UNIVERSITÀ
DI SIENA
1240

University of Siena

Department of Information Engineering and
Mathematical Sciences

PhD in Information Engineering and Sciences – Cycle XXIX

Regularization and Learning in the temporal domain

Advisors:
Prof. Marco Gori
Prof. Marco Maggini

PhD Thesis by
Alessandro Rossi

Siena, May 2017

To my grandfather, missed during my Ph.D.

Acknowledgements

I would like to thank Fondazione Bruno Kessler, Trento - Italy, [1] the research center which provided the founding for my scholarship.

I would like to thank my advisors, Marco Gori for providing many inspiring ideas and Marco Maggini whom has been a constant reference point both from a personal and academic point of view.

I would like to thank my Master Thesis advisor Prof. Cristiano Bocci for the help he provided when starting this experience and Professors Duccio Papini and Paolo Nistri for helpful mathematical insights.

I would like to thank all my colleagues, in particular Stefano Melacci for his constant availability and Dario Zanca for insightful discussions about Physics.

Last but not least, I would like to thank my family and my parents for their patience, and in particular my sister Francesca and my partner Agnese for supporting me and bearing me during my work.

[1] http://www.fbk.eu

Abstract

The main proposals of this thesis concern the formulation of a new approach to classic Machine Learning optimization procedures, so as to inspire some insights about the recent rush to gold in most of the Artificial Intelligence applications.

Crucial aspects we would like to raise up can be associated to the current *thirst of data* that characterizes popular approaches, above all in Deep Learning. These issues are related to the improvements obtained by *Artificial Neural Networks* in last two decades, especially in their more complex forms exploiting *Convolutional* and *Recurrent* architectures. The high representational power of these kind of structures allows us to achieve state of the art results in most of the existent Machine Learning benchmarks, provided that a lot of labeled data are available in order to tune the large number of learnable parameters. All the information about the analyzed phenomena is assumed to be available at the beginning of the training and, usually, provided to the agent in a shuffled order to improve the final result. However, when looking to the fundamental ambition of Artificial Intelligence to simulate human behavior, it is straightforward to note that these problems have been embedded in a framework which is completely different from the biological environment. Indeed, living beings naturally exploit the local temporal coherence in the incoming information, relying on very few supervisions. Bearing these ideas in mind, one could regard the standard learning setting as an over–complication of reality, increasing the complexity of the computations.

We will introduce these concepts in the framework of Computer Vision, being the field in which we have been cultivating these ideas in last years. Moreover, because of the characteristics of the task, it seems (at least from our point of view) to outline more directly the correspondence among theoretical feelings and prac-

tical issues. However, our formulation, inspired by natural behaviors in biology, is related to general Regularization Methods for Machine Learning. As for other natural phenomena, we tried to provide a model described by Laws of Physics, taking into account the existent forces and the final goal of the problem. This approach allows us to recreate, analogously to the studies on systems of particles considered in Classical Mechanics, laws of motion that implement a regularization effect based on the temporal smoothness of the environment. The results came up to be a generalization of classical algorithms for discrete optimization, empirically reinforcing the soundness of the theory.

In Chapter 2 we introduce the theoretical formulation. Starting from the ideas first proposed in [1], the integration of the concept of *dissipation* is analyzed, even if the purely theoretical framework is fully described in [2]. The core of the contribution is indeed focused on the experimental analysis. We derived an algorithmic solution that allows a practical implementation and the possibility to deal with real tasks. This allows us to perform an effective analysis on the dependencies of the model related to its hyper-parameters. In this phase, an extensive experimentation was necessary [3] to validate the theory and reproduce the expected behaviors, working on the arrangement of the details so as to obtain the optimal mode of operation. We then explore applications to general learning structures [4], analyzing a parallel with standard optimization techniques, clearly interpreting the proposed approach as a generalization of classic methodologies. Inspired by the effects of the imposition of these kind of *temporal constraints*, in Chapter 3 we introduce a more abstract theoretical approach, producing a formulation of the learning procedure that strongly relies on the temporal dimension [5]. In this new analysis, the proposed regularization techniques are exploited in order to integrate these constraints in a structure [6] which is capable to extend the classic concepts of Laplacian regularization for semi-supervised learning to a purely on–line configuration.

Contents

Introduction

Motivations and purposes

The core ideas of this work were born within the scope of the project *Learning to see like children (LEASE)* [7], that came up with the assembly of a new kind of learning architecture named *Developmental Visual Agents* (DVAs) [8, 9]. This framework originated in the field of Computer Vision, with the final aim to provide a solution to semantic labeling for full scene understanding in an open world recognition setting [10]. The proposed task is intrinsically complex and represents one of the most studied open problems in Artificial Intelligence. More general motivation in this work were targeted to consolidate some theoretical aspects and to give a practical implementation to Machine Learning approaches driven by an understanding process that simulate natural behaviors inspired by living being in biology.

Even if the dissertation will only skim the Computer Vision field, we will briefly illustrate some steps just to sketch out the evolution of the ideas and to better understand the motivations which generated the theories analyzed in this work.

A first fundamental remark has to be done in order to underline the discrimination between the general semantic scene understanding problem and task oriented solutions concerning the processing of data represented by images or videos. In Figure 1(a) we reported the outcome of an attempt to full scene description obtained with a *DVA* agent[2] on a frame from the *CamVid*[3] database [11],

[2] `http://dva.diism.unisi.it/`
[3] `http://mi.eng.cam.ac.uk/research/projects/VideoRec/CamVid/`

(a) (b)

Figure 1: (a) An example of full scene description outcome taken from the *DVA* site. (b) Samples of 3 digits taken from the MNIST dataset.

providing different labeled videos from the perspective of a moving automobile. In general, the complexity of the task is quite clear, since one has to deal with variability and noise that are difficult to cope with, sometimes even for a human. Moreover, the input is often provided to an agent by RGB channels expressed at the pixel level, without any global information about the scene. To provide a clear comparison with the other perspective, concerning specific tasks as images classification, objects detection and so on, in Figure 1(b) we show three samples of handwritten digits from the MNIST dataset[4], each one to be assigned to the correspondent class. In spite of the apparent triviality, the variability of the provided patterns can introduce some tricky configurations. However, engineering approaches allow the development of a wide variety of techniques to solve this kind of tasks, providing very efficient operative solutions to real-world problems as for example in Object Detection [12] and Tracking [13], Video Surveillance or Medical Image Analysis [14].

Despite of the achieved results, higher ambitions were raised at the birth of Computer Vision with the final goal of full scene understanding in open world scenarios. However, the very first steps were characterized by a great underestimation of the actual problem. Indeed, at the first glance, the continuity of the

[4] `http://yann.lecun.com/exdb/mnist/`

interaction with our visual system hides the intrinsic complexity of reproducing a quite common activity that happens with great naturalness. Until one does not focus on the details of the specific problem, it is difficult to appreciate how the semantics associated with representational concepts makes our perceptual system able to recognize an object in different situations and shapes, regardless of transformations (in shape, color and movements for example), changes of light, possible occlusions and so on. However, the representation of these kind of invariances becomes a crucial issue in the Computer Vision field and, despite of the progress done, the solution of this general problem seems far away, especially concerning methods trying to reproduce human behavior [15].

The first anecdotal event which can be linked to this field was in 1966, when Marvin Minsky *"hired a first-year undergraduate student and assigned him a problem to solve over the summer: connect a television camera to a computer and get the machine to describe what it sees"* [16]. Whether or not Minsky was aware of the ambiguity of the assigned task (and neglecting the misfortune of the student addressed to solve a such riddle), we can locate the beginning of Computer Vision in those years. Since then, extensive studies have been proposed. In the first years, many hand-crafted approaches were developed, based on modeling the skeleton of the objects (Constellation Models [17]), detecting the edges [18] or finding a representation partitioned in the main components [19, 20]. The progress on the studies on living being visual systems [21, 22] inspired the formulation of the *Neocognitron* [23], the first algorithm which tried to reproduce the principles of humans vision. This led, after few years, at the first systematic work on Convolutional Neural Networks [24], introducing the first break through in Handwritten Character Recognition (specifically regarding the task proposed in Figure 1(b)). These advances also contributed to the consolidation of a general common structure among Computer Vision algorithms, composed by a *feature extraction* part and a label-based *classifier*. Whereas general Machine Learning algorithms have been used for the second task, *Support Vector Machines (SVMs)* or *Artificial Neural Networks (ANNs)* in the simplest cases, many specific methods had been developed in order to detect discriminant features, as for example *Scale Invariant Feature Transform (SIFT)* [25], *Histogram of Oriented Gradients (HOG)* [26] and *Speeded-Up Robust Features (SURF)* [27]. The increase in computational capabilities and the improvement of hardware and software envi-

ronments led to an extensive employment of *Parallel Computing* and *Graphics Processing Units* (GPUs), producing the nowadays explosion of Deep Learning, i.e. of architectures structured in several representational layers and millions of learnable parameters. This success is also due to the introduction of new optimization techniques based on unsupervised pre-training [28], exploiting *Random Boltzmann Machines (RBM)* [29], *Stacked Auto–Encoder* [30] and similar methods.

A common feature of these kinds of architectures is the need of a large number of labeled samples in the supervised fine–tuning phase, because of the huge amount of free parameters, allowing us to reach top results in image classification [31, 32], as well as in other fields such as Speech Recognition [33] and Natural Language Processing [34]. The enormous representational power of these structures makes them capable to learn, in some way, the majority of patterns which are statistically well described in the input data distribution. Hence, trying to provide data that contain, with a meaningful statistical evidence, the largest amount of possible transformations, the scientific efforts focused on benchmark-oriented approaches and on the generation of lager and lager dataset, as for example *ImageNet* [31], *Microsoft COCO* [35] and *OpenImage* [36].

In spite of the achieved performances, some criticism has been shown on these methodologies [37]. Particular attention is paid to the importance of finding a feature representation that is suitable for the goal [38]. Some works highlight how, in spite of the variety of the provided cases, the acquired patterns are still related to the presented data. In Figure 2 we report an example from [39], where we can see how a small random perturbation, imperceptible to our sight, could substantially compromise the prediction quality in these learning configurations.

These issues are the core motivations of the proposal of the project *LEASE*. Indeed, when considering the biological approach to vision and the studies on the evolution of children behavior in understanding [40], it is easy to see the relevance of the temporal order of data, both to propagate information at a local level and to present more difficult concepts as time goes by [41, 42]. Despite of the fact that nowadays learning structures does not have the same complexity of the living beings brain, which is able to acquire and process information from different sources, common approaches, processing a lot of labeled data in a shuffled order,

Figure 2: In the middle column of the two figures we can see the noise added to the left column images to obtain its correspondent in the right ones. Even if the images appear unchanged, the reported algorithm classifies correctly the left ones, whereas the right ones are all assigned to the class *ostrich* with good confidence.

could appear as an over complication of the problem. Previous work inspired by biological principles [43, 44] and on the architecture of the human visual cortex [45, 46] are available in the literature. However, the approach proposed by *DVA* [47, 48, 49] is, to the best of our knowledge, the first in this field that tries to overcome the standard Training–Test configuration, by transposing the learning process into a life–long scenario learning and predicting data on–line. The agent is assumed to grow in an environment where a continuous stream of data is available, but paired with very few labels, so as that the extraction of information should rely substantially on unsupervised methods [50].

As already said, one of the most important characteristics of this framework is the assumption of temporal smoothness in the evolution of the data sequence. To give an idea of this, it could be useful to think at how vision works in reptiles, for which the moving object tracking has a key role. The core of this work aims at embedding these kind of information into the learning process, trying to formulate a model that is able to represent similar concepts by means of meaningful constraints.

In [51] we can find one of the first work trying to reproduce similar ideas, exploiting the correlations among consecutive instances by maximizing the Mu-

tual Information. In [52], a coherence constraint is implemented directly on the predictions between adjacent samples, whereas the complete exploitation of the output is imposed by the Entropy maximization overall the outputs. Analogous constraints are applied in [53] and [54] within the feature extraction process, or in [55] on the activation of pooling units among consecutive frames.

The previous work is still bound to the batch configuration, whereas our work completely focuses on the on–line setting, trying to formulate the learning as a smooth process in time, modulating the interaction between the agent and its own environment in real time. Nevertheless, it could be plausible that, in practice, such assumptions could not yields significant benefits for Machine Learning algorithms that, as a matter of facts, represent a mathematical instrument for function approximation.

From a theoretical point of view, the aim of this work is more conceptual. Indeed, we propose a study on a general formulation of the whole learning process described by Physics laws modeled by a reinterpretation of the *Principle of Least Action*, taking inspiration from the classic theory of Regularization Networks. These ideas should produce a learning procedure which is devised as a truly temporal process, providing a flexible approach that can be suitable for different configurations, depending on the specific needs. In this way, one should be able to efficiently deal with typical issues of life-long learning, where data are not available at the beginning of the training and the number of incoming samples rapidly becomes difficult to be managed and store in memory. Moreover, the approach with differential operators in the temporal domain, that allow us to derived a closed form of the solution, should represents a general method to directly include different kinds of constraints, which can be learned in an unsupervised way and can extract intrinsic properties from data. The aim of the work is then not only related to providing a regularization method for on–line learning which is capable to exploit more global informations, but also to introduce a new way to integrate additional knowledge in the learning process, inspired to principles from Physics and Biology.

Main Contributions

A already said, the main ideas have been already introduced in [1], whereas in [2] they provided a complete integration of the concept of dissipation. However, the experimental analysis of this work contributed to a first actual validation and also some theoretical aspects have been explored and consolidated during this process. We provide the closed form of the solution and, in particular, its discretized version, allowing us a feasible implementation, avoiding an overload of memory or numerical approximations. The study of the stability issues produces empirical results on the conditions that regulate the behavior of the model and, hence, the convergence to suitable solutions. The application to existent learning structure, and in particular to Artificial Neural Networks, allowed us to produce results comparable with standard procedures and to provide a further consolidation of the theory as a generalization of the standard *Stochastic Gradient Descent* case. In this phase, an intensive experimentation has been necessary to find hyper-parameters configurations solving well known issues about ANNs training.

The obtained results inspired the formulation of Chapter 3, for which a similar experimental analysis has been explored. We also provide the integration of the obtained dynamic system as an additional regularizer to an incremental learning structure, merging standard regularization techniques on the spatial input domain with the proposed approach. Even if similar agents have been already developed in the framework of Laplacian manifold regularization [56, 57, 58], our formulation generate an optimization procedure that is completely on–line, adapting the response and the memory to the specific problem, allowing to deal with infinite stream of data and easily adaptable to multi-class problems.

List of Publications

Journals

- Gori M., Maggini M., Rossi A, *Neural Network Training as a Dissipative Process*, Neural Networks, 2016

International Conferences

- Maggini M., Rossi A., *On–line Learning on Temporal Manifolds*, 15th International Conference of the Italian Association for Artificial Intelligence (AI*IA), 2016

Technical Reports

- Giannini F., Laveglia V., Rossi A., Zanca D., Zugarini A., *Neural Networks for Beginners. A fast implementation in Matlab, Torch, Tensorflow*, arXiv preprint 2017
- Gori M.,Maggini M., Rossi A., *The principle of cognitive action – A preliminary experimental analysis*, arXiv preprint, 2017
- Gori M.,Maggini M., Rossi A., *Collapsing of dimensionality*, arXiv preprint, 2017

1 Background Literature

1.1 The Principle of Least Action

A first formulation of the concept of Action was introduced by Maupertuis (1744) (see [59]) with the aim of motivating and unifying Physics laws proposed in various fields (Mechanics, Optics, etc.) under the principle that *Nature acts as simply as possible*. After about a century, Hamilton provided a more rigorous and general formulation, related to variational principles. This leads to the modern notions which form the basis of Hamilton Mechanics [60] describing laws of motions and continuous systems. The problem is formulated as the search of the stationary points of a cost functional of the system trajectory, composed in relation to the existent forces. Given a spatio-temporal domain and the description in terms of the Lagrangian coordinate $q(t)$, the *Hamilton Action* S is defined by the integral along the trajectory connecting two events $(t_s, q(t_s))$ and $(t_e, q(t_e))$:

$$S = \int_{t_s}^{t_e} L(t, q, \dot{q})\, dt \tag{1.1}$$

where $\dot{q} = dq(t)/dt$ and L represent the Lagrangian, which in the simplest case is given by:

$$L = K - V \tag{1.2}$$

where K and V indicate the Kinetic and the Potential energies respectively. The Hamilton's principles states that the true trajectories are those that make S stationary, and hence must satisfy the condition:

$$\delta S = 0\,. \tag{1.3}$$

The first variation δS introduced is obtained when operating a small perturbation $\delta q(t)$ on the true trajectory, with fixed endpoints, which gives:

$$S[q(t) + \delta q(t)] = S[q(t)] + \delta S + \delta^2 S + \dots \tag{1.4}$$

When approximating the variation to first order, the stationarity condition comes clearly by satisfying the (1.3). However, further information on the nature of critical points depends on the sign of the second variation $\delta^2 S$. The stationarity can be a minimum ($\delta^2 S > 0$) or a saddle point, but never a maximum [61]. This issue depends on the notation used to define the Lagrangian in (1.2). Indeed, the same problem can be defined as the search of a maximum by a sign flip in (1.2) (we will analyze some practical issues in Section 2.3). However, with this formulation, it is possible to obtain a minimum for the action only when the trajectory is defined for a sufficiently short interval of time [59], whereas in general the stationarity is a saddle point. Within the framework of the Calculus of Variations [62], the condition (1.3) leads to the *Euler-Lagrange equations of motion*:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = 0 \tag{1.5}$$

that generate a second-order differential equation (a system of equations when dealing with systems of particles) if the Lagrangian contains derivatives of q up to first order. The Euler-Lagrange equations in (1.5) are classically expressed as a boundary value problem at the points $q_s = q(t_s)$ and $q_e = q(t_e)$, that provides the conditions to determine a unique solution from the set of the admissible ones. Nevertheless, when analyzing dynamic systems it is more common to have available the initial conditions and, given that the Hamiltonian principle assumes a prescribed travel time, it is possible to impose the conditions at t_s and find the solution of the correspondent boundary value problem.

The classic formulation introduced is developed under the assumption of conservative forces, in which the Lagrangian does not depend explicitly on time and could be written as $L(q, \dot{q})$. However, all the statements can be extended when introducing the dependencies due to a time-variant Lagrangian, expressed by the explicit dependence on time in the potential energy $V(q, t)$ [59]. Even the generalization to the case in which higher order derivatives appear in the Lagrangian can be introduced in a natural way [63]. In addition, in our work we are particu-

larly interested in the management of dissipative forces. In this case the existence of the Lagrangian is not always guaranteed [64], but the formulation and further analysis for our specific case is developed in [2].

1.2 Regularization Networks

Regularization Theory plays a central role in Machine Learning and Pattern Recognition, especially when dealing with very complex algorithms and problems. In [65] Poggio and Girosi provided a nice formulation of Supervised Learning proposing a new paradigm, called Regularization Networks, following a parallel with standard Approximation Theory. In this framework the problem of learning an input–output mapping from a set of data is cast as the approximation of a multidimensional continuos non-linear function. Let us consider the case in which data are described by a set of labeled points organized in a Training Set $\mathcal{L} = \{(x_k, y_k) \in \mathbb{R}^d \times \mathbb{R}\,,\ k = 1, \ldots, N\}$. The function $f : \mathbb{R}^d \to \mathbb{R}$ often depends on a fixed number of parameters, which have to be learned in order to find a suitable approximation. A fundamental issue is represented by the requirement of *generalization*, i.e. to find a solution which is still a good approximation in those regions of space in which data are not available during the training phase. Moreover, collected data could often be affected by different kinds of noise, making it counter-productive to build up solutions which work well only on the provided samples, leading for example to over–fitting phenomena. In order to increase the amount of information, it seems reasonable to exploit the common redundancy of the world. This means that usually, in real problems, information does not come completely at random, but follows a certain degree of coherence in a neighborhood. This suggests the introduction of an *a-priori* smoothness assumption, which seems extremely sound if we look for example at continuous hyper-surface reconstruction problems, characterized by the fact that small changes in the input should produce small changes in the output. Similar approaches are well known in Standard Regularization, in which the smoothness soft-constraint is included in the optimization within the framework of the Calculus of Variations. The problem is formulated as the minimization of a cost functional composed by

two terms [66, 67, 68], that can be written for example as:

$$\Phi(f) = \gamma \|Pf\|^2 + \frac{1}{2}\sum_{k=1}^{N}(f(x_k) - y_k)^2 \,. \tag{1.6}$$

The second term measures the distance of the predictions from the Training data, in this case by exploiting a quadratic loss function. The term $\|Pf\|^2$ embodies the assumed degree of smoothness. P is a general differential operator, whereas $\|\cdot\|$ could be for example the L^2 norm. The coefficient γ is a (positive) regularization parameter balancing the quality of interpolation on the given data with respect to the smoothness requirement. In the simplest case, the differential operator P, used to define the regularization term, is the gradient, but they also considered a more general class of operators based on high-order derivatives depending on some kind of hypothesis about the problem. The solution of (1.6) is obtained by satisfying the correspondent Euler-Lagrange equations:

$$\hat{P}Pf(x) = \frac{1}{\gamma}\sum_{k=1}^{N}(y_k - f(x))\delta(x - x_k), \tag{1.7}$$

where $\hat{P}$ is the formal adjoint operator of P. The equation is developed within the Theory of Distributions [69] and introduces the presence of the *Dirac's Delta Function* δ, allowing us to write the solution exploiting the *Green's Function* [70] of the operator $L = \hat{P}P$, defined as its response to the Dirac's Delta Function[1]. Because of the translation property of δ, it is easy to write the solution as a superposition of Green's Functions centered at the supervised points x_k:

$$f(x) = \frac{1}{\gamma}\sum_{k=1}^{N}(y_k - f(x_k))G(x; x_k) \,. \tag{1.8}$$

This solution lies in the N-dimensional subspace generated from the function $G(x; x_k)$ and it is written neglecting a term from the null space of L. The form of this term depends on the choice of P and of the boundary conditions. However, all the functions belonging to the kernel of P do not affect the value of the functional in (1.6) and then can be neglected in this analysis. The solution in (1.8) can be determined by determining the coefficients $c_k = (y_k - f(x_k))/\gamma$ by solving

[1] If G is the Green's Function for L then $LG = \delta$ must hold.

the linear system:

$$(\mathbf{G} + \gamma \mathbf{I})\mathbf{c} = \mathbf{y} \tag{1.9}$$

where $\mathbf{I}$ is the identity matrix, $(\mathbf{G})_{kj} = G(x_k; x_j)$, $(\mathbf{c})_k = c_k$ and $(\mathbf{y}_k) = y_k$, with $k, j = 1, \ldots, N$. The resolution depends on the characteristics of G, which is often symmetric and positive definite, and can be considered as a Kernel, linking the (1.8) to the classic representer theorem of kernel machines [71, 72].

Needless to say, the related mathematical and algorithmic framework has played a crucial role in the development of the field of Machine Learning in the last twenty years.

Recently, equation (1.7) has been reformulated in a more general framework in which, instead of dealing with intelligent agents that interact with the environment only by supervised examples, their behavior is optimized also to satisfy a given set of constraints [73]. New representer theorems are given to express the optimal solution for a large class of constraints. Some guidelines were provided to parallel the plain kernel-based solution of (1.7) with cases where special kernels arise from the marriage of regularization operators with specific constraints [74]. A classic example of constraint studied in [73] is the one which imposes the brightness invariance in computer vision, which makes possible to estimate the optical flow [75]. Interestingly, the kernel-based approaches used so far, along with their mathematical apparatus, are clearly the only viable solution. As a matter of fact, the differential equation (1.7), as well as the related Euler-Lagrange equations derived in [73], cannot be tackled with an efficient numerical solution. They are in fact formulated in the features space, whose dimension makes it unfeasible to grid the space for the classic discretization of the differential operator[2] $L = \hat{P}P$. This fundamental computational issue suggests that any biological system involved in learning processes should obey to models which somehow circumvent the *curse of dimensionality* related to equation (1.7).

These ideas drive the formulation analyzed in this work, looking at the process of learning as a dynamic system and introducing new kind of regularizers which can be interpret as Temporal Kernels.

[2]Methods like Runge-Kutta have a unmanageable complexity for the dimensions of interest in Machine Learning.

2 Learning as a Temporal Dissipative Process

In this Chapter we analyze the transposition of the Principle of Least Action introduced in Section 1.1 in its temporal counter part. The model, inspired by the developmental process of biological systems, is proposed as driving principle for a dynamical system simulating the learning process of an agent in an on-line setting, as analyzed in [4, 3]. Such agent is modeled by a set of parameters, that evolve according to the information acquired from the environment. These concepts can be naturally interpreted in a framework similar to the one introduced in Section 1.2, by performing the regularization process in the temporal domain. Indeed, from our point of view, it seems reasonable to extend the local redundancy assumption in the spatial input space, to the events in a temporal sequence of a real world scenario. These hypothesis are, as a matter of fact, a fundamental requirement for a living being to develop a meaningful interaction with its own environment. The general idea is that this temporal regularization, together with the coherence of the input data sequences, should allow the agent to yield a smooth representation on the correspondent spatial domain. The process consists of developing refined configurations through time, exploiting the natural evolvution of data with an essential reduction of the number of constraints.

2.1 The Principle of Least Cognitive Action

Since we want to model the behavior of an agent processing an input evolving in a feature space in $\mathbb{R}^d$, it is convenient to describe data by a function of time

Variable	Learning Theory	Mechanics
w	Weight	Particle Lagrangian coordinate
Pw	Weight variation	Generalized velocity
μ	Weight mass	Particle mass
$K(Pw)$	Temporal smoothness	Generalized kinetic energy
$V(w)$	Loss function	Potential energy
$F(t, w, Pw)$	Cognitive Lagrangian	Mechanical Lagrangian
$A[w]$	Cognitive Action	Mechanical Action

Table 2.1: Links between learning on temporal manifolds and classical mechanics.

$u : T \to \mathbb{R}^d,\ T \subseteq [0, +\infty)$. Hence, the agent can be represented as a mapping to the output $z \in \mathbb{R}^m$ by means of a representation function f such that $z(t) = f(w(t), u(t))$. In a quite general case, we can assume $T = [0, +\infty)$, simulating a real life agent starting to live and learn at $t = 0$ and then evolving in an on-line never-ending setting. Even though this hypothesis is consistent, to give a clear formulation according to the one proposed in Section 1.1, it is convenient to analyze the problem in a finite interval of time $T = [\, 0, t_e \,]$, assuming that the information eventually becomes redundant after a finite period. This periodicity assumption allows us to restrict the initial analysis to a finite set of data, even if we will see that this restriction is not necessary in practice. For now, we do not impose any constraint on the structure of f, which, for example, could be implemented by a feedforward neural network, whose weights are stacked into the vector $w(t)$ at time t. The vector $w(t)$ represents, in general, the parameters modeling the behavior of the agent, which in fact will be considered as the particles of the dynamical system to be described. Hence, from a Machine Learning point of view, the task is to learn the weights $w(t) \in \mathbb{R}^n$ giving the best representation for the function f. We formulate this online learning process by providing the constraints that define the interactions with the environment, according to the general framework defined in [73]. A quite general case is the one in which f should satisfy the equation $\phi(\ f(\, w(t),\, u(t)\,)\ ) = 0$, where $\phi(\,\cdot\,) \geq 0$ is the function modeling the information from the agent's environment. The fulfillment of the constraints during the learning process can be enforced by the minimizing the

penalty:

$$V(w) = \phi(\ f(\ w(t),\ u(t)\ )\ ). \tag{2.1}$$

In this work, we will restrict the analysis to the classic case of supervised learning. Supervisions are provided by an external teacher at discrete time instants t_k by specifying a target value y_k, where $k = 1, \ldots, N$ and the instants are sorted in increasing order so as that $0 < t_1 < \cdots < t_k < \cdots < t_N < t_e$. The supervised pairs $(\ u(t_k),\ y_k\ )$ are collected into the set $\mathcal{L} = \{\ (\ u(t_k),\ y_k\ )\ \}_{k=1}^{N}$ and are provided while the agent's input $u(t)$ evolves in time. This formulation can be incorporated into this framework by posing:

$$V(w) = \sum_{k=1}^{N} \mathcal{V}(\ y_k,\ f(w(t), u(t))\ ) \tag{2.2}$$

where $\mathcal{V}(y, s)$ is a general loss function[1]. We can look at V as the potential energy[2] connected with the constraint ϕ, such that the aim of learning is to develop configurations of w with small potential. By following a parallel with Classical Mechanics, the weights w can be thought of as the *Lagrangian coordinates* in a virtual mechanical system, whose kinetic energy is defined as:

$$K = \sum_{i=1}^{n} \mu_i \dot{w}_i^2. \tag{2.3}$$

The value $\mu_i, i = 1, \ldots, n$, represents the mass of the particle associated with w_i. Taking inspiration from equation (1.6), we may generalize the concept of velocity by means of differential operator:

$$P = \sum_{j=0}^{\ell} \alpha_j \frac{d^j}{dt^j}. \tag{2.4}$$

The coefficients α_j of P are the regularization weights related to the correspondent order of derivative applied to $w_i(t)$, where we consider $\alpha_j > 0,\ j = 0, \ldots, \ell$. The generalized velocity turns out to be Pw and the correspondent kinetic energy is:

$$K(Pw) = \sum_{i=1}^{n} \mu_i (Pw_i)^2. \tag{2.5}$$

[1] For instance, $\mathcal{V}(y, s)$ can be the quadratic loss $\mathcal{V}(y, s) = \frac{1}{2}(y - s)^2$.

[2] When clear from the context, we will drop the dependencies to unload the notation.

