$ is the Fourier transform of $W(\vec{x}, R)$. Since that inflation theory does not predict the normalisation of the power spectrum, an alternative approach is to set the value of the mass variance calculated with a filtering of $R = 8\ h^{-1}$ Mpc, where $h = H_0/(100$ km s^{-1} Mpc$^{-1})$, at the present time:

$$\sigma^2_8 = \frac{1}{2\pi^2} \int P(k) k^2 \hat{W}^2(R = 8\ h^{-1}{\rm Mpc}) {\rm d}k \ . \tag{2.29}$$

The value of σ_8 represents the mass fluctuation in spheres with radius 8 h^{-1}Mpc, that is the normalization of the matter power spectrum. It is critical for the prediction of the phenomenology of the low-redshift Universe.

2.1.4 Two-point correlation function estimators

The 2PCF defined in Eq. (2.18) measures the amount of clustering of cosmic matter in space, and can be determined using a statistical approach. We can interpret $\xi(r)$ as the excess probability ${\rm d}P_{12}$ of finding a pair of objects separated by a comoving distance r in two independent volume elements ${\rm d}V_1$ and ${\rm d}V_2$, compared to a random uniform distribution of objects, using a discretised representation of the density field:

$${\rm d}P_{12} = n^2_{\rm V}[1 + \xi(r)]{\rm d}V_1{\rm d}V_2 \ , \tag{2.30}$$

where $n_{\rm V}$ is the comoving number density of objects in the total volume V. In practice, the estimation of the 2PCF of a data sample is usually done by comparing the number of pairs of objects in the data sample to those found in a random distribution that has the same geometry and number density pattern as the data sample.

Let us assume a catalogue of N_{D} objects and its associated random sample of N_{R} objects. The number of pairs in the real catalogue will be $N_{\mathrm{DD}} = N_{\mathrm{D}}(N_{\mathrm{D}} - 1)/2$, and $N_{\mathrm{RR}} = N_{\mathrm{R}}(N_{\mathrm{R}} - 1)/2$ in the random catalogue. The number of pairs that can be obtained by combining the two catalogues is $N_{\mathrm{DR}} = N_{\mathrm{D}} N_{\mathrm{R}}$. Let $\mathrm{dd}(r)$ be the number of pairs in the real catalogue at a distance of r, $\mathrm{rr}(r)$ is the analogous in the random catalogue, and $\mathrm{dr}(r)$ is the number of objects in the random catalogue with a distance r from the objects in the real catalogue. It is useful to normalize the number of pairs at a certain distance r by dividing it by the total number of pairs:

$$\begin{aligned} \mathrm{DD}(r) &= \frac{\mathrm{dd}(r)}{N_{\mathrm{DD}}} \\ \mathrm{RR}(r) &= \frac{\mathrm{rr}(r)}{N_{\mathrm{RR}}} \\ \mathrm{DR}(r) &= \frac{\mathrm{dr}(r)}{N_{\mathrm{DR}}} \ . \end{aligned}$$

Now, the 2PCF in its simplest form can be written as:

$$\hat{\xi}_{\mathrm{N}}(r) = \frac{\mathrm{DD}(r)}{\mathrm{RR}(r)} - 1 \ . \tag{2.31}$$

This is known as the Peebles & Hauser (1974) estimator. This estimator has low accuracy at large scales because of the discreteness of the sample. For this reason, more accurate estimators are usually preferred, which take into account the correlation between the data and random catalogues. A popular estimator of this kind is the one proposed by Landy & Szalay (1993):

$$\hat{\xi}_{\mathrm{LS}}(r) = \frac{\mathrm{DD}(r) - 2\mathrm{DR}(r) + \mathrm{RR}(r)}{\mathrm{RR}(r)} \ . \tag{2.32}$$

This estimator is characterized by a variance that is close to that of a Poissonian process. When the number of objects is very large, it gives an unbiased estimate of the 2PCF with minimal variance. This estimator is widely used in astrophysics and cosmology.

2.1.5 The matter power spectrum evolution

Before the matter-radiation equivalence, $t = t_{\rm eq}$, the Universe was dominated by radiation. The Hubble drag term, which can be thought of as an expansion force, was stronger than during the matter era ($t > t_{\rm eq}$). Thus, the density perturbations that enter the cosmological horizon before to the radiation-matter equivalence were unable to grow indefinitely, as they were halted by the expansion of the Universe, reaching a maximum value. This effect is called stagnation or **Mészàros effect** (Mészáros, 1974).

Since the cosmological horizon expands over time (see Equation (2.2)), larger perturbations will enter it at later times and thus experience less stagnation. Consequently, the power spectrum has a peak at the wavenumber $k_{\rm h,eq}$, associated with the cosmological horizon at the moment of equivalence.

The shape of $P(k)$ at the equivalence time can be reproduced by defining a **transfer function**, which gives us the fraction of the primordial power spectrum that is not affected by the microphysical effects within the horizon. Let us define the transfer function, $T(k)$, such that:

$$P(k, t_{\rm eq}) = P(k, t_i)\; T^2(k) \;, \tag{2.33}$$

where t_i is a generic cosmological time. $T(k)$, in a CDM scenario (see Sect. 1.5), has the following behavior:

$$T(k) = \begin{cases} 1 \quad \text{for} \quad k < k_{h,\rm eq}, \\ \propto k^{-2} \quad \text{for} \quad k > k_{h,\rm eq} \;. \end{cases} \tag{2.34}$$

The transfer function acts like a filter that does not affect the scales with $k < k_{h,\rm eq}$, but reduces those with $k > k_{h,\rm eq}$. The effect of this filter depends on the type of matter we are considering. Figure 2.1 shows the power spectrum trends for CDM and **hot dark matter** (HDM), which is a relativistic dark matter.

The peak at $k_{\rm h,eq}$, showed in Fig. 2.1 mostly depends on $\Omega_{\rm m}h^2$ and $\Omega_{\rm r}h^2$, so on the matter and radiation densities and the Hubble parameter. Therefore the observed shape of $P(k)$ is a powerful tool to constrain the Universe cosmology.

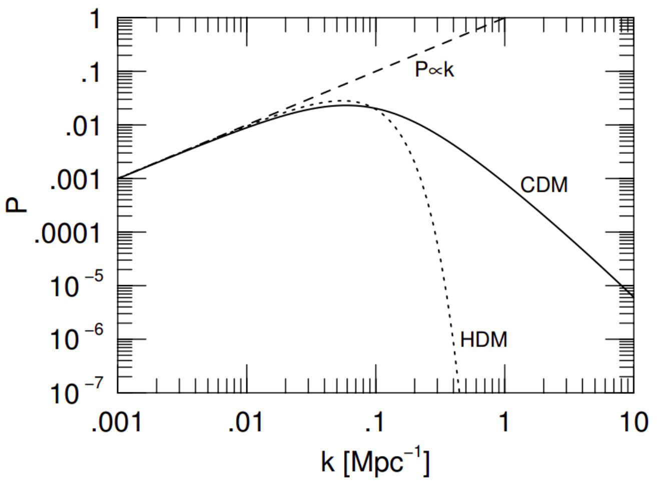

Figure 2.1: Behaviour of the matter power spectrum for different type of matter. The solid line represents the power spectrum at the equivalence time for a matter component made up only of CDM, while the dotted line represents the power spectrum for a matter component made up only of HDM. As a reference, the primordial Zel'dovich power spectrum, having a spectral index $n = 1$ is reported with a dashed line (see Eq. (2.21)). Credits to Ryden (2016).

2.2 Nonlinear theory

The observable local Universe contains cosmic structures such as galaxies, clusters, and DM haloes, which are the result of gravitational instabilities that occurred over the course of cosmic history. The small-perturbations approximation cannot be used to explain the formation of these objects, which are characterized by a very nonlinear regime ($\delta \gg 1$).

Once the linear regime is no longer valid, meaning when δ is close to 1, the **weakly-nonlinear** regime begins. At this stage, the distribution of fluctuations starts to differ from the Gaussian shape. Additionally, the evolution of baryons is not the same as DM since baryons are subject to hydrodynamical effects like star formation, supernovae explosions, and active galactic nuclei (AGN) feedback. These effects make it more difficult to create a comprehensive and accurate theory. Although some approximated analytical models exist to explain this, we mainly use **N-body simulations** to accurately model the weakly-nonlinear and nonlinear perturbation growth.

2.2.1 N-body simulations

Cosmologists use numerical simulations to study the **large-scale structure** (LSS) of the Universe in the nonlinear regime, as it is too complicated to be studied analytically. This involves selecting a cosmological theory and a set of appropriate parameters and then running a simulation to see how the initial system evolves. The result of the simulation can then be compared to observed data sets. This process approximates the formation of cosmic structures as the dynamical evolution of a system of particles, which trace the total mass distribution.

To accurately model the evolution of density fluctuations, the most important factor to consider is the gravitational force, which is most prominent on large scales and affects the majority of the matter of the Universe, i.e. DM. Simulations that only include the gravitational force are called N-body simulations. To get a more realistic representation of the large-scale structure of the Universe, simulations must also take into account the hydrodynamic effects of the baryonic matter. Simulations that include this baryonic component are called **hydrodynamic simulations**. We are only interested in N-body simulations for our purposes, and we will briefly describe their theoretical basis.

Let us consider the simplest kind of N-body simulations, which includes only gravitational effects, solving the following differential equation system:

$$\begin{cases} \vec{F_i} = G M_i \sum_{j\neq i} \frac{M_j}{r_{ij}^2}\hat{r}_{ij} \\ \ddot{\vec{x}}_i = \frac{\mathrm{d}\vec{v}_i}{\mathrm{d}t} = \frac{\vec{F_i}}{M_i} \\ \dot{\vec{x}}_i = \frac{\mathrm{d}\vec{x}_i}{\mathrm{d}t} = \vec{v}_i \end{cases} , \tag{2.35}$$

where for each i-th particles we have that $\vec{F_i}$ is the gravitational force, M_i is the inertial mass, $\hat{r}_{ij}$ is the comoving distance versor, $\vec{v}_i$ is the velocity component and $\vec{x}_i$ is the comoving coordinate vector. Given the system of equations 2.35, the Eq. (2.4) can be re-written as follows:

$$\frac{\mathrm{d}\dot{\vec{x}}_i}{\mathrm{d}t} + 2\frac{\dot{a}}{a}\vec{v}_i = -\frac{1}{a^3}\nabla\Phi = -\frac{G}{a^3}\sum_{j\neq i}\frac{\vec{x}_i - \vec{x}_j}{|\vec{x}_i - \vec{x}_j|^3} = \frac{\vec{F_i}}{a^3} . \tag{2.36}$$

Eq. (2.5) can be calculated, by using Eq. (1.18) (the Second Friedman Equation):

$$\nabla^2\Phi = 4\pi G\overline{\rho}(t)a^2\delta = \frac{3}{2}H_0^2\Omega_0\frac{\delta}{a} \ , \tag{2.37}$$

where $\overline{\rho}(t)$ is the average non-relativistic matter density and δ the local density contrast.

An N-body simulation solves numerically the equations that govern the motion of N objects over time and in discrete intervals. At each interval, δt, the total gravitational force of the system, that is $\vec{F}_i$ for each object i, is calculated. The most accurate and most expensive method to calculate the gravitational force acting on the i-th particle is the particle-particle method, which requires computing the $N(N-1)/2$ distances between the particles. This method takes a huge amount of computational time, as the number of operations scales as $O(N^2)$.

The equation of motion is solved by calculating the new positions, $\vec{x}_i(t \pm \delta t)$, and velocities, $\vec{v}_i(t \pm \delta t)$, using numerical integration. The time is then increased, $t = t + \delta t$, and the process repeated. Different criteria can be used to determine the value of δt, which can be suitable for different approaches. These criteria can be divided into three main categories (Bagla & Padmanabhan, 1997):

i) total energy conservation,

ii) convergence of final positions and velocities,

iii) reproducibility of the initial conditions.

The selection of one of the above criteria for determining δt in an N-body simulation depends upon the numerical integration method used to solve the equations of motion. Generally, the criterion that provides the most accurate estimate of δt while utilizing the fewest computational resources is preferred.

Chapter 3

Machine Learning and Bayesian inference

In this chapter, we will first give a brief overview of the basic foundations of machine learning. Afterwards, we will introduce the python package `CosmoPower`[1] (Spurio Mancini et al., 2022), a set of neural power spectrum emulators that allow us to compute the theoretical 2PCF. At the end of this chapter, we will provide a brief overview of the Bayesian inference and Markov Chain Monte Carlo, which are essential for validating the accuracy of the emulator and estimating the speed up of cosmological analyses.

3.1 Machine Learning: supervised learning

The computer scientist Arthur Samuel used the term **machine learning** (ML) for the first time in 1959 to describe certain computer algorithms. In the present day, ML has developed into one of the mainstays of computer and data science, and it has been incorporated into almost every part of everyday life. For instance, Google, YouTube, and Netflix have all improved their search engine and "recommendation" functions by implementing learning algorithms that record users' choices and preferences to create a theoretically personalised experience for each user. Additionally, ML algorithms, particularly those related to pattern recognition, are commonly used by social networks to identify, e.g., different people in photos. They also have applications in medicine, with 2D/3D images of human tissues and

[1]https://github.com/alessiospuriomancini/cosmopower

organs being employed for diagnostic purposes. ML technologies are also used in a variety of other fields too. The rise of learning algorithms is largely due to the proliferation of data, coupled with advances in storage and computing capabilities, which can be utilized with lower costs for upkeep and materials.

ML has found applications also in various astrophysical and cosmological pursuits, such as image recognition in gravitational lensing studies (Hezaveh, Perreault Levasseur & Marshall, 2017), measuring photometric redshift using galaxy images (Hoyle, 2016), morphological galaxy classification (de la Calleja & Fuentes, 2004), determining the dynamical mass of galaxy clusters (Ntampaka et al., 2015), learning from the two-point correlation function of BOSS galaxies (Veronesi et al., 2023) and exoplanet transit detection (Schanche et al., 2019). In a few years, new instruments such as *Euclid* (Laureijs et al., 2011; Amendola et al., 2018; Euclid Collaboration: Blanchard et al., 2020), the Large Synoptic Survey Telescope (LSST, LSST Dark Energy Science Collaboration, 2012; Graham et al., 2018; Ivezić et al., 2019), and the James Webb Space Telescope (JWST, Gardner et al., 2006; Kalirai, 2018) will provide an unprecedented volume of data for future studies, making ML a pivotal factor in the cosmology of the modern era.

Several methods exist to enable an algorithm to learn. Supervised, unsupervised and reinforcement learning are the key categories of learning algorithms. We will focus on one type of **supervised learning** in the following section, since this is the technique implemented in `CosmoPower`, i.e. one of the numerical libraries that we will employ in this Thesis work.

In supervised learning, the goal is to get a function that maps an input, $\mathbf{x}$, to an output, $\mathbf{y}$, based on observations of several examples of the input-output relationship. This specific learning algorithm aims at identifying a function linking an input vector to an output vector, based on **input-output pairs** that have been provided. This is achieved in the training process: during this stage, along with the input details, a target variable is also presented (acting as the "truth"), which contains the details that the algorithm needs to learn. The term **supervised** originates from the idea of an external supervisor who guides the ML system, providing it with the desired outcome.

3.1.1 Linear regression and loss function

Let us take a look at a concrete example of a ML algorithm to further understand the concept of learning. The goal is to develop a system that takes a vector $\mathbf{x} \in \mathbb{R}^n$ as input and gives an accurate prediction of a scalar $y \in \mathbb{R}$ in output. The predicted value $\hat{y}$ is determined through a **linear regression** model, which is a linear function of the input $\mathbf{x}$:

$$\hat{y}(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b \ , \tag{3.1}$$

where $\mathbf{w} \in \mathbb{R}^n$ is a vector of parameters called **weights**, while b is a scalar parameter called **bias**. This terminology originates from the perspective that the output of the transformation is biased towards a value of b in the absence of any input.

