UNIVERSITÀ DEGLI STUDI DI MILANO

DATA SCIENCE AND ECONOMICS

MODELLING DYNAMIC SYSTEMS WITH NEURAL NETWORKS

Students:
Michele Bartesaghi and Guido Giacomo Mussini
Registration numbers: 17098A and 988273

ACADEMIC YEAR 2021-2022

Abstract

This paper deals with the challenge of predicting the evolution of a dynamic system thanks to neural networks, embedding the conservation of energy in the model. First and foremost, a brief analysis on neural networks is carried out in order to introduce the necessary concepts. In particular, the Universal Approximation Theorem is proved. Consequently, the basics of both the Hamiltonian and the Lagrangian mechanics are presented together with an ideal double pendulum as a real-world application. Furthermore, the novel approaches of both the Hamiltonian Neural Networks (HNN) and the Lagrangian Neural Networks (LNN) are examined from a theoretical point of view. The paper makes use of the aforementioned double pendulum in order to provide a simple, yet meaningful example of the presented techniques, as well as a comparison with the Baseline Neural Networks (BNN) approach.

Contents

Chapter 1

Introduction

Artificial neural networks perform several different tasks, yet they fail at modelling symmetries and conservation laws of paramount importance with regards to dynamic systems, which are physical models describing the evolution of a system in time. In an attempt to find a solution, researchers tried to take advantage of Hamiltonian and Lagrangian mechanics in order to create innovative neural networks models that can learn physics priors in a correct fashion. In particular, both the Hamiltonian and the Lagrangian mechanics have been included in the machine learning framework in order to properly model the conservation of energy. These new approaches represent an utmost enhancement of the ones that have been used so far, and they allow complex systems to be efficiently simulated by the human being. Although in this paper the presented practical example is rather simple, namely an ideal double pendulum, it is riveting as it embeds the characteristics of a chaotic dynamic system. Furthermore, the novel neural networks are able to model more complex cases, such as celestial mechanics, fluid dynamics as well as quantum mechanics ([14], [15]).

Chapter 2

Baseline Neural Networks (BNNs)

Artificial neural networks (NNs) are computational models which can be regarded as a simplification of both the structure and the functioning of the human brain. NNs first appeared in the twentieth century and nowadays they are extensively used in different fields and for diverse applications.

A neural network can be described as a directed graph in which many neurons can be linked by arcs. This graph is denoted by $G = (V, E)$, where V is the set of all *nodes* (neurons) and E is the set of all *edges* (arcs). Observe that V can be partitioned as follows:

$$V = \dot{\bigcup}_{t=0}^{T} V_t.$$

where V_0 is called *input layer*, V_T is called *output layer* and $V_1, \dots, V_{T-1}$ are called *hidden layers.* The input layer contains $n + 1$ input nodes, where n is the dimensionality of the input space (typically $\mathbb{R}^n$), whereas the output layer is made of m *output nodes*, where m is the dimensionality of the output space (typically $\mathbb{R}^m$). Furthermore $|V|$ and $\max_t |V_t|$ are defined as *size* and *width* of the network respectively, while T is called *depth.*

Before continuing with the description of artificial neural networks, it is convenient to make a few assumptions about their structure. First and foremost, this paper focuses exclusively on *feedforward* neural networks, which are acyclic NNs, namely graphs in which edges are only allowed to go from V_t to V_{t+1}. In addition, in this theoretical description the assumption that neural networks are fully connected is made; in other words each node in V_t is connected to each node in V_{t+1}. These premises always hold in this section, unless explicitly stated.

In order to fully specify a neural network, there are two missing crucial elements.

Definition 2.0.1. Given a directed acyclic graph $G = (V, E)$, a function $w : E \to \mathbb{R}$ is called **weight function**.

Let $v_{t,j}$ be the *j-th* node in the *t-th* layer and let $(v_{t,j}, v_{t+1,i})$ denote the edge going from $v_{t,j}$ to the *i-th* node in the following layer V_{t+1}. Then, in case of a fully connected neural network there is no $v_{t,j} \in V_t$ such that the weight associated with the edge $(v_{t,j}, v_{t+1,i})$ is zero. In other words, in this analysis the weight function w is not allowed to take up the value 0.

Definition 2.0.2. Given a neural network $G = (V, E)$, the scalar function $\sigma : \mathbb{R} \to \mathbb{R}$ associated with each node is called **activation function** of the neuron.

Observe that, potentially, each node could have its own activation function, different from the ones associated with other neurons. The most common are the following:

- Sign function:
$$\sigma(z) = \operatorname{sgn}(z) = \begin{cases} 1 & z > 0 \\ 0 & z = 0 \\ -1 & z < 0 \end{cases}$$
- Leaky ReLU (Rectified Linear Unit):
$$\sigma(z) = (\alpha \mathbb{1}\{z < 0\} + \mathbb{1}\{z \geq 0\})z, \qquad \text{where } \alpha \text{ is typically of the order of } 10^{-2}.$$
- Sigmoid (or logistic):
$$\sigma(z) = \frac{1}{1 + e^{-z}}, \qquad \text{that is sometimes rescaled so that } \sigma(z) \in [-1, 1].$$
- Hyperbolic tangent:
$$\sigma(z) = \tanh(z) = \frac{\sinh(z)}{\cosh(z)} = \frac{e^z - e^{-z}}{e^z + e^{-z}}.$$
- Softplus, that can be viewed as a smooth version of the ReLU:
$$\sigma(z) = \log(1 + e^z).$$

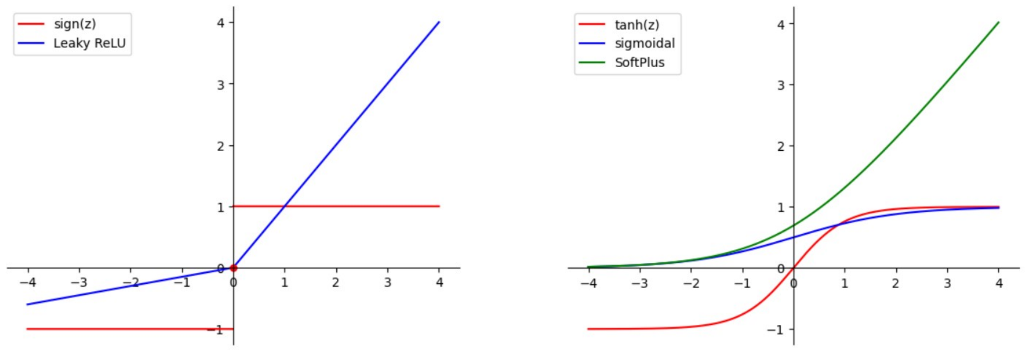


Figure 2.1: Activation functions written defined above.

In a neural network each node has one input and one output. Each edge in the graph links the output of a certain neuron to the input of another neuron in the next layer. Let $o_{t,i}$ be the output of the node $v_{t,i} \in V_t$, $t = 1, \dots, T$. Given a data point $x \in \mathbb{R}^n$, the output of the *i-th* input node in the input layer V_0 is simply x_i. The output of the $n{+}1$-*th* neuron always satisfies $o_{0,n+1} \equiv 1$. On the other hand, the output value of the node $v_{t,i} \in V \setminus V_0$ depends on its activation function and it is defined by

$$o_{t,i} = \sigma(a_{t,i}),$$

where $a_{t,i}$ is the input of that node. Furthermore, the input is defined as the weighted sum of the outputs of the nodes in the previous layer, namely

$$a_{t,i}(x) = \sum_{r:(v_{t-1,r},v_{t,i})\in E} w(v_{t-1,r}, v_{t,i}) o_{t-1,r}(x), \qquad x \in \mathbb{R}^n.$$

The true objective of a neural network is to learn a class of functions, called *hypothesis class* ($\mathcal{H}$), in order to perform different tasks, such as image recognition, classification and dynamic systems modelling, just to name a few.

Definition 2.0.3. Given a graph (V, E) and an activation function $\sigma : \mathbb{R} \to \mathbb{R}$, the set

$$\mathcal{H}_{V,E,\sigma} = \{h_{V,E,\sigma,w} \mid w : E \to \mathbb{R}\}$$

is called **hypothesis class** and it is the set of all the functions

$$h_{V,E,\sigma,w} : \mathbb{R}^{|V_0|-1} \to \mathbb{R}^{|V_T|}$$

for some $w : E \to \mathbb{R}$.