Now we can provide a formulation which resembles the Principle of Least Action in Physics by defining the Lagrangian:

$$F(\, w,\, Pw\,) = K(Pw) + \gamma V(w), \tag{2.6}$$

maintaining the split into the kinetic and the potential energies, respectively. We introduced a *regularization parameter* γ similar to the one in (1.6), that, in classical Machine Learning, balances the smoothness requirement on f with respect to the satisfaction of the constraints. Indeed, while the potential energy represents the environment constraints, the kinetic energy impose, via μ_i and α_j, a uniformity on the learning process along the input trajectory. Because of the presence of these parameters in K, balancing its regularization effect, we can reduce the study to $\gamma = \pm 1$. Notice that in the Lagrangian of (1.2), coming from the formulation proposed in Section 1.1, we have $\gamma = -1$, while here we also assume $\gamma = 1$ depending on the choice of the operator P, as shown in Section 2.3. The learning process, defined on the horizon $[\, 0,\, t_e\,]$, consists of determining the optimal value of w in a given functional space $\mathcal{F}$ by:

$$w^* \;=\; \underset{w \,\in\, \mathcal{F}}{argmin} \quad A[w] \tag{2.7}$$

where $A[w]$ is obtained by the composition of (2.6) with a dissipation function $\psi(t)$, introducing the explicit dependence on time:

$$A[w] = \int_0^{t_e} \psi(t) \cdot F(\, w(t),\, Pw(t)\,)\; dt \tag{2.8}$$

even if we write $A[w]$ instead of $A(\, t,\, w(t),\, Pw(t)\,)$. The functional in (2.8) is referred to as the *Cognitive Action*, accomplishing the parallel with the formulation in Mechanics summarized in Table 2.1. The introduction of a positive monotonically increasing function $\psi(t)$ has an important impact on energy dissipation. It is worth mentioning that the idea of weighing the Lagrangian by a temporally growing exponential term has already been explored in studies on dissipative Hamiltonian systems [76]. A clear interpretation of the concept of dissipation in this framework and the relative formulation of the Lagrangian in (2.6), together with the correspondent action principle, are shown in [2]. Intuitively, we can interpret $\psi(t)$ as a temporal weight, assigning different importance to the

constraints inside $[\,0,\, t_e\,]$ (the latest events are more important). A convenient choice for the dissipation function, which will be assumed in the remainder, is $\psi(t) = e^{\theta t}$. In this case, the parameter $\theta > 0$ measures the dissipation and, as we will see later, the memory of the system. Indeed, the more the weight $\psi(t)$ grows rapidly in time, the more the agent becomes interested in the latest events seen, forgetting, when in contrast with the new data, what learned in the past.

As already said, the formulation of (2.7) as a minimization problem is coherent with standard Machine Learning approaches, even if, by following the Hamilton principle, the incoming solution will satisfy only the condition for stationarity, without any guarantees on the quality of the determined critical point. However, in many Machine Learning and Optimization problems could in general not be straightforward to guarantee the conditions for a solution to be a minimum. In fact, a local minima or a saddle point is usually a good deal. To prove the soundness of this approach, in the following we will experimentally show that the results obtained by this solution are not too far from the standard ones achieved by discrete optimization methods. Indeed, under particular configurations, we find out the proposed formulation as an almost pure generalization of classic procedures.

2.2 Euler-Lagrange Equations

Stationary conditions for the functional in (2.8) can be derived by the Eulero-Lagrange equations, in a similar way to the case introduced in (1.1) and (1.6). In our case it can be easily shown (see Section A.1) that any stationary point of $A[w]$ must satisfy the system of differential equations:

$$\hat{P}(\psi P w_i) + \frac{\gamma}{\mu_i} V_{w_i} = 0, \quad i = 1, ..., n, \tag{2.9}$$

where V_{w_i} represents the partial derivative of the potential V in the (2.2) with respect to the variable w_i. The operator $\hat{P}$ introduced as the formal adjoint of P can be written as:

$$\hat{P} = \sum_{j=0}^{\ell} (-1)^j \alpha_j D^j$$

by posing $D^j = \frac{d^j}{dt^j}$. Notice that, unlike equation (1.7), which is defined in the feature space, equation (2.9) represents a system of ordinary differential equations describing the evolution of the parameters over time. The underlying assumption is that the generation of the features by means of the function $u(t)$ evolves on a temporal manifold, allowing us to convert equation (1.7) into a temporal process for learning. The solutions, developed in the framework of the Theory of Distributions, can be written as:

$$w_i(t) = w_i^o(t) + \eta(i, \ell) \sum_{k=1}^{N} \zeta_{i,k} \cdot G(t; t_k) \ , \quad i = 1, ..., n. \tag{2.10}$$

The second term of the summation is the counterpart of the solution proposed in (1.8), where again $G(\cdot)$ represents the Green's Function of the differential operator Q such that $Qw = \hat{P}(\psi P w)$, for which $QG(t) = \delta(t)$ holds. Interestingly, the Green's function can be viewed as a *temporal kernel*, since it closely resembles classic kernels in the features space [71, 72]. Indeed, this time the notation $G(t; t_k)$ expresses the fact that the function G is centered around the instant t_k in which the correspondent supervision is placed. Within the standard formulation, the obtained kernel G has the well-known properties of positivity and symmetry. However, the transposition of the formulation in the temporal domain, introduces the concept of *causality*. Indeed, in the framework of Circuit Systems and Control Theory [77], the function G is referred to as the *Impulse Response* (and is usually indicated by $g(t)$, as we will do in the remainder underlying the difference with the spatial Green's Function G), since its shape defines the reaction in time of the system to an impulse defined by the Dirac's Delta function $\delta(t)$. The oriented evolution of the temporal dimension and the outcoming concept of *causality* impose the response to be considered null for $t < 0$. This condition, dropping the symmetry property, is coherent with the on–line learning setting, since each supervision does not affect previous predictions (i.e. $w(t)$ depends on the pairs $(u(t_k), y_k)$ only for $t > t_k$). Since we restricted the analysis to the case of a linear differential operator of general order ℓ, the equation in (2.9) consists in a linear differential equation of order 2ℓ, with constant coefficients. The ob-

tained impulse response $g(t)$, calculated by the Laplace Transform, is given by by a summation of exponentials as:

$$g(t) = \sum_{q=1}^{2\ell} c_q e^{\lambda_q t} \tag{2.11}$$

where the λ_q, $q = 1, \ldots, 2\ell$ are the roots of the characteristic polynomial associated to (2.9) and the coefficients c_q are set so as to yield a unitary Dirac Delta function when computing $Qg(t)$. To complete the description of (2.10), the term $\zeta_{i,k}$ represents the partial derivative of the penalty function $\mathcal{V}$ with respect to w_i, evaluated at t_k. The term $\eta(i,l) = (-1)^{\ell+1}\gamma/\mu_i\alpha_\ell^2$ is a constant which depends on the chosen order of the differential operator (as we will see later), leading to the introduced sign-flip issues. The first term $w_i^0(t)$ represents the solution to the homogenous equation associated to (2.9), belonging to the kernel of the operator Q. It can be written as an exponential expansion analogous to (2.11), where the correspondent coefficients are evaluated imposing the given Cauchy's Conditions. Under the hypothesis of asymptotic stability, imposed by the satisfaction of the Routh-Hurwitz conditions, which we will analyze in details for the two simple cases in Sections 2.2.1 and 2.2.2, this term (as well as the impulse response) eventually approaches 0. This allows us to neglect the first part of the (2.10) (as in the classic case) and to drop the dependence from the Initial Cauchy's Conditions, so that the solution can be asymptotically approximated by:

$$w_i(t) \simeq \eta(i,\ell) \sum_{k=1}^{N} \zeta_{i,k} \cdot g(t;t_k) \ , \quad i = 1, ..., n. \tag{2.12}$$

Hence, we can start to understand how all the learning process is affected by the shape of the impulse response $g(t)$, i.e. by the choice of the regularizer P. In the following Sections 2.2.1 and 2.2.2 we will analyze the dependence from the obtained roots λ_q in the odd and even order cases, respectively (first and second order in particular), starting to introduce the sign-flip issue, which we will explore deeper in Section 2.3. In Section 2.4 we will show, by an extensive experimental analysis, how the dissipation can modulate the learning process tailoring a parallel with different methods for standard discrete optimization.

2.2.1 First Order Operator

In this Section we start the analysis on some properties of the impulse response $g(t)$ considering the most simple case in which the order of P is $\ell = 1$. As already said, we restrict to the cases in which $\alpha_j > 0,\ j = 0, \dots \ell$, so that the operator can be generally written as $P = \alpha_0 + \alpha_1 D$, with correspondent formal adjoint $\hat{P} = \alpha_0 - \alpha_1 D$. The Euler-Lagrange equations in (2.9) become (see Section A.2.1):

$$D^2 w_i + \theta D w_i + \beta w_i - \frac{\gamma}{\mu_i \alpha_1^2} U(t) = 0 \ , \quad i = 1, ..., n \tag{2.13}$$

where θ and $\beta = \frac{\alpha_0 \alpha_1 \theta - \alpha_0^2}{\alpha_1^2}$ represent the coefficients of the characteristic polynomial of the associated homogeneous equation, while $U(t) = \sum_{k=1}^{N} \zeta_{i,k} \cdot \delta(t - t_k)$ the signals coming from the external input and the correspondent supervisions, and

$$\zeta_{i,k} = \left. \frac{\partial \mathcal{V}\left(y_k, f(w(t), u(t))\right)}{\partial w_i} \right|_{t=t_k} . \tag{2.14}$$

The *characteristic polynomial* $p(\lambda)$ correspondent to the (2.13) is:

$$p(\lambda) = \lambda^2 + \theta \lambda + \beta = 0 \tag{2.15}$$

for which the Routh-Hurwitz stability conditions implies (see Section A.2.1) the satisfaction of the additional requirement $\theta > \alpha_0/\alpha_1$, which results into the imposition on the roots $\lambda_1,\ \lambda_2$ to have a strictly negative real part, giving an impulse response $g(t)$ composed by exponentially decaying terms. Some examples of impulse responses, obtained varying the model parameters θ, α_0 and α_1 are shown in Figure 2.1. As we can see, the impact of the coefficients of the regularizer P is smaller than the one of the dissipation term θ, which in fact has a central role in the system modeling. Its meaning, linked to the function $\psi(t)$, is quite clear from a theoretical point of view. Here we can observe how, when increasing the dissipation, the impulse response approaches zero faster, forgetting more quickly the information coming from each supervision, according to the role intended for ψ to assign more importance to the latest samples.

Another important issue is represented by the relation among the regularizer parameters α_j and the generated roots λ_q. The cases reported in Figure 2.1 only

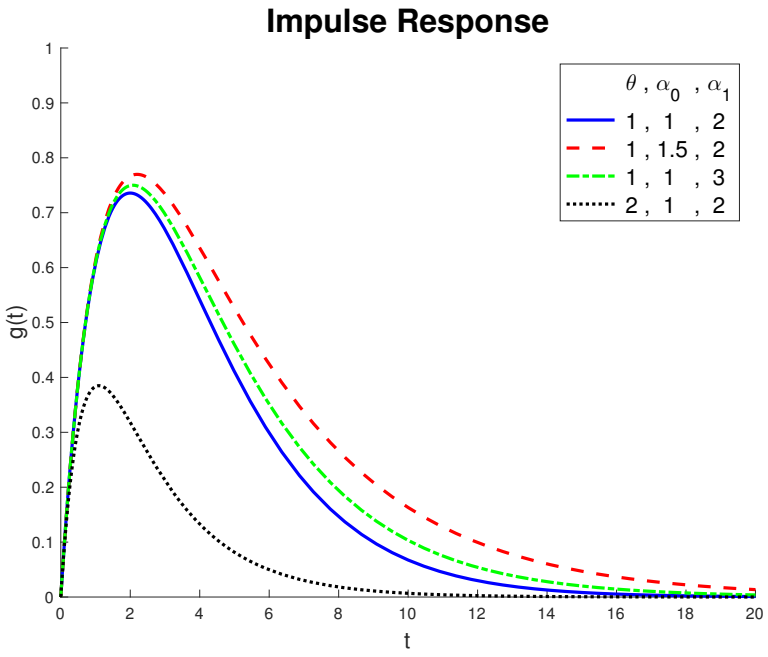


Figure 2.1: Examples of impulse responses $g(t)$ generated by a first order differential operator P, varying the coefficients α_0, α_1 and the dissipation term $\theta = 1$,

analyse cases in which λ_q have null imaginary part. Indeed, in the case in which $\exists q \ : \ \Im(\lambda_q) > 0$, the exponential decay scales a sinusoidal term. The produced oscillations could in some case (as a for instance, when the frequency of the impulses is high) compromise the stability of the system, due to a possible shifting among the oscillation phases. Hence, in practice, it will be more safe to chose the model, and then $g(t)$ actually, by assigning the roots of the characteristic polynomial, instead fo the coefficients θ, α_j, with the only requirement of generating a response coherent with the hypothesis on the model. Obviously, the stability is guaranteed by taking λ_q with negative real part and null imaginary part. However, the roots are related to the choice of θ and α_j too, which are all assumed to be positive. Then, we have to set them so as to generate a characteristic polynomial compatible with these assumptions, as shown in details in Section A.2.1. For example, since θ represents the coefficient of the term of order one in (2.15), is easy to see that:

$$\theta = -(\lambda_1 + \lambda_2)$$

and this always agrees with the conditions $\theta > 0$ when the λ_q satisfy the stability conditions. Moreover, this relation allows us to give an idea of the dissipation

level, since θ can be easily computed from the chosen roots.

A final remark has to be done on the sign-flip introduced by the higher order term in $\hat{P}$, which will be analyzed in Section 2.3.

2.2.2 Second Order Operator

The case $\ell = 2$ exploits the regularizer $P = \alpha_0 + \alpha_1 D + \alpha_2 D^2$, with formal adjoint operator $\hat{P} = \alpha_0 - \alpha_1 D + \alpha_2 D^2$. The correspondent Euler-Lagrange equations are given by the system of fourth order linear differential equations:

$$D^4 w_i + \beta_3 D^3 w_i + \beta_2 D^2 w_i + \beta_1 D w_i + \beta_0 w_i + \frac{\gamma}{\mu_i \alpha_2^2} U(t) = 0 \ , \quad i=1,...,n \tag{2.16}$$

where the coefficents (derived in Section A.2.2) are:

$$\begin{aligned} \beta_0 &= \tfrac{\alpha_0\alpha_2\theta^2 - \alpha_0\alpha_1\theta + \alpha_0^2}{a_2^2} \\ \beta_1 &= \tfrac{\alpha_1\alpha_2\theta^2 + (2\alpha_0\alpha_2 - \alpha_1^2)\theta}{\alpha_2^2} \\ \beta_2 &= \tfrac{\alpha_2^2\theta^2 + \alpha_1\alpha_2\theta + 2\alpha_0\alpha_2 - \alpha_1^2}{\alpha_2^2} \\ \beta_3 &= 2\theta. \end{aligned} \tag{2.17}$$

This time the Routh-Hurwitz stability criterion imposes more conditions (see again Section A.2.2), which however are easy to be satisfied. Because of the exponential form of $g(t)$, the implication is again that the roots λ_q, $q = 1, \ldots, 4$ have a strictly negative real part. However, in this case more configurations are possible for the roots, due to the potential multiplicities. Again, possible problems arise when non null imaginary parts are present. Since at this point we need some computations to find the roots, it is more convenient to use numerical tools, which nevertheless could introduce some numerical error. This could produce a non-null imaginary part in the roots even when the characteristic polynomial has in fact only real roots. To avoid this problems and to simplify the setting up of the model, in the remainder we will choose the desired response $g(t)$ by assigning λ_q, as introduced for the previous case. Again, these roots must be correlated to meaningful values of the regularizer coefficients α_j and the dissipation parameter θ, as shown in Section A.2.2. Even in this case, the dissipation parameter is

directly related to the roots since:

$$\beta_3 = 2\theta = -(\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4)$$

and hence $\theta > 0$ is automatically satisfied by the stability criterion and it is easy to compute by checking directly the dissipation level. The shape of the impulse response is the same of the one shown in Figure 2.1 for the first order. The only difference is at the point $t = 0$ connecting the constant part of $g(t)$ from the negative semi-plane $t < 0$ with the exponentials summation of the positive one. Indeed, the originated solution $g \in C^{2\ell-2}(0)$, since the operator $D^{2\ell}$ introduces the presence of the Dirac Delta function. Hence, the Heaviside Step function[3] appears in the solution when applying $D^{2\ell-1}$, producing a jump discontinuity. This condition on the order of the derivatives affects the smoothness of the response in $g(0)$, but the general trend is similar even when increasing the order ℓ. For these reasons, we restrict the analysis up to the second order, since we are more interested in the relations between even and odd order than in its magnitude.

2.2.3 From Action Principle to Gradient Descent

The Gradient Descent (GD from now on) method plays a central role in many optimization problems [78]. In particular, the pairing of GD with the Back-Propagation rule ([79], [80]) leads to a mayor breakthrough in the training procedure of Multi-Layers Perceptron [81, 82] architectures and is the key to success of the nowadays achievements of Artificial Neural Networks. In [1] the authors propose a first analysis on the similarities between the classic discrete GD method and a reformulation of learning as a continuous temporal process in the framework of Variational Calculus. In the Supervised Learning case, classic GD proposes an iterative solution to the minimization problem for a loss function similar to the potential in (2.2). Starting form an initial configuration of the weight vector $w[0]$ at the step $K = 0$, the configuration at the step $K + 1$ can be computed by:

$$w[K+1] = w[K] - \eta \cdot \Delta \mathcal{V}_{w_K} \tag{2.18}$$

[3] In the classical Theory of Distribution, the Dirac Delta function can be interpreted as the derivative of the Heaviside step function

where $\Delta\mathcal{V}_{w_K}$, in the simplest case, is the gradient vector of the penalty with respect to the weights in w evaluated at step K. The parameter $\eta > 0$, referred to as the *learning rate*, represents a scaling factor for the step of the gradient descent steps. A plethora of enhancements have been studied within the Machine Learning field in order to make this process more efficient, as for example the adding of a *momentum term* [83], techniques to adapt the step during the optimization [84] and different normalizations exploiting second order methods [85, 86]. However, the exploration of such methods is out of the aim of this work and we will focus on the parallel between the standard formula in (2.18) and the one introduced in (2.10). Indeed, the term $\zeta_{i,k}$ in the (2.10) is computed exactly as $\Delta\mathcal{V}_{w_K}$ in the discrete version of (2.18).

As already said, a fundamental issue in the composition of $\eta(i,\ell)$ in (2.10) is the role of the factor $(-1)^{\ell+1}\gamma$. The optimization process, aiming to provide a minimum for the penalty function $\mathcal{V}$, relies on evaluating its gradient, which represents the direction of the steepest increase of the loss. Then, the parameters are moved in the opposite direction to find regions with lower values of the penalty. Assuming that, in the cost functional $A[w]$, we want to minimize te summation of the penalty and the regularizer, weighted by the positive parameter $\gamma = 1$, our formulation is coherent with the formula in (2.18) when operating in the case of even order operator P. Indeed, the case analyzed in Section 2.2.2 leads to an updating formula of the kind of:

$$w_i(t) \simeq -\frac{\gamma}{\mu\alpha_2^2}\sum_{k=1}^{N}\zeta_{i,k}\cdot g(t;t_k)\ , \quad i=1,...,n \tag{2.19}$$

which is analogous to the classic case since all the coefficients are positive. However, when adopting an odd order operator, as explored in Section 2.2.1, the sign between the two terms inside the correspondent solution is flipped, obtaining:

$$w_i(t) \simeq \frac{\gamma}{\mu\alpha_1^2}\sum_{k=1}^{N}\zeta_{i,k}\cdot g(t;t_k)\ , \quad i=1,...,n. \tag{2.20}$$

The relation with the (2.18) clearly leads the optimization towards a configuration of maximum potential (as introduced in Section 1.1) and in practice originates divergent systems. The classic formula can be replicated by assuming $\gamma = -1$, as we will investigated more precisely in Section 2.3. However, this drives our

dissertation towards models composed by even order operators.

At this point, the choice of the symbol $\eta(i, \ell)$ to indicate the scaling factor should appear clear, since it acts in a way similar to the learning rate η in discrete methods. Moreover, from (2.9) and (2.10) we can observe how the mass μ_i affects the balance between regularization and fitting, by modulating the contribution of the terms $\zeta_{i,k}$ and allowing the weight w_i to change as faster as the smaller is μ_i itself, providing a sound interpretation of the concept of *inertia* related to the particle mass. An additional (time-variant) scaling factor for the gradient $\zeta_{i,k}$ is represented by the function $g(t)$. However, given a regularizer and hence a shape for $g(t)$, we can regulate the magnitude of the system response by varying $\eta(i, \ell)$. As already said, is more easy to skip the computation of the model from the coefficients α_j by assigning the roots λ_q defining $g(t)$ and, anyway, given $\alpha_\ell > 0$, we can always find an equivalent model after a normalization to $\alpha_\ell = 1$. For these reasons, in the remainder we will fix the model from the roots λ_q and assign a coefficient η_i which embeds the information about μ_i and α_ℓ, i.e. $\eta_i = 1/\mu_i\alpha_\ell^2$, referring properly to it as the *learning rate* of the model. Another simplification is possible by dropping the dependence of i and posing $\eta_i = \eta$, i.e by assigning the same learning rate, and hence the same mass μ and inertia, to all the weights in w.

2.2.4 On-line solution and discrete computation

The general form in (2.12) provides an explicit way to evaluate the solution at any instant of time t. Indeed, we can compute the exact value $w(t)$ as a summation of the impulse responses centered at $t - t_k$, each one multiplied by the gradient of penalty evaluated at t_k, indicated by $\zeta_{i,k}$. The terms related to future instants of time, for which $t_k > t$, can be neglected since in these cases the causal impulsive response is null, i.e $g(t - t_k) = 0$. This configuration is tailored for on-line learning mode, since we assume to have not in advance all the input samples given, and allows up a continuous evaluation of the function in real-time. However, in this setting, we have to store all the computed gradients and, for each one, evaluate the correspondent value of $g(t - t_k)$, that, in practice, could become unfeasible when the number of samples grows. To reduce this computation, an approximation can be obtained exploiting the asymptotic stability of the response. Indeed,

the function $g(t) \to 0$ when $t \to +\infty$. Hence, a possible approximated solution can be computed by neglecting the term $\zeta_{i,k}$ when we decide that $g(t - t_k) \approx 0$, depending on the decay time of g. As we show in Figure 2.1, the value of θ mainly affects the *peak* of G, i.e. the value of is maximum, fixing an upper bound for the magnitude of the roots. Because of this, the dissipation affects the promptness of the response too, since the higher is the magnitude of the roots, the faster is the response to reach its maximum. This behavior, and in particular its relation with the impulse frequency, will be deeply analyzed in Section 2.4, since it turns out to be fundamental to determine the global learning mode. On the other hand, the decay time is strictly related to the roots of $p(\lambda)$ closer to the origin, that originate the exponential decaying more slowly. This trend is related to the memory of the system, since determines in some way when the contribution of each sample disappears. Hence, in the remainder, we will refer to the root with smaller magnitude as the *memory* one. When dealing with a real task, it seems reasonable to exploit as much information as possible and, hence, to build up systems in order to have the slower decay time as possible. Because of the relations among parameters and roots introduced in Sections 2.2.1 and 2.2.2, is easy to fix the roots in order to have the desired promptness and memory. Nevertheless, the choice of systems with long term memory makes the computation of the explicit solution unfeasible in practice, since the rounding of g to zero is reasonable only after large intervals of time. Like for any constraint, and in general any differential equation, we could always provide a numerical solution.

However, we can derive a discretized version of $w(t)$, which is not an approximation and allows the evaluation at any time instant without storing any of the past values. From now, we drop the dependence on the parameter index i, managing the case of a single weight. Indeed, even in the general case the equations differ only by the initial conditions and, hence, the computation is the same. Let us define $W(t) = \left[\, w(t),\, \ldots,\, D^{2\ell-1} w(t) \,\right]'$ such that any linear differential equation is rewritten as:

$$\dot{W}(t) = A \cdot W(t) + B \cdot U(t) \tag{2.21}$$

where:

$$A = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -\beta_0 & -\beta_1 & -\beta_2 & \dots & -\beta_{2\ell-1} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ (-1)^{\ell+1}\gamma\eta \end{bmatrix} \tag{2.22}$$

being A the $2\ell \times 2\ell$ companion matrix of the 2ℓ order differential equation, mapping it into an equivalent system of 2ℓ differential equations of first order. B is the vector that maps the scalar external input $U(t)$ (see for instance equations (2.13) and(2.16)), in order to add it to the correct equation of the system (i.e. the last one). Exploiting a discretization process typical of classic linear state space models [87], the solution of this system of first order differential equations can be compactly written as:

$$W(t) = e^{At} \cdot W(0) + \int_0^t e^{A(t-s)} \cdot B \cdot U(s)\, ds \tag{2.23}$$

exploiting the matrix exponential form e^{At}. In this expression, the weights at time t still depend on all the supervisions provided at the instants $t_k \leq t$, since the integral in the second term still contains the impulses summation $U(t)$. However, we can take advantage of a crucial simplification by restricting the computation to case in which the solution is evaluated with a discrete time-sampling step. This is not a limitation since in real problems even continuos data are always provided in a discrete sampling, due to elaboration and evaluation processes (as a for instance frames for videos). Hence, we assume the solution $W(t)$ to be updated with a fixed time-sampling step τ, so that it can be computed in an iterative process with a finite number of steps given by T/τ, arbitrarily assuming that the width of the period T is a multiple of τ. Starting from the given initial Cauchy conditions $W[0] = W(0)$, after $K+1$ steps, $K \in \mathbb{N}$, we will have $W[K+1] = W((K+1)\tau)$. By splitting the integral in (2.23) in the two intervals $[0, K\tau]$ and $[K\tau, (K+1)\tau]$, it is easy to see (Section A.3) that

$$W[K+1] = e^{A\tau} \cdot W[K] + \int_{K\tau}^{(K+1)\tau} e^{A[(K+1)\tau - s]} \cdot B \cdot U(s)\, ds \tag{2.24}$$

applying the property that $e^{A(K+1)\tau} = e^{A\tau}e^{AK\tau}$. Without loss of generality, we can assume that each supervision eventually happens in the middle of a sample interval, i.e. $\forall k \, \exists K_k : \; t_k = \left[K_k + \frac{1}{2}\right] \tau$, where we use k and K_k to index a supervision and the correspondent updating instant respectively. Hence, the integral in equation (2.24) computed in the interval $[K\tau, (K+1)\tau]$ is either null, if no supervision is present, or equal to the constant $e^{A\frac{\tau}{2}} \cdot B \cdot \zeta_k$ when a label y_k is provided for $u(t_k)$, this because of the translation property of the Dirac's Delta function $\delta(t)$. The recurrent weights update formula can then be written as

$$W[K+1] = e^{A\tau} \cdot W[K] + e^{A\frac{\tau}{2}} \cdot B \cdot \Delta_K \tag{2.25}$$

where

$$\Delta_K = \begin{cases} \zeta_k \,, & \text{if } \; \exists k : t_k \in [\, K\tau, \, (K+1)\tau \,] \\ 0 & \text{otherwise} \end{cases} \tag{2.26}$$

obtaining the configuration showed in Figure 2.2. As we can see, the instants of updates and the instants of evaluations, in which a possible supervision comes, are shifted by $\tau/2$. By using this update rule, the trend of the solution depends only on its last evaluation and a possible supervision. We do not need neither to store any ζ_k nor compute $g(t - t_k)$, since the impulses are implicitly embedded in the evolution of the system. The matrices $e^{A\tau}$ and $e^{A\frac{\tau}{2}}$ can be precomputed, given the roots λ_q of the characteristic polynomial.

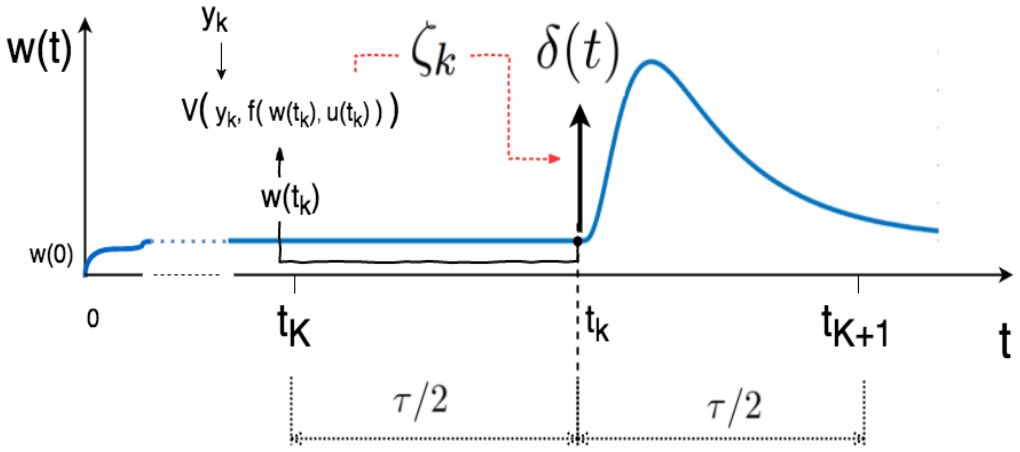


Figure 2.2: Assumed temporal distribution of data and updatings.