These parameters $\mathbf{w}$ and b control the behaviour of the linear regression model. Indeed, if a feature x_i has a positive weight w_i, then increasing the value of that feature will lead to an increase in the value of the prediction $\hat{y}$. Conversely, if the feature has a negative weight, increasing the value of the feature will result in a decrease in the prediction. When the magnitude of the weight is large, it has a large impact on the prediction, while a weight of zero has no effect.

Training a model involves determining the values of all the weights and bias from labeled examples so as to achieve an accurate prediction $\hat{y}$. In supervised learning, the algorithm creates a model by analyzing numerous examples and attempting, iteratively, to identify a model that minimizes the **loss**, i.e. a measure of the difference between the output of the model and the expected output.

In order to calculate the amount of loss incurred, we need to assume a specific **loss function**. The latter is a measure of how well a model is performing and it is usually evaluated on a data set. There are various types of loss functions that can be employed, for instance, the mean squared error (MSE) has the following expression:

$$\mathrm{MSE} = \frac{1}{N} \sum_{(\mathbf{x},y)\in D} (y - \hat{y}(\mathbf{x}))^2 \ , \tag{3.2}$$

where $(\mathbf{x}, y)$ is a labeled example and N is the number of the examples. We will refer to y as **labels** and $\mathbf{x}$ as **features**, while D as a **training data set**. The latter is a set of labeled examples to train $\hat{y}$, to which we will refer as the **model**. The aim is to reduce the MSE (thus the loss) as far as possible, although it does not necessarily have to be a null value, as we will see in the following.

3.1.2 Optimization and gradient descent

Optimization in ML is the process of finding the best set of parameters to minimize a given objective function, for instance the MSE in linear regression model. This involves finding a set of parameters that minimize the loss function of a ML model. Optimization in ML is used to tune the model parameters and improve the accuracy of the model.

Let us now present an example of ML optimization, considering again the linear regression model (see Sect. 3.1.1) and assuming the MSE as the objective function, $l(\mathbf{w})$. Initially, when the training starts, all the weights are randomly set and the value of the loss is calculated over D, as in Eq. (3.2): this is the so-called **forward propagation**.

Let $\mathbf{w}_1$ be the vector of the initial weights value. Considering Eq. (3.2) it turns out:

$$l(\mathbf{w}_1) \equiv \mathrm{MSE} = \frac{1}{N} \sum_{(\mathbf{x},y)\in D} (y - \mathbf{w}_1^{\mathrm{T}}\mathbf{x} + b)^2 \ , \tag{3.3}$$

where we have only explicitly stated the dependence on $\mathbf{w}_1$. Our purpose is to reduce the objective function as much as possible, so we will calculate the gradient of $l(\mathbf{w})$ for $\mathbf{w}_1$ to determine the direction in which it is increasing. At the next step, it will be necessary to select a $\mathbf{w}_2$ in a direction opposite to that of the gradient to ensure that $l(\mathbf{w}_2) < l(\mathbf{w}_1)$:

$$\mathbf{w}_2 = \mathbf{w}_1 - \eta\nabla[l(\mathbf{w}_1)] \ , \tag{3.4}$$

where η is a hyperparameter called **learning rate**, which is lower than 1 and serves to reduce the magnitude of $\nabla[l(\mathbf{w}_1)]$. After computing $\mathbf{w}_2$, the **back propagation** process starts, and the newly calculated values are placed into the model. This optimization technique is called **gradient descent** and is the way the linear regression model moves within the space of weights in a supervised learning. The training phase in ML covers a certain number of **epochs**, including one or more cycles of forward and back propagation, the so-called **training iteration**.

Though it may appear that the gradient descent guarantees loss minimization, this is not always the case. An improper choice of η can lead to the inability to reach the minimum loss, or to a long number of epochs to get there. Indeed, let us suppose to have a loss function with a shape like that in Fig. 3.1: setting a learning rate that is too small takes too many steps to reach the minimum of the curve, while if chosen too large there is a risk that the minimum will be missed.

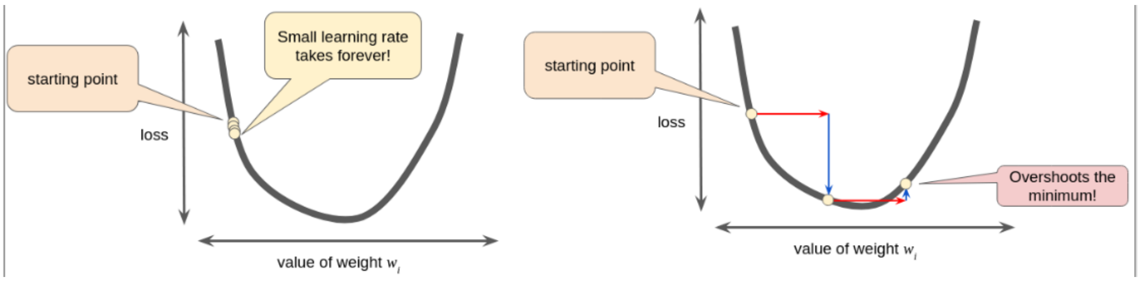

Figure 3.1: Effect of different settings of learning rate on the loss function. Setting a learning rate too small (left) can lead to an indefinitely increased time to reach the minimum, whereas setting a value too large (right) can impede reaching the minimum. Credits to: `https://developers.google.com/machine-learning/crash-course/reducing-loss/learning-rate`.

The choice of the features is crucial when feeding an algorithm. For the model to be able to generalize and make correct predictions on new inputs, these features must provide a thorough description of the data set. For example, if the goal is to emulate a cosmological power spectrum model, appropriate features may include parameters defining the cosmology, such as matter density parameters (see Sect. 1.4.1). It is also essential to provide the algorithm with the correct label **representation.** For instance, `CosmoPower` offers the option of using either $P(k)$ or $\log[P(k)]$ as the label representation of the matter power spectrum. The choice between the two label representation can significantly impact the precision of the output model.

3.2 Machine Learning: deep neural network architecture

As mentioned in the introduction of this chapter, ML algorithms have started to be used extensively in Astrophysics and Cosmology. The implementation of these powerful techniques is expensive though, as it requires a great amount of resources to manage data sets of the order of terabytes needed for training. Luckily, emulators offer an innovative solution to this problem.

Emulators are models derived through ML algorithms that can emulate the output of cosmological models or simulations, with minimal loss of accuracy, but taking much less execution time. This makes the process of deriving scientific results much quicker and also removes the need to handle large data sets, as the realization of such outputs is not overly expensive. Emulators designed to increase efficiency have already been documented in the literature. For instance, emulators have been used to compute the power spectrum for galaxy cluster analyses (e.g. DeRose et al., 2022), the two-point angular statistics for photometry of galaxy surveys (e.g. Bonici et al., 2022) and the power spectrum with massive neutrinos and self-consistent DE perturbations (e.g. Euclid Collaboration et al., 2021).

The aim of this Thesis work is the implementation of a 2PCF emulated model within the `CosmoBolognaLib` (see Sect. 4.1) environment. We start with the computation of the 2PCF model. We chose the `CosmoPower` suite of neural cosmological matter and CMB power spectra emulators for our purpose. This library is not only more convenient to use, but also more versatile than other learning algorithms. In particular, we employ `CosmoPower` to create and train deep neural network models, designed to emulate the power spectrum of LSS.

In Sect. 3.1.1, we examined the linear regression model. Nonetheless, this type of model is not able to replicate a cosmological function such as the power spectrum (see, for example, Eq. 2.33). With `CosmoPower` it is possible to implement more complex learning algorithms that employ nonlinear models known as **deep neural networks** (DNN), thus enabling a greater level of complexity in the processing of features, adequate for our requirements.

A DNN architecture consists of multiple layers of **neurons**, which takes in input data and produces an output, through a mathematical function called **activation function**. Each model layer is connected to the next, in a hierarchical fashion: the input layer at the bottom, the **hidden layers** in the middle and the output layer at the top. Each layer has a certain number of neurons that are connected to the neurons in the adjacent layers. The neurons in each layer receive input from the neurons in the previous one and, once it gets processed, it gives its output to the neurons in the next layer. Each neuron has a set of weights which determine how much influence it has on the output of the layer. Fig. 3.2 schematizes what is described above.

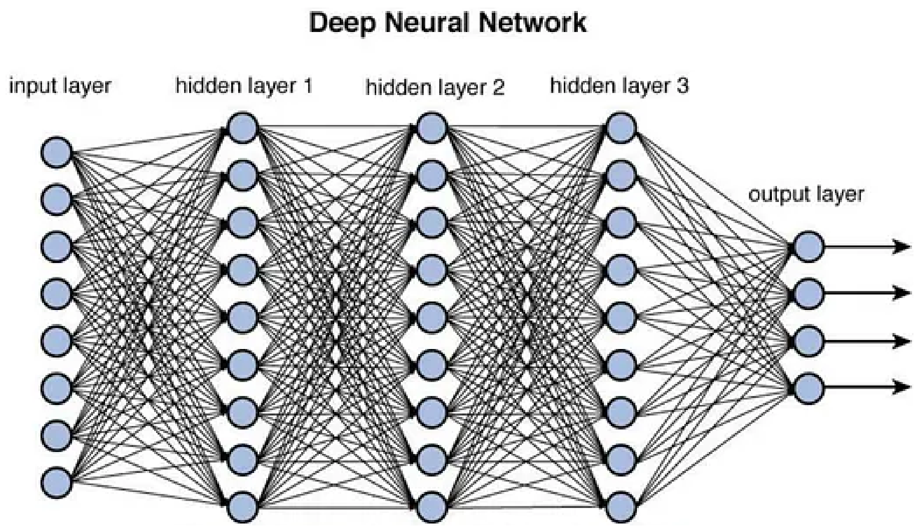

Figure 3.2: An example of DNN. The model takes features in the input layer, which are elaborated in the hidden layers. Finally the output layer is the outcome of the model. Credits to: https://towardsdatascience.com/training-deep-neural-networks-9fdb1964b964.

3.2.1 Activation function

In the whole DNN architecture, one of the most important roles is played by the activation functions. These functions allow each neuron to process the incoming information, which is an essential component for the model success. Different types of activation functions can be used in ML models, but for the purpose of this discussion, we will focus exclusively on the activation functions used in `CosmoPower`. Let n be the number of neuron in a hidden layer, thus the activation function implemented in these libraries is (in matrix form):

$$f(\mathbf{x}) = \left(\boldsymbol{\beta} + \left(1 - \boldsymbol{\beta}\right) \odot \left(1 + e^{-\boldsymbol{\alpha}\odot\mathbf{x}}\right)^{-1}\right) \odot \mathbf{x} \,, \tag{3.5}$$

where $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^n$ are vector parameters of the activation function, $\mathbf{x}$ is the input layer or a hidden layer, while $\odot$ is the wise-element product.

Eq. (3.5) allows us to express the mathematical form of a hidden layer in a DNN model.

If we have I hidden layers in our model, with J neurons in each layer, the j-th neuron of the i-th hidden layer, h^i_j, can be expressed mathematically as:

$$h^i_j(\mathbf{h}^{i-1}) = \sum_{k=1}^{J} \left(w^{i-1}_{jk} f(h^{i-1}_k) \right) + b^i_j = \sum_{k=1}^{J} \left[w^{i-1}_{jk} \left(\beta^i_j + \frac{i - 1\beta^i_j}{1 + e^{-\alpha^i_j h^{i-1}_k}} \right) h^{i-1}_k \right] + b^i_j \ , \tag{3.6}$$

where w^{i-1}_{jk} is the weight parameter, h^{i-1}_k is the k-th neuron of the $(i-1)$-th layer, while b^i_j and the pair (α^i_j, β^i_j) are, respectively, the bias parameter and the coefficients of the activation function. We observe that the weight parameters in Eq. (3.1) form a vector, as the linear regression model consists of an input layer and an output layer with a single value. However, Eq. (3.6) represents a more complex situation, as there is a hidden output layer of J elements processing an input layer of the same amount of components. Therefore, for each layer of the DNN model, the weight parameters are not represented as vectors, but as matrices, while the bias parameters and the activation function coefficients are represented as a vectors.

3.3 Machine Learning: model training

In Sect. 3.1, we delved into the training phase of a linear regression model. Here, we will deepen the discussion on the training of a DNN model in a supervised learning context, with a focus on the techniques that can be implemented through `CosmoPower`.

After the architecture of the model has been determined - i.e. once the number of hidden layers, the number of neurons in each layer and the activation function have been chosen - the first training **epoch** begins. This consists of randomly selecting the trainable parameters, which, in the case of a DNN model like the one of Sect. 3.2, are $\mathbf{w}, \mathbf{b}, \boldsymbol{\alpha}, \boldsymbol{\beta}$ (see Eq. 3.6). Let us refer to $\mathbb{P}$ as the set of all trainable hyperparameters[2]:

$$\mathbb{P} = \{\mathbb{W}, \mathbb{B}, \boldsymbol{\alpha}, \boldsymbol{\beta}\} \ , \tag{3.7}$$

where $\mathbb{W}$ is the set of all weight matrices, $\mathbb{B}$ is the set of all bias vectors, while $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the set of all coefficient vectors belonging to the activation functions. When the forward

[2]We use a different notation for the subsets of hyperparameters, to distinguish those that belong to the model architecture, from those that belong to the activation function.

propagation (see Sect. 3.1.2) starts, all the labeled examples are processed by the DNN model and the loss function is evaluated.

Let us now take a closer look at the loss function, l. Let us assume a training set, D, formed by $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ labeled examples, where $i \in [1, N]$, and let e be a training epoch, where $e \in [1, E]$ the set of all epochs. The loss function implemented in `CosmoPower` is the **root mean square error** (RMSE). For a generic epoch e, RMSE, is written as:

$$l_e(D) = \sqrt{\frac{\sum_{i=1}^{N} |\boldsymbol{y}_i - \hat{\boldsymbol{y}}(\mathbf{x}_i)|^2}{N}} \ , \tag{3.8}$$

where $\hat{\boldsymbol{y}}(\mathbf{x}_i)$ is the model prediction for the i-th example, which this time is not a scalar, but is instead a vector. Before moving to the next epoch, the model must be optimized.

Rather than employing a gradient descent optimization (see Eq. 3.4), `CosmoPower` utilizes adaptive moments estimation (ADAM, Kingma & Ba, 2014) to optimize its models. This optimizer estimates the first and second moments, respectively, $\mathbf{m}$ and v, of the gradient of the loss function, to adjust the learning rate. At the epoch e, ADAM employs the exponential decay rates to compute the gradient moments as follows:

$$\begin{aligned} \mathbf{m}_e &= \beta_1 \mathbf{m}_{e-1} + (1-\beta_1)\nabla l(\mathbf{p}_{e-1}) \\ v_e &= \beta_2 v_{e-1} + (1-\beta_2)|\nabla l(\mathbf{p}_{e-1})|^2 \ , \end{aligned} \tag{3.9}$$

where $\beta_1, \beta_2 \in [0, 1[$ are the exponential decay rates, l is the loss function, and $\mathbf{p} \in \mathbb{P} = \{\mathbf{w}, \mathbf{b}, \boldsymbol{\alpha}, \boldsymbol{\beta}\}$ the set of trainable parameter vectors for the DNN model (see Sect. 3.2). Once $\mathbf{m}_e$ and v_e have been computed, the back propagation (see Sect. 3.1.2) starts, so the parameter vector is updated in the following manner:

$$\mathbf{p}_e = \mathbf{p}_{e-1} - \eta \frac{\hat{\mathbf{m}}_e}{\sqrt{\hat{v}_e} + \epsilon} \ , \tag{3.10}$$

where $\mathbf{m}_e = \frac{\mathbf{m}_e}{1-\beta_1^t}$ and $\hat{v}_e = \frac{v_e}{1-\beta_2^e}$ are, respectively, the unbiased first and second moments of the gradient, while η is the learning rate and ϵ is the ADAM parameter, usually set equal to 10^{-8}.