It follows that the trainable parameters of a learning problem involving artificial neural networks are the weights over the edges of the graph. In particular, one can define W as the $|V| \times |V|$ matrix of such parameters and call it *weight matrix.* This matrix plays a fundamental role in the learning process because setting optimal parameters can lead to significantly better results.

In the following section, the Universal Approximation Theorem is presented. This fundamental result guarantees that any continuous function, which can be learnt in order to model a physical system, can be arbitrarily well approximated by a neural network meeting the hypotheses.

2.1 Universal Approximation Theorem

Definition 2.1.1. Let $X \subseteq \mathbb{R}$ be a set. Then X is said to be **dense** in $\mathbb{R}$ if for any $\varepsilon > 0$ and for any $y \in \mathbb{R}$, there exists $x \in X$ such that $d(x, y)_{\mathbb{R}} < \varepsilon$.

A classical, less generalised version of the theorem is shown in [9]. The statement is the following.

Theorem 2.1.2. *Let $\rho : \mathbb{R} \to \mathbb{R}$ be any continuous function. Let $\mathcal{N}_n^\rho$ represent the class of feedforward neural networks with activation function ρ, n neurons in the input layer, one neuron in the output layer and one hidden layer with an arbitrary number of neurons. Let also $K \subseteq \mathbb{R}^n$ be compact. Then $\mathcal{N}_n^\rho$ is dense in $C(K)$ if and only if ρ is polynomial.*

Hence the classical theorem deals with NNs characterised by arbitrary width and bounded depth. In this paper a different and more general version will be presented, rearranged from [10], where neural networks have bounded width and arbitrary depth. Moreover, it is crucial to study which types of activation functions are admitted in order for the class of feedforward neural networks to be dense in $C(K)$. There are many existing works that prove interesting results relying on specific assumptions and particular activation functions, such as the ReLU ([11], [12], [13]).

Let $\mathcal{NN}_{n,m,k}^\rho$ denote the hypothesis class $\mathcal{H} = \{f : \mathbb{R}^n \to \mathbb{R}^m\}$ computed by neural networks with n input nodes, m output nodes, an arbitrary number of hidden layers, each with k neurons and activation function ρ. Furthermore, let $C(K, \mathbb{R}^m) = \{f : K \to \mathbb{R}^m : f \text{ continuous}\}$. Then, the statement of the theorem is the following.

Theorem 2.1.3 (Universal Approximation). *Let $\rho : \mathbb{R} \to \mathbb{R}$ be any non-affine continuous function, which is continuously differentiable at at least one point, with non-zero derivative at that point. Let $K \subseteq \mathbb{R}$ be compact. Then $\mathcal{NN}_{n,m,n+m+2}^\rho$ is dense in $C(K, \mathbb{R}^m)$ with respect to the uniform norm.*

In order to prove this result it is necessary to introduce a few essential propositions.

First of all, note that a neuron can be seen as an activation function ρ composed with two affine functions[1] ϕ, ψ and this kind of neurons are called *enhanced neurons*. Thanks to this representation it might be easier to present the functions learnt by a neural network. The idea behind this formula is that many activation functions can be used to approximate the identity function, thus allowing the neurons to store information and pass it through the layers (*registers*) as shown in the following proposition.

Proposition 2.1.4. Let $\rho : \mathbb{R} \to \mathbb{R}$ be any continuous function which is continuously differentiable at at least one point, with non-zero derivative at that point. Let $L \subseteq \mathbb{R}$ be compact. Then a single enhanced neuron with activation function ρ may uniformly approximate the identity function $id : \mathbb{R} \to \mathbb{R}$ on L, with arbitrarily small error.

Proof. Let $a \neq b$, with $a, b \in \mathbb{R}$. Since ρ is continuously differentiable, there exists a neighbour of $[a, b] \subseteq \mathbb{R}$ on which ρ is differentiable and $\alpha \in (a, b)$ such that ρ' is continuous at that point and $\rho'(\alpha) \neq 0$. Given $h \in \mathbb{R} \setminus \{0\}$ let

$$\phi_h(x) = hx + \alpha, \qquad \psi_h(x) = \frac{x - \rho(\alpha)}{h\rho'(\alpha)}.$$

Then it is possible to call $id_h = \psi_h \circ \rho \circ \phi_h$ that is the form of an enhanced neuron. By the Mean Value Theorem, for all $v \in [a, b]$ there exist ξ_v between v and α such that $\rho(v) = \rho(\alpha) + (v - \alpha)\rho'(\xi_v)$. Thus, for every h sufficiently small such that $\phi_h(L) \subseteq [a, b]$,

$$\begin{aligned} id_h(x) &= (\psi_h \circ \rho \circ \phi_h)(x) \\ &= \psi_h(\rho(\phi_h(x))) \\ &= \psi_h(\rho(hx + \alpha)) \\ &= \psi_h(\rho(\alpha) + (hx)\rho'(\xi_{hx+\alpha})) \\ &= \frac{\rho(\alpha) + hx\rho'(\xi_{hx+\alpha}) - \rho(\alpha)}{h\rho'(\alpha)} \\ &= \frac{x\rho'(\xi_{hx+\alpha})}{\rho'(\alpha)}. \end{aligned}$$

[1]A function f of the form $f(x) = ax + b$, $a \neq 0$, $b \in \mathbb{R}$.

Let $id : \mathbb{R} \to \mathbb{R}$ be the identity function, then for all $x \in L$

$$\begin{aligned} |id_h(x) - id(x)| &= \left| \frac{x\rho'(\xi_{hx+\alpha})}{\rho'(\alpha)} - x \right| \\ &= \left| \frac{x\rho'(\xi_{hx+\alpha}) - x\rho'(\alpha)}{\rho'(\alpha)} \right| \\ &\leq \frac{|x|}{|\rho'(\alpha)|} \omega(hx), \end{aligned}$$

where $\omega(hx) = |\rho'(\xi_{hx+\alpha}) - \rho'(\alpha)|$ is called *modulus of continuity*. By definition, a modulus of continuity is any increasing function $\omega : [0, \infty] \to [0, \infty]$ such that

$$\lim_{t \to 0} \omega(t) = \omega(0) = 0$$

and it is admitted by a function $f : L \to \mathbb{R}$ if and only if $|f(x) - f(y)| \leq \omega(|x - y|)$, for all x, y in the domain of f. Since $\frac{|x|}{|\rho'(\alpha)|}$ is a product of continuous functions over a compact, it is limited, hence we can write

$$\frac{|x|}{|\rho'(\alpha)|} \leq M_h.$$

Thanks to the property of the modulus of continuity,

$$\lim_{h \to 0} \sup_{x \in L} |id_h(x) - id(x)| = 0,$$

namely $id_h \to id$ uniformly over L as $h \to 0$. $\square$

In the following results, id_h denotes such an approximation of the identity function id. Since uniform convergence is equivalent to the convergence in $\|\cdot\|_\infty$,[2] this fact is denoted by $\|id_h - id\|_\infty \xrightarrow[h \to 0]{} 0$. A simplification of a neural network that actually deploys the identity as activation function, instead of an approximation, is presented as an intuitive model in which some neurons are treated as registers, hence the name.

Proposition 2.1.5 (Register Model)**.** Let $\rho : \mathbb{R} \to \mathbb{R}$ be any continuous non-polynomial function. Let $\mathcal{I}^\rho_{n,m,n+m+1}$ represent the class of neural network with n input nodes, m output nodes and an arbitrary number of hidden layers each with $n + m$ neurons, with the identity activation function, and one neuron with the activation function ρ. Let $K \subseteq \mathbb{R}^n$ be compact. Then $\mathcal{I}^\rho_{n,m,n+m+1}$ is dense in $C(K, \mathbb{R}^m)$.

[2]Let f be defined on a domain K, then $\|f\|_\infty = \sup_{x \in K} |f(x)|$.