2.3 On the sign flip of Back-Propagation

To carry out a preliminary study on the behavior of the learning equation in (2.25) under different settings, we consider the implementation of a simple linear function, which can be computed by $f(\, w(t),\, u(t)\,) = m(t) \cdot u(t) + b(t)$. As already said, the function of time $u : [\, 0,\, T\,] \to \mathbb{R}^d$ describes the evolution of the input in the feature space $\mathbb{R}^d$, whereas the set of weights $w(t) = \{m(t), b(t)\}$ is composed in this case by the slope $m(t)$ and the bias $b(t)$ of the line. Classic discrete optimization methods are based on an iterative process allowing the parameters, starting from random initial values, to converge to their correspondent optimum. The procedure loops over the available supervised data, neglecting the truly dimension of time (samples are often randomly shuffled in practice). The restriction of our formulation to the periodic case, i.e. data are assumed to be distributed in the interval of time $[\, 0,\, T\,]$ and then repeated, can be easily adapted in a similar configuration in order to set up a simple comparison and an in–depth analysis of the main properties. Indeed, for now, it is convenient to generate a static sampling of data, arbitrarily assigning an artificial temporal distribution. Starting by a simple regression task, the learning trajectory is generated by considering a set of 2000 equally spaced points $u(t_k)$ in $[\, -1,\, 1\,]$, such that $u(t_1 = 0) = -1$ and $u(t_N = T) = 1$. Targets are computed as $y(t_k) = 2 \cdot u(t_k) - 1$ and supervisions are provided every 100 steps (only 20 supervisions in the interval). Unsupervised samples do not affect the weights trend but, in this initial phase, are useful to produce plots with a dense sampling in order to have a high resolution in the temporal domain to simplify the analysis. An additional parameter is represented by the time width of the step τ, that determines also the duration of the period $T = \tau N$. In general, our formulation is designed to deal with on-line learning scenarios in which the data distribution could also change in time, overcoming common batch procedures in which the weights should converge to optimal values. However, in this setting the classic case is reproduced, requiring that eventually $m(t) \to m^* = 2$ and $b(t) \to b^* = -1$ exploiting the asymptotic stability assumption. This process can be configured as a standard training by interpreting the concepts of epoch as a sweep of the input function over the period T. Hence,

the whole procedure can be organized in the sequence reported in Algorithm 1.

Algorithm 1: Algorithm Sketch for linear case.

input Data : $\mathcal{L} = \{(\, u(t_k),\, y_k\,)\}_{k=1}^{N}$
output : $m(t),\, b(t)$
parameters : $\lambda_1,\, \ldots,\, \lambda_{2\ell},\, \gamma,\, \eta,\, \tau,\, epochs$
initialization: Computation of A, B, $e^{A\tau}$, $e^{A\frac{\tau}{2}}$ from the chosen parameters;
Cauchy's conditions : $\mathbf{m} = [\,0,\, 0,\, 0,\, 0\,]',\ \mathbf{b} = [\,0,\, 0,\, 0,\, 0\,]'$;
for *epochs* **do**
  for $K = 1, ..., T/\tau$ **do**
    if *supervision* **then**
      $\mathbf{m}^{tmp} = e^{A\frac{\tau}{2}}\mathbf{m}$;
      $\mathbf{b}^{tmp} = e^{A\frac{\tau}{2}}\mathbf{b}$;
      $f = \mathbf{m}_1^{tmp} u(t_k) + \mathbf{b}_1^{tmp}$;
      $\zeta_{m,k} = (f - y_k)u(t_k)$;
      $\zeta_{b,k} = (f - y_k)$;
    else
      $\zeta_{m,k} = \zeta_{b,k} = 0$;
    end
    $\mathbf{m} = e^{A\tau} \cdot \mathbf{m} + e^{A\frac{\tau}{2}} \cdot B \cdot \zeta_{m,k}$;
    $\mathbf{b} = e^{A\tau} \cdot \mathbf{b} + e^{A\frac{\tau}{2}} \cdot B \cdot \zeta_{b,k}$;
  end
end

The initial Cauchy's conditions $\mathbf{m}[0] = [\,0,\, 0,\, 0,\, 0\,]',\ \mathbf{b}[0] = [\,0,\, 0,\, 0,\, 0\,]'$ are arbitrary because of the asymptotic stability, as we will briefly analyze in Section 2.3.3. The temporary variables $\mathbf{m}^{tmp}$ and $\mathbf{b}^{tmp}$ are introduced to compute the values of the parameters at the instant of the gradient evaluation at $t_k = K + \tau/2$ when a supervision is provided. Indeed, since no external forces are applied at $t = K\tau$, they could be computed from their last values exploiting the homogeneous part of the (2.25) with step $\tau/2$.

2.3.1 Second Order Operator

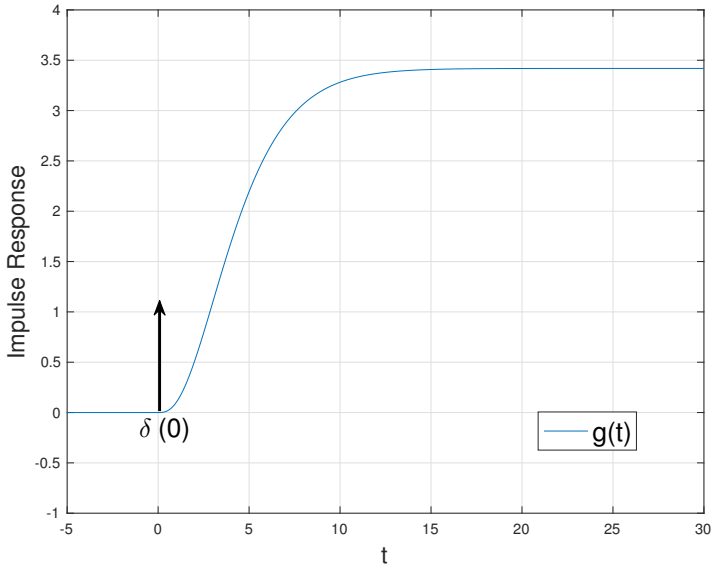
We start the practical test by implementing a model with second order operator P, since, as already said, the obtained solution is coherent with the classic GD rule. We choose the memory root $\lambda_1 = -10^{-8}$, whereas the other ones are selected so as to give $\theta = 1$. This choice is arbitrary, since varying θ we obtain a scaled response and we can always find an equivalent global configuration by adjusting

the other parameters. In this case, we chose $\lambda_2 = -0.6, \lambda_3 = -0.65, \lambda_4 = -0.75 - \lambda_1$ so as to satisfy the condition $2\theta = 2 = -(\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4)$ introduced in Section 2.2.2. From now on, we will always assume $\theta = 1$ in order to drop some variability. The small memory root guarantees the impulse to slowly approach zero, as we can see from Figure 2.3 (b), and hence a model with almost infinite memory. The concept of memory is in general related to the width of the step τ and the total number of updates, but we will adapt the parameters in order to carry out the whole training process before the beginning of the decay of the first impulse. The local behavior of the response near to the origin is shown in Figure 2.3 (a). Fixed θ, the promptness of $g(t)$ can be accelerated by selecting the roots as equal as possible each to the other, so as to not have slow exponential in the increasing phase (except from the one holding memory). This balancing is not fundamental in practice, since we can always adapt the step τ in order to place the next update in the desired position, after the peak or during the rising of $g(t)$. If we set τ enough large to have $\tau/2$ after the peak, we can always exploit the maximum response before the next updating. By setting infinite memory and prompt response, we fully approximate a standard discrete *Stochastic Gradient Descent* (SGD) algorithm. We will better analyze the dependence of the global configuration from the parameters in Section 2.4, but we give this introduction to describe the working environment.

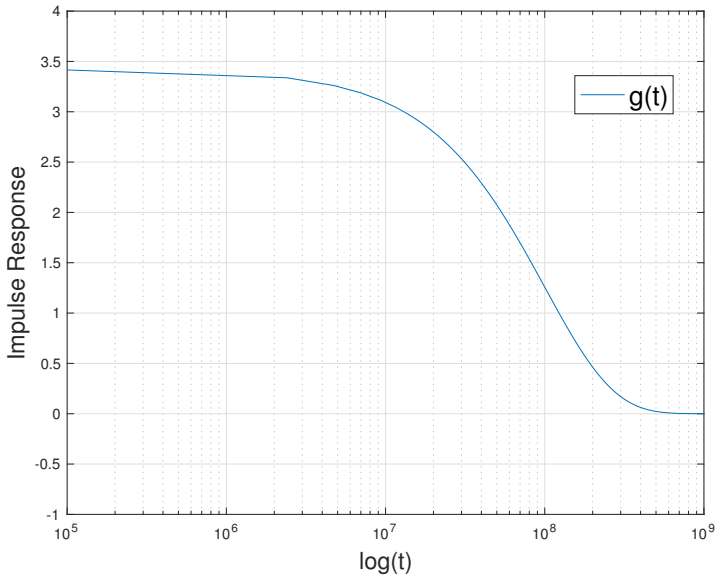
In Figure 2.4 (a) we show the trend of the weights described by the presented configuration. The step $\tau = 40$ guarantees a fully response between consecutive samples. The linearity of the representation function f allows us to set a quite big learning rate $\eta = 0.1$ and a fast convergence in this first case. Only 3 epochs are indeed enough to find the expected values for m and b. The parameter γ is assumed positive according to the previous argumentation on the parallelism with classic GD formula. Indeed, when flipping the sign assuming $\gamma = -1$, we obtain the trend reported in Figure 2.4 (b), exhibiting divergence both for the solution of $m(t)$ and $b(t)$.

2.3.2 First Order Operator

In Figure 2.5 we show how the stability depending on g is flipped with respect to the previous case when adopting a regularizer P of first order. All the parameters

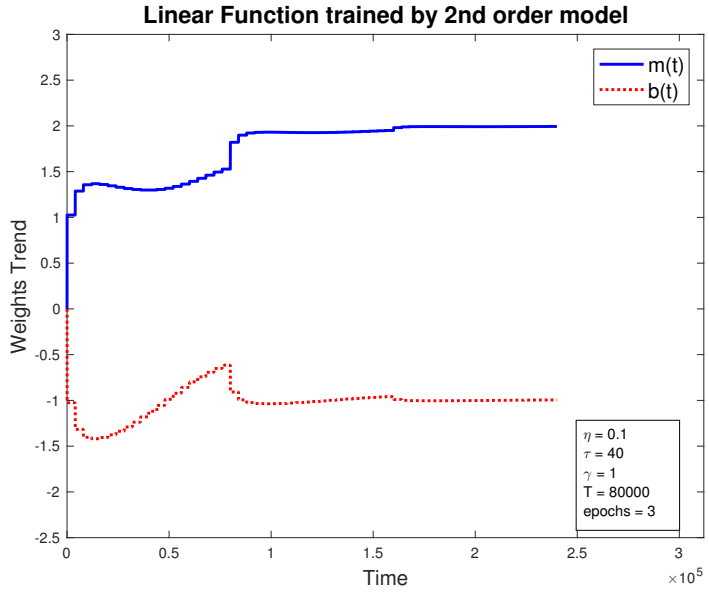


(a)

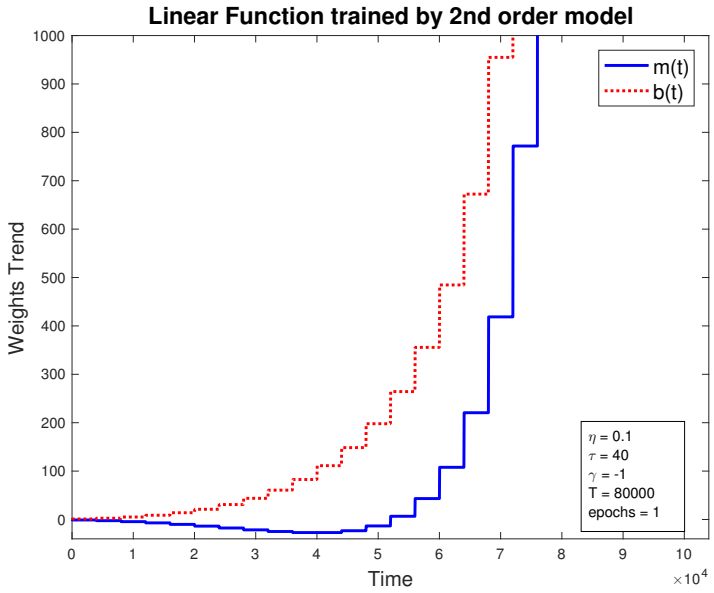


(b)

Figure 2.3: (a) Local behavior of the impulse response around $t = 0$ obtained by a second order operator P originating a system with roots $\{ \lambda_1 = -10^{-8}, \lambda_2 = -0.6, \lambda_3 = -0.65, \lambda_4 \approx -0.7499 \}$ and $\theta = 1$. (b) Correspondent global trend of $g(t)$ obtained with the *memory* root $\lambda_1 = 10^{-8}$. Due to the asymptotic stability $g(t) \to 0$ as $t \to \infty$.



(a)



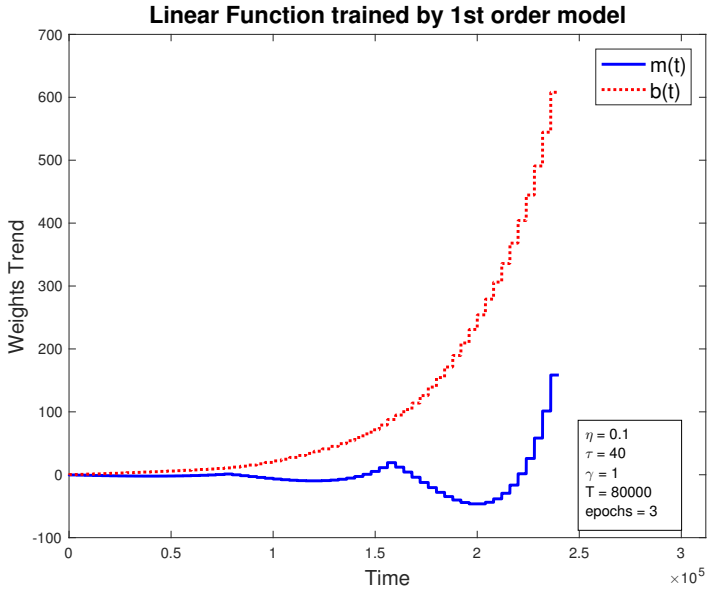
(b)

Figure 2.4: Trajectory of two weights $m(t)$ and $b(t)$ of a linear model evolving under the temporal regularization exploiting a second order differential operator. The roots of the characteristic polynomial are $\{\lambda_1 = -10^{-8}, \lambda_2 = -0.6, \lambda_3 = -0.65, \lambda_4 \approx -0.7499\}$. The learning rate is $\eta = 0.1$, the sampling step $\tau = 40$ and the period width $T = \tau N = 40 \cdot 2000$. We show both a configuration which converges to the optimal values $2, -1$ in (a), assuming $\gamma = 1$, and a divergent system in (b) produced by $\gamma = -1$.

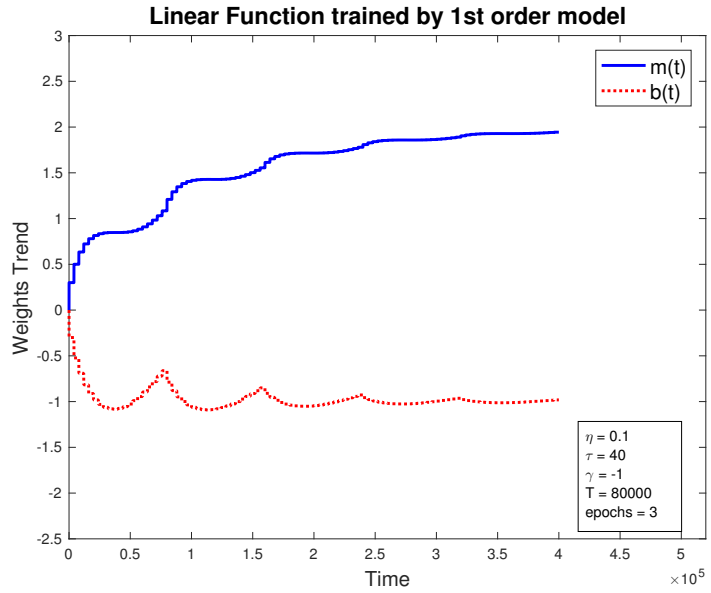
are the same of the previous case, except for the roots of $p(\lambda)$ chosen as $\lambda_1 = -10^{-8}$ and $\lambda_2 \approx -1 - \lambda_1$, so as to satisfy the condition $\theta = -(\lambda_1 + \lambda_2) = 1$. In Figure 2.5 (a) we show how the choice of $\gamma = 1$ produces divergent solutions because of the sign flip reported in (2.20). In this case, suitable solutions can be re-established by assuming $\gamma = -1$ in order to replicate the GD updating formula. Indeed, as shown in Figure 2.5 (b), the composition of the solution as the summation of gradients, scaled by η and $g(t)$, produces the achievement of good results even if, from a theoretical point of view, the minus sign between the two terms in the Lagrangian inside the expression in (2.8) originates a cost functional which seems to be not well defined for a minimization problem. Despite of this result, in the reminder we will focus on regularizer with order $\ell = 2$ to deal with models coherent with their theoretical formulation.

2.3.3 Initial Conditions

In this Section we carry out an experimentation to support the assumption that asymptotic stability allows us to neglect the homogeneous term in the (2.10) and, hence, the dependence of the Initial Cauchy's Conditions. We exploit the same learning environment and parameters proposed in Section 2.3.1. In Table 2.2 we report four different random choices for initialization of the parameter vectors. The obtained solutions are shown in Figure 2.6 (a) and (b) respectively for the slope $m(t)$ and the bias $b(t)$. As we can see, after an initial phase in which the trends are very different, depending on the starting points, eventually they converge to the same value, which is again the desired one. The final values are not clear from the figure because of the magnitude of the initial oscillations, but we have in practice $m(t) \to 2$ and $b(t) \to -1$.

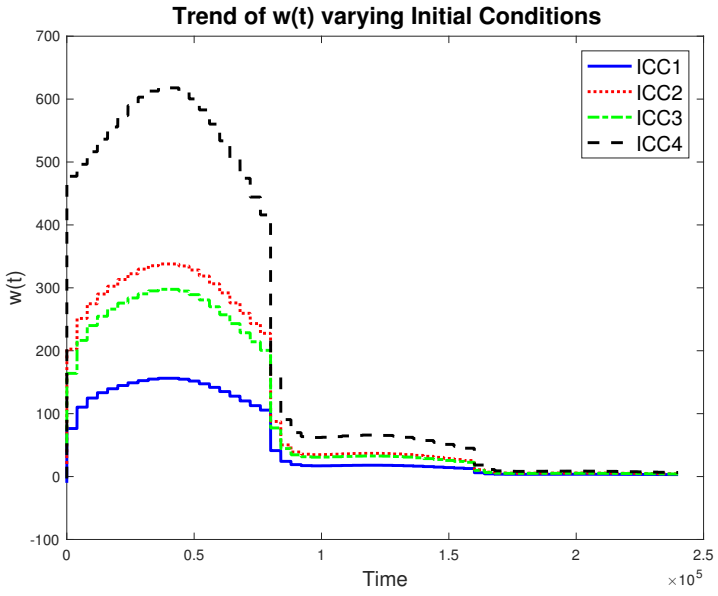


(a)

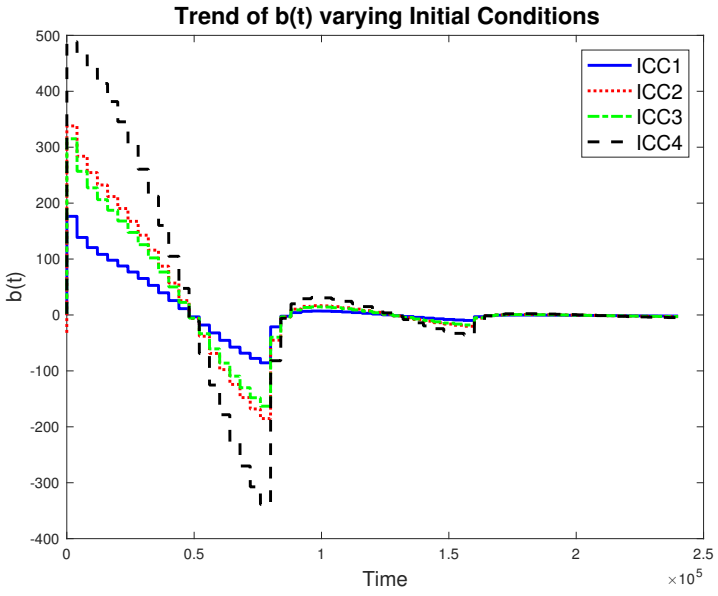


(b)

Figure 2.5: Trajectory of two weights $m(t)$ and $b(t)$ of a linear model evolving under the temporal regularization exploiting a first order differential operator. The roots of the characteristic polynomial are $\{\lambda_1 = -10^{-8}, \lambda_2 \approx -0.9999\}$. The learning rate is $\eta = 0.1$, the sampling step $\tau = 40$ and the period width $T = \tau N = 40 \cdot 2000$. We show the flipped configurations with respect to Figure 2.4, obtaining a divergent system in (a) when $\gamma = 1$ and optimal solutions in (b) with $\gamma = -1$.



(a)



(b)

Figure 2.6: Trajectory of the two weights $m(t)$ and $b(t)$ when training a linear model evolving for 5 epochs with temporal regularization exploiting a second order differential operator using the same parameters of previous section. Different Initial Cauchy's Conditions are reported in Table 2.2

	$m(0)$	$m'(0)$	$m''(0)$	$m'''(0)$	$b(0)$	$b'(0)$	$b''(0)$	$b'''(0)$
ICC1	-10	-7	1	0	-0.1	-30	14	96
ICC2	20	5	27	-51	-32	0	86	-20
ICC3	50	-1	-11	8	12	-45	100	0
ICC4	-3	70	0	43	2	9	66	3

Table 2.2: Configurations of the different Initial Cauchy Conditions tested in Figure 2.6.

2.4 Dissipation and Learning modes

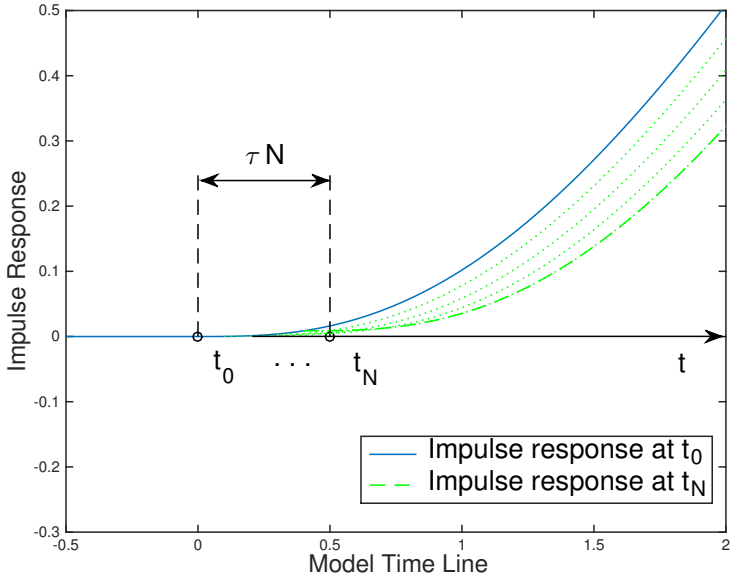
The concept of dissipation is introduced in the definition of the cost functional in (2.8) by means of the function $\psi(t) = e^{\theta t}$. Its effect modulates the contribution of both the loss and the regularizer in $A[w]$, assigning them a varying weight along the temporal trajectory. The exponential function is a convenient choice since the properties of its derivative simplifies the computation of the correspondent Euler-Lagrange equations (see Section A.2). However, fundamental characteristics are the positivity and monotonic growth. These properties allow us to assign an increasing weight to the events in time, such that most recent information is more important with respect to the previous one. The suitability of such behavior could in general depend on the features of the specific task to be faced, but it can be easily tuned by the exponent θ. In particular, the periodic case faced in our setting gives a clear idea of the role of dissipation. As already said, we assume our data, and hence the global information about the environment, to be distributed in the interval $[\,0,\, T\,]$. At the beginning of the process, the weight assigned by the dissipation function $\psi(t)$ can be evaluated as $e^{\theta 0} = 1$, whereas at the end period it is given by $e^{\theta T} = e^{\theta N \tau}$, because of the discretization of the interval with sampling step τ. In other words, given the horizon T, the effect of dissipation can be controlled through the parameter θ. At this point, if we want to set up an environment which equally weights all the information provided along the process, we consider a *low dissipation* configuration in which $e^{\theta N \tau} \approx 1$. When considering the discrete implementation, the additional parameter τ, corresponding to the sampling step, allows us to refer to dissipation as the agent processing rate rather than to the absolute value of the time horizon T. In practice, since

the effect of dissipation depends on the product $\theta N\tau$, in the analysis we can assume that $\theta = 1$ and then τ is accordingly set in order to obtain the desired effect. In practice, this point of view can be explained by considering the time spacing of data with respect to the behavior of the impulse response $g(t)$ that describes the evolution of $w(t)$, whose shape generally depends on θ. In a *low dissipation* configuration, as depicted in Figure 2.7 (a), the sampling instants, in which the function is computed, are close with respect to the response time of $g(t)$. At $t = T = \tau N$ the system has still to start to react to the impulse placed at $t = 0$. In this setting the actual response is a combination of the impulses given by every supervision provided, such that they have nearly the same influence on the initial value. On the other hand, a *high dissipation* arrangement, which in general can be achieved by a larger θ, imposes the response $g(t)$ to be faster than the frequency of the impulses and, hence, to develop itself for a while before the next update. When dealing with responses of the kind of Figure 2.1, approaching zero rather rapidly, this setting can lead to sterile configurations, since the weights tend to go to zero between two consecutive impulses. This kind of behavior can be avoided by the choice of a memory solution proposed in Section 2.2.4, generating responses such a that of Figure 2.3, which slowly approaches zero and allows us to develop all the learning process before the response disappears. In this case, we obtain the configuration depicted in Figure 2.7 (b). Two consecutive supervised points come far enough to allow $g(t)$ to yield a significant effect between the samples, but the response can be considered almost flat after the initial increasing phase. After the agent processed a point, the impulse rapidly (relatively to the impulses rate) reacts and its contribution is already absorbed when receiving the following supervision. The effect of an impulse is then strictly related to effect of the previous one on the trajectory.

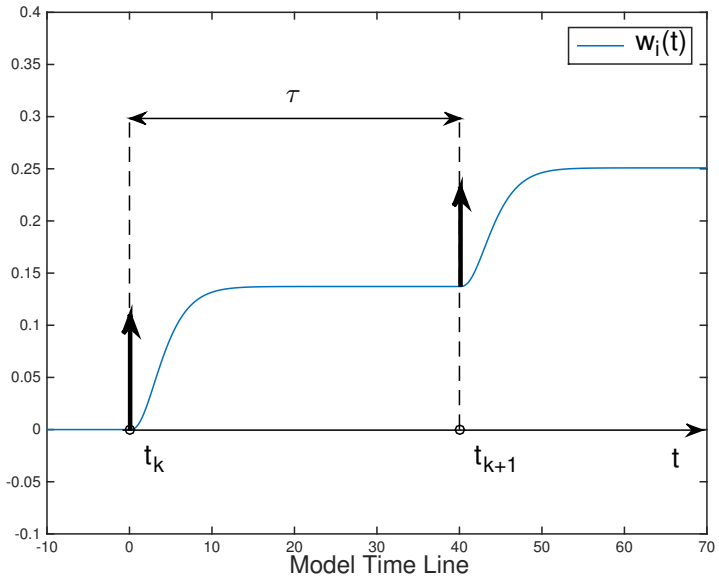
2.4.1 A simple example

To analyze the behavior of the learning equation in different settings, we exploit again the simple learning scenario proposed in Section 2.3. The agent, via the learnable weights $m(t)$ and $b(t)$, evolves according to the proposed differential equations, obtained with $\ell = 2$, $\gamma = 1$ and roots

$$\lambda = \left\{-1 \cdot 10^{-8},\ -0.6,\ -0.65,\ -0.75 + 1 \cdot 10^{-8}\right\} \tag{2.27}$$



(a) Impulse frequency in the Low Dissipation Setting



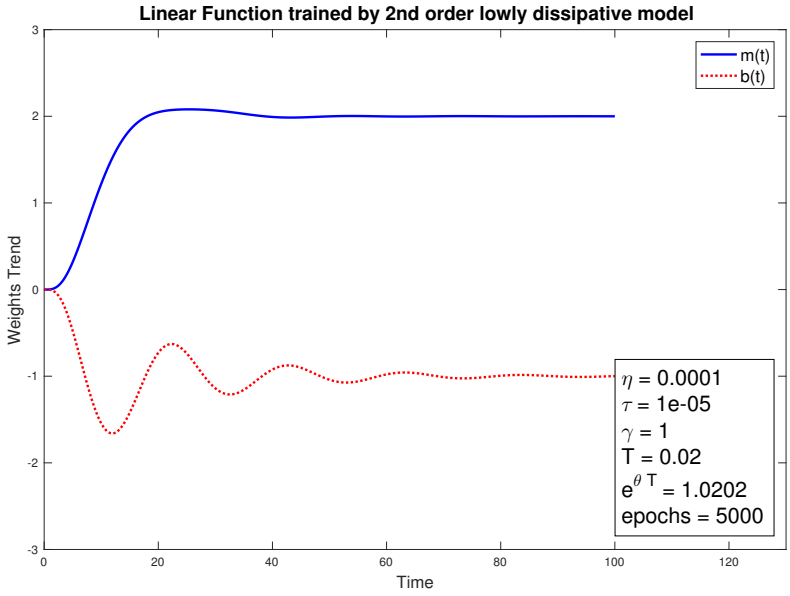
(b) Impulse frequency in the High Dissipation Setting

Figure 2.7: (a) Time spacing of inputs in the *low dissipation* configuration. The supervisions are close one to the other and the correspondent contributions are almost independent. The response of the system embeds the reaction to several impulses together. (b) Time spacing of two consecutive inputs in the *high dissipation* configuration. Each impulse comes after the system has had enough time to respond to the previous one, such that the value of the weights at t_{k+1} significantly depends on the impulse at t_k.