For each epoch, the loss function is calculated on all the N labeled examples, needing both forward propagation and back propagation. This can lead to long gradient calculation times. Therefore, a **batch** is typically included as part of the training process. A batch is a set of examples used for a single training iteration. By selecting a batch size of n, such that $n < N$ and $N/n \in \mathbb{N}$, the loss function summation will have fewer terms, thus speeding up gradient calculation. Moreover, for each epoch e, the number of training iteration grows ($N/n > 1$), as all examples must be evaluated. It is essential to select accurately the batch size, since a small one would lead to the usage of a reduced amount of examples, while large one would not result in any time savings when calculating the gradient.

Selecting the right number of epochs is also essential. If the chosen number is too small, the algorithm might not converge in the minimization of the loss function. On the other hand, if it is too large a lot of computation time might be wasted, as the training will continue for many epochs even after the reaching of the minimum, without any further improvement.

3.3.1 The learning steps

In order to effectively monitoring the training process, it is necessary to break it down into the **learning steps**. This can be done by running a fixed number of total epochs, E, multiple times and evaluating the model accuracy with a **validation set**. The validation set is a subset of the training set that is used to assess the accuracy of the model in generalizing well, that is in accurately predicting data set not used for training. This is done by performing only the forward propagation and measuring the value of the loss function.

By setting the learning steps, we are able to adjust the learning rate and batch size in order to create a training process that gradually decreases the learning rate, while simultaneously increases the batch size, as training progresses. At each learning step, the optimization process becomes more accurate. In fact, it evaluates a greater number of examples to find the minimum loss function and moves in the parameter space with more precision since η is smaller (see Eq. 3.10). Finally, a **patience value** can be set to limit the amount of training time; this value is the max number of epochs that can elapse before the loss function further decreases or else the learning algorithm will move on to the next learning step.

3.4 Machine Learning: model validation

Once the training of the DNN is finished, we test it to check its accuracy in the model prediction. For example, in our case we evaluate how well the DNN is able to emulate a cosmological function. To do this, we use a **validation data set** that contains labels that the model has not seen before. We refer with T to the training set of all $(\mathbf{x}, \mathbf{y})$ features-labels pairs never seen by the DNN model.

We compare the RMSE (see Eq. 3.8) of our model on the training data set, RMSE(D), to the one on the testing data set, RMSE(T). If RMSE$(D) \ll$ RMSE(T) and RMSE(D) $\ll 1$ it means that the model is **overfitting** the training set. Overfitting occurs when the model performs well on training data but generalizes poorly to unseen data. This issue may arise when the architecture of the DNN is too complicated.

Our aim is to make RMSE(T) as similar to RMSE(D) as possible to guarantee the accuracy of our model beyond the training set. Though, this might not be sufficient. Even if both values are similar, there might be **underfitting**, which is the opposite of overfitting, and occurs when the model is not able to obtain a sufficiently low RMSE(D). This could imply that the model is not appropriate for the task or it has not been trained properly. To make sure the model has a **balanced fit**, we should have the lowest RMSE(D) and the smallest gap between training and testing RMSE.

3.5 Bayesian Methods for Cosmology

In modern cosmology, the **Bayesian inference** is the dominant paradigm for determining the parameters of cosmological models. This is accomplished by comparing and combining data from a variety of observations. As data sets become larger and more accurate, it is necessary to apply increasingly accurate analysis techniques to account for both statistical error bars and systematic effects. The primary goal of the cosmological statistical analyses is to determine which parameters of a certain cosmological models provide the best fit to observations.

3.5.1 Bayesian Inference of cosmological parameters

In a Bayesian statistical framework, the probability is a measure of the degree of belief in a given proposition, rather than a property of the event itself. This concept is applicable to any event, including repeated experiments, one-off situations, random variables, or assumptions of a model (Trotta, 2017). Before moving forward, let us attempt to outline this framework in a rigorous and mathematical fashion.

Let (Ω, F, P) be a measure space, with Ω a set, F a σ-algebra[3] on Ω, and P a real-valued function on (Ω, F) such that:

- $P(E) \in \mathbb{R},\ P(E) \geq 0\ \forall E \in F$,
- $P(\Omega) = 1$,
- $P\left(\bigcup_{i=1}^{\infty} E_i\right) = \sum_{i=1}^{\infty} P(E_i)\ \forall E_i \in F$ such that $P(E_i \cap E_j) = 0\ \forall i \neq j$
- $P(E_i, E_j) = P(E_i)P(E_j|E_i)\ \forall i, j \in \mathbb{N}$, where $P(E_i, E_j)$ is the probability that both E_i and E_j occur, i.e. the **joint probability**, while $P(E_j|E_i)$ is the probability that E_j occurs when E_i has already occurred, i.e. the **conditional probability**.

The first three conditions are known as the **Kolmogorov Axioms**, and the fourth is the **Product Rule**. In this context, a (Ω, F, P) measure space is referred to as a **probability**

[3] In mathematical analysis and probability theory, a σ-algebra on a set X is a nonempty collection Σ of subsets of X that is closed under the operations of complement, countable union and countable intersection.

space, with sample space Ω, event space F and the **probability distribution** P. Thanks to these relations we can re-write the conditional probability as:

$$P(E_i|E_j) = \frac{P(E_i)P(E_j|E_i)}{P(E_j)} \ , \tag{3.11}$$

$\forall E_i, E_j \in F \ \wedge \ P(E_j) \neq 0$. This is the so-called **Bayes' theorem**, which describes the probability of an **event**, E_i, based on prior knowledge of conditions, E_j, that might be related to the event.

In Bayesian probability, the term "event", as written above, should be understood broadly: in cosmology we can refer to E_i as $\mathbf{H}$, the parameter vector of a given model, and to E_j as d, the set of observational or simulated data. The importance of the conditional probability, $P(d|\mathbf{H})$, becomes evident when we make the parameters $\mathbf{H}$ vary while keeping d constant. In this case we obtain a function depending on the parameters, which tells us how admissible the data d are when the parameters $\mathbf{H}$ vary, allowing us to define the **likelihood** function as:

$$\mathcal{L}(\mathbf{H}) \equiv P(d|\mathbf{H}) \ . \tag{3.12}$$

Thanks to the above definition we can re-write Eq. (3.11) as:

$$P(\mathbf{H}|d) = \frac{\mathcal{L}(\mathbf{H})P(\mathbf{H})}{P(d)} \ , \tag{3.13}$$

where $P(\mathbf{H}|d)$ is the **posterior** probability distribution, which represents our knowledge of the hypothesis $\mathbf{H}$ after examining the data d. The posterior is proportional to the likelihood $\mathcal{L}(\mathbf{H})$ multiplied by the **prior** probability distribution $P(\mathbf{H})$, which represents the state of knowledge about the hypothesis before the data are analysed. Since the data set is fixed, the posterior distribution is determined by the likelihood function and prior distribution, with $P(d)$ acting as a normalization factor.

The knowledge about the prior can be derived from previous experiments. For example, if these experiments provide a parameter posterior that is normally distributed, we can embed this information thanks to a **Gaussian prior**, fully characterized by mean and standard deviation. Alternatively, the prior may express the theoretically accepted range of values for a given parameter of a cosmological model. In this case it is referred to as a **flat prior**, given that each value within the range is considered equally probable.

Let $\mathbf{H}$ belong to $\mathbb{R}^n$, then we can write the **probability density function** (PDF) of the flat prior distribution as:

$$p(\mathbf{H}) = \begin{cases} \Pi_{j=1}^{n}\left(\frac{1}{H_j^{\mathrm{u}}-H_j^{\mathrm{l}}}\right) & \forall\ \mathbf{H} \in \Omega \\ 0 & \forall\ \mathbf{H} \in \mathbb{R}^n \setminus \Omega \end{cases} \quad , \tag{3.14}$$

where $p(\mathbf{H})$ is defined as:

$$P(\Omega) = \int_{\Omega} p(\mathbf{H})\mathrm{d}\Omega = \int_{H_1^{\mathrm{l}}}^{H_1^{\mathrm{u}}} ... \int_{H_n^{\mathrm{l}}}^{H_n^{\mathrm{u}}} p(\mathbf{H})\ \mathrm{d}H_1...\mathrm{d}H_n = 1 \ , \tag{3.15}$$

where H_j^{l} and H_j^{u} are the lower and upper bound of each parameters validity range, such that $\Omega = [H_1^{\mathrm{l}}, H_1^{\mathrm{u}}] \times ... \times [H_n^{\mathrm{l}}, H_n^{\mathrm{u}}]$. Naturally, the PDF is not an exclusive to the prior probability distribution. In fact each probability distribution has its own PDF, which is defined analogously to Eq. (3.15).

In an inference problem, we look for the parameters that give the **best fit** to the data. In other words, we are interested in the relative probability of the parameters, rather than their absolute value. This means that the normalization constants are not important, as we are looking for the parameters that maximize the probability distribution, rather than the value of the probability at its maximum. Therefore, assuming a flat prior distribution, setting $k = \Pi_{j=1}^{n}\left(\frac{1}{H_j^{\mathrm{u}}-H_j^{\mathrm{l}}}\right)$, Eq. (3.13) becomes:

$$P(\mathbf{H}|d) \propto \mathcal{L}(\mathbf{H}) \ . \tag{3.16}$$

Suppose we have computed the posterior distribution, $P(\mathbf{H}|d)$, and we want to find the posterior probability for a specific parameter. In a general case, the parameter vector will be $\mathbf{H} = (H_1, ..., H_n) \in \mathbb{R}^n$ and we can rewrite the posterior as:

$$P(\mathbf{H}|d) = P(H_1, ..., H_n|d) \ . \tag{3.17}$$

Therefore the posterior for the i-th parameter, H_i, will be:

$$P(H_i|d) = \int_{H_1^{\mathrm{l}}}^{H_1^{\mathrm{u}}} ... \int_{H_{i-1}^{\mathrm{l}}}^{H_{i-1}^{\mathrm{u}}} \int_{H_{i+1}^{\mathrm{l}}}^{H_{i+1}^{\mathrm{u}}} ... \int_{H_n^{\mathrm{l}}}^{H_n^{\mathrm{u}}} p(H_1, ..., H_i, ..., H_n|d)\ \mathrm{d}H_1 ... \mathrm{d}H_{i-1}\mathrm{d}H_{i+1}...\mathrm{d}H_n \ , \tag{3.18}$$

where $p(H_1, ..., H_n|d)$ is the PDF of $P(\mathbf{H}|d)$. This is called **marginalization** and $P(H_i|d)$ is the **marginal posterior**. It is the projection of the posterior along the parameter H_i and is often used to marginalize over the **nuisance** parameters, which are parameters that we must take into account for the statistical analysis, but without physical interest.

Marginalization is also necessary to compute the **credible interval**, i.e. an interval in the marginal posterior domain within a parameter of $\mathbf{H}$ falls with a certain probability. For instance, let H_j be the parameter of interest, then the credible interval, $[H_j^{\mathrm{Cl}}, H_j^{\mathrm{Cu}}]$, will be such that:

$$\int_{H_j^{\mathrm{Cl}}}^{H_j^{\mathrm{Cu}}} p(H_j|d)\ \mathrm{d}H_j = C \ , \tag{3.19}$$

where $p(H_j|d)$ is the PDF of $P(H_j|d)$ defined as in Eq. (3.18), while C is the probability that we want to obtain. The C value can be interpreted as a multiple of σ, the variance of a normal distribution, thanks to the **central limit theorem**. The latter establishes that the sampling of an infinite independent and identically probability distributions tends towards to the Gaussian one. In this case imposing $C \equiv 1\sigma$ and evaluating Eq. (3.19), will yield the credible interval within which there is the 68.3% probability to get the true value of H_j, while imposing $C \equiv 2\sigma$ ($C \equiv 3\sigma$) will yield the interval in which the probability rises up to 95.4% (99.7%).

By extending the above reasoning to a pair of parameters of $\mathbf{H}$, we can also provide **credible regions**:

$$\int_R p(H_i, H_j|d)\ \mathrm{d}H_i\mathrm{d}H_j = C \ , \tag{3.20}$$

where $p(H_i, H_j|d)$ is the marginal posterior of H_i and H_j, while R is the region that gives us the probability C of finding the true values of the pair of parameters.

3.5.2 Markov Chain Monte Carlo

The non-trivial mathematical expression of the likelihood function makes it very difficult to obtain an analytical posterior distribution (see Eq. 3.13), especially for multi-parameter cosmological applications. Alternatively to infer the posterior, it is possible to sample its distribution in the parameter space, particularly the most peaked regions. For this purpose, the **Markov Chain Monte Carlo** (MCMC) method represents a fundamental tool.

Using the MCMC method, we aim at constructing a map of the posterior distribution by generating a sequence (or chain) of points (or samples) in the parameter space. There are various algorithms that can be employed to generate the chains. The key concept is that the chain must follow a Markovian trajectory in the parameter space, meaning that the probability of the $(t+1)$-th element in the sequence is only determined by the value of the t-th element.

Let $\mathbf{H}$ be a parameter vector such that $\mathbf{H} \in \Omega \subset \mathbb{R}^n$. Two consecutive generic elements of the chain will be $\mathbf{H}^{(t)}$ and $\mathbf{H}^{(t+1)}$. The **transition probability** $T(\mathbf{H}^{(t+1)}|\mathbf{H}^{(t)})$ describes the probabilistic generation of chain elements, indicating the likelihood of transitioning from $\mathbf{H}^{(t)}$ to $\mathbf{H}^{(t+1)}$ in the parameter space Ω.

Let us now consider a posterior distribution $P(\mathbf{H}|d)$ (Sect. 3.5.1). Sufficient condition to have a Markov chain is that the transition probability satisfies the **detailed balance condition** (Trotta, 2017):

$$P(\mathbf{H}^{(t)}|d)\ T(\mathbf{H}^{(t)}|\mathbf{H}^{(t+1)}) = P(\mathbf{H}^{(t+1)}|d)\ T(\mathbf{H}^{(t+1)}|\mathbf{H}^{(t)}) \ . \tag{3.21}$$

For the sake of clarity, we re-write this equation as follows:

$$\frac{T(\mathbf{H}^{(t)}|\mathbf{H}^{(t+1)})}{T(\mathbf{H}^{(t+1)}|\mathbf{H}^{(t)})} = \frac{P(\mathbf{H}^{(t+1)}|d)}{P(\mathbf{H}^{(t)}|d)} \ . \tag{3.22}$$

We can now appreciate that the ratio of the transition probabilities is inversely proportional to the ratio of the posterior probabilities, for each pair of consecutive points in the chain.

A sequence that satisfies Eq. (3.21) is called **ergodic chain** (see e.g. Metcalf, 2022), and is characterized by the following proprieties:

- irreducibility, i.e. any point of the chain can be reached in a finite number of steps;
- aperiodicity, i.e. there is not periodicity in the chain;
- positive recurrently, i.e. the expectation value for the number of steps between any two states, $\mathbf{H}'$ fand $\mathbf{H}''$, is finite.

Ergodic chains allow us to get the parameter posterior distribution, since they have a unique **stationary distribution** such that:

$$P(\mathbf{H}^{(t+1)}|d) = \int_0^\infty \mathrm{d}t' \; P(\mathbf{H}^{(t')}|d) \; T(\mathbf{H}^{(t+1)}|\mathbf{H}^{(t')}) \; . \tag{3.23}$$

This means that we are able to sample the posterior once we have sampled an ergodic chain in the parameter space Ω.

There are many MCMC methods, but we will only discuss the **Metropolis-Hastings algorithm** (Metropolis et al., 1953), which is the one used in this thesis work and is also widely used for the Bayesian inference in cosmology. This algorithm is based on finding a transition probability that will have any desired stationary distribution, i.e. $P(\mathbf{H}|d)$ in our case.

