Implementation of Dynamical Systems with Plastic Self-organising Velocity Fields

Xinhe Liu

Department of Mathematical Sciences

Loughborough University

A Doctoral Thesis

Submitted in partial fulfilment of the requirements
for the award of Doctor of Philosophy of Loughborough University
September 2015

Abstract

To describe learning, as an alternative to a neural network recently dynamical systems were introduced whose vector fields were plastic and self-organising. Such a system automatically modifies its velocity vector field in response to the external stimuli. In the simplest case under certain conditions its vector field develops into a gradient of a multi-dimensional probability density distribution of the stimuli. We illustrate with examples how such a system carries out categorisation, pattern recognition, memorisation and forgetting without any supervision.

Crucially, such a plastic dynamical system remains a mathematical abstraction that does not seem to describe any phenomenon or a system existing in the real world. Here we explore various possibilities to implement the plastic dynamical system in hardware. Since neural networks can be implemented in practice, we first apply a neural network comprising N continuous-state neurons undergoing unsupervised Hebbian learning in the attempt to approximate the simplest one-dimensional gradient plastic system. Two coding methods are proposed to represent the one-dimensional state of a plastic system by N neurons. It is shown with examples of numerically generated input that at least with the coding methods used, this neural network does not reproduce evolution of the plastic velocity field with sufficient accuracy.

We therefore consider another approach based on the fact that any polynomial function can in practice be implemented in an electronic circuit. Within this approach we approximate the probability density function of the input by polynomials with coefficients involving various moments of the input. It is shown that a one-dimensional evolving plastic vector field can be approximated by derivatives of Legendre or Hermite polynomials with reasonably high accuracy.

This moment-based approximation method is then generalised to approximate a two-dimensional plastic system. With two-dimensional Hermite polynomials, the evolving plastic vector field can be approximated with satisfactory accuracy.

Key words: Nonlinear dynamics, Learning, Neural networks, Moment-based approximation

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr Natalia Janson, Senior Lecturer in Department of Mathematics, for providing me with the opportunity to undertake this interesting and challenging project as PhD research and suggesting the motivation and some important directions of this research. In addition, I'm extremely thankful and indebted to her not only for her continuous support of my study and research, but also for sharing experience and sincere and valuable guidance of being an independent researcher, which is very helpful for my future career.

I am also grateful to my husband and parents for their continuous encouragement and support in both my life and study.

Abbreviations

CNN: Continuous-state nonlinear neural network

DS: Dynamical system

MBDA: Moment-based density approximation

NN: Neural network

PDF: Probability density function

PDS: Dynamical system with plastic self-organising velocity fields, or plastic dynamical system

Contents

Chapter 1

Introduction

Mathematicians and physicists have been working on creating artificial intelligent devices reproducing some features and functions of the human brain for decades. Artificial neural network (NN) is a popular paradigm and the basis of many modern artificial intelligent devices. The physical and mathematical theory of NNs has been developing rapidly since 1950s. During this period, NNs have been developing from binary to linear and then to continuous nonlinear models of increasing complexity, and from single-layer networks to multi-layer ones. The range of applications of NNs has also broadened to encompass recognition, machine vision, prediction, language processing, etc. To perform these tasks, the parameters of the NNs, such as the strengths of inter-neuron connections or the excitability thresholds of individual neurons, are controlled by certain pre-defined algorithms, which are known as learning algorithms. For example, NNs applying supervised learning, in which target output and feedback are provided for every input pattern from the training set, are capable of classification and optimisation. Also, estimation of statistical distribution and filtering can be achieved by unsupervised learning, which does not require supervision or target output.

However, some natural features of NNs limit their capabilities. One feature is its

rigid architecture. A NN can be regarded as a dynamical system (DS) which consists of individual units (neurons) with rigid structure and flexible coupling between them. In a NN, the only flexibility is provided by control of the strengths of connections, that is, of the parameters of the system. We are not able to modify the whole structure of the vector field freely by modifying only the connection strengths. This leads to problems such as spurious attractors that do not correspond to any categories. Another feature of a NN is its reliance on algorithms. Namely, the NN employs algorithms at least at the stage of learning [35]. The most popular learning algorithm in a NN is a supervised one, while unsupervised learning requires considerable complication of the algorithms. This does not seem to agree with how biological NNs work, which do not perform algorithms. Also, artificial NNs can store memories, but their storage capacity is limited, so that some old memories may be destroyed while new memories are formed in response to the new input.