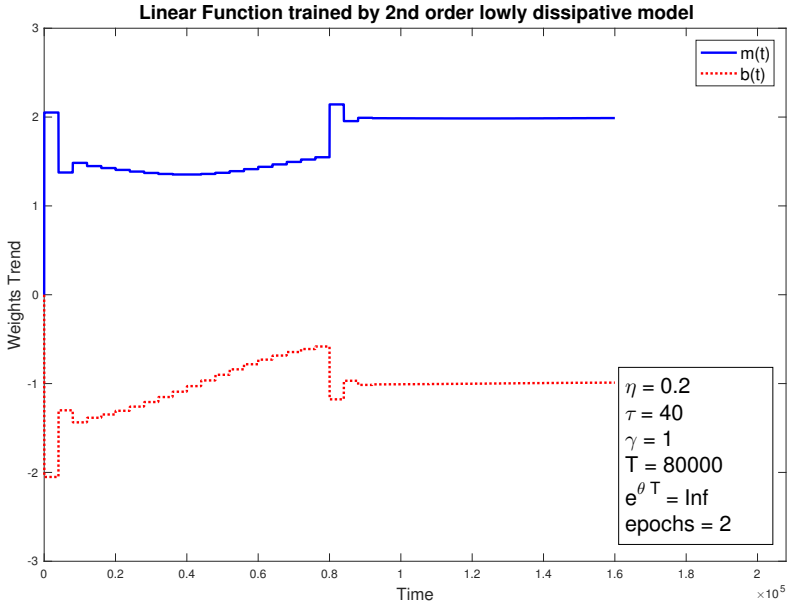
so as to satisfy $\theta = 1$, giving the impulse response shown in Figure 2.3.

In Figure 2.8 (a) we report the trend of the optimization process obtained by a model driven by low dissipative configuration. Assuming $\theta = 1$, the sampling step is set to $\tau = 10^{-5}$ so as to obtain $\psi(T) = \psi(\tau N) \approx \psi(0) = 1$, balancing the importance of the samples all over the interval. The choice of the learning rate η in this case is quite crucial, since an accumulation of the delay of the overlapped impulse responses (see Figure 2.7 (a)) causes an overestimated reaction of the system which compromises the convergence. In order to avoid this kind of problems, a small η has to be set to restore stability, requiring a sensible larger number of training epochs to achieve the optimal values. Even if the oscillatory behavior around the final asymptote (observable especially for the weight $b(t)$) is not fully removed when balancing the delay effects, the obtained solutions are very smooth trajectories reaching the desired values, due to the effective exploitation of the regularization action of the operator P.

In Figure 2.8 (b) we show the outcome of the learning process under a high dissipative configuration, analogous to the one reported in Figure 2.4 (a). The step $\tau = 40$ gives a very large period T, which causes the exponential $\psi(T)$ to explode. Both the parameters show step-like trends, due to the distance between consecutive responses, as depicted in Figure 2.7 (b). This distance and the flatness of the response after its initial peak allows us to set a relatively high learning rate $\eta = 0.2$ that yields a fast convergence (only 2 epochs are enough) to the desired values 2 and -1 respectively for $m(t)$ and $b(t)$. However, if we observe the trend of the weights at the end of the first epoch, i.e. in the middle of the plot, we can notice a local sub-convergence to the values $\{\, 1.5,\, -0.5 \,\}$. These values indeed, are a better solution with respect to $\{\, 2,\, -1\}$ in the neighborhood of $u(t) = 1$, satisfying the target fitting and reducing the cost with respect to the regularizer (since this have a lower magnitude). However the transition over other samples restores the global optimal configuration. Due to the simple structure, this local properties do not affect to much the process. However, when dealing with more complex architectures, these properties could drive the optimization of the parameters towards the basin of attraction of local minima, in which sometimes the agent can get stuck, raising up well known issues related to the classic SGD process.



(a) Low dissipation setting



(b) High dissipation setting

Figure 2.8: Trajectory of the two weights $m(t)$ and $b(t)$ of a linear model in Low (a) and High (b) dissipation settings respectively.

As we can see, in the previous case the regularization effect avoids the attraction of the local minima $\{ 1.5, -0.5 \}$ encountered in the high dissipation case, taking into account the information all over the training data.

2.5 Experiments

Main motivations of this work aim to model a new approach to learning, exploiting the additional information coming from the temporal coherence when dealing with real world problems in which data describe phenomena evolving continuously in time. The most suitable scenario for our formulation is represented by those cases in which data are not available at the beginning of the process and then the system assimilates labels and starts to react with its own predictions at the same time. However, taking advantage of the periodicity assumption on data distribution, we can always set up the agent training in a classic static environment, in order to evaluate the proposed methods on standard benchmarks. Moreover, the smoothness assumption, embedded in the cost functional in (2.8), is reflected in the trajectories of the weights, so that further constraints on the input evolution are in practice not necessary.

In order to enlighten both the generality and the flexibility of our proposal we will test the learning procedure for training of Artificial Neural Networks. This kind of structures had a large diffusion in the last 20 years, achieving state of the art results in most of the Artificial Intelligence trending topics (Computer Vision, Speech Processing, Natural Language Processing and so on). The complexity of such architectures represents a crucial issue, combining a high representative power and an intrinsic difficulty of the training process, which usually follows the GD rule. As already said, several techniques have been developed to overcome these harshnesses, producing a multiplicity of highly efficient methods. However, the analysis of this Section is developed around the two basic approaches of SGD and *Full Batch* mode, even if in practice best results are achieved exploiting the intermediate compromise of *mini-batch* setting [88]. Indeed, we are interested in the investigation of the parallel between these methods and different configurations defined by the dissipation level in our formulation, as introduced in Section 2.4. Since we are not interested in issues related to the architecture,

our tests are bounded to the most simple case of 2-Layer Artificial Neural Networks, optimized using the *Mean Square Error* as penalty function. We exploit the *Rectifier Linear Unit* (ReLU) as activation function for the hidden layer, first introduced as transfer function in [89]. As already said, we are primarily interested in the global behavior of the network within the comparison among standard and proposed learning settings, rather than into the properties of single modules. However it becomes necessary to exploit the capability of the quasi-linearity of the ReLU preventing the neurons to saturate [90], i.e. to get stuck in regions producing zero derivatives in the activation. Indeed, we found out that the accumulation of the delay in the impulse responses and the resulting oscillating dynamic systems in the low dissipative cases, showed in Figure 2.8 (a), often drive sigmoidal activations functions to the saturation, compromising the optimization process. Hence, we will start with a regression case in Section 2.5.1, in which the simple setting allows us to set up the architecture by standard activation functions, quickly exploring some of the saturation issues. We will analyze in Section 2.5.2 and 2.5.3 more complex classification tasks from an artificial dataset and a standard benchmark, respectively.

2.5.1 Regression with ANNs

A preliminary test is carried out on a small artificial data set. We generated the input data by taking a sampling of 1000 point in $[\,-\pi/2,\,\pi/2\,]$, computing the target by the function $\sin(u)$ perturbed by some small random noise, obtaining the distribution shown in Figure 2.9 (a). A sampling of 100 elements evaluated by the noiseless function is used for test. The learning architecture is a 2-Layer ANN with 5 units in the hidden layer and the *hyperbolic tangent* as activation function in both the layers.

In order to unload the notation, we will use the following initials when referring to different settings:

- FB : for standard full batch training
- SGD : for Stochastic Gradient Descent training
- HD : for high dissipative dynamic model

- LD : for low dissipative dynamic model

The dynamical system is obtained, exploiting a regularizer P of order $\ell = 2$, with the same parameters of the previous experiments by choosing the roots of the characteristic polynomial as in (2.27), so as to satisfy $\theta = 1$. The parameter γ is then fixed to $+1$. In the high dissipation setting we still set $\tau = 40$, whereas in the low dissipation we choose $\tau = 0.001$ so as to have $e^{\theta\tau N} = e^1 = e$. In standard settings the learning rate η is usually scaled up by a factor of $max(\, g(t)\,) \approx 3.5$, because of the value of the impulse response of Figure 2.3, so as to obtain similar modulation of the gradients and convergence of the training error, at least in the HD and SGD comparison. To analyze the dependence of the algorithm on the order in which data are presented, we trained the network under each configuration both in the original order and after a random shuffling of the input points. Indeed, even if data are not originated as a temporal sequence, we can simulate a similar situation by feeding samples to the agent scanning the input axis from the leftmost point $-\pi/2$ to the last one $\pi/2$, moving among consecutive instances. In this way we can reproduce a stream presenting a good coherence of information at a local level.

In Figure 2.9 (b) we show the introduced analogy between the HD and SGD settings which, after a rescaling of η, lead to very similar configurations. We can also notice the same behavior with respect to different sorting of training data. The benefits achieved when presenting input samples in a random order are a well known feature of the stochastic optimization process, maintaining the same effect on the HD setting too, as we can observe by the sensible improvements in the convergence speed (*random data* trends).

In Figure 2.10 (a) we showed how the same learning rate used in the HD configuration is not suitable for the LD setting. Indeed, as already said, the accumulation of the delay of the impulse response compromises the stability of the trajectory of the weights, as well as the optimization process.

In Figure 2.10 (b) we can observe how a significant reduction of the parameter η restores stability and allows the achievement of good fitting results in the LD setting too. Despite of the fact that the idea behind the FB and LD configurations are the same, is in fact impossible to obtain a specular behavior. Indeed, even if the responses are delayed, in the dynamic system the optimization happens in

a continuos fashion, because of the smooth weight coming from the dissipation, whereas in the FB setting, the accumulation of the gradients is cut off at the end of the training set at each update (and, in our case, the component of the gradients with highest magnitude normalized to 1). However, the effect of the data shuffling is the same in both settings. Since each sample in the interval $[\,0,\,T\,]$ has more or less the same weight and the system will react after the whole training set has been seen at least once, changing the order of the samples does not affect the training convergence, as exactly for the FB case.

2.5.2 Vowel classification

To generate the dataset for the first classification task, we recorded two different audio tracks with the sequential pronunciations of the five Italian vowels. The audio signal was processed to extract 40 auditory spectral coefficients using the RASTA-PLP algorithm proposed in [91], which will be used as feature representation in the input space. The first track, of about 20 seconds, with one utterance of each vowel, is used to train the model. We select 5 supervised instances per class[4], neglecting the unsupervised samples. Indeed, when no label is provided in the middle of the update interval, the system follows its natural evolution and the computation becomes redundant if on-line predictions are not necessary (considering a rescaling of the sampling so as to obtain the desired dissipation configuration). The second track of about 20 seconds, with different pronunciation of the vowels, is used as test set, providing about 250 labeled samples per class to compute the performance[5].

We run the experiments on these data by using a 2-layer Artificial Neural Network with 100 hidden units and 5 outputs, the ReLU as activation for the hidden layer, whereas the output function is linear. The cost function is the MSE. The dynamical system is obtained, exploiting a regularizer P of order $\ell = 2$ with the roots of (2.27) and $\theta = 1$, $\gamma = +1$. In the high dissipation setting we still set $\tau = 40$, whereas in the low dissipation we choose $\tau = 0.01$ so as to have $e^{\theta\tau N} = e^{0.25} \approx 1.28$. Again, the learning rate η is scaled up by a factor

[4]Targets are provided in the one-hot encoding form.

[5]MatLab® scripts used for the implementation are available at the GitHub repository `https://github.com/alered87/NN-Training-as-Dynamical-Process`

of $max(\, g(t)\,) \approx 3.5$ in the HD because of the value of the impulse response of Figure 2.3, so as to obtain similar convergence of the training error. In FB and LD the learning rate is the same since the relation is not direct as in the SGD–HD case. As already said, the learning rate has to be reduced in presence of low dissipative configurations to balance the delay accumulation of the response and avoid instability.

In Figure 2.11 we show the results obtained in the comparison, starting from the same initialization of the weights of the network in each different setting. In (b) and (d) we show the evaluation during training of MSE and Classification Accuracy on the test set for SGD and HD cases, obtaining again very similar outcomes. In (a) and (c) we report the same analysis on the FB and LD settings, finding a similar behavior even if the global setting can not be completely replicated.

2.5.3 MNIST

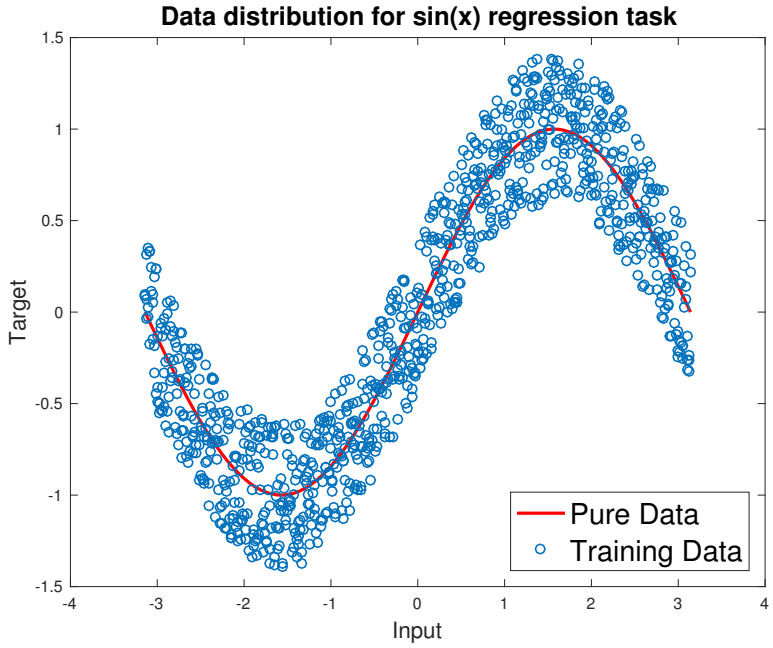
In this Section we propose a similar set of experiments on the MNIST data set [24]. It is a well known classification problem on images representing handwritten digits and is extensively used to test and compare general Machine Learning algorithms and Computer Vision methods. Data are provided as 28×28 pixels images of the grayscale levels, split in training and test sets containing respectively 60,000 and 10,000 instances. Data are available at the official site[6], together with a list of results obtained by most common algorithms. We use the same MatLab® implementation of the previous Section, exploiting the functions from the `mnistHelper`[7] package to extract data so as to obtain for each sample a 1-Dimensional vector with 784 elements, containing the gray levels of the images normalized from $[\,0,\ 255\,]$ to $[\,0,\ 1\,]$. To speed up the computation, we used a portion of the complete training set by randomly selecting $1,000$ instances for each of the 10 classes (for a total of $10,000$ samples, randomly sorted in the training sequence). Apart from computational issues, the excess of supervised data could lead to some problems in the convergence of our formulation in the low

[6] `http://yann.lecun.com/exdb/mnist/`.

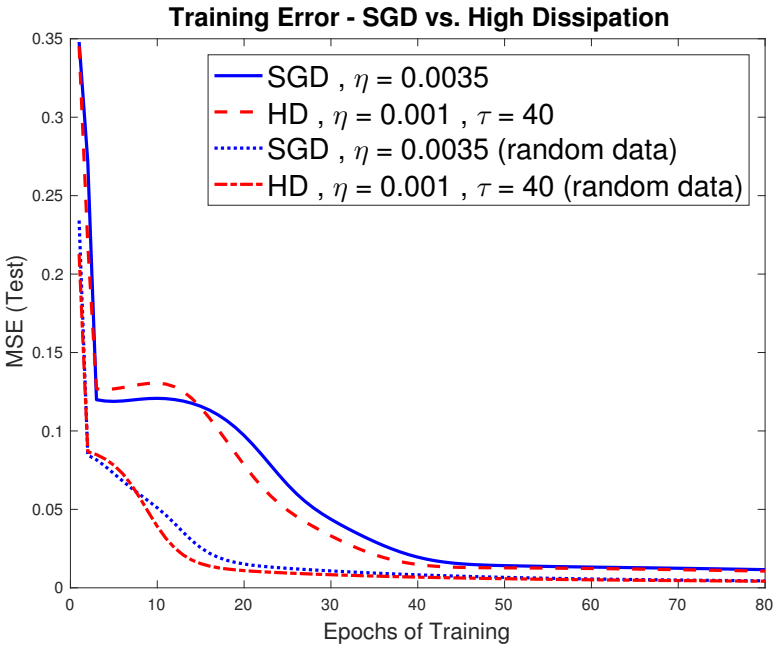
[7] `http://ufldl.stanford.edu/wiki/index.php/Using_the_MNIST_Dataset`

dissipation setting. Indeed, the accumulation of the delay of the huge amount of impulses requires a very small learning rate, compromising the convergence to a good configuration. Even after the data reduction, the LD setting required a learning rate of the order of 10^{-5}, against more standard values ($\eta \in \{ 10^{-3}, 10^{-4} \}$) used for SGD, HD and FB. In the dissipative cases, we used the same second order operator P of the previous section (roots from (2.27)). Using the same notation of Section 2.5.2 to refer to each experimental setting, the HD configuration is obtained by $\tau = 40$, whereas the LD one by $\tau = 10^{-4}$, so as that $e^{\tau N} = e$.

Again, the experiments in the same comparison are carried out by starting from the same initial configuration of the ANN weights. Figure 2.12 shows the classification accuracy on the test set measured during the learning process in both settings. In plot (a) we reported the comparison between the LD version of the proposed algorithm and the FB gradient descent technique. In (b) we compared the HD and SGD settings. Plot (a) shows that the results obtained with the low dissipation configuration are similar to what attained in the classic batch mode, even if the convergence is slower. As in the previous cases, plot (b) clearly shows the analogy between the HD and SGD. We do not report the experiments on the complete dataset, since the high number of data requires a small time sampling-step ($\tau = 10^{-5}$) and then a small learning rate (η of order of 10^{-7}) in the LD case, that makes the convergence too slow. However, in the high dissipation case, we obtain an accuracy similar to the one presented in [24] for the same architecture (about 3% of error).

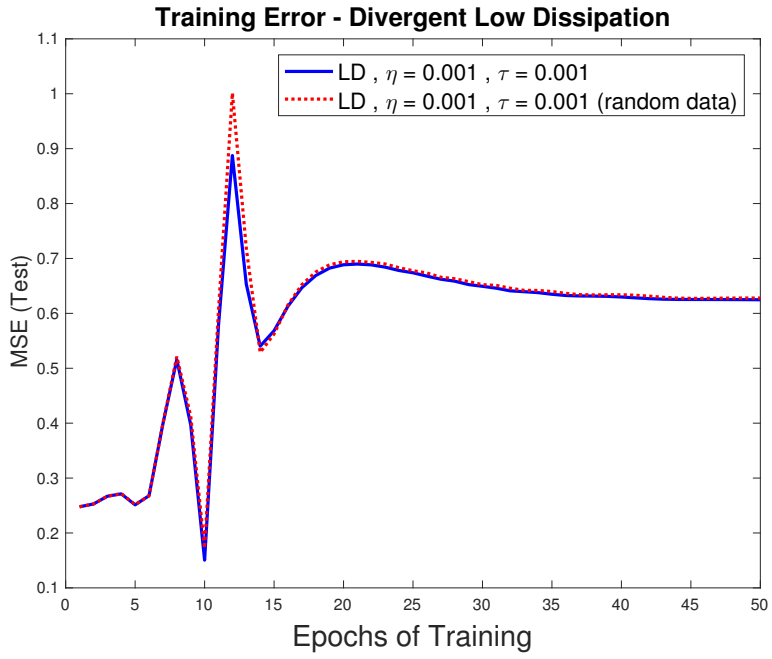


(a)

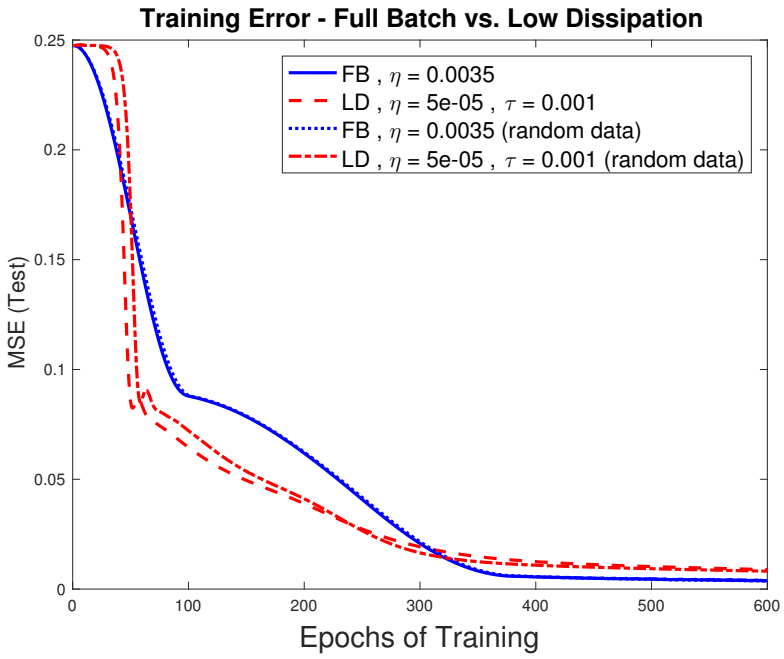


(b)

Figure 2.9: (a) The employed sampling of the training data and the true value for the function $\sin(u)$. (b) Trend of the MSE evaluated on the test data during the training in the SGD–HD comparison.



(a)



(b)

Figure 2.10: Trend of the MSE evaluated on the test data during the training in : (a) LD setting in unstable configuration (d) FB–LD comparison.

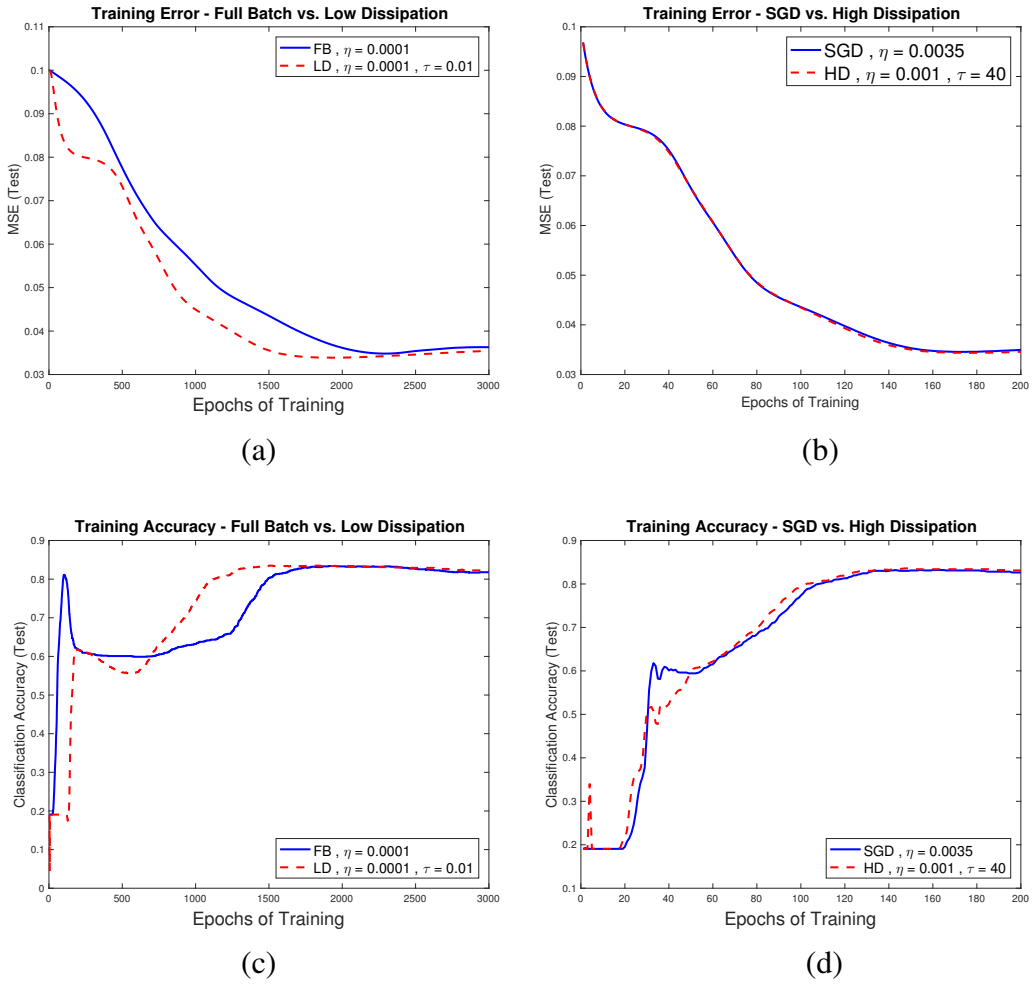
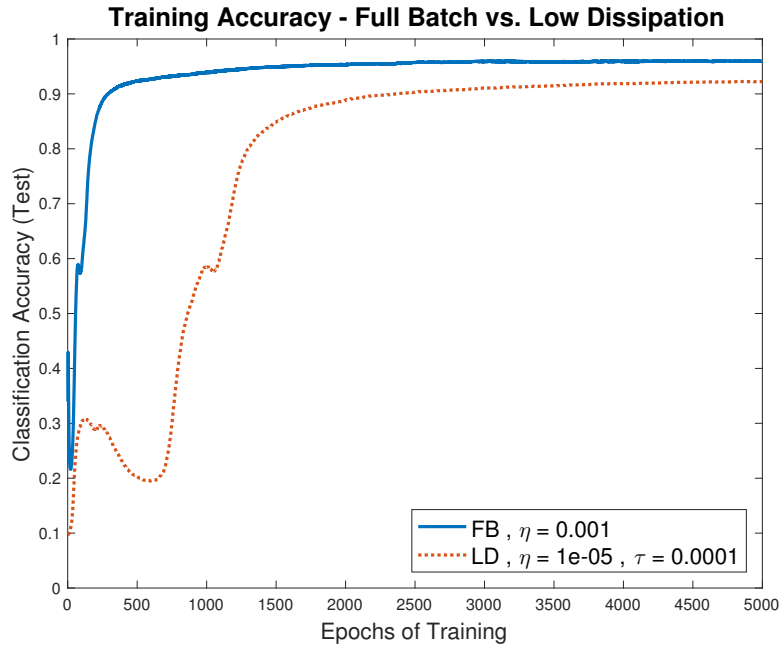
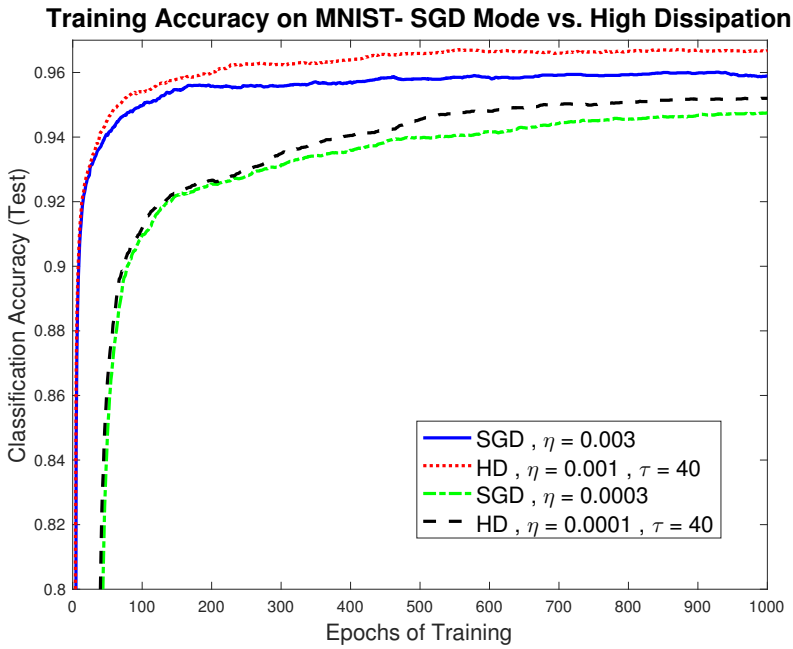


Figure 2.11: Comparison of results achieved on the Vowel classification task by a 2-layer ANN trained with the proposed formulation and standard methods in different settings. In (a) and (c) we compare the classic Batch mode (FB) and the Low Dissipation setting (LD) showing the trend of Classification Accuracy and MSE respectively, evaluated on the test set during training. In (b) and (d) we report the same comparison between the classic Stochastic Gradient Descent method (SGD) and the High Dissipation setting (HD).



(a)



(b)

Figure 2.12: Accuracy performance on classification evaluated on the test set at the end of each epoch of training on the MNIST data. For each case, we indicate the learning rate η. (a) Low dissipation setting obtained with $\tau = 10^{-4}$ (LD), compared with the batch version of the classic Gradient Descent method (FB). (b) High dissipation setting obtained with $\tau = 40$ (HD), compared with the Stochastic Gradient Descent technique (SGD).