The transition probability is, for each successive pair of sampled parameter vectors, $\mathbf{H}^{(t+1)}$ and $\mathbf{H}^{(t)}$:

$$T(\mathbf{H}^{(t+1)}|\mathbf{H}^{(t)}) = q(\mathbf{H}^{(t+1)}|\mathbf{H}^{(t)}) \; \alpha(\mathbf{H}^{(t+1)}, \mathbf{H}^{(t)}) \; , \tag{3.24}$$

where $q(\mathbf{H}^{(t+1)}|\mathbf{H}^{(t)})$ is the **proposal distribution**, while α is equal to:

$$\alpha(\mathbf{H}^{(t+1)}, \mathbf{H}^{(t)}) = \min\left\{1, \; \frac{q(\mathbf{H}^{(t)}|\mathbf{H}^{(t+1)})}{q(\mathbf{H}^{(t+1)}|\mathbf{H}^{(t)})} \frac{P(\mathbf{H}^{(t+1)}|d)}{P(\mathbf{H}^{(t)}|d)}\right\} \; . \tag{3.25}$$

Thanks to this assumption, the chain that we sample will be ergodic (it easy to show that Eq. 3.24 satisfies Eq. 3.21) with the stationary distribution $P(\mathbf{H}|d)$.

The following steps can be used to schematize the Metropolis-Hastings algorithm:

(i) starting from a random point $\mathbf{H}^{(0)}$;

(ii) proposing a candidate point $\mathbf{H}^{(c)}$ by drawing from the proposal distribution q;

(iii) evaluating α and generating a random number u from the uniform distribution $[0, 1[$. Accepting candidate sample $\mathbf{H}^{(c)}$ if $\alpha > u$, rejecting otherwise;

(iv) if the candidate point has been accepted, adding to the chain. Otherwise $\mathbf{H}^{(0)}$ will be counted twice in the sequence. Go back to point (ii).

The selection of the proposal distribution q is essential for the successful investigation of the posterior. If the scale of q is too small when compared to the scale of the target distribution, then exploration will be inadequate as the algorithm will take too much time at a local level. On the other hand, if the q scale is too large, the chain will get stuck as it will not move frequently enough.

The Metropolis-Hastings algorithm is implemented in the `CosmoBolognaLib` (see Sect. 4.1) and will be used in Sect. 5.2 to sample the posterior distribution of the cosmological parameters investigated in this Thesis work. We refer the interested reader to Trotta (2017) for further details about the MCMC technique and methods.

Chapter 4

A new two-point correlation function emulator

In this chapter we present the main result of this Thesis work, i.e. a new code for emulating functions in Cosmology. The main goal is to provide a public and user-friendly tool, useful to speed up the analysis of cosmological data sets. To this purpose, we begin by introducing the `CosmoBolognaLib`, i.e. the numerical environment in which our code is inserted. Then, we discuss in detail the class `Emulator`, representing the core of our implementation. Finally, we present an example of the usage of this class through a concrete example.

4.1 `CosmoBolognaLib`

The `CosmoBolognaLib`[1] (CBL, Marulli, Veropalumbo & Moresco, 2016) is a comprehensive collection of *free software* numerical libraries, tailored for cosmological calculations. These libraries are based on object-oriented programming and implemented in C++. Moreover, their usage is possible in Python language too, thanks to wrapping procedure applicable. The CBL is a continuously evolving project, which has been utilized by many astrophysicists and cosmologists worldwide. This set of libraries is ideal for manipulating extra-galactic source catalogues, both real and simulated, and extracting cosmological parameters from statistical analyses.

[1]We considered V6.1, which is available at `https://gitlab.com/federicomarulli/CosmoBolognaLib`

For example, they offer a valuable tool to measure the 2PCF (see Eq. 2.19) and predict its trend for different cosmological parameters, as well as for different theoretical models. The 2PCF models are public members of the class `Cosmology`. This class allows the user to easily define a cosmological model by specifying the values of all the standard cosmological parameters. The latter can then be used to calculate the theoretical 2PCF model, selecting a number of input parameters (e.g. the redshift and the spatial scale) to obtain the desired output. Furthermore, various Bayesian inference techniques have been implemented in the CBL, such as MCMC posterior sampling methods, like the Metropolis-Hastings algorithm discussed in Sect. 3.5.2.

The primary goal of this Thesis is to expand these already existing set of libraries with a code for emulating functions of the class `Cosmology`. This is done through the new class `Emulator`, which offers the possibility to reproduce a given function in a fast but still accurate manner. We focus in particular on emulating the spherically averaged 2PCF of DM, $\xi(r)$. The 2PCF represents in fact one of the primary statistics in Cosmology, and it is fundamental to improve the numerical tools for its analysis. We underline, however, that the code presented in this Thesis work is applicable to any model function, paving the way to a number of different cosmological applications.

Using the CBL, we calculate the 2PCF by Fourier transforming the DM power spectrum, $P(k)$. The latter can be computed by exploiting a Boltzmann solver, a code that derives a number of observables by solving the linearized Einstein-Boltzmann equations on an expanding background (Dodelson, 2003). Two examples of widespread Boltzmann solvers are the Code for Anisotropies in the Microwave Background[2] (CAMB, Lewis, Challinor & Lasenby, 2000) and the Cosmic Linear Anisotropy Solving System[3] (CLASS, Lesgourgues, 2011; Blas, Lesgourgues & Tram, 2011). Both the codes are incorporated in the environment of the CBL, but are external and independent to these libraries.

In this work, we will focus on the usage of CAMB to compute the power spectrum, but our analysis can be easily extended to other Boltzmann solvers. It is important to highlight that, albeit very accurate, these codes require a considerable amount of time to run. Even if a single run may take a time frame of the order of the second, this becomes

[2]`https://camb.info/`

[3]`https://lesgourg.github.io/class_public/class.html`

very important when used in the context of a MCMC, where power spectrum – or, more in general, the cosmological function derived from it – is typically computed millions of times. Hence the need for a tool to reproduce the same output provided by the CBL functions in a dramatically reduced amount of time, i.e. an emulator. In fact, an emulator would allow any type of function to be calculated for a certain range of validity of the input parameters, avoiding the direct solving of complex and time-consuming calculations, such as derivatives and integrals.

4.2 The `Emulator` class

We present here the class `Emulator`, whose full name in the CBL environment is `cbl::emulator::Emulator`[4]. This new implementation allows the user to create an object of class `Emulator`, with which it is possible to reproduce a given function (the 2PCF in this case). An emulator is a DNN model built up by the setting of weights, $\mathbb{W}$, bias values, $\mathbb{B}$, and the matrices $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, whose entries are the parameters of the activation function (as described in Sect. 3.2.1). All these parameters are embedded in the set of parameters $\mathbb{P}$, as described in Sect. 3.3.

A C++ object of class `Emulator` can be created via two main constructors: the first requires to insert manually each element of $\mathbb{P}$, while the second automatically loads them from a specified directory containing all the hyperparameter details. In the following, we will provide a more detailed description of $\mathbb{P}$.

Let us assume that we set the DNN architecture (see Sect. 3.2) with N hidden layers, each containing M neurons. Therefore, the set of weights is given by:

$$\mathbb{W} = \{\mathbf{w}_0, ..., \mathbf{w}_N\} \ , \tag{4.1}$$

where each $\mathbf{w}_i$ is a matrix whose entries are the coefficients that connect the neurons of a single layer with the neurons of the next one. In particular, the $\mathbf{w}_0$ is a $n \times M$ matrix, which links n parameters[5] of the input layer to the M neurons of the first hidden layer. Then, $\mathbf{w}_i$ with $i \in [1, N-1]$ are $M \times M$ matrices, which connect the M neurons of the

[4]`cbl` indicates the global namespace of the CBL.

[5]We will describe in detail in Sect. 4.3.1 the parameters we are interested in, comprising cosmological parameters and redshift.

i-th hidden layers to the $i+1$-th one. Finally, $\mathbf{w}_N$ is a $M \times k$ matrix, which links the M neurons of the last hidden layer to the spatial scale range where the 2PCF is evaluated (or, more in general, the x input of whatever $f(x)$ function). We assume the input quantity of the function we want to emulate to be a vector of S elements.

The set of bias parameters is given by:

$$\mathbb{B} = \{\mathbf{b}_0, ..., \mathbf{b}_N\} \ , \tag{4.2}$$

where $\mathbf{b}_i$ are vectors. Those vectors have M elements for $i \in [0, N-1]$, which are the b_j values belonging to the hidden layers (see Eq. 3.6), while the N-th vector, $\mathbf{b}_N$, has S elements and belongs to the output layer.

Lastly, the coefficients of the activation functions are given by the matrices $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$:

$$\begin{aligned} \boldsymbol{\alpha} &= \{\alpha_{ij}\}_{i\in[1,N],\ j\in[1,M]} \\ \boldsymbol{\beta} &= \{\beta_{ij}\}_{i\in[1,N],\ j\in[1,M]} \end{aligned} \ , \tag{4.3}$$

where α_{ij} and β_{ij} are the coefficients of the activation function (see again Eq. 3.6). $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are $N \times M$ matrices, where N is the number of the hidden layers and M the number of the neurons belonging to each layer.

Once the model hyperparameters are loaded, it is possible to compute the 2PCF model using the implemented function `cbl::emulator::Emulator::model`, by setting the following function parameters:

- `cosmologicalParameters`, a vector containing the input cosmological parameters, to fix the desired cosmology;
- `outputModes`, a vector containing the values of the spatial scale at which we want to compute the 2PCF;
- `redshift`, the redshift at which the 2PCF is evaluated.

This function computes the forward propagation (see Sect. 3.1.2) of the emulated model thanks to two private functions implemented in the `Emulator` class: `m_layerOperation` and `m_activactionFunc` (where `cbl::emulator::Emulator::` is implicit). The first one defines the computational flow between the model layers, while the second the operations done by each neuron (see Eq. 3.6), returning:

$$\boldsymbol{\xi}_{\rm emu}(\boldsymbol{\Omega}, \mathbf{r}, z) = \{\xi_{\rm emu}(\boldsymbol{\Omega}, r_1, z), ..., \xi_{\rm emu}(\boldsymbol{\Omega}, r_S, z)\} \ , \tag{4.4}$$

where the subscript "emu" refers to the emulated model, while $\boldsymbol{\Omega}$, $\mathbf{r}$, and z are, respectively, the inputs `cosmologicalParameters`, `outputModes` and `redshift`.

The structure of the function `cbl::emulator::Emulator::model` is independent of the choice of the learning algorithm chosen to calculate the DNN model. Indeed, the `m_layerOperation` function is common to every DNN model, and if we want to use a pre-trained model, which activation function is different from Eq. (3.6), we only have to add a new different `m_activactionFunc`, needed to compute the new different activation function operations. This flexibility allows our class to work with trained models computed with any learning algorithm, not just `CosmoPower`.

In the following section we present the training and validation operations performed to prepare the DNN implemented in the CBL to emulate the 2PCF model. The outcome of this procedure is the set of hyperparameters $\mathbb{P}$ needed to emulate the target model. We emphasize that the whole process is extensible to any cosmological function, so the results presented have to be interpreted as just one of the many applications that the methodology offers.

4.3 Training and validation

Our goal is to construct a map between the space of the input parameters and the space of the corresponding 2PCF model outputs. In particular, we aim at using the ML to provide a partial representation of this map. In fact, we prefer to limit the input parameter space to its most relevant sub-set, as increasing the number of parameters mapped implies a decrease in the emulator accuracy. For example, we train our DNN to emulate the 2PCF model in those cosmological parameter ranges that are favoured by the recent survey analyses or that we are most interested in.

Moreover, we focus on two 2PCF models: linear and nonlinear. The latter includes those nonlinear effects affecting mainly the small spatial scales. Both are derived with the

CBL function `xi_matter` and by exploiting the Bolztmann solver CAMB (see Sect. 4.1). Hereafter, we will refer to them with `xi_matter`(CAMB linear) and `xi_matter`(CAMB nonlinear), respectively. Otherwise we will use `xi_matter`(CAMB) if no distinction is needed.

4.3.1 Sampling the training and testing data sets

In Sect. 3.1.1, we explained that a training data set is needed to compute a DNN model (see Sect. 3.2). It consists in our case of a sample of parameter values and the corresponding 2PCFs computed for each set of these parameters. In particular, we choose to focus on the cosmological parameters and the redshift. These are the features of our training set, while the computed 2PCF models represent the labels of the ML algorithm.

We aim at emulating `xi_matter`(CAMB linear) and `xi_matter`(CAMB nonlinear) for different values of the redshift z and three cosmological parameters: the matter density parameter, $\Omega_{\rm m}$, the DE density parameter, $\Omega_{\rm de}$, and the scalar amplitude, $A_{\rm s}$. We leave the other cosmological parameters fixed, but we underline that the procedure can be easily extended to them, although with the challenge of keeping the accuracy of the emulator sufficiently high. For completeness, we report in the following the list of cosmological parameters that we leave unchanged in our analysis:

- the density parameters such as baryon density, $\Omega_{\rm b}$, neutrino density, Ω_ν, radiation density, $\Omega_{\rm r}$, cold dark matter density, $\Omega_{\rm cdm}$, and the density of curvature energy $\Omega_{\rm k}$;
- the values of $N_{\rm eff}$ and N_ν concerning, respectively, the effective number and the number of degenerate massive neutrino species belonging to the cosmic neutrino background;
- the dimensionless Hubble parameter h, i.e. $H_0/100$;
- the spectral index, $n_{\rm s}$, and pivot scale (in unit of Mpc^{-1}), k_0, of the primordial matter power spectrum;
- the coefficients w_0 and $w_{\rm a}$ of the Chevallier-Polarski-Linder parametrization of the DE equation of state (Chevallier & Polarski, 2001).

Parameter	prior range / value
$\Omega_{\rm m}$	$[0.2423, 0.3883]$
$\Omega_{\rm b}$	0.0486
Ω_{ν}	0.00141975
$\Omega_{\rm r}$	$9.26535 \cdot 10^{-5}$
$\Omega_{\rm cdm}$	0.26528
$\Omega_{\rm de}$	[0.572, 0.812]
$\Omega_{\rm k}$	0
$N_{\rm eff}$	2.04
N_{ν}	1
h	0.6736
$A_{\rm s}$	[1.8e-9, 2.4e-9]
k_0	0.05
$n_{\rm s}$	0.9649
w_0	1
$w_{\rm a}$	0
τ	0.0544
z	[0, 2]

Table 4.1: Parameters used to generate the features and label pairs of the training and testing sets. These parameter values are used for both `xi_matter`(CAMB linear) and `xi_matter`(CAMB nonlinear).

We report in Table 4.1 the parameters considered in the training of the DNN: $\Omega_{\rm m}$, $\Omega_{\rm de}$ and $A_{\rm s}$ are allowed to vary in a range spanning ± 10 times the standard deviation provided by Planck Collaboration et al. (2020), while the redshift value covers the range $0 \leq z \leq 2$, suitable for treating the data of ongoing and next generation wide-field surveys. We set the other parameters to their best-fit values provided by Planck Collaboration et al. (2020).

Given the limited nature of our computational resources, it is necessary to cover the parameter space properly by working with a finite number of parameter values. We employ the **Latin hypercube sampling** (LHS) method to sample the parameter space: it consists in generating quasi-random samples extracted from a multidimensional distribution of values, which in our case is uniform. This is achieved by dividing the parameter space, composed of the cosmological parameters and redshift values, into a hypercube grid, such that each grid cell contains only one element of the parameter space.