Proof. This is a constructive proof. Let $f \in C(K, \mathbb{R}^m)$, with $f = (f_1, \dots, f_m)$. The classical version of the Universal Approximation Theorem (2.1.2) ensures that there exist m single-hidden-layer neural networks $g_1, \dots, g_m \in \mathcal{N}_n^\rho$ with activation function ρ, which approximate $f_1, \dots, f_m$ respectively. Fix $\varepsilon > 0$, then each approximation is to within error ε, with respect to $\|\cdot\|_\infty$ on K, that is denoted by $\|f_i - g_i\|_\infty < \varepsilon$ for $i = 1, \dots, m$. Let each g_i have β_i hidden neurons and, as neurons are being represented as functions, let $\sigma_{i,j}$ be the operation of the j-th hidden neuron, for $j = 1, \dots, \beta_i$. Recalling the form of an enhanced neuron, let each $\sigma_{i,j}$ be the composition of its activation function and the affine function of the single node in the output layer of g_i, so that $g_i = \sum_{j=1}^{\beta_i} \sigma_{i,j}$. Ultimately, define $M = \sum_{i=1}^{m} \beta_i$ which is the sum of all the hidden neurons of all the m neural networks.

The objective is to build a neural network $N \in \mathcal{I}_{n,m,n+m+1}^\rho$ that for a given input $x = (x_1, \dots, x_n) \in \mathbb{R}^n$ will output

$$G = (G_1, \dots, G_m) = (g_1(x), \dots, g_m(x)) \in \mathbb{R}^m.$$

Namely, N will compute all the shallow neural networks, thus achieving

$$\|f - N\|_\infty < \varepsilon.$$

The construction works as follows. Imagine a neural network with the input layer at the bottom and the output layer at the top. The resulting structure present M hidden layers, that can be naturally divided in groups each one β_i-neurons high and $n + m + 1$-neurons wide. Fix the first group with β_1 hidden layers in order to make the notation more clear. Moving along the horizontal direction (2.2) for each of the β_1 layers, the first set with n neurons, which are called *in-register neurons*, is made of nodes that simply stores the input $x_1, \dots, x_n$ by deploying an identity activation function. The second set is made of a single neuron, characterised by the operation $\sigma_{1,j}$, where j is the height at which that neuron is located inside the group. Each computation neuron of the fixed group performs its computation on the inputs preserved and sent by the in-register neurons. To sum up, this group is made by arranging vertically the hidden layer of g_1. Ultimately, the third set is composed by m neurons (*out-register neurons*) that use the identity activation function as well, but their affine parts gradually sum up the results of the computations made by the neurons in the previous set. In other words the (l, M_1)-th out-register neuron computes $\zeta_{l,M_1} = \sum_{t=1}^{\beta_1 - 1} \sigma_{l,t}$, for $l = 1, \dots, m$, where $M_k = \sum_{t=1}^{k} \beta_t$. Finally, the neurons in the output layer receive information from the out-register neurons in the M-th hidden layer. This concludes the construction as well as the proof. $\square$

In other words, a layer with ρ as an activation function is equivalent to multiple layers in which a single neuron per layer uses ρ as an activation function, whereas the other ones deploy the identity.

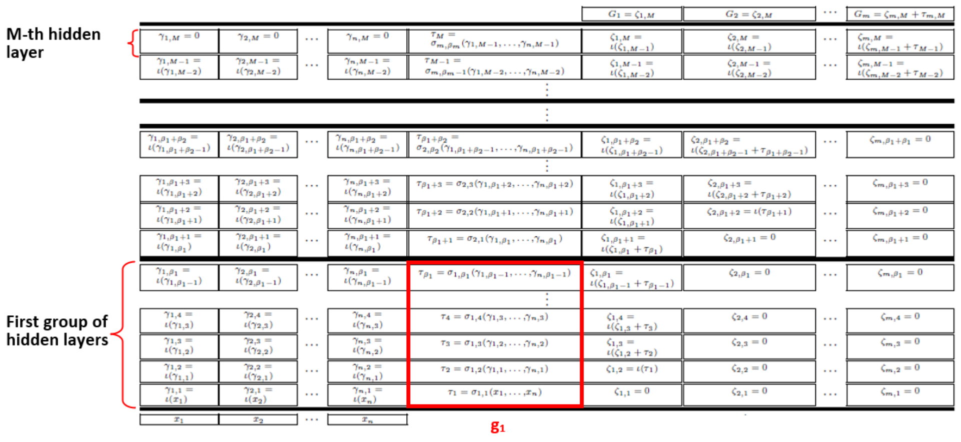


Figure 2.2: Register model from [10].

Now it is crucial to present another simplified model, which is called *Square Model* whose extensively constructive proof can be found in the aforementioned paper ([10]).

Proposition 2.1.6 (Square Model)**.** Let $\rho(x) = x^2$ and let $K \subseteq \mathbb{R}^n$ be compact. Then $\mathcal{NN}^{\rho}_{n,m,n+m+1}$ is dense in $C(K,\mathbb{R}^m)$.

The next one is the first key result in order to prove the Universal Approximation Theorem.

Proposition 2.1.7. Let $\rho : \mathbb{R} \to \mathbb{R}$ be any continuous non-polynomial function which is continuously differentiable at at least one point, with non-zero derivative at that point. Let $K \subseteq \mathbb{R}^n$ be compact. Then $\mathcal{NN}^{\rho}_{n,m,n+m+1}$ is dense in $C(K,\mathbb{R}^m)$ with respect to the uniform norm.

Proof. Let $f \in C(K,\mathbb{R}^m)$ and $\varepsilon > 0$. Thanks to the Register Model (2.1.5) it is possible to build a neural network[3] $N \in \mathcal{I}^{\rho}_{n,m,n+m+1}$ such that

$$\|f - N\|_\infty < \frac{\varepsilon}{2}.$$

However, in this case let the identity activation function id, required by the neurons in N, be approximated by id_h, as in 2.1.4.

[3]With abuse of notation $N \in \mathcal{NN}^{\rho}_{n,m,n+m+1}$ denotes the fact that a function approximated by N is in the hypothesis class.

Ultimately, let $N_h = N \circ id_h$ be this new model. Then

$$\|f - N_h\|_\infty \leq \|f - N\|_\infty + \|N - N_h\|_\infty.$$

Note that $\|f - N\|_\infty \leq \varepsilon/2$ as stated above and $\|N - N_h\|_\infty \leq \varepsilon/2$, too. In fact N is uniformly continuous, id_h is uniformly convergent and the composition N_h uniformly converges to N. Therefore, the new model approximates arbitrarily well f on a compact, since the image of a compact set through a continuous function is still compact, thus concluding the proof. □

The second key result is the following proposition, which consider polynomial activation functions.

Proposition 2.1.8. Let $\rho : \mathbb{R} \to \mathbb{R}$ be any non-affine polynomial function. Let $K \subseteq \mathbb{R}^n$ be compact. Then $\mathcal{NN}^{\rho}_{n,m,n+m+2}$ is dense in $C(K, \mathbb{R}^m)$ with respect to the uniform norm.

Proof. First of all, fix $\alpha \in \mathbb{R}$ such that $\rho''(\alpha) \neq 0$. Observe that ρ'' exists since ρ is non-affine by assumption. Let $h \in (0, \infty)$ and let $\sigma_h : \mathbb{R} \to \mathbb{R}$ be a function defined by

$$\sigma_h(x) = \frac{\rho(\alpha + hx) - 2\rho(\alpha) + \rho(\alpha - hx)}{h^2 \rho''(\alpha)}.$$

Consider the Taylor expansion centred in α

$$\rho(\alpha + hx) = \sum_{i=0}^{n-1} \frac{\rho^{(i)}(\alpha)}{i!}(hx)^i + \mathcal{O}(h^n x^n).$$

Then,

$$\begin{aligned}\sigma_h(x) = &\frac{\rho(\alpha) + hx\rho'(\alpha) + h^2x^2\rho''(\alpha)/2 + \mathcal{O}(h^3x^3)}{h^2\rho''(\alpha)} - \frac{2\rho(\alpha)}{h^2\rho''(\alpha)} + \\ &\frac{\rho(\alpha) - hx\rho'(\alpha) + h^2x^2\rho''(\alpha)/2 + \mathcal{O}(h^3x^3)}{h^2\rho''(\alpha)},\end{aligned}$$

and performing some simplifications leads to $\sigma_h(x) = x^2 + \mathcal{O}(hx^3)$, which strongly resembles a square activation function and will be used later in the proof in order to approximate $\gamma(x) = x^2$.