3 On-line Learning on Temporal Manifolds

In this part, starting from the approach to Regularization and Learning proposed in Chapter 2, we formulate a transposition of the optimization procedure which completely relies on the continuity of the input data sequence. The whole process is still expressed in the temporal domain, maintaining the fundamental structure of the cost functional. We enforce the assumption that the input trajectory lies on a temporal manifold that models the evolution of the function, thus reducing the algorithm complexity. The hypothesis of local coherence in the input temporal sequence is assumed to impose smoothness constraints directly on the behavior of the representation function. Indeed, the previous formulation forces the development of the learning parameters to be a smooth process in time. In fact, the dependence on the input, left to the function itself an additional degree of freedom, producing possibly abruptly changes in the computed output stream due to occasional discontinuities in the input trajectory. The proposed approach models the trajectory of the prediction function by applying the Principle of Least Action. The integration of the differential equations is moved from space to time, without assuming the function to have a specific structure. The requirement of a dissipative environment is still necessary to guarantee the stability to the resulting system, even if few modifications can be applied to the dissipation function $\psi(t)$ in order to simplify the theoretical derivation. The model of the learning function that is independent of both the input and the parameters leads to a *collapsing of the dimensionality* generating a single Eulero-Lagrange equation. This differs from the system found in (2.9) which yields as many equations as the number of parameters. The obtained solution still represents a dynamic system, describing in this case the output trajectory. Again, the dependence on time produces an

explicit expression allowing a continuous evaluation, but the computation can be easily discretized to embed implicitly all the impulses and fit the data sampling.

In this formulation, the poorness of the representation function comes up to be insufficient to embody the behavior of any phenomena. In fact, such equations only model the smoothness in the output trajectory. To cope with this shortcoming, we exploit the obtained trajectory itself as a regularizer for a graph, which represents the skeleton of the architecture modeling the input-output mapping. The nodes of the graph, developed in an incremental fashion, are visited in time and store the information extracted from data during the learning process. The access policy to the graph nodes allows an easy on–line update during the input processing and a fast extraction of additional knowledge both from the input evolution and provided supervisions, allowing a perfect synchronization with the updates of the function $f(t)$. The introduced structure makes it straightforward to implement both spatial and temporal constraints on the function mapping, but further generic knowledge can be added to the learning process, as for example logical [92, 93] and statistical constraints [94]. To provide a set of data which evolves according with the hypothesis of the model, we evaluate the proposed learning scheme on a benchmark created from the MNIST dataset, generating an artificial video that allows us to analyze the properties of the designed structure and to have a feedback on the behavior with respect to target environments, as reported in [6]. A first preliminary analysis of this work can be also found in [5].

3.1 Regularization in the temporal domain

The input is still assumed to be described in a proper feature space $\mathbb{R}^d$ by a function of time $u : T \rightarrow \mathbb{R}^d$, with $T \subseteq [0, +\infty)$. External teaching signals are expressed by labels y_k provided in supervised pairs $(u(t_k), y_k)$ coming at $t_k \in T$. Again, in the theoretical formulation is convenient to restrict the analysis to the periodic case, so that we can assume $T = [0, t_e]$ exploiting a redundancy hypothesis on the global information on a widen horizon. Hence, data can be collected in the finite set $\mathcal{L} = \{ (u(t_k), y_k) \}_{k=1}^{N}$, where $k = 1, \ldots, N$ and $0 < t_1 < \cdots < t_k < \cdots < t_N < t_e$. The task is to learn a function $f(t, u(t))$ defined on $f : [0, t_e] \times \mathbb{R}^d \rightarrow \mathbb{R}^m$, whose output represents some kind of predic-

tion on data according to the provided labels. We work under the simplification of one dimensional mapping space $\mathbb{R}$ (i.e $m = 1$), since we can easily generalize to the multi-dimensional output case by extending the same model. In the new formulation the confidence on the local smoothness on the input requires to enforce the assumptions on the continuity of $u(t)$, which can be mathematically expressed by the condition $u \in C^{\ell+1}(\,[\,0,\,t_e\,],\,\mathbb{R}^d\,)$. The structure of the cost functional is similar to the one reported in (1.6) for standard Regularization Theory, following the derivation presented in [95]. If we start by considering that, relying on the $C^{\ell+1}$ requirement, the input trajectory defines a temporal manifold $\mathscr{M}$ given by its graph:

$$\mathscr{M} = \{\,(\,t,\,u(t)\,)\,:\,t \in [\,0,\,t_e\,]\,\}$$

which is in fact the domain of $f(\,t,\,u(t)\,)$. If we consider the correspondent mapping $\pi\,:\,[\,0,\,t_e\,] \to \mathscr{M}$ such that $\pi(t) = (\,t,\,u(t)\,)$, this can be viewed as a C^ℓ–diffeomorphism from the manifold $[\,0,\,t_e\,]$ to the manifold $\mathscr{M}$. Skipping analytical details, π can be used to generalize the fact that $f : \mathscr{M} \to \mathbb{R}$ is of class C^ℓ at a point $p \in \mathscr{M}$, simply by requiring that $\bar{f} = f \circ \pi : [\,0,\,t_e\,] \to \mathbb{R}$ is of class C^ℓ in a neighborhood of $\pi^{-1}(p)$. In a similar way (see [96]), it can be used to translate the general linear differential operator $P : C^\ell(\,[\,0,\,t_e\,]\,) \to C^0(\,[\,0,\,t_e\,]\,)$ of order ℓ defined in (2.4) into the correspondent operator $P_{\mathscr{M}}(D) : C^\ell(\mathscr{M}) \to C^0(\mathscr{M})$ such that:

$$P_{\mathscr{M}}\left(\frac{d}{dt}\right)(f \circ \pi) = P\left(\frac{d}{dt}\right)\bar{f}\,.$$

With these assumptions, the parametrization π can be used to *push forward* tangent vectors from $[\,0,\,t_e\,]$ to tangent vectors in $\mathscr{M}$. Since the usual derivation operator $\frac{d}{dt}$ is a smooth vector field in $[\,0,\,t_e\,]$, its pushforward $D = \pi * \frac{d}{dt}$ gives a way to translate a first order derivation operator from smooth real valued functions to $\mathscr{M}$:

$$D : C^\ell(\mathscr{M}) \to C^{\ell-1}(\mathscr{M}) \text{ such that } \; Df(p) = \frac{d}{dt} f(\,\pi(t)\,)\bigg|_{t=\pi^{-1}(p)}$$

and in the same way its general powers:

$$D^j f(p) = \frac{d^j}{dt^j} f(\,\pi(t)\,)\bigg|_{t=\pi^{-1}(p)} \quad \text{for } j = 1, \ldots, \ell\,.$$

The same concept can be extended to a general linear differential operator P by following the link:

$$[P(D)] \circ \pi = P\left(\frac{d}{dt}\right)(f \circ \pi) = P\left(\frac{d}{dt}\right)\bar{f}\,.$$

From now on we will indicate a general differential operator simply by P since its domain will be specified from the context. At this point, joining the cost functional in (1.6) and the concept of dissipation introduced in (2.8) by means of the function $\psi(t)$, we can formulate the supervised learning environment as a temporal process composing the regularization and the penalty term, so as to obtain the functional:

$$\Phi(f) = \int_{\mathscr{M}} (P_{\mathscr{M}} f)^2 \psi \, ds + \gamma \sum_{k=1}^{N} \psi(p_k) \, (\, f(p_k) - y_k \,)^2 \tag{3.1}$$

where $p_k = \pi^{-1}(\, t_k,\, u(t_k)\,)$ are points in the domain $\mathscr{M}$ of f and γ is the well known regularization parameter. According to the principle analyzed in Chapter 2, a solution to (3.1) can be formulated in order to find stationary points of the functional $\Phi(f)$ by the Euler-Lagrange equations. However, finding a stationary point f of Φ is equivalent to find $\bar{f}$ as a stationary point of:

$$\bar{\Phi}(\bar{f}) = \int_0^T (P\bar{f})^2 \bar{\psi} \sqrt{1 + \|u'(t)\|^2} dt + \gamma \sum_{k=1}^{N} \bar{\psi}(t_k) \, \big(\, \bar{f}(t_k) - y_k \, \big)^2 \tag{3.2}$$

where we pose $\bar{\psi} = \psi \circ \pi$. The Euler-Lagrange equations, obtained by a derivation specular to the one used for (2.9), require a stationary point of $\bar{\Phi}$ to satisfy the necessary condition:

$$\hat{P}(\bar{\psi} \sqrt{1 + \|u'(t)\|^2} \, P\bar{f}) + \gamma \sum_{k=1}^{N} \bar{\psi}(t_k) \, \big[\, \bar{f}(t_k) - y_k \, \big] \, \delta(t - t_k) = 0\,. \tag{3.3}$$

Despite of the similarity with the (2.9), in this form it is not straightforward to obtain the explicit solution. Indeed the term $b = b(\dot{u}) = \sqrt{1 + \|u'(t)\|^2}$ introduces a direct dependence on $u'(t)$ in the resolution of (3.3). This derives from the transposition of (3.1) in the temporal domain, representing the magnitude of the distance in local variations of the input function $u(t)$ (where $\|\cdot\|$ is intended to be the Euclidean Norm). However, to deal with a manageable version of (3.3),

a substantial simplification can be achieved by choosing:

$$\bar{\psi}(t, \dot{u}) = \frac{e^{\theta t}}{\sqrt{1 + \|u'(t)\|^2}} = \frac{\varphi(t)}{\sqrt{1 + \|u'(t)\|^2}}$$

where we write $\varphi(t) = e^{\theta t}$ to indicate the function embodying the same concept of dissipation introduced in (2.8) modulated by the parameter $\theta > 0$. Hence, we can rewrite the (3.3) in a form more similar to the (2.9):

$$\hat{P}(\varphi\, P\bar{f}) \;+\; \gamma \sum_{k=1}^{N} \psi|_{t_k} \left[\bar{f}(t_k) - y_k\right] \delta(t - t_k) \;=\; 0 \tag{3.4}$$

where again the term $\left[\, \bar{f}(t_k) - y_k \,\right]$ represents the derivative of the penalty function, this time calculated with respect to $\bar{f}(t)$. The solution to (3.4) is analogous to the one in (2.10) and hence asymptotically given by:

$$\bar{f}(t) \simeq \frac{(-1)^{\ell+1}\gamma}{\alpha_\ell^2 \varphi(t)} \sum_{k=1}^{N} \bar{\psi}|_{t_k} \left[\bar{f}(t_k) - y_k\right] g(t; t_k) \tag{3.5}$$

keeping also the sign-flip term $(-1)^{\ell+1}$. An apparent further complication is introduced by the modification of the dissipation function from $\psi(t) = e^{\theta t}$ to $\bar{\psi}(t, \dot{u}) = e^{\theta t}(\sqrt{1 + \|u'(t)\|^2})^{-1}$. However, the expression in (3.5) still provides an explicit formula to evaluate the trajectory $\bar{f}(t)$ at any instant of time, re-introducing the computational issues seen in Section 2.2.4. Again, for same reasons regarding the computation of the roots from α_j, we will set the model by assigning the correspondent roots. In this case, the parameter γ will be used as constant parameter modulating the response, acting with the same role of the learning rate η in Chapter 2. From now on, with a light abuse of notation, we will write γ instead of $(-1)^{\ell+1}\gamma/\alpha_\ell^2$. In an analogous way, we can replicate the discretization process proposed in Section 2.2.4, so as to obtain a formula to update the solution $\bar{f}(t)$ with a discrete sampling step τ analogous to the one in (2.25):

$$\mathbf{f}[K+1] = e^{A\tau} \cdot \mathbf{f}[K] + e^{A\tau/2} \cdot B \cdot \Delta_K \tag{3.6}$$

where $\mathbf{f}(t) = \left[\, \bar{f}(t),\, \ldots,\, D^{2\ell-1}\bar{f}(t) \,\right]'$ is the vector storing the derivatives of $\bar{f}(t)$ from order 0 to $2\ell - 1$. The matrices A and B come from the transformations of the 2ℓ order differential equation in (3.5) to a system with 2ℓ equations of first

order. A slightly modification appears in Δ_K generated by the change in $\bar{\psi}(t)$, giving:

$$\Delta_K = \begin{cases} \dfrac{\gamma}{b|_{t_k}} \left[\bar{f}(t_k) - y_k\right] , & \text{if } y_k \text{ is provided at } t_k \\ 0 \quad \text{otherwise.} \end{cases} \tag{3.7}$$

Apart from the scaling term $1/b|_{t_k}$, a formal modification has to be done in the rate between the instant of time representing input samples and system updates, generally referred to as t_k and K respectively. Indeed, in Section 2.2.4 it is not required that at each updating corresponds an input sample. On the contrary, in this setting the updates and the input sampling has to be in one to one correspondence, even if we have still to maintain the hypothesis that each sample comes in the middle instant between two updates K and $K+1$. Eventually we have $T/\tau = N$. From now on we pose $t_k = K\tau - \tau/2$, so that the first evaluation of the system $\mathbf{f}[1] = \mathbf{f}(\tau)$ is computed using $u_1 = u(t_1)$, where we assumed $t_1 = \tau/2$. In the remainder, we will use the notation t_K and t_k to indicate the instants $K\tau$ and $K\tau - \tau/2$ respectively, so that the (3.7) will be modified by shifting from $\bar{f}(k)$ to $\bar{f}(k+1)$ when computing Δ_K.

The expression of $\bar{f}$ obtained in (3.6) clearly shows the dependences describing its evolution. We assumed a general form of $\bar{f}$, neglecting the structure of the function that processes the input, bringing to a collapse of the dimensionality that models the evolution of the output. At each instant of time in the discretized sampling, the prediction of $\bar{f}$ depends on the incoming input $u(t)$ only via the term $b(t)$ that modulates the contribution, taking into account the variations in the input space. The input–output relation is only supported by the provided targets y_k. When the system crosses unsupervised regions, the coherence in the predictions should be kept by the imposed regularization, following the assumption that the input is distributed on a trajectory describing a smooth temporal manifold. The term $b(t)$ helps to keep a connection between the two sources of information. Indeed, when a supervision is given, the magnitude of the impulses received by the system is inversely proportional to the input derivative, and, hence, to the distance between the two points. That is, even in presence of an external teaching signal, the system avoids abrupt changes when the input crosses high variability regions, reducing problems due to possible noise in the input data. Nevertheless, the system is not aware of specific information on the input, but only of the variability of

the current region of the manifold. In fact, we can interpret $\bar{f}$ as an agent without memory, since is not capable to recognize information already processed in the past. With this formulation, it seems to be able just to track the provided target, performing good predictions in constant regions and possibly inaccurate predictions in the transition phase among regions with different targets. To circumvent this limitation, in the remainder we show how to exploit these regularization abilities by blending different techniques in a more sophisticated learning structure.

3.2 Developmental Learning Structure

The dissertation of the previous Section underlines the necessity to pair the derived prediction function with a scheme suitable to represent a some form of memory, taking into account the landmarks in the input feature space. Such a storage should be characterized by an incremental nature, possibly dealing with a never-ending learning scenario. At the same time, a fast computation would be desirable in on–line setting, allowing real–time reactions when processing the incoming stream of data. In the Data Mining literature, several methods for data splitting, allowing an efficient *Nearest Neighbor* search, have been widely studied (see [97] for a survey), as for example many different graphical structures as KD-trees [98], Ball trees [99], Cone trees [100] and Cover trees [101], together with very efficient searching techniques [102], [103], [104]. The proposed approach is based on existing techniques formulated in the same learning configuration, developing an incremental memory paired with the on–line regularization formulation. The work in [105], that represents a previous analysis assuming the same on–line scenario, exploits the management of the memory with *Random Projection Trees*, first formulated in [106], that represent a very flexible structure and provide a fast search procedure. However, we will adopt an implementation similar to [58], which we found out to better perform in the analyzed cases. The memory is developed as ϵ-Network, by linking fixed radius hyper-spheres in the input feature space. Even if the search criteria could be in general slower with respect to the architecture of [106], several techniques have been developed to make the computation more efficient when dealing with a large amount of data [107]. Even so, a common peculiarity of previous works is the fact that the optimization of the representation function is accomplished on a buffer of the last samples

in the data streaming, adding an hyper-parameter which is strongly dependent on the data distribution. Indeed, the buffer size should, from a theoretical point of view, be adapted so as to cut pieces of information far enough to represent a meaningful statistics on data, finding a compromise which guarantees a feasible computational cost.

In our approach, this issue is overcome by relying on the statistical redundancy of the input environment. The main idea is to develop in time a graph $\mathcal{G}$, whose nodes represent a flexible sampling of the input space, whereas different kind of edges are used to diffuse information among these nodes. This process has to interact, as time goes by, both with the sequence of the input data and the dynamic system in (3.6). Anyway, this could be achieved in a natural way since the sampling of the input sequence and the discretized updating of $\bar{f}(t)$ are already synchronized by the sampling step τ. Hence, when an input sample $u(t_k)$ comes, it is associated with a suitable node, in which the correspondent prediction $\bar{f}(t_k)$ is saved too. At this point, a crucial modification can be done to the computation of the magnitude of the impulse Δ_K to be returned to the dynamic system. The formula derived in (3.7), analogous to the one in (2.26) provided in Section 2.2.4, only computes Δ_K from the possible supervision on the current sample. In particular, with the setting proposed in Section 2.2.4, the computation on unlabeled data come up to be not necessary in the tested off–line learning approach, since no further modifications would be produced on the dynamic system evolution. However, the new setting is strictly designed to process streams of data which are not completely supervised. Indeed, if a label is never provided on a node (i.e. all the samples assigned to it are unlabeled), the value of $\bar{f}(t)$ predicted at the instant of the last visit is saved. This value is not only useful to provide a prediction for the unsupervised samples associated to the node, but also to propagate this information to its neighborhood. Indeed, to keep track of the input trajectory in its feature space, the current output is compared with the predictions stored in the closest[1] nodes. Moreover, since in general this kind of measure in high dimensional spaces could be plagued by well known ambiguities, the evaluation on adjacent nodes in the input sequence, with respect to all the previous occurrences of the current node, are taken into account too. From a theoretical point of view, all these actions can be easily embedded in the cost

[1] with respect to the Euclidean distance in the space $\mathbb{R}^d$.

functional in (3.2) by writing:

$$\Phi_{\mathcal{S}}(\bar{f}) = \Phi(\bar{f}) + (1-\eta)\,\mathcal{S}(\bar{f}) + \eta\,\mathcal{T}(\bar{f}) \tag{3.8}$$

where $\eta \in [0,1]$ is used to tune the global contribution of terms $\mathcal{S}(\bar{f})$ and $\mathcal{T}(\bar{f})$, representing the spatial and temporal contributions respectively. In the remainder we will show the details of the computations and the interpretation of all these concepts maintaining the conciseness of the (3.6). Furthermore, we will see how this derivation is quite general and can be easily extended to embody additional constraints.

3.2.1 Representational Graph

The graph $\mathcal{G}$ is defined and evolves as a fixed radius ϵ-*Radial Network*. The set of nodes will be referred to as V_G, whereas we will introduce two different sets of edges described by the correspondent adjacency matrices W_G and A_G, representing the spatial and the temporal connections respectively. The elements of $\mathcal{G}$ are updated with the same sampling step τ of the input and the system, starting from $V_G = \emptyset$ at $K = 0$. At each step of computation $K+1$, the input sample $u(t_{k+1}) = u(K\tau + \tau/2) = u_{K+1}$ becomes available, together with its supervision $y(t_{k+1}) = y(K\tau + \tau/2) = y_{K+1}$. Let us pose $y_{K+1} = \texttt{NULL}$ to indicate a missing label for the point u_{K+1}. At first step, the node v_1 is created by assigning $u[v_1] = u(t_1) = u_1$, where we introduce the notation $u[v_j]$ to refer to the representative input of the node v_j. This could in fact be interpreted as the center of the hypersphere associated to the node in $\mathbb{R}^d$, since the norm used to compute the distance will be the Euclidean one in the input space $\mathbb{R}^d$. Assuming that $v_K \in V_G$ is the node selected at step K, a new node of v_{K+1} is determined at $K+1$ by a procedure similar to the one proposed in [58] as:

$$v_{K+1} = \begin{cases} v_K\,, \;\; \text{if } \|u_{K+1} - u[v_K]\| \leq \epsilon_K\,,\; \epsilon_K \in \mathbb{R} \\ v^* = \underset{v_j \,\in\, V_G}{\arg\min}\; \|u_{K+1} - u[v_j]\|\,, \;\; \text{if } \exists j:\; \|u_{K+1} - u[v_j]\| \leq \epsilon_j\,,\; \epsilon_j \in \mathbb{R} \\ v_{new}\; s.t.\; u[v_{new}] \leftarrow u_{K+1} \;\; \text{otherwise.} \end{cases} \tag{3.9}$$

The parameter ϵ represents the radius of the spheres of the graph. The indexes assigned to ϵ in (3.9) means that, in a very general case, we could assume a varying radius depending on the sampling density of the region around each node. From now on, we will drop the index from the notation since we exploit a fixed radius. It is worth noticing that the update rule implements a sort of on–line Nearest Neighbor, in order to speed up the computation. Indeed, at each step we do not search V_G for the closest node, but we first try to assign the new input to last selected one, provided that they are close enough, i.e. if the input u_{K+1} belongs to the hypersphere related to v_K, that can also be written by $u_{K+1} \in \mathcal{B}(\, u[v_K],\, \epsilon\,)$. Otherwise, the algorithm searches for a suitable sphere over the whole set V_G. If none of the two is verified, a new element is added to V_G using u_{K+1} as representative vector. A graphical explanation of the process is depicted in Figure 3.1. Once we have selected the current node v_{K+1}, the prediction $\bar{f}[v_{K+1}]$ that will be associated with the node is determined by:

$$\bar{f}[v_{K+1}] = \begin{cases} \bar{f}(t_{k+1}), & \text{if} \quad h[v_{K+1}] = 0 \quad \text{and} \quad y_{K+1} = \texttt{NULL} \\ \bar{f}[v_{K+1}], & \text{if} \quad h[v_{K+1}] > 0 \quad \text{and} \quad y_{K+1} = \texttt{NULL} \\ \frac{\bar{f}[v_{K+1}] \cdot h[v_{K+1}] + y_{K+1}}{h[v_{K+1}]+1} & \text{otherwise.} \end{cases} \tag{3.10}$$

The counter $h[\,\cdot\,]$ represents the number of supervised hits, counting the number of supervised points assigned to each node. With this definition, each supervision falling in a node is saved, and averaged if the node was labeled with a different target in previous occurrences. If a supervision is never provided for a node, the prediction of the system evaluated at the moment of the last visit will be stored. This configuration is supposed to be the most frequent in our formulation. Indeed, the main idea is to find suitable predictions for unlabeled elements by refining this evaluation as time goes by, so as to converge to optimal values when the representational power of the graph increases together with the stored amount of knowledge.

After the node is selected and the correspondent prediction updated, the graph $\mathcal{G}$ outputs a feedback of the current output $\bar{f}(t)$ to the diagonal system, based on its own awareness of the environment. If the correspond label is provided, a value dependent on the derivative of the loss function with respect to $\bar{f}$ is evaluated and scaled by the input derivative, as reported in equations (3.6) and (3.7). However, according to the (3.8), two additional terms $\mathcal{S}(\bar{f})$ and $\mathcal{T}(\bar{f})$ are taken into account,

that will be described in the following, together with the combination of all the contributions.

3.2.2 Spatial Regularization

The term $\mathcal{S}(\bar{f})$ represents a constraint on the output function $\bar{f}$ forcing it to be coherent with respect to the predictions stored in the nodes of the graph $\mathcal{G}$ that are spatially close to the current node in the input space. As already said, $\bar{f}(t)$ does not depend explicitly on the input $u(t)$ and is not able to memorize the regions of the space that have been already visited in the past. The function $\bar{f}(t)$ can be then interpreted as a short-term memory component. The information coming from $\mathcal{S}(\bar{f})$ should encode the previous predictions in the neighborhood of the current input, yielding a certain degree of smoothness in the spatial dimension. Thanks to the synchronization assumption among the sampling of $u(t)$ and the updates of $\mathcal{G}$ and $\bar{f}(t)$, this penalty can be expressed by:

$$\mathcal{S}(\bar{f}) = \sum_{k=1}^{N} \frac{\bar{\psi}|_{t_k}}{\rho} \sum_{s=1}^{\rho} w_{k,n_{ks}} (\bar{f}(t_k) - \bar{f}[v_{n_{ks}}])^2 \tag{3.11}$$

where $\mathcal{N}_k = \{v_{n_{k1}}, ..., v_{n_{k\rho}}\}$ represent the set of the nodes in the neighborhood of node $v[K]$, sorted in decreasing order depending on their weights w. The parameter $\rho \in \mathbb{N}_{|V_G|}$ sets the number of the nodes to be assigned to the neighborhood. The quadratic penalty $(\bar{f}(t_k) - \bar{f}[v_{n_k}s])^2$ requires the value of $\bar{f}(t_k)$ to be as more similar to the value stored in a node as more the node is close to $v[K]$. Indeed, each term in the internal summation is scaled by the weight $w_{k,n_{ks}}$ and averaged with respect to the size of $\mathcal{N}$. The weights $w_{\cdot,\cdot}$ represent the elements of the *spatial adjacency matrix* $W_G \in \mathbb{R}^{|V_G| \times |V_G|}$, containing a measure which is inversely proportional to the distance between each pair of nodes in V_G. In the remainder, we will consider the elements of W_G calculated by the Gaussian distance as in standard Manifold Regularization [56, 57]. This means that w_{ij} representing the correlation between the nodes v_i and v_j is given by:

$$w_{ij} = e^{-\frac{\left\| u[v_i] - u[v_j] \right\|^2}{2\sigma^2}} \tag{3.12}$$

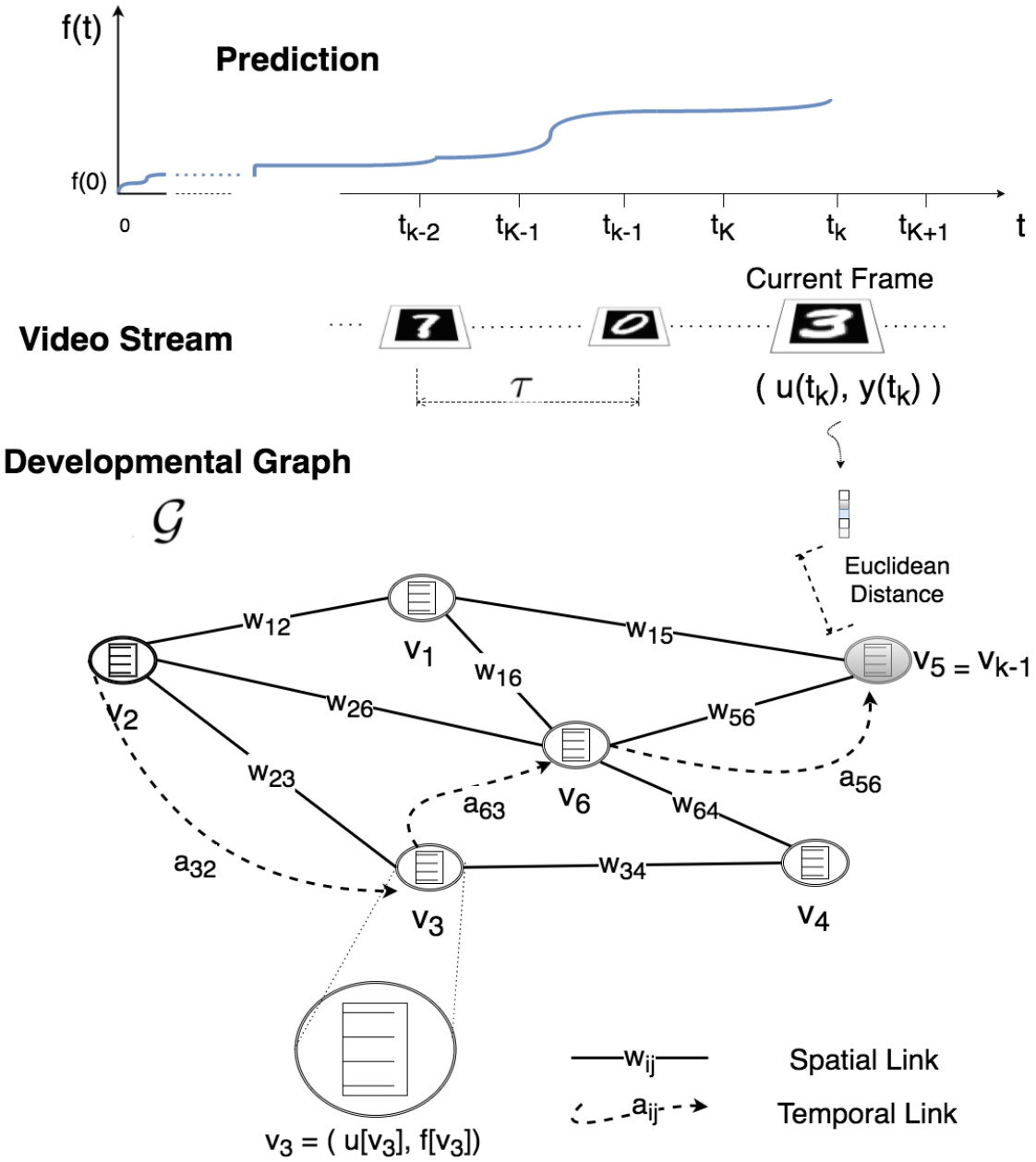


Figure 3.1: Simplified visualization of the node selection process. At the top we show the evolution of the prediction $\bar{f}$ in time. Below, the input sequence, in which each frame is synchronized with the system updates. Then, at the bottom, we show the structure of the graph. The dashed arrows (labeled by $a_{.,.}$) indicate the path generated by the input sequence on the graph nodes. The solid connections (labeled by $w_{.,.}$) represent the spatial links. We report the zoom of a node (v_3) in order to remark the fact that each node contains both a representative vector ($u[v_3]$) and the correspondent prediction ($\bar{f}[v_3]$).

which decreases with respect to $\|u[v_i] - u[v_j]\|^2$. The deviation σ is a hyper-parameter that, in our case, is computed as the mean of the Euclidean distances among the nodes in V_G.