Recently an alternative model of an artificial learning system was introduced [42; 43]. This model can serve the same purposes as artificial NNs but without their limitations such as the bounded phase space or a finite memory size. Mathematically it is a DS, which shapes its velocity vector field in response to external stimuli automatically according to some rule. In such a system the velocity vector fields is fully plastic and self-organising and for brevity we call it a Plastic Dynamical System (PDS). In the simple example introduced in [42; 43], the vector field converges to the gradient of the probability density distribution of the input process, taken with negative sign. Such a system represents a mathematical model that naturally learns without supervision and can perform pattern recognition, categorisation, memorisation and forgetting simultaneously. However, at this stage, this model is only a mathematical concept. Hence the purpose of this work is to explore the possibility to implement this abstract model in practice to make it possible to eventually develop an intelligent device of a new type.

In Chapter 2, we provide the background knowledge related to this work, including some historical remarks on neuroscience and some basic concepts associated with artificial NNs. The structure and learning algorithms of NNs are introduced, followed by the discussion of their limitations. Finally, the motivation for this work is explained.

In Chapter 3, it is reminded about the concept of a system with plastic self-organising velocity field. The simplest form of a one-dimensional PDS is illustrated by examples of numerically generated random input data to demonstrate how such DSs self-organise their velocity vector fields and perform categorisation and pattern recognition without any supervision. The advantages of such systems as compared to models of NNs are discussed.

To implement the PDS, one possible way is to represent its velocity field in terms of some well defined analytical functions, which can be implemented in electronic circuits. Two approaches are considered. One is by means of a conventional model of a NN, which is also a DS that modifies its velocity field in response to stimuli by means of adjusting its inter-neuron couplings. In Chapter 4, a NN consisting of continuous-state neurons is trained using the conventional unsupervised Hebbian learning with the same sequence of one-dimensional input stimuli as the one given to the PDS whose performance we are trying to imitate. In order to compare the performance of the "ideal" PDS, whose phase space is unbounded with the one of the NN with a fundamentally bounded phases space, we need to find a way to represent the state of the former by the state of the latter. The question about memory representation in biological NNs remains open so we developed two methods by which a NN could code the state of a system with an unbounded phase space. The plastic vector field is approximated by NN with both coding methods.

Given that the vector field of the simplest perfectly PDS converges to some approximation of the gradient of the probability density distribution of the input, taken with negative sign, we consider another way to implement such a system in hardware,

that is, to approximate its vector field by a series of polynomials, whose coefficients would involve the moments of the input. The advantage of this approach as compared to NNs would consist in not needing to code the state in an unbounded space by the state in a bounded one. In Chapter 5, to start with, we explore the possibility to use this approach in order to reproduce the evolution of a plastic velocity field of a one-dimensional DS. The one-dimensional probability density function of the input is approximated by Legendre and Hermite polynomials, respectively. We show that with either polynomial, this method can approximate the plastic velocity vector field with acceptable error. We also demonstrate the analytical calculations and numerical simulations of the error.

In Chapter 6, we extend the approach developed in Chapter 5 for a one-dimensional PDS to a multi-dimensional one. Although all analytical expressions obtained here can be derived for a general multi-dimensional case, we illustrate this method for two-dimensional PDS for clarity. Specially, we approximate the vector field of the two-dimensional plastic system by Hermite polynomials, which is usually called Gram-Charlier series in multi-dimensional case. The evolving joint probability density distribution of two correlated random variables is approximated by Gram-Charlier series and by the PDS, respectively. We demonstrate how the quality of this approximation depends on the parameters of the series used, and obtain an analytical estimate of the approximation error for a general case.

In Chapter 7, we draw some main conclusions of this work. Namely, it seems that the idea of using polynomials with evolving parameters, that are determined by various statistical characteristics of the input received earlier, is promising for the purpose of implemental plastic velocity fields. In this respect, polynomial approximations seem to have more potential than the conventional artificial NNs with a rigid structure of each neuron and of each inter-neuron connections, in which flexibility is limited to only the values of couplings.

Chapter 2

Background

This Chapter gives a general introduction to the evolution of our understanding of the brain and some historical remarks about the artificial NNs. Some basic concepts in biological and artificial NNs are introduced. The advantages and limitations of artificial NNs are discussed and thereafter the motivation for this work is described in detail.

2.1 Historical background of neuroscience

2.1.1 Understanding of the brain based on natural philosophy

The mystery of the brain has been attracting people since the ancient Egypt. The first written reference to the word ”brain” was found in the Edwin Smith Surgical Papyrus, which was an ancient paper-like document written by an ancient Egyptian around 1700 BC [27]. By the middle of the fifth century BC, a Greek medical scientist Alcmaeon first described sensory nerves and concluded that the brain is the seat of sensation and cognition [49]. Brain's controlling role in sensation and intelligence was also appreciated by the Greek physician, Father of Western Medicine, Hippocrates, and by the famous Greek philosopher Plato [27]. However, the brain was not always held in high regard. Plato's student, an ancient Greek philosopher and a biologist Aristotle, believed that heart controls mental activity, while the brain was cold and bloodless and therefore without sensation [52]. In the second century, the Greek physician and philosopher Galen described the anatomy of the brain accurately and in detail, and confirmed by systematic experiments that sensation and motion are controlled by the brain through the nerves [59].