Observe that by definition σ_h exactly performs two operations on affine transformations of x via ρ, namely $\rho(\alpha \pm hx)$. Therefore, two enhanced neurons with activation function ρ can approximate the operation of a single enhanced neuron with square activation function.

Let N be a neural network satisfying the Square Model properties (2.1.6) and ℓ be any hidden layer containing $n+m+1$ neurons. Denote with η the $(1 \times n+m+1)$ vector made of the values of the neurons in the previous layer. Moreover, define ϕ_i as the affine part of the i-th neuron in ℓ. By doing so, η is transformed into

$$\Phi = (\gamma \circ \phi_1(\eta), \ldots, \gamma \circ \phi_{n+m+1}(\eta)) = (\phi_1(\eta)^2, \ldots, \phi_{n+m+1}(\eta)^2)$$

by ℓ. Notice that Φ can be obtained by using $n+m+1$ layers with $n+m+1$ neurons each. In every layer, only one neuron uses the square activation function, whereas the others deploy the identity function. One can visualise this structure by arranging the neurons with the square activation function along the diagonal of a $(n+m+1) \times (n+m+1)$ matrix, so that the the value of the i-th neuron is squared by the i-th layer. Furthermore, this process is repeated for every layer ℓ of N, resulting in a new network $\widetilde{N}$ which is $n+m+1$ times deeper than N but that computes exactly the same function, yet using only a single squaring operation in each layer. Now, let $\widetilde{N}_h$ be a copy of this new network with id_h instead of the identity function, making use of the activation function ρ, which is possible thanks to 2.1.4. The next step is to use σ_h defined above to approximate the square activation function found only once in each layer. Since σ_h can be approximated by two enhanced neurons with activation function ρ, this last passage requires to add a neuron in each hidden layer, thus leading to a network with $n+m+2$ neurons in each hidden layer. Furthermore, $\widetilde{N}_h$ only uses ρ, meaning that $\widetilde{N}_h \in \mathcal{NN}^{\rho}_{n,m,n+m+2}$ as desired. Ultimately,

$$\|f - \widetilde{N}_h\|_\infty \le \|f - \widetilde{N}\|_\infty + \|\widetilde{N} - \widetilde{N}_h\|_\infty \le \varepsilon,$$

similarly to the proof of 2.1.7 and this concludes the proof. $\square$

The universal approximation theorem 2.1.3 follows directly from both proposition 2.1.7 and 2.1.8.

Observe that this result gives the mathematical foundation to the fact that it is possible to approximate both the Lagrangian and the Hamiltonian functions, which will be presented later. In other words, the dynamics of the systems which are described by functions that satisfy all the conditions of the Universal Approximation Theorem can be approximated arbitrarily well by a neural network. Nevertheless, the higher the desired level of approximation, the more complex it is for the neural network to solve the problem, which easily becomes computationally infeasible.

Chapter 3

Dynamics of mechanical systems

In this chapter, a little insight into the basic concepts upon which both the Hamiltonian and the Lagrangian neural networks are modelled will be presented.

Definition 3.0.1. A **vector field $\mathbf{F}$** in $\mathbb{R}^n$ is an application $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^m$ that maps each point $x \in \mathbb{R}^n$ in a vector $\mathbf{F}(x)$ applied in x.

Definition 3.0.2. A parametrised curve $x_i = x_i(t),\ t \in [a,b]$ is called **integral curve** of the vector field $\mathbf{F}$ if in every point in which the curve is defined, its tangent vector equals $\mathbf{F}$ evaluated at the same point. In other words, integral curves satisfy

$$\frac{dx_i}{dt} = F_i(x_1, \ldots, x_n), \quad i = 1, \ldots, n.$$

This system is called *system of first order ordinary differential equations (ODEs) associated to the vectorial field.* When the second derivative of the unknown function appears in the system, this is a system of second order differential equations.

3.1 Lagrangian mechanics

The Euler-Lagrange equations are a system of n second order differential equations of the form

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = 0, \quad k = 1, \ldots, n, \tag{3.1}$$

where L is a function of the $2n$ variables $(q_1, \dots, q_n, \dot{q}_1, \dots, \dot{q}_n)$ (and eventually of the time t) called *Lagrangian.* In case of conservative forces[1] $\mathbf{F} = -\nabla U$ Newton equations $\mathbf{F} = m\mathbf{a}$, describing the dynamics of a system, can be written in the Lagrangian form as in 3.1, with $L = T - U$.

Note that L can be expressed as a function of arbitrary coordinates instead of the classic euclidean ones $(x_1, \dots, x_n)$ without changing the form of the Euler-Lagrange equations. In fact, with euclidean coordinates $L = \frac{1}{2}m(\dot{x}_1^2 + \cdots + \dot{x}_n^2) - U(x_1, \dots, x_n)$ and it is trivial to see that

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{x}_k} - \frac{\partial L}{\partial x_k} = m\ddot{x}_k + \frac{\partial U}{\partial x_k} = 0 \qquad k = 1, \dots, n.$$

On the other hand, with arbitrary coordinates $(q_1, \dots, q_n)$, if the position is uniquely identified by the vector $\mathbf{r} = (x_1(q_1, \dots, q_n), \dots, x_n(q_1, \dots, q_n))$ the form of the Euler-Lagrange equations is preserved with the kinetic energy written as

$$T = \frac{1}{2}m\mathbf{v} \cdot \mathbf{v} = \frac{1}{2}m \sum_{i,j=1}^{n} (\frac{\partial \mathbf{r}}{\partial q_i} \cdot \frac{\partial \mathbf{r}}{\partial q_j})\dot{q}_i\dot{q}_j$$

and the potential energy

$$U = U(q_1, \dots, q_n, \dot{q}_1, \dots, \dot{q}_n).$$

Definition 3.1.1. A quantity that is constant on the solutions of a system of differential equations is called **conserved quantity**.

Proposition 3.1.2. If the Lagrangian does not depend explicitly on time, the Euler-Lagrangian equations admit a conserved quantity called *Jacobi's integral.*

Proof. Consider the derivative of L on the solutions of the Euler-Lagrange equations

$$\frac{dL}{dt} = \sum_{i=1}^{n} \frac{\partial L}{\partial q_i}\frac{dq_i}{dt} + \sum_{i=1}^{n} \frac{\partial L}{\partial \dot{q}_i}\frac{d\dot{q}_i}{dt} = \sum_{i=1}^{n} \left(\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k}\right)\frac{dq_i}{dt} + \sum_{i=1}^{n} \frac{\partial L}{\partial \dot{q}_i}\frac{d\dot{q}_i}{dt} = \sum_{i=1}^{n} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\dot{q}_i\right).$$

Therefore the quantity

$$\sum_{i=1}^{n} \frac{\partial L}{\partial \dot{q}_i}\dot{q}_i - L$$

[1]Forces such that the total work done is independent of the path taken. They only depend on the position of the object, hence they change the potential energy of the object by an amount that does not depend on the path.

is constant on the solutions of the Euler-Lagrange equations and it coincides with the total energy of the system.

$$E(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \sum_{i=1}^{n} \frac{\partial L}{\partial \dot{q}_i} \dot{q}_i - L \tag{3.2}$$

□

3.2 Hamiltonian mechanics

Consider the coordinate system that has been used so far $(q, \dot{q})$ and a Lagrangian L. In addition, consider the mapping $\mathcal{L}$ called *Legendre transformation*, which associates $(q, \dot{q})$ to (q, p), where

$$p_i = \frac{\partial L}{\partial \dot{q}_i}, \quad i = 1, \dots, n$$

are called *canonical momenta*.