In our implementation the matrix W_G is evaluated on–the–fly by computing only the weights related to the last added node to speed up the process, even if this approach could introduce an approximation in the estimation of the neighborhood of the oldest nodes.

3.2.3 Statistics on Temporal Correlations

The introduction of this term is strictly related to the development of this framework aiming to face the processing of sequences of data and can be easily explained when considering the case of video streams. Let us consider for example two consecutive frames representing the same moving object, provided that the representation is not too different (i.e. the subject changes slowly with respect to the frame rate). Let us assume that such a object can be located into the image window in different frames, so as that we have to compare two different representations for which we would like to predict the same class. A simplified case is the one in which each image represents only one object, such that the comparison between the two frames is straightforward. However, even when undergoing to slow changes, two consecutive representations could be very different at pixels level, because of possible translations, scaling, occlusions, changes of light, rotations and so on. As a matter of fact, the comparison between input images from consecutive frames by a distance measure could produce not very informative results. Apart from the specific example presented, various problems, generalized under the concept of the *curse of dimensionality* [108], arise when data are represented in a high dimensional feature space. The input space could also be characterized by regions with different density and, hence, the plain spatial information could be sometimes not completely informative, especially regarding points close to the class boundaries. As a for instance, the exploitation of the temporal adjacency could provide an approach trying to correct these variations and discover further relations among the samples of the same class. Focussing on the architecture of the network defined in Section 3.2.1, each node represents various occurrences of similar input samples provided in different splits of the sequence.

We have to consider the possibility that points close to the boundary are probable to be adjacent sometimes to nodes of other classes. Since the input trajectory is assumed to be continuos, this happens each time the function $u(t)$ crosses a separation surface between two classes. In practice however, these kind of problems are usually introduced by some occasional discontinuity in the input (for example a change of scene or an occlusion, considering again the video processing). The assumption to process a large (and potentially infinite) amount of real data, regardless of the number of labels, should allow us to acquire a statistically meaningful variety of temporal patterns containing the same node. Eventually, the occurrences of the correspondent class should overcome the others correcting the spatial information. We introduced this idea in (3.8) by the constraint:

$$\mathcal{T}(\bar{f}) = \sum_{k=1}^{N} \bar{\psi}|_{t_k} \cdot a_k \cdot \sum_{j=1}^{|V_G|} a_{kj} \cdot (\bar{f}(t_k) - \bar{f}[v_j])^2. \tag{3.13}$$

Again, we impose a quadratic penalty on the output, weighted in this case by the elements[2] a_{kj} of the *temporal adjacency matrix* $A_G \in \mathbb{R}^{|V_G| \times |V_G|}$, whose entries represent the temporal correlations between nodes. The entries of A_G are initialized to zero, then an element a_{kj} is incremented by one every time the input moves from node v_j to node v_k. The coefficient a_k in (3.13) represents a statistical normalization coefficient computed by:

$$a_k = \frac{\max_j(a_{kj})}{\max_{ij}(a_{ij}) \cdot \sum_{j=1}^{|V_G|} a_{kj}} \,. \tag{3.14}$$

The summation term $\sum_{j=1}^{|V_G|} a_{kj}$ averages the contributions with respect to the elements in the row of the node v_k. The element $\max_{ij}(a_{ij})$ scales the single weights with respect to the global absolute value of the temporal correlation. It is an index of how much confident are the statistics of a node with respect to the most frequent pair. Each row k is multiplied by its maximum $\max_j(a_{kj})$, so as to assign maximum weight 1 to the nodes in the row with the most significant statistics. This normalization is also useful to balance the wrong correlations introduced in the transition zones of the data sequence. An ideal situation should arise when, at a certain point during the training, the size of V_G becomes more or less stable.

[2] We remark that, at this point, we defined the correlation between the indexes k and K and they are freely interchangeable to refer to the same node at step K.

At this point, the increasing of the statistics reliability, together with the spatial regularization, should allow the system to refine the stored predictions at each passes on the nodes, especially on the unlabeled ones, so as to improve final performances by taking into account more and more informations. As already said, this approach still relies on the number of available data, but, as a matter of fact, it mainly exploits data without supervision, improving its confidence as time goes by, without requiring to have all the data available at the beginning or to reset the trained variables.

3.2.4 Composing the Impulse Response

As already said, the synchronization of all the contributions allows a simple way to compose an iterative updating formula. Despite of the introduction of multiple terms in the cost functional in (3.8), the correspondent distributional Euler–Lagrange equation can be derived by simply adding the derivatives of the quadratic loss in the new terms multiplied by the Dirac's Delta impulse, as for the the term coming from the supervision already present in (3.2), since all the others elements are in fact constants with respect to $\bar{f}$. Once we define the details of the computation of $\mathcal{S}(\bar{f})$ and $\mathcal{T}(\bar{f})$, we can derive the Euler–Lagrange equation for (3.8) as:

$$\begin{aligned} \frac{1}{\gamma}\,\hat{P}(\varphi\, P\bar{f}) + \sum_{k=1}^{N} \bar{\psi}|_{t_k}\,\left[\,\bar{f}(t_k) - y_k\,\right]\delta(t-t_k) \quad + \\ +(1-\eta)\sum_{k=1}^{N}\frac{\bar{\psi}|_{t_k}}{\rho}\sum_{s=1}^{\rho} w_{k,n_{ks}}[\,\bar{f}(t_k) - \bar{f}[v_{n_{ks}}]\,]\delta(t-t_k) \quad + \\ +\eta\sum_{k=1}^{N}\bar{\psi}|_{t_k}\cdot a_k\cdot\sum_{j=1}^{|V_G|} a_{kj}\cdot[\,\bar{f}(t_k) - \bar{f}[v_j]\,]\delta(t-tk) \quad = \quad 0. \end{aligned} \tag{3.15}$$

Starting from the (3.6), the discretized solution of the previous Euler-Lagrange equation can be computed on–line with the updating formula:

$$\mathbf{f}[K+1] = e^{A\tau}\mathbf{f}[K] + e^{A\tau/2}\cdot B\cdot\mathbf{E}[K]. \tag{3.16}$$

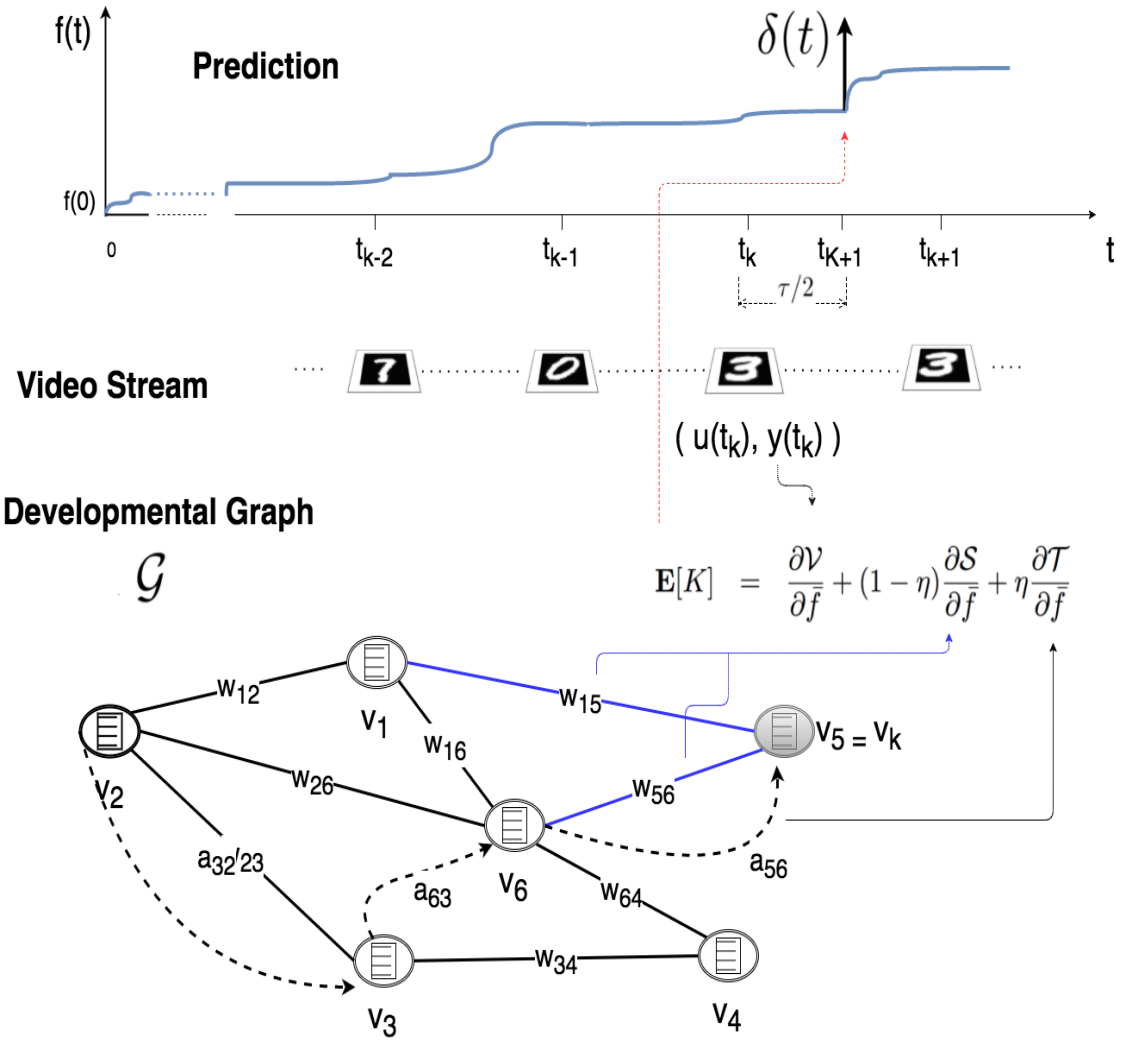


Figure 3.2: Simplified visualization of the impulse generation process. After the input sample presented at t_K is associated to a node v_K, the system reacts with an impulse ideally delayed by a step of $\tau/2$. The magnitude of the impulse is obtained by modulating the possible supervision, the contribution $\mathcal{S}(\bar{f})$ coming from the spatial weights w and the term $\mathcal{T}(\bar{f})$ coming from the temporal connections. Then, the system is updated such that the prediction on the next sample at t_{K+1} takes into account the information processed at t_K.

The term $\mathbf{E}[K]$ represents the extension of the term Δ_K in the (3.7) and collects the derivatives from all the contributions in (3.8). It can be composed by a simple summation as:

$$\begin{aligned}
\mathbf{E}[K] &= \frac{\partial \mathcal{V}}{\partial \bar{f}} + (1-\eta)\frac{\partial \mathcal{S}}{\partial \bar{f}} + \eta \frac{\partial \mathcal{T}}{\partial \bar{f}} = \\
&= \left[\, \bar{f}(t_k) - y_k \,\right] + \\
&+ \frac{(1-\eta)}{\rho} \sum_{s=1}^{\rho} w_{k,n_{ks}} [\, \bar{f}(t_k) - \bar{f}[v_{n_{ks}}] \,] + \\
&+ \eta \cdot a_k \cdot \sum_{j=1}^{|V_G|} a_{kj} \cdot [\, \bar{f}(t_k) - \bar{f}[v_j] \,]. \qquad (3.17)
\end{aligned}$$

In Figure 3.2 we tried to give a graphical explanation of the computation of all the terms composing $\mathbf{E}[K]$.

It worths to remarking that the computation of the derivative of the input function $u'(t)$ in the evaluation of the term $b(t) = \sqrt{1 + \|u'(t)\|^2}$. Indeed, in general is impossible to have the explicit form of a function which describes the input trajectory. However, an approximation can be computed by the Finite Differences Method [109]. Exploiting the first order formula, and the chosen sampling step τ, we can write:

$$u'(t_k) = \frac{u(t_k) - u(t_{k-1})}{\tau}. \qquad (3.18)$$

Anyway, the precision of this approximation is not a central issue, since the term represents a scaling factor among the impulses that should maintain the same rate by changing the approximation order.

3.3 Experiments

The proposed theoretical formulation assumes the periodicity of data in time, restricting the initial analysis to the trajectory optimization on a finite interval. Nevertheless, the practical implementation allows a full open scenario, in which the dynamic system elaborates data at a local level. Hence in this Section we

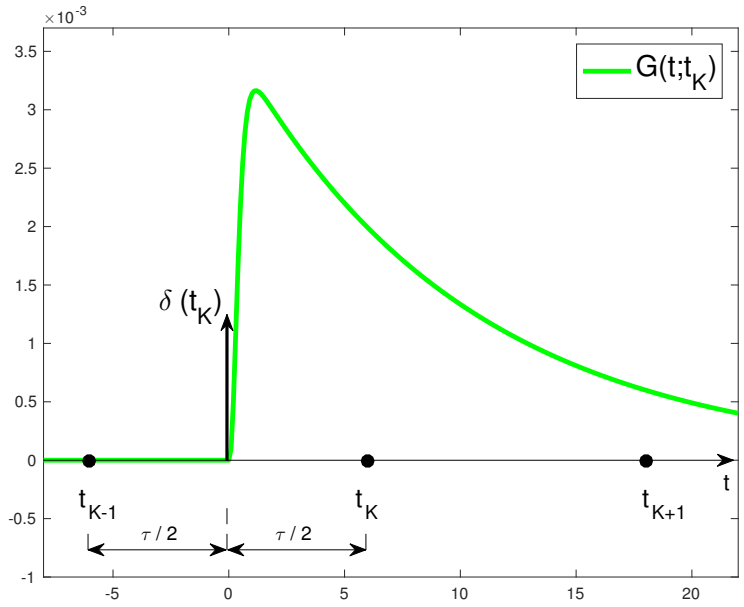


Figure 3.3: Impulse Response and sample spacing.

propose an experimental analysis on data that embody a good statistical representation without requiring to set up a direct periodicity by iterating over them. Indeed, data will be organized in a single stream, in which all the information is already assumed to be repeated many times in different situations, in order to reply as well as possible real scenarios. We also try to propose a purely on–line evaluation, in which we overcome the separation between training and test data. Again, we start in Section 3.3.1 by a first exploration on an artificial dataset, in order to better analyze the behavior of the algorithm with respect to each parameter. Hence, in Section 3.3.2 we propose a similar evaluation on a real data set.

In this case we found convenient to configure the dissipative model by setting $\theta = 10$ and a fourth order system with roots:

$$\lambda = \{ -0.1,\ -6,\ -6.5,\ -7.4 \}$$

that give the impulse response of Figure 3.3. Even if in practice the dissipation is not only related to the absolute value of θ, but also to its product with the sampling step τ, we found useful to increase θ to guarantee stability. Indeed, an impulse is produced at each step, due to summation of all the contributions, requiring the

magnitude of the response to be smaller, even if a similar regularization effect could had been achieved by reducing the regularization parameter γ too. The *memory* of the system is set again by the first roots $\lambda_1 = -0.1$. This time the value is not too small, since we would like each impulse to affect the model predictions only on few incoming samples.

3.3.1 MNIST

To create an artificial sequence, which is easy to manage and, at the same time, follows a natural-like behavior, we generated a video from the MNIST[3] dataset [24]. This choice is due to several simplifications. The transformations on data are quite simple and intuitive, obtaining a set of smooth deformations that seem to be enough natural. The samples are completely labeled, allowing a precise evaluation and, since the proposed algorithm is at this step a pure classifier, to skip a feature extraction part or others preprocessing that could, in some way, make the comparison more difficult. We randomly selected 5240 points (equally distributed per class) from the original training data (60000 examples of 28×28 images of handwritten digits). We generated a sequence by applying 60 consecutive small transformations on each sample. We obtained a video of 32400 frames in which each sample, i.e. its various transformations, appears for 2 seconds.The sequence of transformations is created by randomly selecting at each step among:

- rotation, angle in $[-20°, 20°]$ with a step of $4°$
- translation, up to 3 pixels in each possible direction
- scaling, few pixels bigger or smaller
- blurring, obtained by a Gaussian filter with parameter in $\{0.25, 0.5, 0.75, 1\}$

Since the images are 28×28 pixels and we feed the classifier with the 1-D vector of grey levels (normalized in $[\,0,\,1\,]$), these small transformations are sufficient to yield a quite different descriptor, as shown in the few consecutive images in Figure 3.4. The changes are not so evident at the first sight, but they are relevant at the feature level, as shown by the reported Euclidean distances from the correspondent predecessor.

[3] http://yann.lecun.com/exdb/mnist/

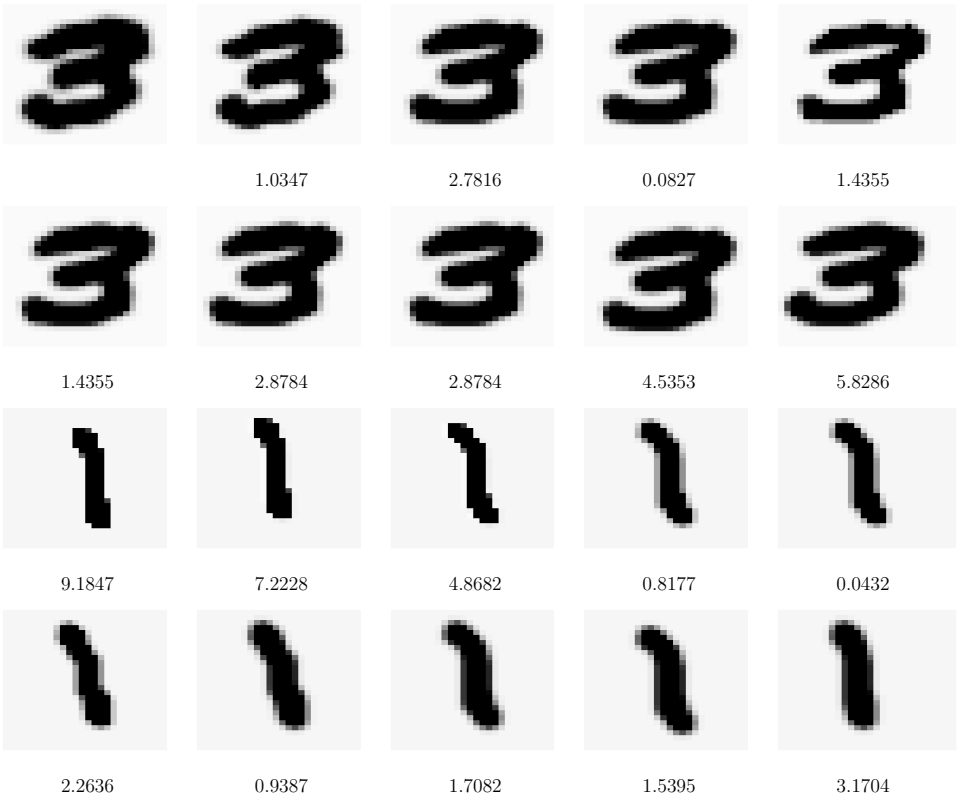


Figure 3.4: Examples of consecutive images from the training video generated by random transformations on a subset of the MNIST digits. Starting from the second frame we indicate the magnitude of the Euclidean distance from the previous one, used to approximate u' in (3.18). To compare the distances among samples from different classes, we report 20 frames from positions 51 to 70 in the sequence moving from class 3 to 1, ordered left to right and top to bottom.

We trained the agent with this sequence in different settings, by varying the number of supervisions along the stream. In order to perform the comparisons, the classification accuracy is evaluated on the original MNIST test set. After the processing of the training sequence, the predictions are given by the values of $\bar{f}$ stored in the graph, pairing each test sample with the best matching node, after a simple Nearest Neighbor search among the nodes of V_G. We report the comparison with the plain k–Nearest Neighbor algorithm and a 2-layer Artificial Neural Network, exploiting the correspondent Matlab built-in functions[4]. For the first one we set $k = 10$, which we found to achieve the best test performance in $\{1, 5, 10, 25, 50, 100, 1000\}$. The network has 300 hidden units and the *hyperbolic tangent* as activation in the hidden layer, the *softmax* as output function

[4] `fitcknn` and `patternnet` respectively.

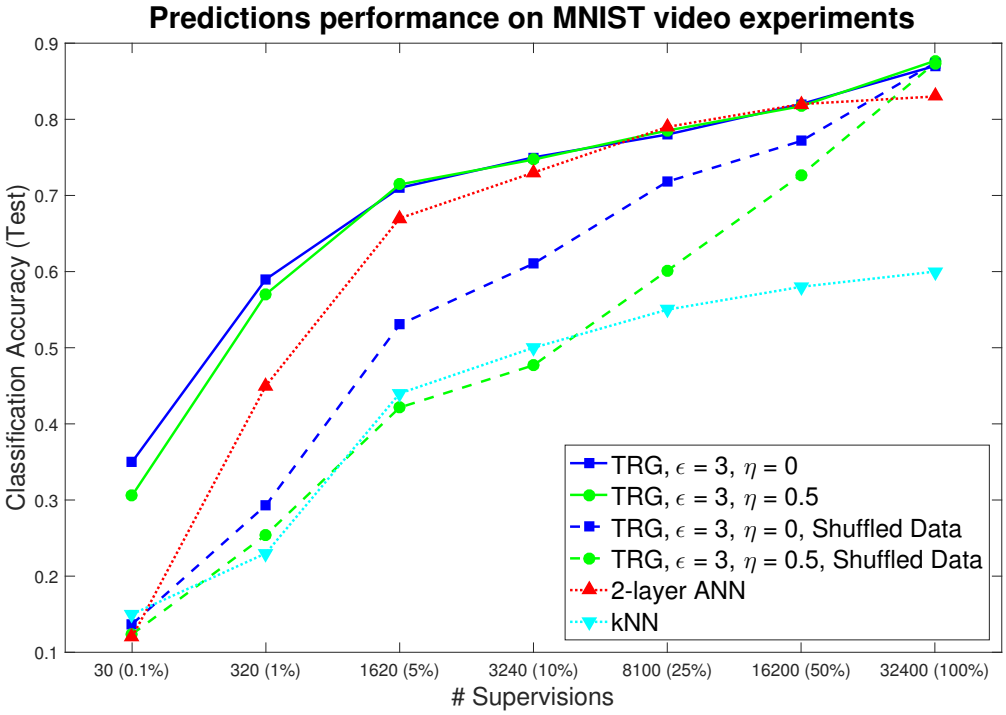


Figure 3.5: Plots of classification accuracy on the MNIST test set, decreasing the number of supervisions among the samples of the training sequence. We compare our algorithm (TRG), varying $\eta \in \{ 0, 0.5 \}$, with a k-Nearest Neighbors (kNN) classifier and a 2-layer Artificial Neural Network (2-layer NN). We propose also the results on TRG after the training on a sequence obtained by operating a random permutation on the original one (Shuffled Data).

and the *Cross-Entropy* as penalty. The training function automatically selects a part of data for the validation phase, preventing overfitting (which is useful because of the reduction of the training set). The radius of the ϵ-Net spheres had been tested in $\{10^{-4}, 1, 3, 5, 8\}$. The smaller values generate graphs with more nodes, increasing the representation capability and the computational cost but not always the prediction performances. A good tradeoff was found at $\epsilon = 3$, that generates a graph with about 8000 nodes ($\approx 25\%$ of the available samples). The remaining parameters are chosen in a validation phase on the first 10% of the sequence. A small regularization parameter γ amplifies the response (improving the performance) but leads to divergence (because of an accumulation of the delay of the impulse responses). A good balance is found at $\gamma = 0.01$ (combined with the used impulse response). We selected $\sigma = 3$ from $\{0.5, 1, 2, 3\}$, $\tau = 12$ from $\{1, 2, 4, 6, 12, 18\}$, and $\rho = 5$ from $\{1, 5, 10, 25, 50\}$ so as to achieve the

best accuracy on the last 20% of the validation sequence. The value of η was tried in $\{0,\ 0.5\}$. We neglected to report the case $\eta = 1$ since the spatial information came out to be fundamental for reasonable predictions. The results are reported in Table 3.1 and shown in Figure 3.5 with respect to the number of provided supervisions. We report also the running time for each experiment (for a correct reading, notice that the proposed algorithm exploits also the unsupervised samples). Each test is carried out 3 times with supervisions on different samples (randomly chosen) and the results are averaged[5].

The first consideration we would like to do regards the fact that the proposed structure does not need to iterate on data to improve performance, despite of standard algorithms (as for example the compared ANNs) which in general require several epochs of training. By analyzing results from Table 3.1, we can notice that the ANN is obviously faster when decreasing the number of supervisions (since the unlabeled data are discarded), but the higher computational efficiency does not balance the performances decay. On the opposite, the graph shows, for a given ϵ, the same running times and, even with the completely supervised setting, time and performance are still good. We report the outcomes of training the graph with a random shuffling of the sequence, leading to a worsening of results. Indeed, despite of the formulation proposed in Chapter 2, in this case the smoothness of data variations through the temporal dimension is an essential hypothesis. Moreover, the computational time is quite bigger for the same case. This is due to the fact that the on–line nearest neighbor search defined in (3.9), avoids to search for the actual nearest neighbor at each step, relying again in the temporal smoothness of data. By randomizing the process, we forced the algorithm to leave the last node and perform an exhaustive search almost at each step. We can notice that the statistics about temporal patterns introduced in Section 3.2.3 assumed a positive effect only in few cases, especially when decreasing both the labels and the size of V_G. This could be due to the fact that the sequence is artificially generated, such that is does not provide a large variety of different patterns containing the same node to build up a meaningful statistics.

Starting to implement the learning scenario discussed in the introductory steps, we propose an on–line evaluation of the algorithm directly on the train-

[5]MATLAB scripts and a part of the generated video from the MNIST dataset are available at `https://github.com/alered87/Graph-Regularization`

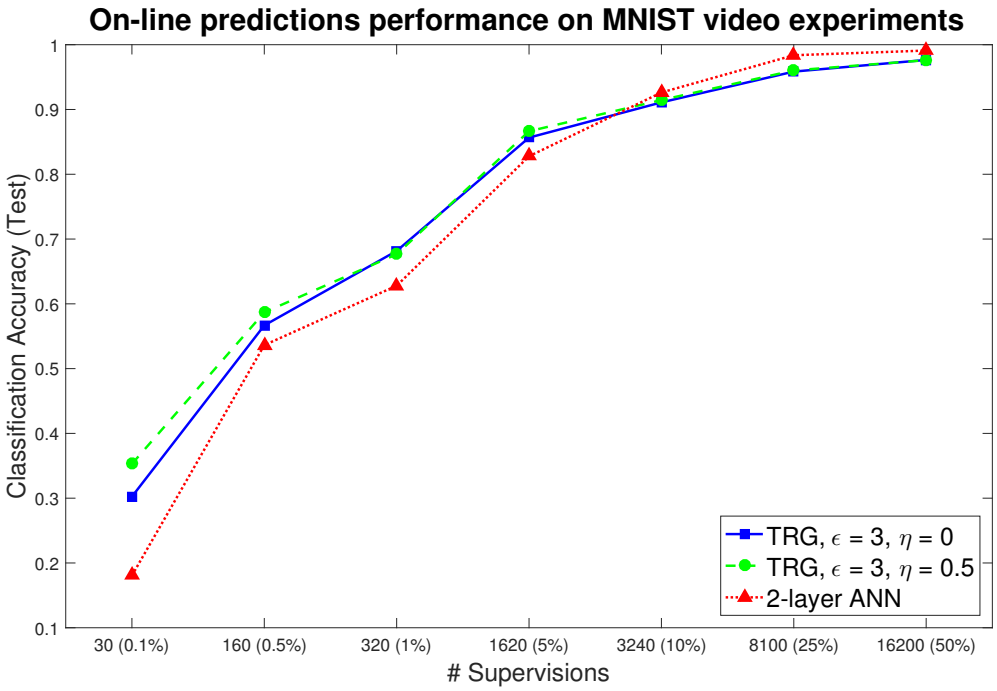


Figure 3.6: Plots of classification accuracy on half of the MNIST sequence, decreasing the number of supervisions among the others samples (test and training samples are separated). We compare the on–line predictions of the proposed algorithm (TRG) varying $\eta \in \{ 0, 0.5 \}$ and a 2-layers Artificial Neural Network (2-layer NN), trained in the standard setting.

ing sequence. We repeated 3 runs, averaging the results, for each different setting obtained decreasing the number of supervisions. At each attempt, half of the training data are randomly selected for the test set, halving the maximum number of supervisions available (from 32400 to 16200). Because of the poorness of previous results, we neglected the k–NN algorithm in the comparison. In this configuration, the graph processes all the data once, and the prediction are evaluated at the frame instants, so that the predictions at the beginning of the sequence are in some way worse. However, to allow a feasible comparison the ANN is trained in the standard setting, extracting and shuffling the supervised data from the labeled ones. More than before, a crucial issue is a meaningful employment of the validation set, preventing the network to overfit the training samples. This time, we can notice, from Table 3.2 and Figure 3.6 how the use of temporal statistics allows us to improve the performance when decreasing the number of supervisions.

3.3.2 MusicNet

We evaluated the proposed model also on a benchmark based on audio signals that are characterized by a natural continuous distribution and, at the same time, allow us a straightforward feature extraction procedure. We employ data from MusicNet [110], a large corpus containing 34 hours of chamber music performances from 10 authors and 11 instruments under various studio and microphone conditions, organized in 330 records. Over 1 million of labels are provided on segments of these tracks, indicating instrument/note annotations. Following the procedure explained at the official site[6], we extracted the first 5 sequences together with the correspondent notes divided into 128 classes with possible multi-labels.