We choose to allocate 2×10^5 points for the training set, $D_{\rm train}$, and 2×10^4 points for the validation set, $D_{\rm valid}$, getting the respective feature sets for every element. Each element has the following structure:

$$\mathbf{x}_i = \{\Omega_{\mathrm{m}_i}, \Omega_{\mathrm{de}_i}, A_{\mathrm{s}_i}, z_i\} \ , \tag{4.5}$$

where $i \in [1, 2 \times 10^5]$ if $\mathbf{x}_i$ belongs to D_{train}, while $i \in [1, 2 \times 10^4]$ to D_{valid}. The difference in the length of the two data sets is due to a twofold reason: first, to accelerate the computation of the testing set, and second, to cover points of the parameter space that have not been seen by the emulator during the training phase, ensuring the effectiveness of the validation test (see Sect. 3.4).

To get the labels of the training and testing set we compute the function `xi_matter`(CAMB) setting the fixed values reported in Table 4.1 and using the features seen in Eq. (4.5). Moreover, we set the range scale, r, to [0.1, 200] h^{-1} Mpc with a custom binning: 75 bins in [0.1, 90] h^{-1} Mpc, 150 bins [90, 130] h^{-1} Mpc and 75 in [130, 200] h^{-1} Mpc, for a total of 300 bins. This difference in the bin spacing allows us to give more importance to those scales that are harder to model during the learning optimization phase (see Sect. 3.1.2), such as **baryonic acoustic oscillation**[6] (BAO) scale. The BAO feature is visible around 105 h^{-1} Mpc and the position of its peak and shape is a powerful tool to constrain the expansion rate of the Universe and to investigate the nature of DE (see e.g. Crocce & Scoccimarro, 2008; Percival et al., 2010; Beutler et al., 2011). Therefore, it is fundamental to sample densely this region in order to emulate accurately the 2PCF.

Following the work of Spurio Mancini et al. (2022), to better emulate the matter power spectrum we apply a change in the label representation. By utilizing `CosmoPower`, it is possible for example to convert the label values' representation to a decimal logarithm ones. However, this is not a viable solution for 2PCF models, due to the presence of non-positive values. Therefore, we choose to use a custom representation by multiplying the values of 2PCF by the square of the spatial scale r, resulting in the following labels:

$$\mathbf{y}_i = \{r_0^2 \cdot \xi_{0,i}, ..., r_{299}^2 \cdot \xi_{299,i}\} \ . \tag{4.6}$$

[6]It refers to the fluctuations visible in the distribution of the baryonic matter on large scales, generated by the acoustic waves propagating in the primordial plasma before the recombination epoch (see Sect. 1.5).

Parameter	**Value for each learning step**
`learning_rates`	$[10^{-1}, 10^{-2}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}]$
`batch_sizes`	$[10^2, 2\cdot10^2, 10^3, 2\cdot10^3, 10^4, 2\cdot10^4]$
`patience_values`	$[20, 20, 20, 20, 20, 20]$
`max_epochs`	$[2000, 2000, 2000, 2000, 2000, 2000]$

Table 4.2: Table showing the values chosen to optimize the learning phase during the building of the emulator (see Sect. 3.3), for both `xi_matter`(CAMB linear) and `xi_matter`(CAMB nonlinear).

This approach is indeed commonly used in cosmological analyses to highlight the BAO feature of the 2PCF. As a final result, we obtain the four desired data sets: two training data sets, i.e. $\mathrm{D}_{\mathrm{train}}^{\mathrm{CAMB\ linear}}$ and $\mathrm{D}_{\mathrm{train}}^{\mathrm{CAMB\ nonlinear}}$, with 2×10^5 pairs $(\mathbf{x}_i, \mathbf{y}_i)$ of tuples, and two validation data sets, i.e. $\mathrm{D}_{\mathrm{valid}}^{\mathrm{CAMB\ linear}}$ and $\mathrm{D}_{\mathrm{valid}}^{\mathrm{CAMB\ nonlinear}}$, with 2×10^4 pairs $(\mathbf{x}_i, \mathbf{y}_i)$ of tuples.

4.3.2 Model training

We now set up the `CosmoPower`-implemented learning algorithm in order to obtain the emulated models of `xi_matter`(CAMB linear) and `xi_matter`(CAMB nonlinear). This will provide the sets of hyperparameters, $\mathbb{P}$, seen in Sect. 4.2. We define the model architecture by configuring **4** hidden layers, each comprising **20** neurons, with the activation function outlined in Sect. 3.2.1.

We configure 6 learning steps (see Sect. 3.3.1) and we fix the learning hyperparameters (`learning_rates`, `batch_sizes`, `patience_values` and `max_epochs`) as shown in Table 4.2. We set these parameters on the basis of the work of Spurio Mancini et al. (2022), adjusting then their values according to the output our model. This fine-tuning procedure is aimed at reaching, as quickly as possible, the lowest value for the RSME, i.e. the loss function (see Eq. 3.8). Let us note how during each learning step the values of the `learning_rates` decrease, while those of the `batch_sizes` increase. This optimizes and speeds up the search for the minimum value of the loss function.

We compute the DNN models using the parallel computing cluster of the Open Physics Hub[7] at the Physics and Astronomy Department in Bologna. In particular, utilizing 56 CPUs with a clock frequency of 2.2 GHz, the algorithm took approximately 1 hour to

[7] https://site.unibo.it/openphysicshub/en

process each model. However, we underline that the execution time is closely related to the architecture of computed models, to the number of labelled examples of the training test and to the computational resources employed. For example, the usage of a GPU node would dramatically reduce the computation time required for the algorithm training.

Once the best configuration of the hyperparameters is achieved and the loss value is satisfactory, it is possible to proceed with the testing phase, which we will present in Sect. 3.4. Otherwise, further investigations must be conducted to improve the outcome, such as changing the DNN architecture parameters or label representation, or both. After implementing the necessary modifications, the training phase must be repeated to verify if the desired result has been achieved.

4.3.3 Model validation

Now that the learning phase is completed, we move on to the testing phase to evaluate the accuracy of the generated models. In principle, this phase could be skipped, simply loading the models into the `Emulator` class. In that case, the model validation would be performed applying the emulator directly to a Bayesian analysis (as we will see in Chapter 5), comparing the posterior distributions obtained with the original `xi_matter`(CAMB) and its emulated version. However, if the model turns out to have low accuracy, it is more difficult in that case to understand how to improve the emulator.

To test the accuracy of the emulated models, we firstly compute the RMSE over the testing data set, D_{test}, and compare them with those of the training data set, D_{train}. This is useful to check if a balanced fit is achieved (Sect. 3.4). In Table 4.3 we show the RMSE values for both the CAMB linear and the CAMB nonlinear emulated models. These values indicate that a balanced fit has been achieved, as the gap between the RMSE for D_{test} and D_{train} is of the order of 10^{-5}.

To evaluate the accuracy of the models, we compute the **percent error** as follows:

$$\Delta_{\%}(r_j) = \left| \frac{\xi_{\text{emu}}(r_j) - \xi_{\text{true}}(r_j)}{\xi_{\text{true}}(r_j)} \right| \cdot 100 \ , \tag{4.7}$$

where r_j is the range scale value, in unit of h^{-1} Mpc, such that $j \in [0, 299]$ (see Eq. 4.6) and the expression is evaluated over the whole D_{test}. We aim at reaching a percent error

CAMB linear	**RMSE**
D_{train}	$5.05 \cdot 10^{-3}$
D_{test}	$5.12 \cdot 10^{-3}$

CAMB nonlinear	**RMSE**
D_{train}	$5.56 \cdot 10^{-3}$
D_{test}	$5.64 \cdot 10^{-3}$

Table 4.3: RMSE values computed over the whole training and testing data set, for the emulated models of CAMB linear and CAMB nonlinear. In both case the gap between the RMSE is small enough to ensure a balanced fit for the sets of hyperpamarameters of the DNN models.

lower than 4% for each function over the whole D_{test}, as achieved in the work of Spurio Mancini et al. (2022).

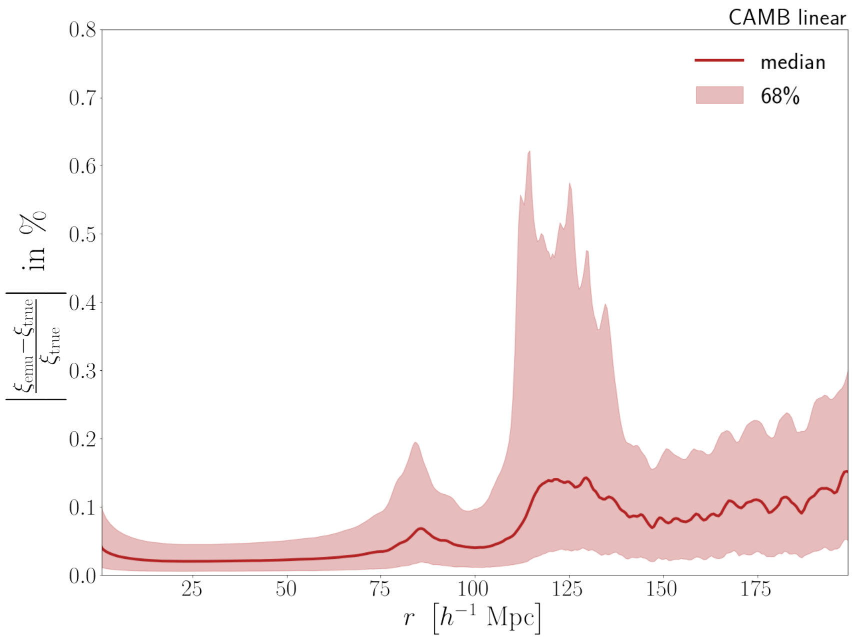

Figure 4.1: Emulator percent error, computed evaluating `xi_matter`(CAMB linear) on the $2 \cdot 10^4$ combinations of parameters of the sample $\text{D}_{\text{test}}^{\text{CAMB linear}}$, on the scale range $r \in [0.1, 200]\ h^{-1}$ Mpc. We represent the median percentage error with a red line and the 68% confidence region around this value with a shaded area.

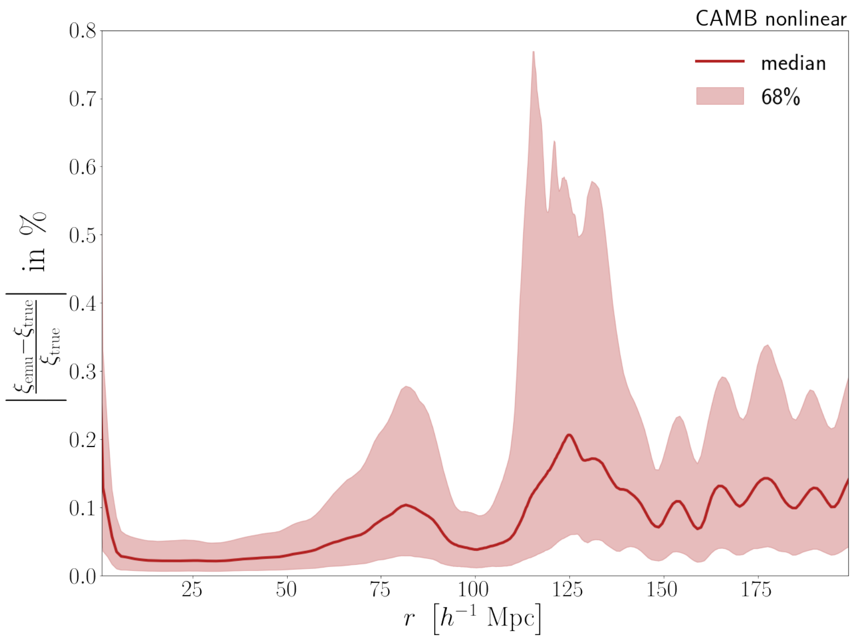

Figure 4.2: As Fig. 4.1, but for the model `xi_matter`(CAMB nonlinear).

We show in Fig. 4.1 and Fig. 4.2 the percentage error computed for the `xi_matter`(CAMB linear) and `xi_matter`(CAMB nonlinear) DNN models, respectively. We represent the median of the distribution of percentage errors, obtained by testing all the combination of values extracted with the LHS technique. With a shaded band we represent the 68% of probability around the median, so the interval between the 16-th and 84-th percentile. The median percent error of the emulator is less than $\approx 0.2\%$ in the scale range $r \in [0.1, 200]$ h^{-1} Mpc, for both models. When considering the 68% probability region, the percentage error reach the values $\approx 0.6\%$ and $\approx 0.7\%$, for the cases CAMB linear and nonlinear, respectively. However, this happens only at the spatial scales in the range $r \in [100, 150]$ h^{-1} Mpc, i.e. the region characterized by the BAO peak. Given the complexity of this 2PCF feature, the accuracy of the emulator is indeed expected to be lower in this zone, despite the denser scale sampling we applied area during training.

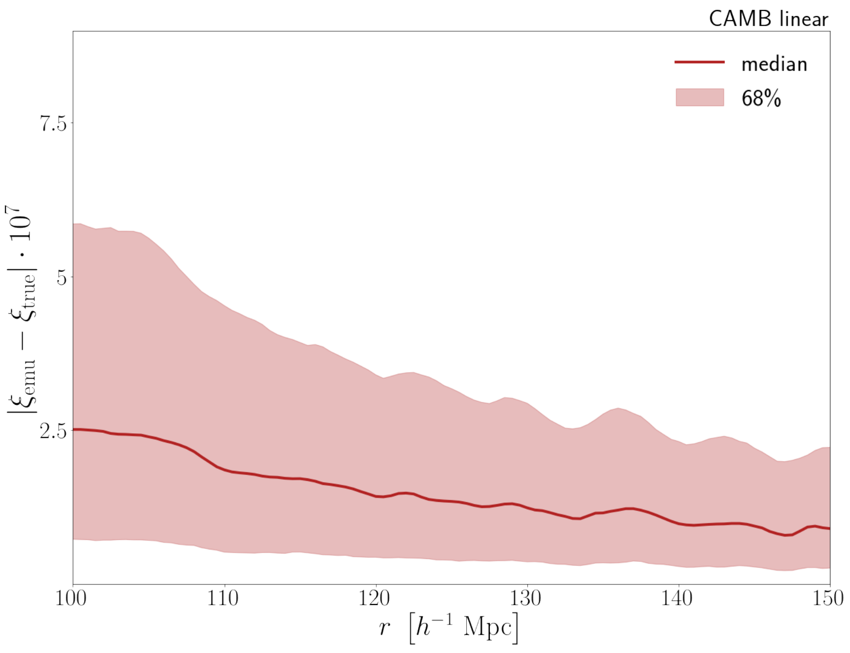

Figure 4.3: Emulator absolute error computed for `xi_matter`(CAMB linear) and multiplied by 10^7. The description of the plot is analogous to the one of Fig. 4.1.

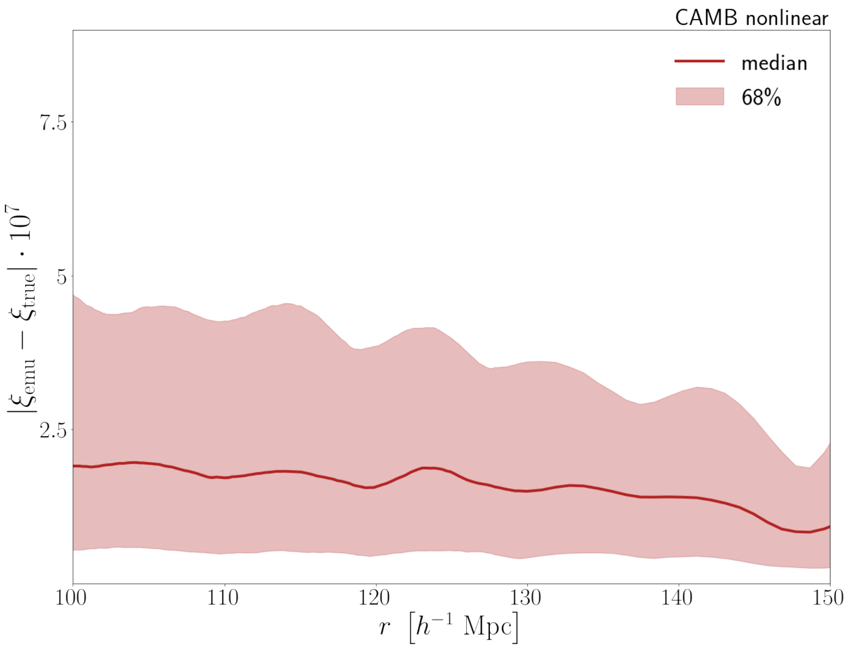

Figure 4.4: As Fig. 4.3, but for the model `xi_matter`(CAMB nonlinear).