Definition 3.2.1. The coordinates (q, p) are called **canonical** if they satisfy the following relations:

$$\{q_i, q_j\} = 0, \qquad \{p_i, p_j\} = 0, \qquad \{q_i, p_j\} = \delta_{ij},$$

where

$$\{g, h\} = \sum_i \left(\frac{\partial g}{\partial q_i} \frac{\partial h}{\partial p_i} - \frac{\partial g}{\partial p_i} \frac{\partial h}{\partial q_i} \right)$$

is a function called *Poisson bracket.*

Thanks to this transformation, 3.2 becomes a well-known function called *Hamiltonian*, which depends on p, q (and eventually on t),

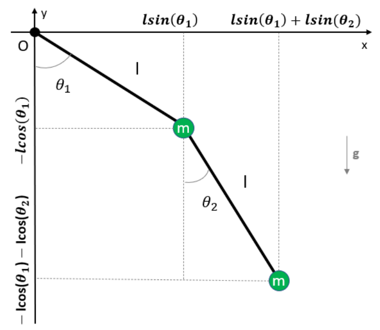
$$H(p, q) = E(q, \dot{q}) = \sum_{i=1}^{n} p_i \dot{q}_i - L(q, \dot{q}).$$

Hence, the Euler-Lagrange equations 3.1 become the system of $2n$ first order differential equations, called Hamilton equations

$$\begin{aligned} \frac{dq_i}{dt} &= \frac{\partial H}{\partial p_i}, \qquad i = 1, \dots, n, \\ \frac{dp_i}{dt} &= -\frac{\partial H}{\partial q_i}, \quad i = 1, \dots, n. \end{aligned} \tag{3.3}$$

3.3 Double pendulum

In order to provide a real-world application of this theoretical concepts, this paper will describe the dynamics of the double pendulum. A double pendulum is made of two pendulums directly attached to each other. Suppose that each pendulum P_i is made of a a mass m_i connected to a massless rigid rod l_i, which is only allowed to move along the vertical plane. The hinge of P_1 is the fixed point O and the angles between the two rods l_1, l_2 and the vertical direction are called θ_1 and θ_2 respectively (3.1). In order to simplify the analysis, let's consider $m = m_1 = m_2$ and $l = l_1 = l_2$ Finally, let all motion be frictionless.



y
$lsin(\theta_1)$
$lsin(\theta_1) + lsin(\theta_2)$
x
O
θ_1
l
m
$-lcos(\theta_1)$
g
l
θ_2
$-lcos(\theta_1) - lcos(\theta_2)$
m

Figure 3.1: Double pendulum.

In order to compute both the Lagrangian L and the Hamiltonian H of the system, consider the components $x_i, y_i, \dot{x}_i, \dot{y}_i$, under the assumptions stated above, which also make the notation more clear. Let

$$\begin{cases} x_1 = l\sin(\theta_1) \\ y_1 = -l\cos(\theta_1) \end{cases} \qquad \begin{cases} x_2 = l\sin(\theta_1) + l\sin(\theta_2) \\ y_2 = -l\cos(\theta_1) - l\cos(\theta_2) \end{cases} .$$

Deriving x_i and y_i with respect to the time t, one obtains

$$\begin{cases} \dot{x_1} = l\dot{\theta_1}\cos(\theta_1) \\ \dot{y_1} = l\dot{\theta_1}\sin(\theta_1) \end{cases} \qquad \begin{cases} \dot{x_2} = l\dot{\theta_1}\cos(\theta_1) + l\dot{\theta_2}\cos(\theta_2) \\ \dot{y_2} = l\dot{\theta_1}\sin(\theta_1) + l\dot{\theta_2}\sin(\theta_2) \end{cases} .$$

The Lagrangian L of the system is defined as

$$L = T - U$$

where T is the kinetic energy

$$T = \frac{1}{2}mv_1^2 - \frac{1}{2}mv_2^2 \tag{3.4}$$

and U is the potential energy

$$U = mgy_1 + mgy_2. \tag{3.5}$$

Observe that v_1^2 and v_2^2 can be decomposed as

$$v_1^2 = \dot{x_1}^2 + \dot{y_1}^2, \quad v_2^2 = \dot{x_2}^2 + \dot{y_2}^2. \tag{3.6}$$

By substituting 3.6 in 3.4, one can get

$$T = \frac{1}{2}ml^2[2\dot{\theta_1}^2 + \dot{\theta_2}^2 + 2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1 - \theta_2)]$$

and it is clear that in the same manner one can write

$$U = -mgl[2cos(\theta_1) + cos(\theta_2)].$$

Therefore the Lagrangian of the system becomes

$$L = \frac{1}{2}ml^2[2\dot{\theta_1}^2 + \dot{\theta_2}^2 + 2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1 - \theta_2)] + mgl[2\cos(\theta_1) + \cos(\theta_2)].$$

In order to determine the Hamiltonian of the system, the canonical momenta are given by

$$p_{\theta_1} = \frac{\partial L}{\partial \dot{\theta}_1} = ml^2(2\dot{\theta}_1 + \dot{\theta}_2\cos(\theta_1 - \theta_2))$$
$$p_{\theta_2} = \frac{\partial L}{\partial \dot{\theta}_2} = ml^2(\dot{\theta}_2 + \dot{\theta}_1\cos(\theta_1 - \theta_2))$$

In this case the Hamiltonian H of the system becomes

$$H = \sum_{i=1}^{2} \dot{\theta}_i^2 p_{\theta_i} - L$$

from which one can obtain the Hamilton equations of the system

$$\dot{\theta}_i = \frac{\partial H}{\partial p_{\theta_i}}, \quad \dot{p}_{\theta_i} = -\frac{\partial H}{\partial \theta_i} \quad \text{for} \quad i = 1, 2.$$

They require to express H in terms of the variables θ_i, p_{θ_i} only, therefore it is necessary to write $\dot{\theta}_i$ and L as functions of these variables. After a few simple algebraic passages, the details of which will not be included, the resulting Hamiltonian is

$$H = \frac{p_{\theta_1}^2 + 2p_{\theta_2}^2 - 2p_{\theta_1}p_{\theta_2}\cos(\theta_1 - \theta_2)}{2ml^2[1 + sin^2(\theta_1 - \theta_2)]} - mgl(2\cos\theta_1 - \cos\theta_2).$$

Therefore, the Hamiltonian equations of the motion of the double pendulum are the following:

$$\begin{aligned}\dot{\theta}_1 &= \frac{p_{\theta_1} - p_{\theta_2}\cos(\theta_1-\theta_2)}{ml^2[1+sin^2(\theta_1-\theta_2)]}\\ \dot{\theta}_2 &= \frac{-p_{\theta_1}\cos(\theta_1-\theta_2) + 2p_{\theta_2}}{ml^2[1+\sin^2(\theta_1-\theta_2)]}\end{aligned}$$

$$\begin{aligned}\dot{p}_{\theta_1} &= -2mgl\sin(\theta_1) - h_1 + h_2\sin[2(\theta_1-\theta_2)]\\ \dot{p}_{\theta_2} &= -mgl\sin(\theta_2) + h_1 - h_2\sin[2(\theta_1-\theta_2)],\end{aligned}$$

where

$$\begin{aligned}h_1 &= \frac{p_{\theta_1}p_{\theta_2}\sin(\theta_1-\theta_2)}{ml^2[1+\sin^2(\theta_1-\theta_2)]}\\ h_2 &= \frac{p_{\theta_1}^2 + 2p_{\theta_2}^2 - 2p_{\theta_1}p_{\theta_2}\cos(\theta_1-\theta_2)}{2l^2m[1+\sin^2(\theta_1-\theta_2)]^2}.\end{aligned}$$

These equations form a set of four first order differential equations which can be solved numerically using a Runge-Kutta method ([5]).

Chapter 4

Hamiltonian Neural Network (HNN)

As stated before, baseline neural networks fail at learning and respecting conservation laws. More precisely, they learn an approximation of these laws, whose imperfection causes them to drift over time to higher or lower energy states, as small errors accumulate. Conversely, Hamiltonian neural networks, which are based on Hamiltonian mechanics, aim and succeed at precisely conserve energy in a dynamic system. Furthermore, HNNs are effective in the way they deal with the problem of "discrete time steps" in the sense that follows. The study of a dynamic system consists of predicting the future states of the system, given the current one. However, BNNs typically discretise time in their approach, whereas time is actually continuous. Hamiltonian neural networks are more coherent with the real nature of time as they express the dynamics of a system as a set of differential equations and then integrate them from an initial state t_0 to a final state t_1. Let $S_H = \left(\frac{\partial H}{\partial p}, -\frac{\partial H}{\partial q}\right)$ be the time derivatives of the coordinates of the system, according to 3.3, which tells that moving the coordinates in the direction S_H gives the time evolution of the system. Then

$$(q_1, p_1) = (q_0, p_0) + \int_{t_0}^{t_1} S_H(q, p)\ dt, \tag{4.1}$$

where (q_1, p_1) are the coordinates at the final state t_1 and S_H is a vector field over the inputs of H, called *symplectic gradient*. Moving along the gradient of H, which points towards the direction of maximum growth, changes the output as quickly as possible, whereas moving in the direction of S_H keeps the output exactly constant. This approach applied to Hamiltonian mechanics is efficient in modelling the conservation of energy because the output is exactly the total energy of the system $E = H(q, p)$.