We exploited the spectral representation of the digital sampling in the Mel frequency scale, as computed by the package *librosa* [111]. We generated a vector of 40 features from each 2048 samples window, sliding with a step of 512 elements. We dropped several classes because of the dataset reduction and, hence, we focused on the most frequent 30, obtaining a total of 123877 samples, from which the last 20% (24775) is used for the test phase. We employ the dynamic system described at the beginning of Section 3.3, whereas the other parameters are determined in a validation step exploiting the first 10% of the training sequence. This part of data is used to compute *min* and *max* to normalize the features in $[\,0,\,1\,]$, which we found more convenient with respect to another normalization since we have to compute Euclidean distances in a high dimensional space. The mean of the distances among validation data (which we found ≈ 1) is used both to estimate σ and ϵ, that we fixed to 0.45 (slightly lower than the half of its value) so as to store about 23000 nodes in the graph over the 99102 samples from the training sequence ($\approx 25\%$). In a setting analogous the one proposed in previous experiments, we selected $\sigma = 1$ from $\{0.75, 1, 1.25\}$, $\tau = 3$ from $\{3, 6, 12, 18\}$, $\rho = 5$ from $\{5, 10, 15\}$ and $\lambda = 0.01$ from $\{0.02, 0.01, 0.005\}$ so as to achieve the best accuracy on the last 20% of the validation sequence. We compared the proposed model with a 2-Layer Artificial Neural Network, for which we found the best configuration (searched by varying the transfer function and both the size and the number of the layers) with 300 units in the hidden layer and the ReLU as activation. The output function and the penalty are forced to

[6] `https://homes.cs.washington.edu/~thickstn/musicnet.html`

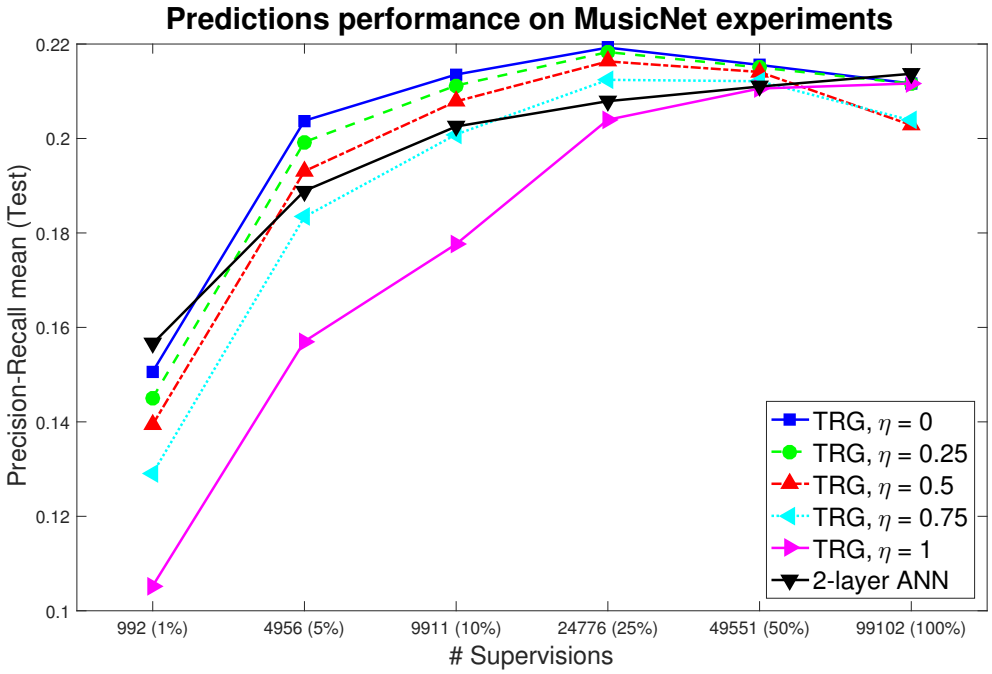


Figure 3.7: Plots of Precision-Recall average on the MusicNet test set (the last 20% portion of the sequence), decreasing the number of supervisions. We compare our algorithm (TRG), varying η, with a 2-layer Artificial Neural Network (2-layer ANN).

logistic sigmoid and *Mean Square Error* respectively, since in this case we have to deal with multi-label prediction (i.e. more than one note could appear in the same frame). The optimization process exploits the *resilient back-propagation* algorithm [112] and GPU computing to speed up the computation (and hence in this case we avoided the comparison on the training time). We found that a standard statistical normalization (zero-mean and standard deviation 1) yields a better performance for this architecture. A first remark has to be done on this part. In the neural network optimization the test set is normalized by using the mean and the standard deviation evaluated on the training set, since the global normalization should be more favorable in this case. Nevertheless, since we propose to deal with a pure on–line setting, it is in practice not possible to have in advance the true parameters for the global normalization of the training sequence for the graph. Hence, both the training and the test sequence are normalized (almost) in $[\,0,\,1\,]$ with respect to *max* and *min* evaluated on the correspondent validation set.

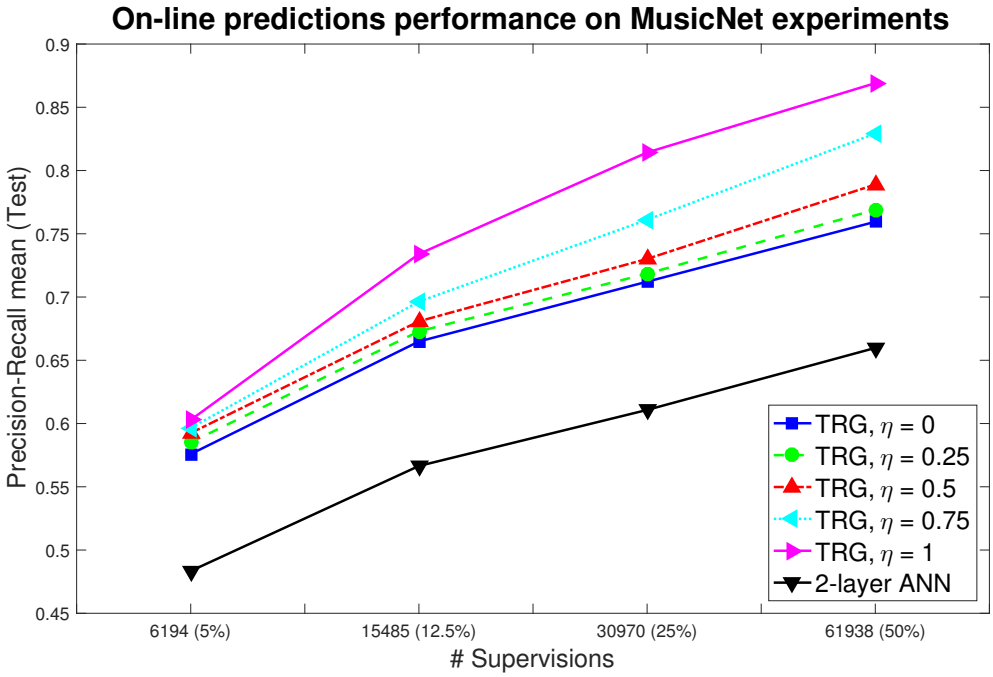


Figure 3.8: Plots of Precision-Recall average on half of the MusicNet sequence, decreasing the number of supervisions randomly selected among the others samples. We compare our algorithm (TRG), varying η, with a 2-layer Artificial Neural Network (2-layer ANN).

In Table 3.3 and Figure 3.7 we report the average of Precision and Recall evaluated over the 30 classes. All the proposed settings are tested on 5 trials by randomly selecting the labels. The parameter η, balancing spatial and temporal contributions, is tested in $\{0, 0.25, 0.5, 0.75, 1\}$. However, even in this setting, we can not appreciate any improvement when adding statistics on the sequence temporal patterns.

As in the case of Section 3.3.1, we report the results of an on-line evaluation in Table 3.4 and Figure 3.8. We considered the whole stream of 123877 elements and randomly selected half of the samples for test, evaluating the predictions of the graph on–line during the training, varying the number of supervisions on the other half of data, averaging the performance over 5 trials for each setting. At each run the 2-Layer Artificial Neural Network is trained and tested on the same sample split in the standard setting. This time the contribution of the penalty term $\mathcal{T}$ leads to a considerable improvement of performance, especially in the most supervised case when the information propagated along the sequence is

more reliable. In this case, the representation in the feature space provided by the graph nodes seems to be quite informative, making the contribution of the spatial regularization from the term $\mathcal{S}$ not very significant.

ϵ	η	Supervisions 0.1 30	 1 320	 5 1620	 10 3240	 25 8100	 50 16200	 100 32400	 % #	Nodes
10^{-4}	0	0.30	0.58	0.72	0.75	0.78	0.82	0.87	Accuracy	32400
	0.5	0.28	0.57	0.72	0.75	0.79	0.82	0.87		
		5638.3	5860.8	5821.7	5590.6	5799.5	5395.4	5326.6	Time (sec.)	
1		0.25	0.59	0.71	0.75	0.79	0.81	0.87		16500
		0.27	0.56	0.71	0.76	0.77	0.80	0.87		
		1018.2	1337.5	1041.0	998.9	1423.6	1463.3	573.3		
3		**0.35**	**0.59**	**0.71**	**0.75**	**0.78**	**0.82**	**0.87**		8000
		0.30	**0.57**	**0.71**	**0.74**	**0.78**	**0.81**	**0.87**		
		360.95	306.91	329.42	412.67	431.24	414.35	320.05		
3		0.13	0.29	0 .53	0.61	0.71	0.77	0.87		8600
Shuffled		0.12	0.25	0.42	0.47	0.60	0.72	0.87		
Data		1434.6	1258.3	1369.8	1598.4	1441.4	1604.7	1430.1		
5		0.34	0.55	0.67	0.73	0.77	0.79	0.85		3500
		0.35	0.56	0.69	0.73	0.76	0.80	0.85		
		258.05	255.24	254.93	226.35	254.59	258.48	228.81		
8		0.27	0.50	0.61	0.66	0.67	0.69	0.70		300
		0.33	0.52	0.64	0.65	0.68	0.68	0.70		
		13.3	11.31	11.68	10.87	10.59	10.77	10.95		
k-NN	$(k = 10)$	0.15	0.23	0.44	0.50	0.55	0.58	0.60		–
		0.44	1.26	4.81	13.64	34.70	62.43	98.11		
2-layer ANN	(HU= 300)	0.12	0.45	0.67	0.73	0.79	0.82	0.83		–
		2.14	4.05	17.09	73.98	204.87	373.59	640.01		

Table 3.1: Classification accuracy on the MNIST test set and training time decreasing the number of supervisions. We compare our algorithm (TRG), varying ϵ, with a k-Nearest Neighbors (kNN) classifier and a 2-layer Artificial Neural Network (2-layer NN). The best case $\epsilon = 3$ is trained also on a sequence obtained by a random permutation of the original one (Shuffled Data).

		Supervisions							
		0.1	0.5	1	5	10	25	50	%
	η	30	160	320	1620	3240	8100	16200	#
TRG	0	0.30	0.56	0.68	0.85	0.91	0.95	0.97	
	0.5	0.35	0.58	0.67	0.86	0.91	0.96	0.97	
2-layer ANN (HU=300)		0.18	0.53	0.62	0.82	0.92	0.98	0.99	

Table 3.2: Classification accuracy on half of the MNIST sequence, decreasing the number of supervisions among the others samples (test and training samples are separated). We compare the on–line predictions of the proposed algorithm (TRG) varying $\eta \in \{0, 0.5\}$ and a 2-layer Artificial Neural Network (2-layer ANN) trained in the standard setting.

		Supervisions						
		1	5	10	25	50	100	%
	η	992	4956	9911	24776	49551	99102	#
TRG	0	**0.1507**	**0.2038**	**0.2135**	**0.2192**	**0.2156**	**0.2116**	
	0.25	0.1451	0.1992	0.2112	0.2183	0.2150	0.2116	
	0.5	0.1395	0.1931	0.2079	0.2163	0.2141	0.2028	
	0.75	0.1291	0.1834	0.2009	0.2124	0.2121	0.2038	
	1	0.1053	0.1570	0.1776	0.2040	0.2106	0.2116	
2-layer ANN (HU=300)		0.1566	0.1889	0.2025	0.2079	0.2110	0.2137	

Table 3.3: Precision-Recall average on the MusicNet test set (the last 20% portion of the sequence), decreasing the number of supervisions. We compare our algorithm (TRG), varying η, with a 2-layer Artificial Neural Network (2-layer ANN).

		Supervisions				
		5	12.5	25	50	%
	η	6194	15485	30970	61938	#
TRG	0	0.5758	0.6648	0.7121	0.7594	
	0.25	0.5852	0.6729	0.7184	0.7690	
	0.5	0.5925	0.6808	0.7301	0.7889	
	0.75	0.5965	0.6964	0.7610	0.8293	
	1	**0.6029**	**0.7340**	**0.8143**	**0.8691**	
2-layer ANN (HU=300)		0.4834	0.5664	0.6107	0.6597	

Table 3.4: Precision-Recall average on half of the MusicNet sequence, decreasing the number of supervisions randomly selected among the others samples. We compare the on–line predictions of the proposed algorithm (TRG) varying $\eta \in \{ 0, 0.25, 0.5, 0.75, 1 \}$ and a 2-layer Artificial Neural Network (2-layer ANN), trained in the standard setting.

4 Conclusions

We delimited the foundations of a new approach to Machine Learning optimization methods aiming to face real configurations in which there is not a sharp cut between training and test set. The learning agent is assumed to be immersed in its own environment and should be able to process and interact with the incoming information in real time. The formulation is proposed by devising a framework typical of Classical Mechanics, in which the involved variables are interpreted as a system of particles. Hence, outlining a parallel with the Principle of Least Action, we found the equations of motions embedding the forces from the environment, so as to obtain the resulting description of the optimal trajectory which depicts the learning process. In this analysis, a fundamental hypothesis is the interpretation of the optimization procedure in the dimension of time, allowing to derive a manageable form of the solution. Moreover, this change of domain simplifies the matching to real world scenarios in which data are characterized by a precise temporal order. The result is an optimization procedure suitable for almost any kind of parametric learning structure.

Starting from a clear outline of the theoretical formulation, we derived a closed form of the equations describing the evolution of the analyzed system, allowing us a feasible practical implementation and an empirical validation of the soundness of the theory. In particular, the proposed experimental analysis focused on the behavior of the model with respect to the different parameters, showing the correspondence between practical results and theoretical principles. Particular attention is paid to the crucial role of *dissipation*, which should embody the counter part of the memory in living being and regulate, in fact, the whole learning mode. The exploration allow us to examine the stability condition and, moreover, the convergence towards suitable solutions. The application

of the model to standard learning algorithms confirm the interpretation of the theory as a good representation for their general formulation. Indeed, under specific conditions, it allows us to express well known optimization procedures, as for example the *Stochastic Gradient Descent* technique, by laws of motion described within classical Physics frameworks.

The obtained results allow us to step up to a higher level of abstraction, making use of the same laws to model a learning function characterized by an intrinsic temporal smoothness as a fundamental driving principle. The nature of this agent is then paired with a representational structure built as an incremental graph, acting as an additional regularization term similar to the ones proposed in classic Manifold Regularization on the input feature space. In this approach, the formulation coming from Laplacian regularization, which allows to obtain the optimal solution by defining a convex optimization problem, is lost. However, the proposed framework easily integrate the on–line formulation, making it possible to deal in a feasible way with infinite stream of data. Fundamental ideas exploiting smoothness assumptions on spatial manifold are preserved by the inclusion of correspondent constraints in the described cost functional. Similar concepts are then extended to the temporal dimension by means of the introduced dynamic system. New kind of constraints, learned in an unsupervised way, are embedded in a simple and flexible way, making straightforward the exploitation of additional knowledge.

4.1 Future Directions

The theory introduced in Chapter 2 provides a method to tune the memory of the reaction to the incoming data in the dynamical system. Even if could be interesting to inspect similar applications in the fields of classical Control Theory, an exploration in the field of Machine Learning could concern those frameworks in which data evolves under the concept of drift [113, 114]. A crucial issue is related to the changes in time of the data distribution and the consequent adaptation of the learning agent. Indeed, the relation between the development of a global permanent knowledge and a good flexibility to local behaviors represents a well known problem, a.k.a. the *stability-plasticity dilemma* [115]. The employment

of the proposed method to control the memory and the promptness of the learning system with respect to external changes could be straightforward, taking into account the generality of the applicable constraints.

Similar proposals can be done on the work described in Chapter 3. The developmental structure proposed can be easily adapted to deal with problems which require to tune the memory of the learning agent by varying the considered temporal window (because of the drift). This problem is faced in many cases by the employment of architectures similar to the one proposed in Section 3.1, that are developed at different levels defining the concepts of Long and Short Term Memory [116, 117].

Desired developments surely regard the embedding in more complex architectures focused on Computer Vision tasks. Indeed the structure is inspired in some way to the high level classifier employed in [8] and the flexibility of the proposed structure could be reason of interest. However, to obtain a faster validation, we would like to investigate the application to simpler tasks in which data ara characterized by a natural temporal evolution and provide a range of temporal patterns allowing meaningful statistics. Another field of investigation could be obtained by the application to configurations in which many objects are assumed to appear in the same image. In this case, an approach similar to the one in [118] can be exploited to obtain a first roughly partitioning of the scene by bounding boxes. Then, the development of logic reasonings on the environment can be easily embedded in the graph of Section 3.1 exploiting the same procedure analyzed for spatial and temporal constraints. As already said, important features of the structure are its lightness and promptness of response, allowing us to process data on–line. Even if the agent is in fact employed as a classifier, the capability to interact with the environment in real time, taking into account both external teaching and unsupervised constraints, could be exploited to provide feedbacks for the creation of the feature descriptors or the image segmentation.

Appendix

A.1 Euler-Lagrange Equations

Stationary points for a functional of the kind of the one in (2.8) can be found by the satisfaction of the Euler-Lagrange equations, derived by imposing the first variation to be null. In this section, in order to not make the notation too heavy, we will restrict the analysis to a system with only one particle, referring to the only element of the parameter vector by w. This does not affect the general derivation since there are not interdependencies among the variables. We will use the same approach proposed in the classic case introduced in Chapter 1.1. The formulation is set as a fixed end-point boundary value problem requiring to find $w \in \mathcal{F}$ to be the optimal trajectory for the functional $A[w]$, where $\mathcal{F}$ is a suitable functional space. If $w(t_0) = w_0$ and $w(t_e) = w_e$ are the boundary conditions for w, we restrict the set of the solutions to:

$$S = \{f \in \mathcal{F} \ : f(t_0) = w_0 \text{ and } f(t_e) = w_e\} \ .$$

We will consider the correspondent space of test functions:

$$H = \left\{h \in \mathcal{F} \ : h^{(k)}(t_0) = h^{(k)}(t_e) = 0 \, , \ k = 1, \ldots, \ell - 1\right\}$$

such that if we take $h \in \mathcal{S}$ and $\epsilon \in \mathbb{R}$, with $\epsilon > 0$ small enough, we have that a perturbation $\hat{w}$ of the original trajectory w can be written as $\hat{w} = w + \epsilon h \in S$. In this way $\hat{w}$ is still a solution belonging to the hypothesis space, i.e. $\hat{w} \in S$. The condition for w to be a stationary point for $A[w]$ is expressed by the *first variation* $\delta(A)$ to be null:

$$\delta A = \lim_{\epsilon \to 0} \frac{A[\hat{w}] - A[w]}{\epsilon} = 0 \ , \qquad \forall h \in H \, . \tag{A.1}$$

If we rewrite $A[\hat{w}]$ by the Taylor expansion approximated to the first order, exploiting the linearity of P so as that $P[w + \epsilon h] = Pw + \epsilon Ph$, we have:

$$\begin{aligned} A[\hat{w}] &= A[\, t,\, w + \epsilon h,\, Pw + \epsilon Ph \,] = \\ &= A[\, t,\, w,\, Pw \,] + \epsilon \left(h \cdot A_w + Ph \cdot A_{Pw} \right) + o(\epsilon^2) \end{aligned}$$

so that the (A.1) can be written as:

$$\delta A = \epsilon \left(h \cdot A_w + Ph \cdot A_{Pw} \right) = 0 \tag{A.2}$$

where $A_w = \partial A / \partial w$ and $A_{Pw} = \partial A / \partial Pw$. At this point, in order to provide a general solution and a manageable notation, it is convenient to rewrite the (A.2) within the framework of the Theory of Distributions [69]. The basic idea is to consider a class of functionals (the so called *distributions*) which map a set of suitable functions into real numbers. In our case it is convenient to consider the domain of such a functional as the set $\mathcal{I}_T$ of the functions from $\mathbb{R}$ to $\mathbb{R}^n$ (even if in practice we restricted the analysis to $n = 1$) that are integrable in $T = [\, t_0,\, t_e \,]$. Following the classic notation, we can generally define such a functional $\Gamma_h : \mathcal{I}_T \to \mathbb{R}$ as:

$$\Gamma_h(f) = \langle\, \Gamma_h,\, f \,\rangle = \int_R h(t) \cdot f(t)\, dt$$

where $h \in \mathcal{I}_T$ and $\langle\, \cdot,\, \cdot \,\rangle$ defines a scalar product. Exploiting this notation, we can write the (A.2) in the distributional form as:

$$\langle\, h,\, (\psi F)_w \,\rangle + \langle\, Ph,\, (\psi F)_{Pw} \,\rangle = 0. \tag{A.3}$$

By considering the properties of the adjoint operator $\hat{P}$ and its relation with the the integration by part formula[1] and the boundary conditions imposed on h belonging to the set H, the (A.3) can be written as:

$$\langle\, h,\, (\psi F)_w \,\rangle + \langle\, h,\, \hat{P}(\psi F)_{Pw} \,\rangle = 0 \tag{A.4}$$

[1] In the simplest case of $P = \frac{d}{dt}$ and $\hat{P} = -\frac{d}{dt}$ we have that $\int_a^b v \cdot Pu = vu|_a^b - \int_a^b Pv \cdot u$, that can be written in its distributional form as $\langle\, v,\, Pu \,\rangle = vu|_a^b + \langle\, \hat{P}v,\, u \,\rangle$

and since we required the condition to hold $\forall h \in H$, we obtain that:

$$\begin{aligned}(\psi F)_w + \hat{P}(\psi F)_{Pw} &= \psi \cdot (K + \gamma V)_w + \hat{P}[\psi \cdot (K + \gamma V)_{Pw}] = \\ &= \psi \cdot (\gamma V)_w + \hat{P}[\psi \cdot (K)_{Pw}] = \\ &= \gamma\psi \cdot V_w + \hat{P}\left[\psi \cdot \left(\frac{1}{2}\mu(Pw)^2\right)_{Pw}\right] = \\ &= \frac{\gamma}{\mu}\psi \cdot V_w + \hat{P}(\psi Pw) = 0 \end{aligned} \tag{A.5}$$

where the (2.9) is derived by considering that K and V only depend on Pw and w respectively.

A.2 Solutions to Euler-Lagrange and Routh-Hurwitz Conditions

A.2.1 First Order Operator

Characteristic Polynomial

Starting from (2.9) that was derived in the previous Section, we can substitute the general differential operator with the first order one $P = \alpha_0 + \alpha_1 D$, with formal adjoint $\hat{P} = \alpha_0 - \alpha_1 D$. By dropping the explicit dependence from t to make the notation lighter we obtain:

$$\begin{gathered}\hat{P}(\psi Pw) + \tfrac{\gamma}{\mu}\psi V_w = [\alpha_0 - \alpha_1 D]\left[\psi\left(\alpha_0 w + \alpha_1 Dw\right)\right] + \tfrac{\gamma}{\mu}\psi V_w = \\ = \alpha_0\left(\alpha_0\psi w + \alpha_1\psi Dw\right) - \alpha_1 D\left[\alpha_0\psi w + \alpha_1\psi Dw\right] + \tfrac{\gamma}{\mu}\psi V_w = \\ = \alpha_0^2\psi w + \alpha_0\alpha_1\psi Dw - \alpha_0\alpha_1 D[\psi w] - \alpha_1^2 D[\psi Dw] + \tfrac{\gamma}{\mu}\psi V_w = \\ = \alpha_0^2\psi w + \alpha_0\alpha_1\psi Dw - \alpha_0\alpha_1(\dot{\psi} w + \psi Dw) - \alpha_1^2(\dot{\psi} Dw + \psi D^2 w) + \tfrac{\gamma}{\mu}\psi V_w\end{gathered}$$

from which we can remove the second and the fourth elements as opposite. Because of the choice of $\psi(t) = e^{\theta t}$, we can notice that $\dot{\psi}(t) = \theta e^{\theta t} = \theta\psi(t)$, so

that we can divide the equation by $\psi(t)$ such that:

$$\begin{aligned}\alpha_0^2\psi w - \alpha_0\alpha_1\dot{\psi}w - \alpha_1^2\dot{\psi}Dw - \alpha_1^2\psi D^2w + \tfrac{\gamma}{\mu}\psi V_w = \\ = \alpha_0^2 w - \alpha_0\alpha_1\theta w - \alpha_1^2\theta Dw - \alpha_1^2 D^2w + \tfrac{\gamma}{\mu}V_w = \\ = D^2w + \theta Dw + \beta w - \tfrac{\gamma}{\mu}V_w\end{aligned}$$

where $\beta = \frac{\alpha_0\alpha_1\theta - \alpha_0^2}{\alpha_1^2}$ and the last line is derived by dividing by $-\alpha_1^2$, obtaining the form of (2.13).

Routh-Hurwitz Conditions

The Routh-Hurwitz stability criterion for a second order characteristic polynomials of the kind of the one in (2.15) requires all the coefficients to be strictly positive. This is always true for the coefficients of the second and first order terms, which are 1 and $\theta > 0$ respectively. Hence, the stability conditions require β to be positive so as to obtain the restriction:

$$\theta > \frac{\alpha_0}{\alpha_1} \tag{A.6}$$

assuring all the roots to have negative real part.

From solutions to parameters

In this Section we will derive the formulas linking the parameters θ, α_j to the produced roots λ_q of the characteristic polynomial in (2.15). This allows us to freely generate the desired impulse response directly by selecting λ_q so as to restrict the analysis to real roots cases[2]. Indeed $p(\lambda)$ can be written as:

$$\lambda^2 + \theta\lambda + \beta = (\lambda - \lambda_1)(\lambda - \lambda_2) = \lambda^2 - (\lambda_1 + \lambda_2)\lambda + \lambda_1\lambda_2$$

[2]Because of what said in Section 2.2.1 about the rate between the sinusoidal trend of the impulse response, due to possible complex roots, and the stability of the system.

and hence, by matching the coefficients of the same order, we find the conditions:

$$\begin{aligned} \theta &= -(\lambda_1 + \lambda_2) \\ \beta = \frac{\alpha_0\alpha_1\theta - \alpha_0^2}{\alpha_1^2} &= \lambda_1\lambda_2 \end{aligned} \tag{A.7}$$

so that it is straightforward to select the roots in order to satisfy the first one. To simplify the second condition we can always assume, as already said in Section 2.2.3, the coefficients $\alpha_\ell = 1$, since a similar regularization effect can be restored by a proper rescaling of a_j and γ. The (A.7) becomes:

$$\begin{aligned} \frac{\alpha_0\alpha_1\theta - \alpha_0^2}{\alpha_1^2} &= \lambda_1\lambda_2 \\ \alpha_0\theta - \alpha_0^2 &= \lambda_1\lambda_2 \\ \alpha_0(-\lambda_1 - \lambda_2) - \alpha_0^2 - \lambda_1\lambda_2 &= 0 \\ \alpha_0^2 + \alpha_0(\lambda_1 + \lambda_2) + \lambda_1\lambda_2 &= 0 \end{aligned} \tag{A.8}$$

which yields the discriminant:

$$\begin{aligned} \Delta_\lambda &= (\lambda_1 + \lambda_2)^2 - 4\lambda_1\lambda_2 = \\ &= \lambda_1^2 + 2\lambda_1\lambda_2 + \lambda_2^2 - 4\lambda_1\lambda_2 = \\ &= \lambda_1^2 + \lambda_2^2 - 2\lambda_1\lambda_2\,. \end{aligned}$$

We can then impose the condition to achieve a solution for the parameter α_0 by $\Delta_\lambda \geq 0$ which means:

$$\begin{aligned} \lambda_1^2 + \lambda_2^2 - 2\lambda_1\lambda_2 &\geq 0 \\ \frac{\lambda_1^2}{\lambda_2^2} + \frac{\lambda_2^2}{\lambda_2^2} - \frac{2\lambda_1\lambda_2}{\lambda_2^2} &\geq 0 \end{aligned}$$

from which we can derive the extremum for the rate $x = \lambda_1/\lambda_2$ as:

$$x_{1,2} = \frac{2 \pm \sqrt{8}}{2} = 1 \pm \sqrt{2}$$

and finally we have that:

$$\frac{\lambda_1}{\lambda_2} \leq 1 - \sqrt{2} \quad \vee \quad \frac{\lambda_1}{\lambda_2} \geq 1 + \sqrt{2}$$

By solving (A.8) we can always find a suitable value for the free parameters as:

$$\alpha_0 = \frac{-\lambda_1 - \lambda_2 + \sqrt{\Delta_\lambda}}{2}\,.$$

Since we want $\alpha_0 > 0$, an additional possibility is available when[3]:

$$\begin{aligned} -\lambda_1 - \lambda_2 - \sqrt{\Delta_\lambda} &> 0 \\ \left(|\lambda_1| + |\lambda_2|\right)^2 &> \Delta_\lambda \\ \lambda_1^2 + \lambda_2^2 + 2\lambda_1\lambda_2 &> \lambda_1^2 + \lambda_2^2 - 2\lambda_1\lambda_2 \\ 4\lambda_1\lambda_2 &> 0 \end{aligned} \tag{A.9}$$

which is always true when we take roots with negative real part.