To better understand if the discrepancy we detected on the scales $r \in [100, 150]\ h^{-1}$ Mpc is relevant for statistical analyses, we compute the **absolute error** between the two models:

$$\Delta(r_j) = |\xi_{\rm emu}(r_j) - \xi_{\rm true}(r_j)| \ . \tag{4.8}$$

We report the absolute values calculated for CAMB liner and CAMB nonlinear in Fig. 4.3 and Fig. 4.4, respectively, along the restricted range $r \in [100, 150]\ h^{-1}$ Mpc. We note in this case that median of distribution of absolute errors is of the order of magnitude of 10^{-7}. We can conclude that the loss of accuracy we found is not relevant to standard cosmological analyses, where the uncertainty on the 2PCF measures is usually larger than the discrepancy between the emulated model and the original function. Being these results very encouraging, we move now to practical applications of the trained DNN models by using the class `Emulator` for example cosmological analyses.

4.4 Examples of usage

Let us now present a practical demonstration of the usage of the class `Emulator`. The following Python script is named `xi_CAMB_emulator.py` and is provided among the examples stored in the CBL. It shows how to calculate the 2PCF model with both a standard CBL function and by using the implementations of the class `Emulator`.

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import gridspec
import CosmoBolognaLib as cbl

# Create a Cosmology object
cosm = cbl.Cosmology(cbl.CosmologicalModel__Planck18_)

# Define the spatial scales and the redshift
rr = np.linspace(30., 120., 100)
zz = 1.

# compute the 2pt correlation function with CAMB
xi_r2_matter = [cosm.xi_matter(RR, "CAMB", False, zz)*RR*RR for RR in rr]

```

```python
# Create an Emulator object
emu = cbl.Emulator("xi_r2_CAMB")

# compute the 2pt correlation function with CAMB
xi_r2_emu = emu.model([cosm.OmegaM(), cosm.OmegaDE(), cosm.scalar_amp()],
    rr, zz)

# compute the percent error
perc_error = np.zeros(100)

for i in range(len(perc_error)):
    perc_error[i] = (xi_r2_emu[i]-xi_r2_matter[i])/xi_r2_matter[i]*100

# Plot results
fig, ax = plt.subplots(2,1, figsize=(20, 16), sharex=True, gridspec_kw={'
    height_ratios':[2,1], 'hspace':0, 'wspace':0.15})

# main plot
ax[0].plot(rr, xi_r2_emu, label="emulated \texttt{xi\_matter}", ls="-", lw
    =4)
ax[0].plot(rr, xi_r2_matter, label="\texttt{xi\_matter}", ls="--", lw=4)
ax[0].set_ylabel("$\xi(r) \ r^2$", fontsize=40)
ax[0].legend(fontsize=35)

# sub-plot
ax[1].axhline(y=0, color='grey', ls=':', lw=2.5)
ax[1].plot(rr, perc_error, lw=4)
ax[1].set_ylim([-0.75,0.75])
ax[1].set_xlabel("$r \ [h^{-1} \ \mathrm{Mpc}]$", fontsize=40)
ax[1].set_ylabel("$\Delta_{\rm \%}(r)$", fontsize=40)