There are a few more relevant properties that should be mentioned. First and foremost, the strength of hamiltonian neural networks is the capability of handling systems with many degrees of freedom. In fact, these models are efficiently used to represent also complex systems such as cosmological ones. Moreover, it is possible to simulate changes in the levels of energy of a system by integrating along the gradient of H, as moving in this direction intuitively corresponds to adding energy. Finally, it is important to notice that all the trajectories are reversible in time, namely any mapping $(q_0, p_0) \to (q_1, p_1)$ is bijective.

4.1 HNN framework

In the HNNs approach a parametric function H_ϕ is learned, in order to exactly learn the conserved energy. This method works as follows: the neural network takes as an input a set of coordinates (p, q) and outputs a single scalar representing the energy of the system, which is the Hamiltonian H_ϕ of the model, given the coordinates. Afterwards, the gradient $S_H^\phi = (\frac{\partial H_\phi}{\partial p}, -\frac{\partial H_\phi}{\partial q})$ is computed before minimising the L_2 loss function, defined as

$$\mathcal{L}_{HNN} = \left|\left|\frac{\partial H_\phi}{\partial p} - \frac{\partial q}{\partial t}\right|\right|_2 + \left|\left|\frac{\partial H_\phi}{\partial q} + \frac{\partial p}{\partial t}\right|\right|_2.$$

Observe that, when possible, analytic time derivatives are used as targets, otherwise finite difference approximations are preferred. In other words, the objective is to find

$$\phi^\star = \arg\min_{\phi\in\Phi} \mathcal{L}_{HNN},$$

where Φ is the set of all possible parameters of the learning algorithm. Ultimately, $S_H^{\phi^\star}$ is integrated according to 4.1 in order to obtain the new state of the system.

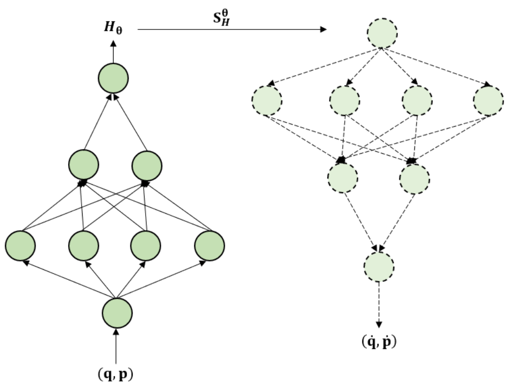

$\boldsymbol{H_\theta}$
$\mathbf{S}_H^{\theta}$
$(\mathbf{q}, \mathbf{p})$
$(\dot{\mathbf{q}}, \dot{\mathbf{p}})$

Figure 4.1: Illustration on the functioning of a HNN.

Eichelsdörfer et al. ([8]) proposed a new regularisation term for the loss function, based on the difference in energy levels of the training data. More precisely, this new term that is added to the loss function penalises the difference between the learned Hamiltonian H_ϕ and actual energy level of the system H. The true values of H have to be either collected or computed analytically only once for every initial condition in the training data, as it is a conserved quantity. Therefore, the novel loss function can be written as

$$\mathcal{L}_{HNN} = \left|\left|\frac{\partial H_\phi}{\partial p} - \frac{\partial q}{\partial t}\right|\right|_2 + \left|\left|\frac{\partial H_\phi}{\partial q} + \frac{\partial p}{\partial t}\right|\right|_2 + \lambda_H(H_\phi - H),$$

where λ_H is a hyperparameter which depends on the problem and whose optimal value has to be found with cross-validation.

Observe that the bigger λ_H the more energy levels different from reality are penalised, therefore the parameters of the model are changed in order to optimise the performance while also considering this discrepancy. This approach leads to an improvement of the long term accuracy and of the generalisation ability of the model. Generalising is the capability of the model to achieve accurate results on unseen data despite having been trained with few training observations. This aspect is extremely important since in the real world the generation of large amounts of data is tremendously expensive. In other words, the regularisation becomes fundamental in a so-called *small-data regime*, in which there are few training data available.

4.2 Double pendulum experiment

As anticipated before, the chaotic non-linear physical system represented by the double pendulum is studied as an application of the techniques discussed so far.

This example is presented in [8], where a neural network with two hidden layers consisting of 128 hidden nodes each is used together with a softplus activation function. However, starting from the code published by the authors of the paper, we implemented a model with 64 neurons in each hidden layer[1]. Starting from the equation of the motion, during training the system is simulated twice for 150 seconds for four distinct initial conditions. In the first case ten data points per second are collected, thus leading to a "large" dataset l (6000 observations), whereas in the other case one data point per second is collected in order to obtain a "small" dataset s (600 observations). On the other hand, during testing the system is simulated for 100 seconds from ten random initial conditions that were different from the ones in the training data. Ultimately, λ_H is set to the value of 0.2, which is said to be optimal by the author of the paper, who used cross-validation in order to tune this parameter.

Let ΔE be the absolute value of the discrepancy between the predicted and the actual energy. ΔE is written as its mean value, where the mean is with respect to the ten different initial positions, corrected by its standard deviation. Moreover, let E_l and E_s denote the energy of the system in the large and small dataset respectively.

In the following table, one can see the accuracy of the predictions made by the HNNs both in the standard and in the regularised framework. Although the performance of the regularised version on the large dataset is similar to the one of the standard HNN, it is clear that the regularised approach presents remarkably better results for the small dataset. Moreover, the standard HNN on the small dataset performs worse than the BNN on the large one. However, the baseline neural network diverges with a small number of training observations. Ultimately, it can be observed that the BNN achieves a maximum error on the large dataset which is approximately twenty and ten times bigger than the $\max \Delta E_l$ presented by the HNN and the regularised one respectively. Similar considerations hold for $\max \Delta E_s$. On the whole, the regularised approach proposed by Eichelsdörfer appears to outperform all the others.

[1]The reason for this choice is to achieve a less complex model, hence to avoid some potential overfitting, as well as reduce the computational time. In fact, we are interested in providing a rather immediate example, that simply gives the intuition behind the results of the various approaches.

Scheme	ΔE_l	$\max \Delta E_l$	ΔE_s	$\max \Delta E_s$
Baseline	13.49 $\pm$ 11.839	47.51	-	-
HNN	**0.517** $\pm$ 0.5098	2.287	33.69 $\pm$ 133.07	1867
HNN Reg.	0.577 $\pm$ 0.857	4.274	**4.517** $\pm$ 8.0907	88.61

Table 4.1: Numerical results from the simulation of the double pendulum system, representing the errors for networks trained on both the large and the small dataset. Values are reported as percentages of the maximum potential energy.

In pictures 4.2 and 4.3 one can observe the predicted states of the system plotted against the actual ones.

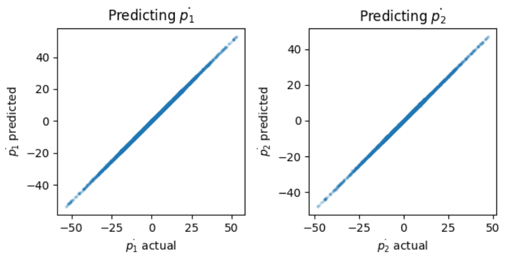


Figure 4.2: States of the system predicted by the HNN trained on the large dataset.

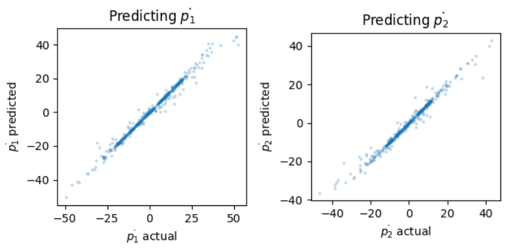


Figure 4.3: States of the system predicted by the regularised HNN trained on the small dataset.