Hence, since in general we would like to set the model from the roots λ_q, in the case of the differential operator of order $\ell = 1$, the conditions to verify if the chosen λ_q are consistent with the model and the formula to find the correspondent parameters are:

$$\begin{aligned} \theta &= -\lambda_1 - \lambda_2 \\ \alpha_0 &= \frac{-\lambda_1 - \lambda_2 \pm \sqrt{\lambda_1^2 + \lambda_2^2 - 2\lambda_1\lambda_2}}{2} \\ \alpha_1 &= 1 \end{aligned} \tag{A.10}$$

which would have generated the same model.

A.2.2 Second Order Operator

Characteristic Polynomial

When we decide to set up the model with the to the second order linear differential operator $P = \alpha_0 + \alpha_1 D + \alpha_2 D^2$, with formal adjoint $\hat{P} = \alpha_0 - \alpha_1 D + \alpha_2 D^2$,

[3]The first step is possible since the stability conditions require λ_1, λ_2 to have negative real part.

the (2.9) can be developed as:

$$\hat{P}(\psi Pw) + \tfrac{\gamma}{\mu}\psi V_w = [\alpha_0 - \alpha_1 D + \alpha_2 D^2]\left[\psi\left(\alpha_0 w + \alpha_1 Dw + \alpha_2 D^2 w\right)\right] + \tfrac{\gamma}{\mu}\psi V_w =$$
$$= \alpha_0^2\psi w + \alpha_0\alpha_1\psi Dw + \alpha_0\alpha_2\psi D^2 w - \alpha_0\alpha_1 D[\psi w] - \alpha_1^2 D[\psi Dw] - \alpha_1\alpha_2 D[\psi D^2 w] +$$
$$+\alpha_0\alpha_2 D^2[\psi w] + \alpha_1\alpha_2 D^2[\psi Dw] + \alpha_2^2 D^2[\psi D^2 w] + \tfrac{\gamma}{\mu}\psi V_w =$$
$$= \alpha_0^2\psi w + \alpha_0\alpha_1\psi Dw + \alpha_0\alpha_2\psi D^2 w +$$
$$-\alpha_0\alpha_1(\dot{\psi}w + \psi Dw) - \alpha_1^2(\dot{\psi}Dw + \psi D^2 w) - \alpha_1\alpha_2(\dot{\psi}D^2 w + \psi D^3 w) +$$
$$+\alpha_0\alpha_2 D^2[\psi w] + \alpha_1\alpha_2 D^2[\psi Dw] + \alpha_2^2 D^2[\psi D^2 w] + \tfrac{\gamma}{\mu}\psi V_w\,.$$

The application of the second order derivative operator D^2 gives:

$$\begin{aligned}
D^2[\psi w] &= D\left[\dot{\psi}w + \psi Dw\right] = \ddot{\psi}w + \dot{\psi}Dw + \dot{\psi}Dw + \psi D^2 w \\
D^2[\psi Dw] &= D\left[\dot{\psi}Dw + \psi D^2 w\right] = \ddot{\psi}Dw + \dot{\psi}D^2 w + \dot{\psi}D^2 w + \psi D^3 w \\
D^2[\psi D^2 w] &= D\left[\dot{\psi}D^2 w + \psi D^3 w\right] = \ddot{\psi}D^2 w + \dot{\psi}D^3 w + \dot{\psi}D^3 w + \psi D^4 w
\end{aligned}$$

again from $\psi(t) = e^{\theta t}$ we can derive $\dot{\psi}(t) = \theta e^{\theta t} = \theta\psi(t)$ and in the same way $\ddot{\psi}(t) = \theta^2\psi(t)$, so that, by dividing by $\psi(t)$ we have:

$$\alpha_0^2 w + \underline{\alpha_0\alpha_1 Dw} + \overbrace{\alpha_0\alpha_2 D^2 w} +$$
$$-\alpha_0\alpha_1(\theta w + \underline{Dw}) - \alpha_1^2(\theta Dw + D^2 w) - \alpha_1\alpha_2(\underbrace{\theta D^2 w + D^3 w}) +$$
$$+\alpha_0\alpha_2(\theta^2 w + 2\theta Dw + \overbrace{D^2 w}) + \alpha_1\alpha_2(\theta^2 Dw + \underbrace{2\theta D^2 w + D^3 w}) +$$
$$\alpha_2^2(\theta^2 D^2 w + 2\theta D^3 w + D^4 w) + \tfrac{\gamma}{\mu}V_w =$$
$$\alpha_2^2 D^4 w + 2\alpha_2^2\theta D^3 w + \left(\alpha_2^2\theta^2 + \alpha_1\alpha_2\theta + 2\alpha_0\alpha_2 - \alpha_1^2\right)D^2 w +$$
$$+\left(\alpha_1\alpha_2\theta^2 + 2\alpha_0\alpha_2\theta - \alpha_1^2\theta\right)Dw + \left(\alpha_0\alpha_2\theta^2 - \alpha_0\alpha_1\theta + \alpha_0^2\right)w + \tfrac{\gamma}{\mu}V_w =$$
$$D^4 w_i + \beta_3 D^3 w_i + \beta_2 D^2 w_i + \beta_1 Dw_i + \beta_0 w_i + \tfrac{\gamma}{\mu}V_w$$

where we obtain the (2.16) by dividing the second to last line by α_2^2 and by posing:

$$\begin{aligned}
\beta_0 &= \tfrac{\alpha_0\alpha_2\theta^2 - \alpha_0\alpha_1\theta + \alpha_0^2}{\alpha_2^2} \\
\beta_1 &= \tfrac{\alpha_1\alpha_2\theta^2 + (2\alpha_0\alpha_2 - \alpha_1^2)\theta}{\alpha_2^2} \\
\beta_2 &= \tfrac{\alpha_2^2\theta^2 + \alpha_1\alpha_2\theta + 2\alpha_0\alpha_2 - \alpha_1^2}{\alpha_2^2} \\
\beta_3 &= 2\theta.
\end{aligned} \tag{A.11}$$

Routh-Hurwitz Conditions

For a fourth order linear differential equation with constant coefficients, the stability conditions require[4]:

$$\begin{aligned} \beta_i &> 0\ ,\ i = 0, ..., 3 \\ \beta_3\beta_2 &> \beta_1 \\ \beta_3\beta_2\beta_1 &> \beta_1^2 + \beta_3^2\beta_0 \end{aligned} \tag{A.12}$$

We avoid further derivations since these conditions should in fact be imposed on the parameters θ, α_j so as to obtain roots with negative real part. As already said, we found more convenient to build the model by directly assigning the roots and, hence, the (A.12) become in practice useless.

From solutions to parameters

In Section A.2.1 we showed an explicit relation among the roots of the characteristic polynomial and the parameters of the model generated by the first order operator P. In this case, we will not able to write a similar explicit relation. However, we will show, assuming some simplifications, that it is in general possible to find more than one admissible configuration of parameters generated by the chosen roots. Indeed, by observing the (A.11), we can notice that the same polynomial can be obtained when varying the α_j, provided that the rates defined in the coefficients β_l are the same. A first strictly condition can be derived on β_3 by considering $p(\lambda) = \prod_{q=1}^{4}(\lambda - \lambda_q)$. We have that, once the roots λ_q are chosen, θ is determined by:

$$2\theta = -(\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4)$$

The expression of the others coefficients β_l are in general more difficult to be managed and some simplifications are required. For example, we empirically found out that a suitable response[5] is generated in some cases in which $\alpha_0 = \alpha_2$. Assuming this restriction, we can simplify the form of (A.11) so as to obtain

[4] We obtain this configuration by substituting $\beta_4 = 1$ in the general form.

[5] We mean roots with negative real part and null imaginary part.

general conditions both on the rate α_1/α_2 and on λ_q. For example:

$$\beta_0 = \frac{\alpha_0\alpha_2\theta^2 - \alpha_0\alpha_1\theta + \alpha_0^2}{a_2^2} = \theta^2 - \frac{\alpha_1}{\alpha_2}\theta + 1$$

from which:

$$\frac{\alpha_1}{\alpha_2} = \frac{\theta^2 + 1 - \beta_0}{\theta}.$$

We can not only evaluate the rate α_1/α_2 from β_0 and θ (which are available from λ_q), but also give conditions on λ_q. Indeed, we want $\alpha_j > 0$ and then $\alpha_1/\alpha_2 > 0$, so that we can impose:

$$\theta^2 + 1 - \beta_0 > 0.$$

Exploiting the fact that β_0 is the numeric term of p, it can be written as:

$$-\lambda_1 - \lambda_2 - \lambda_3 - \lambda_4 + 1 > \lambda_1\,\lambda_2\,\lambda_3\,\lambda_4$$

that is easy to be satisfied when one root is close to zero or, in general, when all the roots are smaller than 1. The expression of the second coefficient gives:

$$\beta_1 = \frac{\alpha_1\alpha_2\theta^2 + (2\alpha_0\alpha_2 - \alpha_1^2)\theta}{\alpha_2^2} = \frac{\alpha_1}{\alpha_2}\theta^2 + 2\theta - \frac{\alpha_1^2}{\alpha_2^2}\theta$$

By posing $\nu = \alpha_1/\alpha_2$, we can find admissible solutions as:

$$\nu_{1,2} = \frac{\theta^2 \pm \sqrt{\theta^4 - 4[\theta(-2\theta + \beta_1)]}}{2\theta}$$

that provides at least a positive value for the rate ν when:

$$\theta^4 - 4[\theta(-2\theta + \beta_1)] = \theta^4 + 8\theta^2 - 4\theta\beta_1 > 0.$$

Again, it is easy to satisfy this condition when the roots λ_q are smaller than one. Similar conditions could be derived by β_2. Finally, we should check all these conditions to be compatible all together, but this is not feasible in practice, because of the number of variables involved. Nevertheless, many configurations are in fact similar and the same global behavior can be restored by adapting the parameters. Hence, in the presented experiments we chose a model without proving the actual consistency, since an equivalent consistent model should always exist.

A.3 Discrete Computation

As introduced in Section 2.2.4, in order to deal with a solution of the differential equations coming from (2.10) that is manageable from a computational point of view, it is more convenient to work with its discretized version. The analysis will be again restricted to a single parameter w. This is not an approximation at any level. Indeed, we provide a formulation that allows us to compute the exact value of $w(t)$ with a fixed time sampling step, according to the composition of the training set $\mathcal{L}$ and, in general, of any stream of data which will be discretized. Regardless of the order of the chosen operator P, the (2.10) leads to a differential equation of order greater than one, that can be generally written as:

$$D^{2\ell}w + \beta_{2\ell-1}D^{2\ell-1}w + \cdots + \beta_0 w + (-1)^{\ell}\gamma\eta \cdot U = 0.$$

Given a such equation of order 2ℓ, it is always convenient to rewrite it as a system of 2ℓ first order equations by the substitution:

$$\begin{cases} w_1 &= w_0' \\ w_2 &= w_1' \\ & \vdots \\ w_{2\ell} &= -\beta_{2\ell-1}D^{2\ell-1}w_{2\ell-1} - \cdots - \beta_0 w_0 + (-1)^{\ell+1}\gamma\eta \cdot U \end{cases}$$

where $w_q = D^{(q)}w$. By posing $W(t) = \left[\, w(t),\, \ldots,\, D^{2\ell-1}w(t) \,\right]'$, the system can be written as:

$$\dot{W}(t) = A \cdot W(t) + B \cdot U(t) \tag{A.13}$$

where:

$$A = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \\ -\beta_0 & -\beta_1 & -\beta_2 & \ldots & -\beta_{2\ell-1} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ (-1)^{\ell+1}\gamma\eta \end{bmatrix}$$

By using the classic approach of linear state space models, it is convenient to express the solution to the (A.13) by the Lagrange formula:

$$W(t) = e^{At} \cdot W(0) + \int_0^t e^{A(t-s)} \cdot B \cdot U(s)\ ds \tag{A.14}$$

given the initial Cauchy's conditions $W(0)$. If we decide to evaluate the obtained equation at a discrete time-sampling step τ, starting from $W[0] = W(0)$, we can write:

$$W[K] = W(K\tau) = e^{AK\tau} \cdot W[0] + \int_0^{K\tau} e^{A(K\tau - s)} \cdot B \cdot U(s)\ ds$$

and, hence, the next step $K+1$ can be computed by:

$$\begin{aligned}
W[K+1] &= e^{A(K+1)\tau} \cdot W[0] + \int_0^{(K+1)\tau} e^{A((K+1)\tau - s)} \cdot B \cdot U(s)\, ds = \\
&= e^{AK\tau} \cdot e^{A\tau} \cdot W[0] + \int_0^{K\tau} e^{A(K\tau + \tau - s)} \cdot B \cdot U(s)\, ds + \\
&\quad + \int_{K\tau}^{(K+1)\tau} e^{\mathbf{A}((K+1)\tau - s)} \cdot \mathbf{B} \cdot U(s)\, ds = \\
&= e^{AK\tau} \cdot e^{A\tau} \cdot W[0] + \int_0^{K\tau} e^{A\tau} \cdot e^{A(K\tau - s)} \cdot B \cdot U(s)\, ds + \\
&\quad + \int_{K\tau}^{(K+1)\tau} e^{\mathbf{A}((K+1)\tau - s)} \cdot \mathbf{B} \cdot U(s)\, ds = \\
&= e^{A\tau} \left(e^{AK\tau} W[0] + \int_0^{K\tau} e^{A(K\tau - s)} \cdot B \cdot U(s) ds \right) \\
&\quad + \int_{K\tau}^{(K+1)\tau} e^{A[(K+1)\tau - s]} \cdot B \cdot U(s) ds = \\
&= e^{A\tau} W[K] + \int_{K\tau}^{(K+1)\tau} e^{A[(K+1)\tau - s]} \cdot B \cdot U(s) ds\,.
\end{aligned} \tag{A.15}$$

At this point, it is straightforward to find a formula of feasible computability by the arbitrarily restriction that supervisions eventually come in the middle of two updating. Such a condition can be formally written by:

$$\forall k\ \exists K_k :\ t_k = K_k + \tau/2.$$

We can notice that the term $U(t) = \sum_{k=1}^{N} \zeta_k \cdot \delta(t - t_k)$ inside the integral is

composed by a summation of Dirac's Delta functions $\delta(t)$ and, because of the translation property:

$$\int_a^b f(x)\delta(x-c) = f(c)$$

the integral term in the (A.15) is either null, if no supervision is provided between $K\tau$ and $(K+1)\tau$, or equal to $e^{A\frac{\tau}{2}} \cdot B \cdot \zeta_k$ if $K = K_k$ for some $k = 1, \ldots, N$. In this way, we finally obtain:

$$W[K+1] = e^{A\tau} \cdot W[K] + e^{A\frac{\tau}{2}} \cdot B \cdot \Delta_K \tag{A.16}$$

by posing

$$\Delta_K = \begin{cases} \zeta_k\,, & \text{if } \exists k : t_k \in [\,K\tau,\,(K+1)\tau\,] \\ 0 & \text{otherwise} \end{cases} \tag{A.17}$$

Bibliography

[1] Salvatore Frandina, Marco Gori, Marco Lippi, Marco Maggini, and Stefano Melacci. Variational foundations of online backpropagation. In *Artificial Neural Networks and Machine Learning - ICANN 2013 - 23rd International Conference on Artificial Neural Networks, Sofia, Bulgaria, September 10-13, 2013. Proceedings*, pages 82–89, 2013.

[2] Alessandro Betti and Marco Gori. The principle of least cognitive action. *Theoretical Computer Science*, pages 83 – 99, 2015.

[3] Marco Gori, Marco Maggini, and Alessandro Rossi. The principle of cognitive action-preliminary experimental analysis. *arXiv preprint arXiv:1701.02377*, 2017.

[4] Marco Gori, Marco Maggini, and Alessandro Rossi. Neural network training as a dissipative process. *Neural Networks*, 81:72 – 80, 2016.

[5] Marco Gori, Marco Maggini, and Alessandro Rossi. Collapsing of dimensionality. *arXiv preprint arXiv:1701.00831*, 2017.

[6] Marco Maggini and Alessandro Rossi. *On-line Learning on Temporal Manifolds*, pages 321–333. Springer International Publishing, Cham, 2016.

[7] Marco Gori, Marco Lippi, Marco Maggini, and Stefano Melacci. Learning to see like children: proof of concept. *CoRR*, abs/1408.2478, 2014.

[8] Marco Gori, Marco Lippi, Marco Maggini, and Stefano Melacci. Semantic video labeling by developmental visual agents. *Computer Vision and Image Understanding*, 146:9–26, 2016.

[9] Salvatore Frandina, Marco Gori, Marco Lippi, Marco Maggini, and Stefano Melacci. *Developmental Visual Agents* learning to see like children. `http://dva.diism.unisi.it/publications.html`, 2014.

[10] Abhijit Bendale and Terrance E. Boult. Towards open world recognition. *CoRR*, abs/1412.5687, 2014.

[11] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV (1)*, pages 44–57, 2008.

[12] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[13] Pierre F Gabriel, Jacques G Verly, Justus H Piater, and André Genon. The state of the art in multiple object tracking under occlusion in video sequences. In *Advanced Concepts for Intelligent Vision Systems*, pages 166–173. Citeseer, 2003.

[14] T Logeswari. Automatic brain tumor detection through mri–a survey. *Digital Image Processing*, 8(9):303–305, 2016.

[15] James J DiCarlo, Davide Zoccolan, and Nicole C Rust. How does the brain solve visual object recognition? *Neuron*, 73(3):415–434, 2012.

[16] Daniel Crevier. *AI: The tumultuous history of the search for artificial intelligence*. Basic Books, Inc., 1993.

[17] Martin A Fischler and Robert A Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on computers*, 22(1):67–92, 1973.

[18] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, PAMI-8(6):679–698, 1986.

[19] David G Lowe. Perceptual organization and visual recognition. *ISBN 089838172X*, 1985.

[20] Alex P Pentland. Perceptual organization and the representation of natural form. *Artificial Intelligence*, 28(3):293–331, 1986.

[21] David H Hubel and Torsten N Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574–591, 1959.

[22] Daniel J Felleman and David C Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex*, 1(1):1–47, 1991.

[23] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988.

[24] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[25] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[26] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[27] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[28] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

[29] Geoffrey E Hinton and Terrence J Sejnowski. Learning and releaming in boltzmann machines. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1:282–317, 1986.

[30] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[31] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[33] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

[34] Oriol Vinyals and Quoc V. Le. A neural conversational model. *CoRR*, abs/1506.05869, 2015.

[35] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[36] Ivan Krasin, Tom Duerig, Neil Alldrin, Andreas Veit, Sami Abu-El-Haija, Serge Belongie, David Cai, Zheyun Feng, Vittorio Ferrari, Victor Gomes, Abhinav Gupta, Dhyanesh Narayanan, Chen Sun, Gal Chechik, and Kevin Murphy. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://github.com/openimages*, 2016.

[37] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1521–1528. IEEE, 2011.

[38] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[39] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[40] Jean Piaget, Brbel Inhelder, and Jean Piaget. *The growth of logical thinking from childhood to adolescence: An essay on the construction of formal operational structures*, volume 84. Routledge, 2013.

[41] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[42] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197, 2010.

[43] David Marr. Vision: A computational investigation into the human representation and processing of visual information. *San Fransisco, CA: Henry Holt & Company*, 1982.

[44] Jake Bouvrie, Lorenzo Rosasco, and Tomaso Poggio. On invariance in hierarchical models. In *Advances in Neural Information Processing Systems*, pages 162–170, 2009.

[45] Thomas Serre. *Learning a dictionary of shape-components in visual cortex: comparison with neurons, humans and machines*. PhD thesis, Massachusetts Institute of Technology, 2006.

[46] Thomas Serre, Lior Wolf, and Tomaso Poggio. Object recognition with features inspired by visual cortex. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 994–1000. IEEE, 2005.

[47] Stefano Melacci, Marco Lippi, Marco Gori, and Marco Maggini. Information-based learning of deep architectures for feature extraction. In

International Conference on Image Analysis and Processing, pages 101–110. Springer, 2013.

[48] Marco Gori, Stefano Melacci, Marco Lippi, and Marco Maggini. Information theoretic learning for pixel-based visual agents. In *European Conference on Computer Vision*, pages 864–875. Springer, 2012.

[49] Marco Gori, Marco Lippi, Marco Maggini, and Stefano Melacci. On-line video motion estimation by invariant receptive inputs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 712–717, 2014.

[50] Stefano Melacci and Marco Gori. Unsupervised learning by minimal entropy encoding. *IEEE transactions on neural networks and learning systems*, 23(12):1849–1861, 2012.

[51] Suzanna Becker. Learning to categorize objects using temporal coherence. In *Proceedings of the 5th International Conference on Neural Information Processing Systems*, NIPS'92, pages 361–368, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[52] Hossein Mobahi, Ronan Collobert, and Jason Weston. Deep learning from temporal coherence in video. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 737–744. ACM, 2009.

[53] David Stavens and Sebastian Thrun. Unsupervised learning of invariant features using video. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1649–1656. IEEE, 2010.

[54] Ross Goroshin, Joan Bruna, Jonathan Tompson, David Eigen, and Yann LeCun. Unsupervised feature learning from temporal data. *arXiv preprint arXiv:1504.02518*, 2015.

[55] Will Y Zou, Andrew Y Ng, and Kai Yu. Unsupervised learning of visual invariance with temporal coherence. In *NIPS 2011 workshop on deep learning and unsupervised feature learning*, volume 3, 2011.

[56] Misha Belkin, Partha Niyogi, and Vikas Sindhwani. On manifold regularization. In *AISTATS*. Citeseer, 2005.

[57] Stefano Melacci and Mikhail Belkin. Laplacian support vector machines trained in the primal. *Journal of Machine Learning Research*, 12(Mar):1149–1184, 2011.

[58] Salvatore Frandina, Marco Lippi, Marco Maggini, and Stefano Melacci. On-line laplacian one-class support vector machines. In Valeri Mladenov, Petia D. Koprinkova-Hristova, Gnther Palm, Alessandro E. P. Villa, Bruno Appollini, and Nikola Kasabov, editors, *ICANN*, volume 8131 of *Lecture Notes in Computer Science*, pages 186–193. Springer, 2013.

[59] C. G. Gray. Principle of least action. 4(12):8291, 2009. revision 150617.

[60] Herbert Goldstein, Charles P. Poole, and John L. Safko. *Classical Mechanics (3rd Edition)*. Addison-Wesley, 3 edition, June 2001.

[61] CG Gray and Edwin F Taylor. When action is not least. *American Journal of Physics*, 75(5):434–458, 2007.

[62] B. van Brunt. *The Calculus of Variations*. Universitext. Springer New York, 2006.

[63] Charles Fox. *An introduction to the calculus of variations*. Courier Corporation, 1950.

[64] V. K. Chandrasekar, M. Senthilvelan, and M. Lakshmanan. On the lagrangian and hamiltonian description of the damped linear harmonic oscillator. *Journal of Mathematical Physics*, 48(3), 2007.

[65] Tomaso Poggio and Federico Girosi. A theory of networks for approximation and learning. Technical report, MIT, 1989.

[66] John B. Bell. Reviewed work: Solutions of ill-posed problems. *Mathematics of Computation*, 32(144):1320–1322, 1978.

[67] M. Bertero. *Regularization methods for linear inverse problems*, pages 52–112. Springer Berlin Heidelberg, Berlin, Heidelberg, 1986.

[68] Eric L. W. Grimson. *From Images to Surfaces: A Computational Study of the Human Early Visual System*. MIT Press, Cambridge, MA, USA, 1981.

[69] F.G. Friedlander and M.S. Joshi. *Introduction to the Theory of Distributions*. Cambridge University Press, 1998.

[70] I. Stakgold and M.J. Holst. *Green's Functions and Boundary Value Problems*. Pure and Applied Mathematics: A Wiley Series of Texts, Monographs and Tracts. Wiley, 2011.

[71] B. Schoelkopf and A.J. Smola. From regularization operators to support vector kernels. In Morgan Kaufmann, editor, *Advances in Neural Information Processing Systems*, 1998.

[72] Giorgio Gnecco, Marco Gori, and Marcello Sanguineti. Learning with boundary conditions. *Neural Computation*, 25(4):1029–1106, 2013.

[73] G. Gnecco, M. Gori, S. Melacci, and M. Sanguineti. Foundations on support constraint machines. *Neural Computation*, 27(2):388–480, 2015.

[74] S. Melacci and M. Gori. Learning with box kernels. *Transactions on PAMI*, 35(11):2680–2692, 2013.

[75] B. K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203, 1981.

[76] L. Herrera, L. Núñez, A. Patiño, and H. Rago. A variational principle and the classical and quantum mechanics of the damped harmonic oscillator. *American Journal of Physics*, 54(3):273–277, 1986.

[77] R. Isermann, K.H. Lachmann, and D. Matko. *Adaptive Control Systems*. Ellis Horwood Series in Physical Chemistry. Prentice Hall, 1992.

[78] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Neurocomputing: foundations of research. *JA Anderson and E. Rosenfeld (Eds.)*, pages 696–699, 1988.

[79] Paul Werbos. Beyond regression: New tools for prediction and analysis in the behavioral sciences. 1974.

[80] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[81] Frank Rosenblatt. *Perceptions and the theory of brain mechanisms*. Spartan books, 1962.

[82] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[83] Martin T Hagan, Howard B Demuth, Mark H Beale, and Orlando De Jesús. *Neural network design*, volume 20. PWS publishing company Boston, 1996.

[84] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Neural Networks, 1993., IEEE International Conference On*, pages 586–591. IEEE, 1993.

[85] Jorge J Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

[86] David G Luenberger. *Introduction to linear and nonlinear programming*, volume 28. Addison-Wesley Reading, MA, 1973.

[87] R. A. DeCarlo. *Linear Systems: A State Variable Approach with Numerical Implementation*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[88] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, pages 689–696. ACM, 2009.

[89] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947–951, 2000.

[90] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[91] Daniel P. W. Ellis. PLP and RASTA (and MFCC, and inversion) in Matlab, 2005. online web resource.

[92] Michelangelo Diligenti, Marco Gori, Marco Maggini, and Leonardo Rigutini. Bridging logic and kernel machines. *Machine learning*, 86(1):57–88, 2012.

[93] Luciano Serafini and Artur S dAvila Garcez. Learning and reasoning with logic tensor networks. In *AI* IA 2016 Advances in Artificial Intelligence*, pages 334–348. Springer, 2016.

[94] Giorgio Gnecco, Marco Gori, Stefano Melacci, and Marcello Sanguineti. Foundations of support constraint machines. *Neural computation*, 27(2):388–480, 2015.

[95] Duccio Papini. Variational laws of cognition. Technical report, Department of Information Engineering and Mathematical Sciences, University of Siena, 2015.

[96] John Marshall Lee. *Introduction to smooth manifolds*. 2002.

[97] Nitin Bhatia et al. Survey of nearest neighbor techniques. *arXiv preprint arXiv:1007.0085*, 2010.

[98] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[99] Ian H Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[100] Parikshit Ram and Alexander G. Gray. Maximum inner-product search using cone trees. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 931–939, New York, NY, USA, 2012. ACM.

[101] Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 97–104, New York, NY, USA, 2006. ACM.

[102] Kenneth L Clarkson. Nearest-neighbor searching and metric space dimensions. *Nearest-neighbor methods for learning and vision: theory and practice*, pages 15–59, 2006.

[103] Vincent Garcia, Eric Debreuve, Frank Nielsen, and Michel Barlaud. K-nearest neighbor search: Fast gpu-based implementations and application to high-dimensional feature matching. In *2010 IEEE International Conference on Image Processing*, pages 3757–3760. IEEE, 2010.

[104] Lawrence Cayton. Accelerating nearest neighbor search on manycore systems. In *Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*, pages 402–413. IEEE, 2012.

[105] Andrew B Goldberg, Ming Li, and Xiaojin Zhu. Online manifold regularization: A new learning setting and empirical study. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 393–407. Springer, 2008.

[106] Sanjoy Dasgupta and Yoav Freund. Random projection trees and low dimensional manifolds. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 537–546. ACM, 2008.

[107] Tomáš Skopal, Jaroslav Pokorný, and Václav Snášel. *Nearest Neighbours Search Using the PM-Tree*, pages 803–815. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[108] R.E. Bellman. *Dynamic Programming*. Dover Books on Computer Science Series. Dover Publications, 2003.

[109] Christian Grossmann, Hans-Görg Roos, and Martin Stynes. *Numerical treatment of partial differential equations*. Springer, 2007.

[110] John Thickstun, Zaid Harchaoui, and Sham Kakade. Learning features of music from scratch. *arXiv preprint arXiv:1611.09827*, 2016.

[111] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, 2015.

[112] H Braun and M Riedmiller. Rprop: a fast adaptive learning algorithm. In *Proceedings of the International Symposium on Computer and Information Science VII*, 1992.

[113] Daniel L Silver, Qiang Yang, and Lianghao Li. Lifelong machine learning systems: Beyond learning algorithms. In *AAAI Spring Symposium: Lifelong Machine Learning*, pages 49–55. Citeseer, 2013.

[114] Cesare Alippi. Learning in nonstationary and evolving environments. In *Intelligence for Embedded Systems*, pages 211–247. Springer, 2014.

[115] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4):44, 2014.

[116] Viktor Losing, Barbara Hammer, and Heiko Wersing. Knn classifier with self adjusting memory for heterogeneous concept drift. In *Data Mining (ICDM), 2016 IEEE International Conference on*. IEEE, 2016.

[117] Lydia Fischer, Barbara Hammer, and Heiko Wersing. Online metric learning for an adaptation to confidence drift. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 748–755. IEEE, 2016.

[118] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, and Scott Reed. Ssd: Single shot multibox detector. *arXiv preprint arXiv:1512.02325*, 2015.