plt.show()
```

We start with the definition of an object of the class `Cosmology`, built with the cosmological parameters provided by Planck Collaboration et al. (2020). Then we compute the linear 2PCF model at $z = 1$ on the spatial range $[30, 120]$ h^{-1} Mpc via the function `xi_matter`. In this case we exploit the output of CAMB to predict the 2PCF model, $\xi(r)$, multiplying it by a factor r^2 to highlight the region of the BAO.

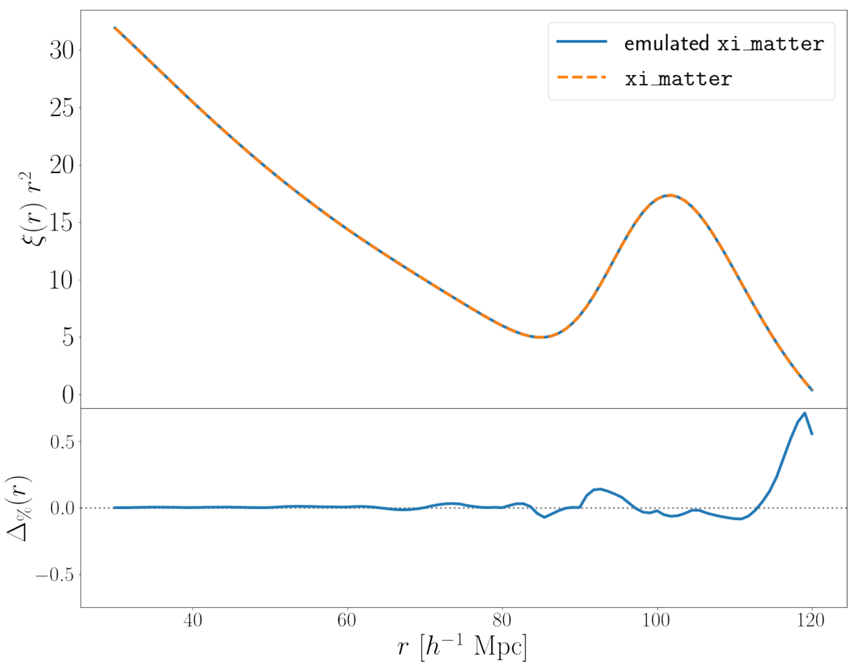

Figure 4.5: Output plot of the `Emulator` class Python example. We compare the linear 2PCF model, $\xi(r)$, multiplied by r^2, computed with two approaches: through the standard function `xi_matter` of the class `Cosmology` (orange dashed line) and with the function of the `Emulator` class (blue solid line). We show in the sub-plot the discrepancy between the two models, computed as the percentage error with respect to the original `xi_matter` function.

To compare this output with the emulated model we build and object of the class `Emulator` by loading all the DNN hyperparameters from the directory "xi_r2_CAMB" we prepared after the training procedure. Then we call the function `model` by giving as input the list of cosmological parameters of our interest ($\Omega_{\rm m}, \Omega_{\rm de}, A_{\rm s}$, in this case), the list of the spatial scales and the redshift at which computing the model. When running the code, a summary table of the DNN model is printed, containing all the details of emulated starting model. In this case we trained the DNN to emulate the function $\xi(r)\ r^2$ derived from the CBL function `xi_matter` and the range of validity parameters of the emulator is the one reported in Table 4.1.

Finally, a plot is produced to compare the output of the two functions, i.e. `xi_matter` and its emulated version. We report it in Fig. 4.5, where it is easy to appreciate the perfect matching between the two curves. The residuals show indeed that the discrepancy between the emulated model and the original one is lower than 0.7%.

To test the speed of the emulated function, another CBL exampled is provided, named `xi_CAMB_emulator.cpp`. Being conceptually similar to the example just presented, we provide here just a rapid overview of the code.

It is written in C++ language and computes the execution time needed to calculate 1000 emulated linear $\xi(r)$, for different cosmological parameters and on 150 points of the spatial scale r. The execution time obviously depends on the machine on which the code is run, but considering a standard portable computer it takes about 5 ms. The same function, computed only one time with the standard `xi_matter` function would have taken 2 s. The improvement is roughly of a factor 400. Although the difference between the two computation times could appear irrelevant, in Chapter 5 we will demonstrate the importance of this improvement, by applying the `Emulator` class in the context of MCMC analyses.

Chapter 5

Application to cosmological simulations

In this chapter, we present an application of the `Emulator` class to a more complex cosmological analysis. We model the 2PCF measured in different snapshots of a cosmological simulation, applying an MCMC analysis (see Sect. 3.5.2) to sample the posterior distribution of the cosmological parameters $\Omega_{\rm m}$, $\Omega_{\rm de}$, and $A_{\rm s}$. The main goals of this test are: (i) to assess the precision of the emulator we developed, and (ii) to quantify the improvement in the computational efficiency of the code, compared to the implementation based on the original 2PCF model function (see Sect. 4.3).

5.1 2PCF measure

We consider a sub-set of the N-body simulation suite named DUSTGRAIN-*pathfinder* (Dark Universe Simulations to Test GRAvity In the presence of Neutrinos, Giocoli, Baldi & Moscardini, 2018; Hagstotz et al., 2019). Although these simulations are designed to test different cosmological models, we focus on the standard flat ΛCDM catalogues, at redshifts $z = 0, 1$ and 2. This interval covers indeed the redshift range explored during the training of our emulator.

The DUSTGRAIN-*pathfinder* are built with the cosmological simulation code `GADGET2` (Springel, 2005), which guides the evolution of 768^3 DM particles within a periodic cosmological box of 750 h^{-1} Mpc per side. The ΛCDM simulation is characterized by DM parti-

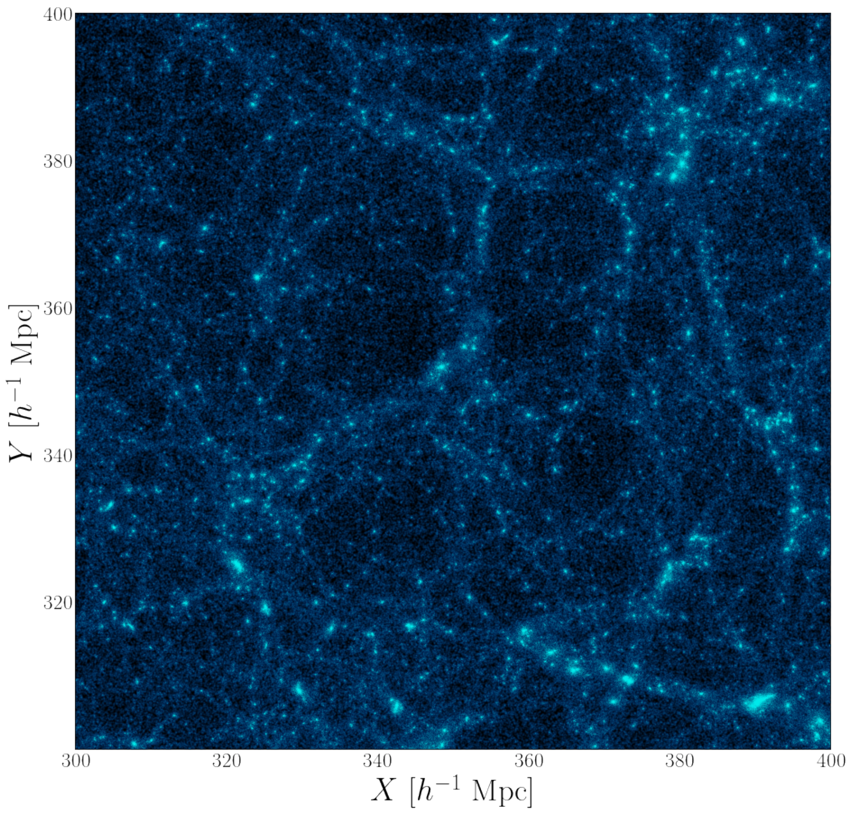

Figure 5.1: Distribution of DM particles in the central parts of the DUSTGRAIN-*pathfinder* ΛCDM snapshot at $z = 0$. We consider a sub-box with edges $[300, 400]$ h^{-1} Mpc and we represent the X and Y positions of each DM particle with a blue dot.

cles of mass 8.1×10^{10} h^{-1} $M_\odot$. We show a zoom-in in the central region $(100h^{-1}$ Mpc$)^3$ of the $z = 0$ snapshot in Fig. 5.1, where we represent the spatial distribution of the DM particles projected along the Z-axis.

The cosmological parameters of the simulation are in agreement with the Planck Collaboration et al. (2016) results: $\Omega_\mathrm{m} = 0.31345$, $\Omega_\mathrm{de} = 0.68655$, $h = 0.6731$, $A_\mathrm{s} = 2.199 \times 10^{-9}$ and $n_\mathrm{s} = 0.9658$. We underline that h and n_s are slightly different to those reported by Planck Collaboration et al. (2020), on which we trained our emulator. In particular, they differ from those of Table 4.1 by 0.07% and 0.09%, respectively. Nevertheless, we will see that these differences do not have a statistically significant impact on our results.

To reduce computational time during the measure of the 2PCF, we sub-sample the original particle catalogue down to the 1%. This procedure lowers the spatial resolution of the sample, but does not affect the 2PCF measure on the spatial scales of our interest. Indeed, we measure the 2PCF on a range scale $r \in [15, 120]$ h^{-1} Mpc, calculating it on 30

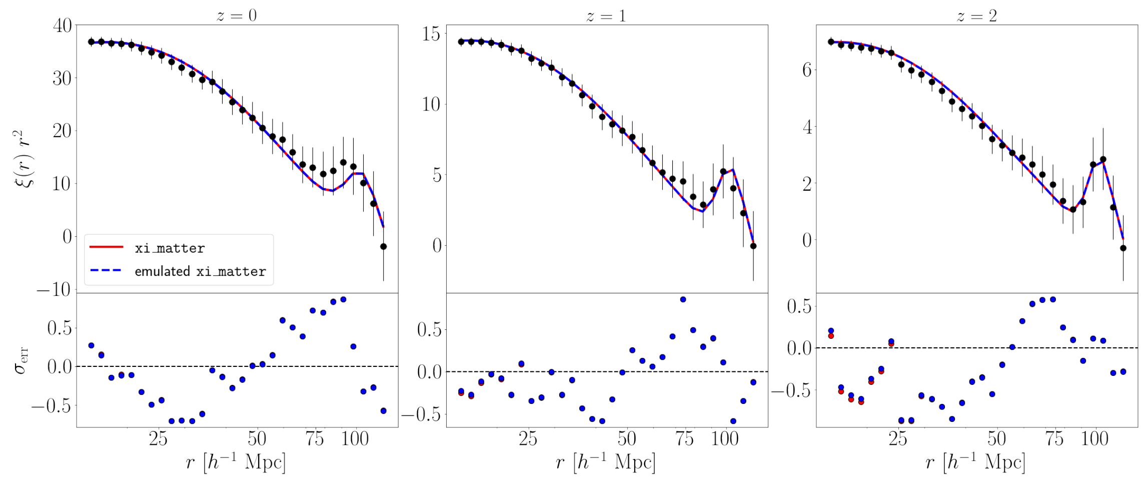

Figure 5.2: Models and measures of the 2PCF in the ΛCDM DUSTGRAIN-*pathfinder* simulations at $z = 0$ (*left*), $z = 1$ (*center*) and $z = 2$ (*right*). We represent the measured 2PCF with black dots and error bars. The nonlinear model computed with `xi_matter` is reported with a red solid line, and the emulated model with a blue dashed line. The bottom panels show the residuals, calculated in terms of $\sigma_{\rm err}$, between the measured 2PCF values and those of `xi_matter` and emulated `xi_matter`, in red and blue respectively.

bins. This range is widely covered by our emulator and includes the BAO region. Smaller scales are more difficult to model because of the stronger nonlinear effects, while on larger scales the 2PCF signal becomes very low and noisy.

We measure the 2PCF with the Landy & Szalay estimator (see Eq. 2.32) and we evaluate the statistical errors applying the Bootstrap resampling method (Efron, 1982; Norberg et al., 2009). In particular, we divide the original catalogues in 125 sub-boxes, constructing 300 realisations by resampling from the sub-catalogues, with replacement.

To model the measured 2PCF at $z = 0, 1$ and 2, we can use `xi_matter`(CAMB non-linear) function, since our matter tracers are unbiased and in real space. In practice, no additional corrections to the theoretical model are included to take into account the effects of the tracer bias, geometrical and redshift-space distortions. In future works, we will also extend the emulation to models of the two-point correlation function that include these effects.

We start by assuming the true cosmology of the simulation and compute the nonlinear 2PCF model with the original CBL function `xi_matter` and its emulated version. Once again, we multiply $\xi(r)$ by r^2 to highlight the BAO scale.

We show in Fig. 5.2 the comparison between the 2PCF measures and the two versions of the model, at $z = 0,\ 1,\ 2$. We note that the data are well reproduced by the theory at all the redshifts, indeed the residuals show a discrepancy lower then $1\sigma_{\rm err}$. The latter is computed as the difference between the model and the data, divided by the data error, at each value of r. The small differences between the two models (`xi_matter` and emulated `xi_matter`) are barely visible in the residuals, confirming again the accuracy of the emulator we developed.

5.2 Bayesian Analysis

In this section we apply the Bayesian statistical framework and methods seen in Sect. 3.5 to infer test cosmological constraints from the 2PCF measured in DUSTGRAIN-*pathfinder* snapshots. We compare then the MCMC results obtained by using the original `xi_matter` with those derived by exploiting the new emulated function, both in terms of accuracy and efficiency.

5.2.1 MCMC preparation

Let us indicate the set of cosmological parameters of our interest as $\mathbf{H} = \{\Omega_{\rm m}, \Omega_{\rm de}, A_{\rm s}\}$. Using the notation introduced in Sect. 3.5.1, we can express the posterior distribution that we aim at sampling as follows:

$$P(\mathbf{H}|\xi_{\rm sim}) = \frac{\mathcal{L}(\xi_{\rm sim}|\mathbf{H})P(\mathbf{H})}{P(\xi_{\rm sim})} \,. \tag{5.1}$$

In this equation, $\xi_{\rm sim}$ is the data set composed of the 2PCF measures extracted from the three redshifts of the DUSTGRAIN-*pathfinder* simulation. Then, $\mathcal{L}(\xi_{\rm sim}|\mathbf{H})$ is the likelihood, assumed in this work to be Gaussian. The priors on the cosmological parameters, $P(\mathbf{H})$, are imposed to be uniform (see Eq. 3.14) such that the parameter space is:

$$[\Omega^l_{\rm m}, \Omega^u_{\rm m}] \times [\Omega^l_{\rm de}, \Omega^u_{\rm de}] \times [A^l_{\rm s}, A^u_{\rm s}] \,, \tag{5.2}$$

where the superscripts l and u denote, respectively, the lower and upper values of the range in which we trained our emulator. This means that, in this MCMC analysis, $\Omega_{\rm m}, \Omega_{\rm de}$ and $A_{\rm s}$ are allowed to vary independently inside the interval defined by the prior reported in Table 5.1, while the other cosmological parameters are kept constant to the true values of the simulation (see Sect. 5.1).

Free parameter	**Prior range**
Ω_m	[0.2423, 0.3883]
Ω_de	[0.572, 0.812]
A_s	[1.8e-9, 2.4e-9]

Table 5.1: Free parameters of the model (Ω_m, Ω_de and A_s) and corresponding priors used in the Bayesian cosmological analysis. The prior distributions are flat and cover the entire validity range of the emulator.

We compute the nonlinear 2PCF model, multiplied by r^2, by using both the original `xi_matter` function derived with CAMB and its emulated version. The model is calculated on the same spatial scales of the measured 2PCF and at the redshifts $z = 0,\ 1,\ 2$, consistently with the data set analyzed.

Then, we sample the posterior distribution of the model free parameters with the Metropolis-Hastings method (see Sect. 3.5.2), which is available in the CBL with an optimized implementation. At each step of the MCMC, the computation of the model and the associated posterior probability is entrusted to a number of **walkers**, which run in parallel on different threads. This allows us to reduce consistently the computational time, being different steps of the MCMC computed by different CPUs simultaneously.

We set the MCMC code with 28 walkers, each performing 7000 steps. As the initial steps of the MCMC are usually necessary for the walkers to identify the region in which the posterior probability is higher, we remove the first 1000 steps of the output MCMC. This chunk is commonly denoted **burn-in**, and its removal is necessary to ensure the results to be computed with steps in which the MCMC has already converged. The final MCMC is therefore composed of 168 000 steps. We run the posterior sampling code on the cluster of calculus of the Physics and Astronomy Department in Bologna, already mentioned in Sect. 3.3, using also in this case 56 CPUs with a clock frequency of 2.2 GHz.

5.2.2 MCMC results

We present now the outcome of the MCMC analysis, comparing the results obtained by using the CBL function `xi_matter` and its emulated version. We show in Fig. 5.3 the values of the cosmological parameters we left free to vary and the corresponding MCMC

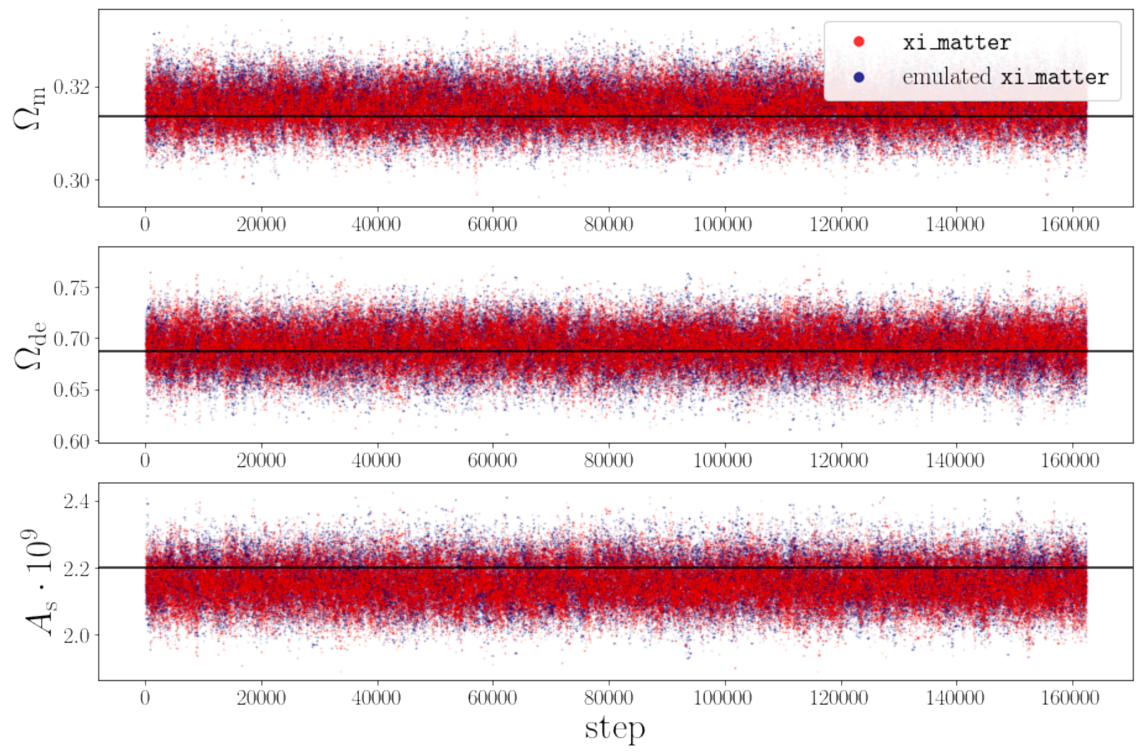

Figure 5.3: Values of the parameters $\Omega_{\rm m}, \Omega_{\rm de}$ and $A_{\rm s}$ for every MCMC step considered in the Bayesian analysis. We represent in red the output for `xi_matter` and in blue the one of its emulated version. We report the true cosmological value of the DUSTGRAIN-*pathfinder* as black horizontal lines.

steps, for the two models considered. We can note that the distribution of points is very similar for the two cases, and in agreement with the true values of the parameters, i.e. $\Omega_{\rm m} = 0.31345$, $\Omega_{\rm de} = 0.68655$ and $A_{\rm s} = 2.199 \times 10^{-9}$. Moreover, the points show a stable trend with the step number, indicating that the MCMC are converged.

We report in Fig. 5.4 the parameter posterior distributions we sampled through the MCMC analysis. We highlight how all the 2D confidence contours we obtained are neat and closed, for both the model functions considered (i.e. `xi_matter` and emulated `xi_matter`). This means that our MCMC algorithm was able to sample accurately the full posterior distribution of the parameters and that the cosmological degeneracies between them are not very strong. However, looking at the orientation of the 2D contours, we can note a small correlation between $\Omega_{\rm m}$–$A_{\rm s}$, and an anti-correlation between $\Omega_{\rm m}$–$\Omega_{\rm de}$ and $\Omega_{\rm de}$–$A_{\rm s}$. The projected 1D distributions represent instead the marginalized posterior for each parameter (see Sect. 3.5.1), which is well represented by a Gaussian.

Thanks to the test constraints we derived, we can conclude that the model we assumed is suitable to reproduce the 2PCF measured in the DUSTGRAIN-*pathfinder* simulations. Indeed, the true value of all the cosmological parameters we considered falls within the 68% of confidence level.

In this Thesis work, however, the important thing to notice is the similarity between the posterior distributions obtained with the two functions used to compute the theoretical 2PCF model. Indeed, the confidence contours derived with our emulator are almost perfectly overlapped to those of the original CBL function `xi_matter`. This further demonstrates the accuracy of the DNN model we trained. The small differences we see are totally within the statistical uncertainly, thus not significant. As an additional validation, we report in Table 5.2 the mean values and the standard deviations of the cosmological parameters we derived with the two model functions.

Now the most important comparison to make: the computational time spent to perform the MCMC analysis in the two cases. The code takes approximately **40 h** for the posterior distribution sampling when using `xi_matter`, while only **16 s** with the emulated version we provided. The improvement shown in the code efficiency is of utmost relevance. The lowering of the MCMC execution time is indeed of a factor 9 000. However, as already mentioned, this time values are dependent on the computational resources employed, like

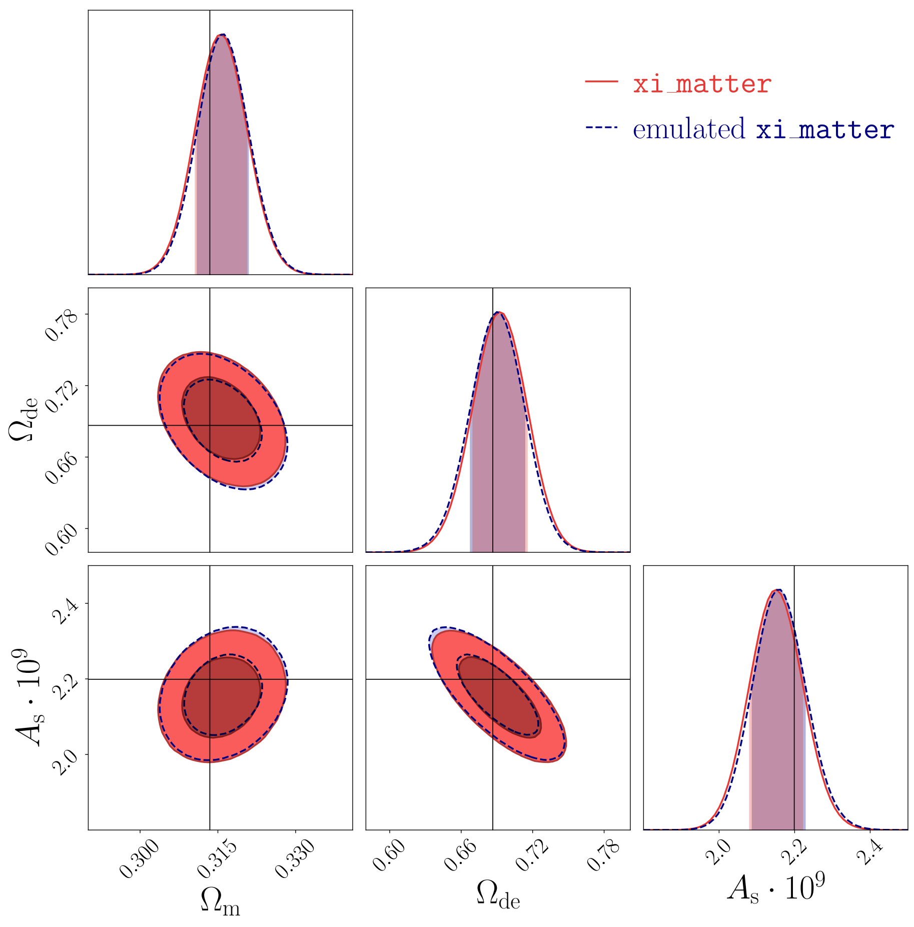

Figure 5.4: 68% and 95% 2D confidence regions obtained by modelling the 2PCF measures with `xi_matter` (blue) and corresponding emulated function (red). The projected 1D distribution of the parameters is shown in the upper panel of each column, with a shaded area representing the 68% of probability around the maximum. The solid black lines indicate the true cosmological values of the DUSTGRAIN-*pathfinder* simulation. These distributions are obtained with an MCMC algorithm, which takes in approximately 40 hours when run with the CBL function `xi_matter`, whereas in only 16 seconds with the corresponding emulated function.

Parameter	`xi_matter`	**emulated** `xi_matter`	**true value**
$\Omega_{\rm m}$	0.3156 ± 0.0046	0.3160 ± 0.0043	0.31345
$\Omega_{\rm de}$	0.692 ± 0.020	0.690 ± 0.020	0.68655
$A_{\rm s}\cdot 10^9$	2.15 ± 0.10	2.16 ± 0.10	2.199

Table 5.2: Mean and standard deviation of the parameters $\Omega_{\rm m}$, $\Omega_{\rm de}$, and $A_{\rm s}$, obtained by using as model function `xi_matter` (*second* column) and its emulated version (*third* column). As a comparison, we report the true values of the DUSTGRAIN-*pathfinder* simulations in the last column.

the number of CPUs and the clock frequency.

For example, running the same MCMC on a laptop having 8 CPUs with a frequency of 2.4 GHz takes about 30 s, when using the emulator. The application to an MCMC analysis of the standard `xi_matter` function (paired with the Boltzmann solver CAMB) is instead almost precluded to portable personal computers. Therefore, the methodology we propose in this Thesis work represents a very powerful tool to perform fast and precise statistical analyses, and will be applied in the near future to a number of cosmological applications.