4.2.1 Further development

Whilst examining the results obtained by the authors of the aforementioned paper, we thought of changing only the time on which the model was trained, in order to study the behaviour of the various models when trained on different time intervals. We asked ourselves whether training the model on less or more seconds, while keeping all the rest unchanged, could significantly affect the validation results on 100 seconds. Since everything else has been left unchanged, we have been able to make also a rough comparison of the computational time. Naturally, training the networks on a larger time interval required more time. In the following tables one can see the numerical results obtained with a dataset of 6000 and 600 observations collected in 15 seconds (first table), and 300 seconds (second table) respectively.

Scheme	ΔE_l	$\max \Delta E_l$	ΔE_s	$\max \Delta E_s$
Baseline	-	-	**42.92** $\pm$ 236.59	7405
HNN	**41.04** $\pm$ 56.654	218.5	-	-
HNN Reg.	254.4 $\pm$ 909.45	6360	-	-

Scheme	ΔE_l	$\max \Delta E_l$	ΔE_s	$\max \Delta E_s$
Baseline	16.03 $\pm$ 12.438	52.53	454.8 $\pm$ 681.34	4336
HNN	0.706 $\pm$ 0.9171	4.625	-	-
HNN Reg.	**0.373** $\pm$ 0.4927	2.949	**4.472** $\pm$ 5.5836	23.72

Table 4.2: Numerical results from the simulation of the double pendulum system, representing the errors for networks trained on both the large and the small dataset. Values are reported as percentages of the maximum potential energy.

We observed that errors in the approach with 15 seconds were extremely high, especially as far as the small dataset is concerned. Perhaps we might impute these results to the fact that it is hard for the trained model to correctly infer the dynamics of the system in 100 seconds, which are much larger than 15 seconds. In other words, this scenario affects the capability of the network to generalise. These aspects might be worth further empirical studies. On the other hand, the approach with 300 seconds proves to achieve better results on both the larger dataset and the smaller dataset with respect to the model trained on 150 seconds. Moreover, coherently with what has been observed above, the regularised version significantly outperform the standard HNN on the small dataset.

In the following pictures one can observe both the training and the validation losses of the model that performed best on either datasets, namely the regularised HNN trained for 300 seconds.

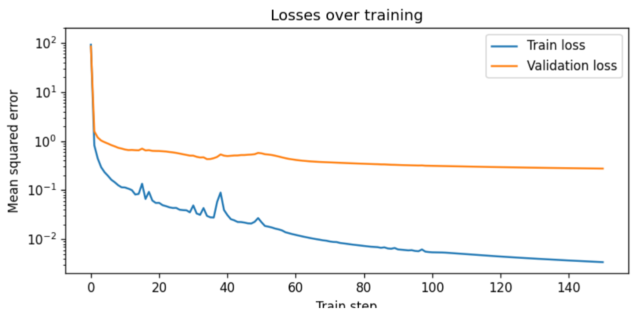


Figure 4.4: Train loss and validation loss for the regularised HNN trained on the large dataset for 300 seconds.

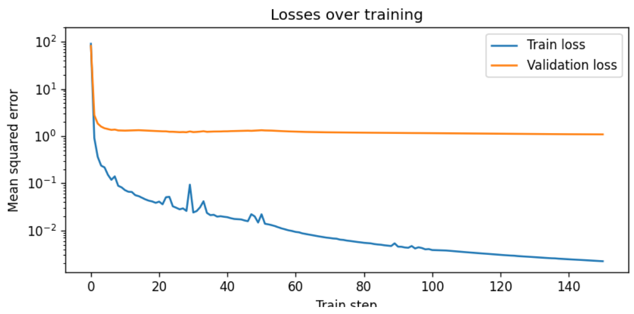


Figure 4.5: Train loss and validation loss for the regularised HNN trained on the small dataset for 300 seconds.

Chapter 5

Lagrangian Neural Network (LNN)

So far it has been shown that HNNs overcome the incapability of BNNs to mirror the true dynamics of the physical world, such as the conservation of energy. However, the canonical coordinates used by HNNs are likely to be unknown as well as really complex in their form, especially as far as canonical momenta are concerned. In fact, often it is really challenging to retrieve their analytical expression. Lagrangian neural networks improve this aspect, as they take as inputs arbitrary coordinates, as stated in chapter 4.

The LNN framework is similar to the one that has been studied for the HNN, with subtle differences. It is possible to model a generic classical physical system with the Lagrangian formalism in which the chosen coordinates at time t are $x_t = (q, \dot{q})$[1]. The dynamics of the system are described by the change in coordinates from a state x_0 to another state x_1. Observe that there are innumerable "paths" that could take the system from state x_0 to x_1. However, only one of these possibilities is actually physically practicable.

Definition 5.0.1. Let $(q_t, \dot{q}_t)$ be the coordinates of the system in the state x_t at time t. Let x_0 and x_1 be two consecutive states at time t_0 and t_1 respectively. The functional associated with every possible path between x_0 and x_1

$$S = \int_{t_0}^{t_1} \left(T(q_t, \dot{q}_t) - U(q_t)\right)\, dt$$

is called **action**, where T is the kinetic energy and U is the potential energy of the system. Observe that, similarly as before, since time is continuous the action is defined as an integral, hence LNNs solve the problem of time discretisation.

[1]Position and velocity.

The only accessible path for the dynamic system is the one corresponding to the stationary value of S, namely the one satisfying $\delta S = 0$. Recalling that the Lagrangian is defined as $L = T - U$, since a path q is a stationary point of S if and only if

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = 0 \qquad k = 1, \dots, n,$$

it is clear that this condition corresponds to the Euler-Lagrange equations 3.1, which describe the path of the system.

5.1 LNN framework

It is often infeasible to derive analytically the Euler-Lagrange equations from the analytical Lagrangian L, nevertheless a numerical expression for the dynamics of the system can be derived learning L with a neural network with four main steps. The first two are rather immediate and consist of obtaining data from a physical system and parametrising the Lagrangian ($L \equiv L_\phi$) with a neural network. The third step is more complex and requires to apply the Euler-Lagrange equations. Let

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} = \frac{\partial L}{\partial q_k}$$

be the Euler-Lagrange equation for $x_k = (q_k, \dot{q}_k)$. Then, for the chain rule[2] the left-hand side becomes

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} = \frac{\partial^2 L}{\partial q_j \partial \dot{q}_k}\frac{dq_j}{dt} + \frac{\partial^2 L}{\partial \dot{q}_j \partial \dot{q}_k}\frac{d\dot{q}_j}{dt} = \frac{\partial^2 L}{\partial q_j \partial \dot{q}_k}\dot{q}_j + \frac{\partial^2 L}{\partial \dot{q}_j \partial \dot{q}_k}\ddot{q}_j.$$

Considering the vectorised form on both the left and the right side, one would get

$$(\nabla_q \nabla_{\dot{q}}^\top L)\dot{q} + (\nabla_{\dot{q}} \nabla_{\dot{q}}^\top L)\ddot{q} = \nabla_q L$$

where $\nabla_q \nabla_{\dot{q}}^\top L$ and $\nabla_{\dot{q}} \nabla_{\dot{q}}^\top L$ are two matrices whose jk-th components are defined as

$$(\nabla_q \nabla_{\dot{q}}^\top L)_{jk} = \frac{\partial^2 L}{\partial q_j \partial \dot{q}_k} \qquad (\nabla_{\dot{q}} \nabla_{\dot{q}}^\top L)_{jk} = \frac{\partial^2 L}{\partial \dot{q}_j \partial \dot{q}_k}$$

[2]Let $y = f(u), u = f(x)$, then $\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$.

respectively. Hence, assuming that the matrix is invertible[3], namely det $\frac{\partial^2 L}{\partial q_j \partial \dot{q}_k} \neq 0$ and det $\frac{\partial^2 L}{\partial \dot{q}_j \partial \dot{q}_k} \neq 0$, the Euler-Lagrange equations can be written in the so-called *normal form*

$$\ddot{q} = (\nabla_{\dot{q}} \nabla_{\dot{q}}^\top L)^{-1} \big(\nabla_q L - (\nabla_q \nabla_{\dot{q}}^\top L) \dot{q} \big) \tag{5.1}$$

In conclusion, for a given set of coordinates $x_t = (q_t, \dot{q}_t)$, the equation above presents a method to compute $\ddot{q}_t$, that can be integrated in order to predict the dynamics of the system. However, note that this process requires a matrix inversion at each step, which might be computationally hard. The inversion of the Hessian matrix $(\nabla_{\dot{q}} \nabla_{\dot{q}}^\top L)$ scales as $\mathcal{O}(d^3)$, where d is the number of coordinates. Since the Hessian matrix needs to be calculated, the second order derivative of the activation function is crucial, therefore functions with non-zero second order derivatives are preferred. After computing the inverse, the last step consists of performing backpropagation in order to find the best parameters that minimise the loss function, thus approximating the true Lagrangian.