Chapter 6

Conclusions

In modern Cosmology, Bayesian inference (see Sect. 3.5) is the primary statistical framework for determining the parameters of the cosmological model. This is achieved by sampling the posterior probability distribution of the model parameters. Such sampling procedure often needs of a big computational effort, as it requires to repeat the calculation of cosmological functions millions of times. Even with optimized codes and large computational resources, this process can take several hours.

For this reason, recent years have seen a surge in the use of ML techniques to reduce the execution time of such analyses (see Sect. 3.1). This has been achieved by reproducing the original cosmological functions with DNN models (see Sect. 3.2). These models are known as emulators and are developed to reduce the computation time while maintaining a negligible loss of accuracy, when compared to the original functions. As a result, many resources have been conducted to provide accurate and efficient emulators (e.g. Euclid Collaboration et al., 2021; Bonici et al., 2022; DeRose et al., 2022; Nygaard et al., 2022).

In this Thesis work we built an emulator of the models implemented in the C++/Python set of libraries `CosmoBolognaLib` (CBL, Marulli, Veropalumbo & Moresco, 2016, see Sect. 3.1), with the aim of speeding up the cosmological analyses performed with codes based on these libraries. We added to the CBL a new C++ class to create and use emulated functions. We focused in particular on emulating the CBL function that provides the basic theoretical model for the 2PCF (see Sect. 2.1.3), `xi_matter`. This function is based on the output of a Boltzmann solver, which computes the matter power spectrum as a function of the cosmological model parameters, in both linear and nonlinear theory. Although very precise, `xi_matter` is limited by the computation time required by the Boltzmann solver. In this work, we relied for example on CAMB (Lewis, Challinor & Lasenby, 2000), which takes some seconds to execute in the configuration of our interest.

To emulate the output of the function `xi_matter` obtained with CAMB, we prepared a DNN architecture with 4 hidden layers, composed of 20 neurons each, and with the activation function presented in Eq. (3.5). We trained the DNN model in a supervised learning context, exploiting the methods and algorithms implemented in the numerical libraries `CosmoPower` (Spurio Mancini et al., 2022).

We focused on the emulation of the 2PCF linear and nonlinear models on the scale range $r \in [0.1, 200]$ h^{-1} Mpc, and for the variation of the redshift z and the cosmological parameters $\Omega_{\rm m}$, $\Omega_{\rm de}$ and $A_{\rm s}$. We trained the DNN model on 2×10^5 points, created with the LHS method (see Sect. 4.3.1) to cover the parameter space defined in Table 4.1. We imposed $\Omega_{\rm m}$, $\Omega_{\rm de}$ and $A_{\rm s}$ to span in a range wide ± 10 times the standard deviation of the parameters presented in Planck Collaboration et al. (2020), while the redshift in the range $z \in [0, 2]$.

Then, we validated our DNN model on another set of 2×10^4 points, comparing the RMSE to the one of the training data set. We found a gap of 10^{-5} between the two, which demonstrates the achievement of a balanced fit. Moreover, the median percentage error between the original function and its emulated version, computed in the full range of parameters we analyzed, was demonstrated to be lower than the 0.2% at all spatial scales. We detected a slight worsening in the precision of the emulator only on the BAO region, which is in fact more difficult to model. The discrepancy is however lower than the 0.7% for the 68% of the parameter combinations we tested.

Finally, we applied our emulator to a Bayesian analysis. We measured the 2PCF at three redshift of the DUSTGRAIN-*pathfinder* simulations (Giocoli, Baldi & Moscardini, 2018; Hagstotz et al., 2019) and we applied the MCMC technique (see Sect. 3.5.2) to sample the posterior distribution of the parameters $\Omega_{\rm m}$, $\Omega_{\rm de}$ and $A_{\rm s}$. We assigned uniform prior distributions to these parameters, spanning on the same ranges used in the DNN training phase. We compared then the outcomes achieved by assuming the nonlinear 2PCF model with `xi_matter` and its emulated version. In both the cases, we recovered the true values of the cosmological parameters within the 68% of uncertainty. Moreover, the two sets of the obtained confidence contours were almost perfectly overlapped and the mean parameter values were almost identical, and totally compatible within the statistical errors.

The most important result, however, was the improvement measured in the computational time required to run the MCMC analysis for the two model functions. When using the original CBL function `xi_matter`, the code took approximately 40 h, while with the implemented emulator only 16 s. With an increment of a factor 9 000, our emulator has been confirmed to provide an extraordinary tool to speed up statistical analyses, laying the groundwork for a number of future applications in Cosmology.

6.1 Future perspectives

The `Emulator` class we developed will be made public with the release of the new CBL version. With it, the two examples outlined in Sect. 4.4, one to appreciate the accuracy of the model and the other to appreciate its computational speed, will be available to help the user to get familiar with the code. Thanks to the extraordinary small computational resources required, our emulator can be exploited on any laptop or on-line server (e.g. Google Colab).

Future developments of our emulator will first concern the extension of the validity range of the DNN model, involving both a larger number of parameters and wider parameter intervals. This, however, will undermine the accuracy of the emulator, as the output behaviour of the target function will become comprehensively more complex, and so more difficult to reproduce. To address this issue, we will test different DNN architectures to optimize even further the performances of our code. Alternatively, we can provide different trained models for the same function, so as to lower the complexity of the single emulated model and consequently select the most suitable one for a given analysis.

Another natural improvement of our emulator will involve the usage of different Boltzmann solvers, like CLASS or MGCAMB (i.e. a modified version CAMB applicable to a number of alternative cosmological scenarios, Zhao et al., 2009; Hojjati, Pogosian & Zhao, 2011; Zucca et al., 2019). Then, we will extend our implementation by emulating more complicated 2PCF models, including e.g. the effects of the tracer bias, redshift-space and geometrical distortions (see e.g. Scoccimarro, 2004; Sánchez et al., 2014). But also by emulating different cosmological functions, to model e.g. the 2PCF multipoles (Taruya, Nishimichi & Saito, 2010; Pezzotta et al., 2017), the 3PCF (Slepian & Eisenstein, 2017), the matter power spectrum (Beutler et al., 2017; Vargas-Magaña et al., 2018), the halo mass function (Tinker et al., 2008; Despali et al., 2016), the void size function (Sheth & van de Weygaert, 2004; Jennings, Li & Hu, 2013), and many others. All these models are already present in the CBL and our methodology can be easily applied to all of them. Our goal will be therefore to prepare a large set of pre-trained models to emulate the cosmological functions implemented in the CBL, and at the same time to provide the users with all the tools necessary to train, validate, and use any model of their choice.

The methodology we presented in this Thesis work could be potentially applied to an unlimited number of cases, allowing us to validate the theoretical models of our interest for different cosmological models, as well as on different scales and redshifts, without the wasting of execution time and computational resources. This will be of key importance in the perspective of the large amount of data expected from the new-generation telescopes, like JWST, *Euclid* and LSST.

Ringraziamenti

Il merito non esiste. Arrivare alla fine degli studi è come terminare la costruzione di una *rete* di strade e percorsi che connettono conoscenze ed abilità di un individuo. Questa rete, beninteso, verrà ampliata negli anni a venire, ma adesso è il momento di fermarsi un attimo a contemplarla per prenderne coscienza.

La rete che ho costruito in questi anni non me la sono meritata, perché in ogni suo nodo e componente c'è l'impronta di qualcuno che insieme a me ha contribuito ad edificarla. In vero non dico nulla di nuovo, è fatto noto fra gli scienziati, i quali sanno che ogni scoperta fatta, ogni tassello aggiunto al sapere, si posa su tutte quelle precedenti e negarne la connessione è solo un atto di hybris. Quindi voglio ringraziare tutte le persone che hanno dato la loro impronta, il loro contributo per costruire la mia rete. Purtroppo non riuscirò a nominarle tutte ed a tutte le escluse chiedo scusa.

Ringrazio Emanuela che da quasi sei anni contribuisce a rendere le mie giornate *belle*, dal mattino alla sera, conferendo ulteriore forza e desiderio di fare, di conoscere e scoprire, perché *tutto è più autentico se condiviso con lei.*

Ringrazio gli amici della Sicilia. Da Angelo con il quale condividiamo la passione per il basket ed il piacere del confronto sui temi più disparati. Martina ed Emiliano, il cui affetto ed attenzioni fraterne mi scaldano sempre l'animo a prescindere in quale parte del mondo mi trovi. Marcello, col quale fare polemica è un piacere raro. Ad Ale, Sebi ed Anto compagni di avventure e disavventure.

Ringrazio gli amici di Bologna. Dai "Sardi": Fabio, Albi e Fabri con i quali ho passato le cene bolognesi più piacevoli, una fusione di pietanze e persone squisite. Un ringraziamento particolare va a Fabri, il quale è stato di supporto essenziale nel mio passaggio dal corso di Fisica Teorica ad Astrofisica e Cosmologia, sempre presente e disponibile a condividere la sua esperienza e le sue conoscenze. Ringrazio i miei coinquilini ed amici, Vincenzo, Giorgia e Sara con i quali ho vissuto il periodo della quarantena in un ambiente meraviglioso, nonostante la tragedia che il mondo stava vivendo in quei giorni.

Ringrazio la mia famiglia. Mia madre, Adelù, e mio padre, Ciccio, che hanno sacrificato il loro tempo per concedermi il privilegio di disporre del mio. Mia sorella Matilde, che nonostante sia più giovane di me, è un riferimento importante per la mia crescita personale.

Ringrazio tutte le Professoresse ed i Professori incontrati in questi anni, ma in particolare i miei relatori, il Prof. Marulli e la Dott.ssa Contarini. Ringrazio Federico per avermi dato la possibilità di studiare e cimentarmi nell'ambito del Machine Learning nonostante non avessi alcuna conoscenza pregressa. Inoltre lo ringrazio per essere sempre stato disponibile a sciorinare e sbrogliare i miei dubbi ed avermi prestato la sua attenzione per ogni idea che mi passava per la mente, anche se queste il più delle volte, non avevano né capo e né coda. Ringrazio Sofia, la quale è stata fin da subito una mentore attenta e paziente, sempre pronta ad insegnare e condividere il suo sapere con estrema capacità e bravura. Inoltre, rispetto alle mie difficoltà ha mostrato una sensibilità ed una comprensione che costituiscono un enorme valore aggiunto. Senza loro due questo lavoro non sarebbe lo stesso.

Bibliography

Abdalla E. et al., 2022, Journal of High Energy Astrophysics, 34, 49

Amendola L. et al., 2018, Living Reviews in Relativity, 21, 2

Bagla J. S., Padmanabhan T., 1997, Pramana, 49, 161

Beutler F. et al., 2011, Mon. Not. R. Astron. Soc., 416, 3017

Beutler F. et al., 2017, Mon. Not. R. Astron. Soc., 464, 3409

Blas D., Lesgourgues J., Tram T., 2011, J. Cosm. Astro-Particle Phys., 2011, 034

Bonici M., Biggio L., Carbone C., Guzzo L., 2022, arXiv e-prints, arXiv:2206.14208

Chevallier M., Polarski D., 2001, International Journal of Modern Physics D, 10, 213

Coles P., Lucchin F., 2002, Cosmology: The Origin and Evolution of Cosmic Structure, Second Edition

Crocce M., Scoccimarro R., 2008, Phys. Rev. D, 77, 023533

de la Calleja J., Fuentes O., 2004, Mon. Not. R. Astron. Soc., 349, 87

DeRose J., Chen S.-F., White M., Kokron N., 2022, J. Cosm. Astro-Particle Phys., 2022, 056

Despali G., Giocoli C., Angulo R. E., Tormen G., Sheth R. K., Baso G., Moscardini L., 2016, Mon. Not. R. Astron. Soc., 456, 2486

Dodelson S., 2003, Modern Cosmology

Efron B., 1982, The Jackknife, the Bootstrap and Other Resampling Plans. Society for Industrial and Applied Mathematics

Ellis G. F. R., Maartens R., MacCallum M. A. H., 2012, Relativistic Cosmology

Euclid Collaboration et al., 2021, Mon. Not. R. Astron. Soc., 505, 2840

Euclid Collaboration: Blanchard A. et al., 2020, Astron. Astrophys., 642, A191

Fixsen D. J., Cheng E. S., Gales J. M., Mather J. C., Shafer R. A., Wright E. L., 1996, Astrophys. J., 473, 576

Gardner J. P. et al., 2006, Space Science Reviews, 123, 485

Giocoli C., Baldi M., Moscardini L., 2018, Mon. Not. R. Astron. Soc., 481, 2813

Graham M. L., Connolly A. J., Ivezić Ž., Schmidt S. J., Jones R. L., Jurić M., Daniel S. F., Yoachim P., 2018, Astron. J., 155, 1

Guth A. H., Pi S.-Y., 1982, Physical Review Letters, 49, 1110

Hagstotz S., Costanzi M., Baldi M., Weller J., 2019, Mon. Not. R. Astron. Soc., 486, 3927

Hezaveh Y. D., Perreault Levasseur L., Marshall P. J., 2017, Nature, 548, 555

Hojjati A., Pogosian L., Zhao G.-B., 2011, J. Cosm. Astro-Particle Phys., 2011, 005

Hoyle B., 2016, Astronomy and Computing, 16, 34

Ivezić Ž. et al., 2019, Astrophys. J., 873, 111

Jennings E., Li Y., Hu W., 2013, Mon. Not. R. Astron. Soc., 434, 2167

Kalirai J., 2018, Contemporary Physics, 59, 251

Kingma D. P., Ba J., 2014, arXiv e-prints, arXiv:1412.6980

Landy S. D., Szalay A. S., 1993, Astrophys. J., 412, 64

Laureijs R. et al., 2011, ArXiv e-prints

Lesgourgues J., 2011, arXiv e-prints, arXiv:1104.2932

Lewis A., Challinor A., Lasenby A., 2000, Astrophys. J., 538, 473

Linder E. V., Jenkins A., 2003, Mon. Not. R. Astron. Soc., 346, 573

LSST Dark Energy Science Collaboration, 2012, arXiv e-prints, arXiv:1211.0310

Marulli F., Veropalumbo A., Moresco M., 2016, Astronomy and Computing, 14, 35

Metcalf R. B., 2022, Notes: Practical Statistics for Physics & Astronomy

Metropolis N., Rosenbluth A. W., Rosenbluth M. N., Teller A. H., Teller E., 1953, J. Chem. Phys., 21, 1087

Mészáros P., 1974, Astron. Astrophys., 37, 225

Norberg P., Baugh C. M., Gaztañaga E., Croton D. J., 2009, Mon. Not. R. Astron. Soc., 396, 19

Ntampaka M., Trac H., Sutherland D. J., Battaglia N., Póczos B., Schneider J., 2015, Astrophys. J., 803, 50

Nygaard A., Brinch Holm E., Hannestad S., Tram T., 2022, arXiv e-prints, arXiv:2205.15726

Peebles P. J. E., Hauser M. G., 1974, Astrophys. J. Suppl., 28, 19

Percival W. J. et al., 2010, Mon. Not. R. Astron. Soc., 401, 2148

Pezzotta A. et al., 2017, Astron. Astrophys., 604, A33

Planck Collaboration et al., 2016, Astron. Astrophys., 594, A13

Planck Collaboration et al., 2020, Astron. Astrophys., 641, A6

Ray D., Vickers J., 2022, Introducing Einstein's Relativity: A deeper understanding

Ryden B., 2016, Introduction to Cosmology. Cambridge University Press

Sánchez A. G. et al., 2014, Mon. Not. R. Astron. Soc., 440, 2692

Schanche N. et al., 2019, Mon. Not. R. Astron. Soc., 483, 5534

Scoccimarro R., 2004, Phys. Rev. D, 70, 083007

Sheth R. K., van de Weygaert R., 2004, Mon. Not. R. Astron. Soc., 350, 517

Slepian Z., Eisenstein D. J., 2017, Mon. Not. R. Astron. Soc., 469, 2059

Springel V., 2005, Mon. Not. R. Astron. Soc., 364, 1105

Spurio Mancini A., Piras D., Alsing J., Joachimi B., Hobson M. P., 2022, Mon. Not. R. Astron. Soc., 511, 1771

Taruya A., Nishimichi T., Saito S., 2010, Phys. Rev. D, 82, 063522

Tinker J., Kravtsov A. V., Klypin A., Abazajian K., Warren M., Yepes G., Gottlöber S., Holz D. E., 2008, Astrophys. J., 688, 709

Trotta R., 2017, arXiv e-prints, arXiv:1701.01467

Vargas-Magaña M. et al., 2018, Mon. Not. R. Astron. Soc., 477, 1153

Veronesi N., Marulli F., Veropalumbo A., Moscardini L., 2023, Astronomy and Computing, 42, 100692

Zeldovich Y. B., 1972, Mon. Not. R. Astron. Soc., 160, 1P

Zhao G.-B., Pogosian L., Silvestri A., Zylberberg J., 2009, Phys. Rev. D, 79, 083513

Zucca A., Pogosian L., Silvestri A., Zhao G. B., 2019, J. Cosm. Astro-Particle Phys., 2019, 001