5.2 Double pendulum experiment

The framework is the same used for the HNN example, namely a neural network with two hidden layers with 64 hidden nodes each using the softplus activation function, whose second order derivative is non-zero.

$$\begin{aligned} \sigma_h(x) &= log(1 + e^x) \\ \sigma_h'(x) &= \frac{e^x}{1 + e^x} \\ \sigma_h''(x) &= \frac{e^x}{(1 + e^x)^2} \neq 0 \qquad \forall x. \end{aligned}$$

Once again the model is trained both on a large dataset of 6000 data points, obtained with ten observations per second, and on a small dataset, which was created by collecting one observation per second. Then the model was validated by simulating the trajectories of the system for 100 seconds from 10 initial conditions that were not contained in the training data. In the following table one can see the results achieved by both the BNN and the LNN approach.

[3]In practice the pseudo-inverse matrix is used, in order to avoid potentially singular matrices.

Scheme	ΔE_l	$\max \Delta E_l$	ΔE_s	$\max \Delta E_s$
Baseline	13.49 ± 11.839	47.51	-	-
LNN	$\mathbf{0.998} \pm 1.400$	7.445	-	-

Table 5.1: Numerical results from the simulation of the double pendulum system, representing the errors for networks trained on both the large and the small dataset. Values are reported as percentages of the maximum potential energy.

As expected, the LNN achieved better results than the BNN on the large data set, whereas it performed in an extremely poor way on the small data set[4]. By comparing the performances of the LNN with the ones of the HNNs in 4.1, one can see that the last approach managed to achieve better results. However, this comes at the cost of a more complex model that requires to use highly specific coordinates, which are often difficult to compute. Moreover, we trained the LNN on a smaller time interval of 15 seconds as well and we found out that it performed in a tremendously poor way. Furthermore, since the LNN approach is more computationally expensive because of the inversion of the Hessian matrix required at every step, it was not feasible to train the model on a time interval of 300 seconds.

In picture 5.1, it is interesting to study the behaviour of both the training and the validation losses of the BNN trained on the small dataset. Moreover, the predicted states of the system are plotted against the actual ones in 5.2.

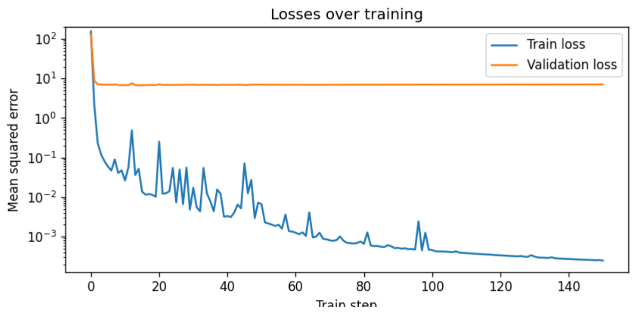


Figure 5.1: Train loss and validation loss for the BNN trained on the small dataset. Note that one can observe a situation of extreme overfitting, which might be the cause of the poor performance of the baseline approach.

[4]There is no regularisation term, which allowed the HNN to achieve good results on a small dataset.

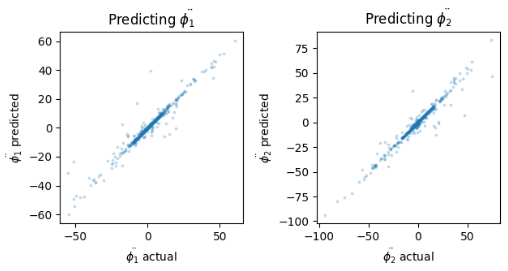


Figure 5.2: States of the system predicted by the BNN trained on the small dataset.

Chapter 6

Conclusions

On the whole, this paper presented different approaches to the problem of precisely describing the evolution of a dynamic system whilst conserving its energy. First and foremost, the Universal Approximation Theorem guarantees that neural networks are able to approximate any continuous function on a compact set arbitrarily well. In particular, this means that it is possible to learn the functions involved in the description of the physical systems considered in this paper. It has been shown that the Hamiltonian approach can be convenient whenever the canonical momenta are easily obtained. In addition, the loss function used for the HNN can be regularised with a term that explicitly embeds the discrepancy between the real and the predicted energy, thus leading to a significantly increased accuracy, especially with regards to small datasets. On the other hand, LNNs are to be preferred when the canonical momenta do not have a trivial expression, which makes them hard to compute. However, LNNs are computationally more demanding than HNNs, as they require a matrix inversion at each step.

6.1 Future works

In future works it could be interesting to analyse the problem of optimising the number of observations required and the frequency of collection of each data point in the fields in which these approaches might be used, such as celestial mechanics or quantum physics. Ultimately, it could be useful to experiment with different regularisation terms in the LNN case, in order to see if they would be of use when dealing with small datasets and frameworks with difficult canonical momenta.

Bibliography

[1] Shalev-Shwatrz S. and Ben-David S., *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press, 2014.

[2] Friedman J. H., Tibshirani R. and Hastie T. *Elements of Statistical Learning.* Springer, 2008.

[3] Cesa-Bianchi N. Machine Learning handouts for DSE Master's degree course, A.Y. 2021/2022. (https://cesa-bianchi.di.unimi.it/MSA/)

[4] James G., Witten D., Hastie T. and Tibshirani R. *An Introduction to Statistical Learning.* Springer, 2021.

[5] Butcher J.C. *A history of Runge-Kutta methods.* Applied Numerical Mathematics, Volume 20, Issue 3, 1996, Pages 247-260, 46(2):167–178, 1895.

[6] Cranmer M., Greydanus S., Hoyer S., Battaglia P., Spergel D., Ho S., *Lagrangian Neural Networks.* arXiv:2003.04630v2 [cs.LG] 30 Jul 2020.

[7] Greydanus S., Dizamba M., Yosinski J., *Hamiltonian Neural Networks.* arXiv:1906.01563 [cs.NE] 2019.

[8] Eichelsdörfer J., Kaltenbach S. and Koutsourelakis P. *Physics-enhanced Neural Networks in the Small Data Regime.* arXiv:2111.10329 (2019). (https://github.com/pkmtum/physics-enhanced_nn_smalldata)

[9] Cybenko G. *Approximation by superpositions of a sigmoidal function.* Mathematics of Control, Signals and Systems volume 2, pages303–314 (1989).

[10] Kidger P., Lyons T. *Universal Approximation with Deep Narrow Networks.* Proceedings of Machine Learning Research vol TBD:1–22, 2020.

[11] Lu Z., Pu H., Wang F., Hu Z. and Wang L. *The Expressive Power of Neural Networks: A View from the Width.* Advances in Neural Information Processing Systems 30, pages 6231–6239. Curran Associates, Inc., 2017.

[12] Lin H. and Jegelka S. *ResNet with one-neuron hidden layers is a Universal Approximator.* Advances in Neural Information Processing Systems 31, pages 6169–6178. Curran Associates, Inc., 2018.

[13] Hanin B. and Sellke M. *Approximating Continuous Functions by ReLU Nets of Minimal Width.* arXiv:1710.11278, 2017.

[14] Salmon, R. Hamiltonian fluid mechanics. Annual review of fluid mechanics, 20(1):225–256, 1988.

[15] Sakurai, J. J. and Commins, E. D. Modern quantum mechanics, revised edition. AAPT, 1995.

[16] Biscari P. *Introduzione alla meccanica razionale.* Springer, 2015.

[17] Notes from the lectures of *Sistemi dinamici e meccanica classica*, UNIMIB, A.Y. 2018/2019.