The discovery of the neuron

Due to the development of the surgery techniques in anatomy and physiology, scientists were able to conduct some groundbreaking research from the 1700s onward. In the middle of the 19th century, the idea of a German histologist Gerlach that the brain and the nervous system could not be split up into distinct structural units gained popularity. These ideas were rejected in 1887 by a Spanish neuroanatomist Cajal who improved the Italian physician Goldi's experiments to investigate nervous tissue and showed that the nervous system was made up of what he called ”absolutely autonomous units” [46]. Soon afterwards in 1891, a German anatomist Waldeyer com-

bined Cajal's research with some previous findings in cell theory to form the neuron doctrine: the nervous system is composed of individual units, which were named "neurons" [67]. Each neuron comprises the cell body, the axon and the dendrites to transmit signals between neurons as will be further explained in Section 2.2.1. Early in the 20th century, it was already understood that neurons have an electrical potential across their membrane, and the signal transmission along the synapse occurs by a propagated electrical pulse. These findings provided the fundamental theory of neuroscience.

2.1.2 Interdisciplinary studies of the brain in the 19th century

In the 19th century, the study of the brain became an interdisciplinary field. Famous scientists in physics, such as Helmholtz, Maxwell and Mach, made significant contributions to the understandings of vision [32]. German Physicist Helmholtz measured the velocity of electrical signals in nerve axons and provided empirical theories on colour vision, visual perception and the sensation of a tone [9]. However, mathematics of the 19th century was not developed sufficiently to help understand these phenomena. The mismatch between a relatively advanced experimental physics and the state of mathematical theories underlied the crisis in the attempts to understand the brain. Meanwhile, a schism between physics and psychology occurred towards the end of the 19th century [32]: theoretical physicists and psychologists abandoned the knowledge of the other field and there was a barrier between theories and experiments. Despite all these difficulties, relevant theories of the brain were still developing during the 19th century. In 1890, an American psychologist James described the functions of different parts of the brain and offered a physiological explanation for how the brain develops habits [27]. This provided an idea of how the brain learns.

2.1.3 Development of artificial neural networks

In the middle of the 20th century, the work of Helmholtz has been re-evaluated, and researchers started to utilize the approaches from theoretical physics and mathematics to describe cognitive phenomena. At the same time, improvements in electronic techniques gave rise to the hopes to reproduce the key functions of the brain in machines. With the assistance of digital computers, scientists started to model the brain and gradually built the field of artificial intelligence. While some work considers biologically relevant models, much work in this area has been focused on the fundamental mechanisms of cognition rather than on biological details. Specifically, it was well appreciated that in biological brains learning is accompanied by the creation and destruction of inter-neuron connections, and by modification of their strengths. With this, modelling of a NN involves two stages: formulating models of individual neurons and of couplings between them, and mathematically specifying rules according to which these couplings evolve while the network is learning, i.e. the training rules. Each of these stages is discussed below.

Development of models of network units and connections

A highly simplified model of a discrete-time binary-state neuron constructed by McCulloch and Pitts in 1943 [53] opened the modern era of NNs. In the McCulloch-Pitts model, each neuron has two states characterised by either 1 (”firing”) or 0 (”not firing”) in response to whether the weighted sum of the input signals received from other neurons is above (firing) or below (not firing) some threshold value. The strength of a synaptic connection from one neuron to another is modelled as a factor (called ”weights”) by which the input from one neuron is multiplied before it is applied to the other neuron. This model has made a seminal contribution to the development of artificial NNs. McCulloch and Pitts proved that it is possible in principle to perform any universal computation that an ordinary digital computer can, by assembling neu-

rons into a network with a certain architecture with suitably chosen weights [35]. It is noteworthy that von Neumann's idea of construction of EDVAC (Electronic Discrete Variable Automatic Computer) was derived from the McCulloch-Pitts neural elements [7], and was later developed to build the digital computer.

In 1958, a new approach to the NNs was introduced by Rosenblatt. He designed and developed the perceptron, which was constructed according to biological principles and displayed the ability to learn [63]. A perceptron consists of three layers: an input layer, a middle layer and an output layer, with each layer fully connected to the next one uni-directionally. The perceptron was thought to be a great improvement as compared to McCulloch-Pitts model and made it possible to produce a machine. However, in 1969, Minsky and Papert highlighted the limitations of the perception with one middle layer, that it could not be trained to recognise a certain type of functions, those which are not linearly separable, i.e. those which cannot be separated by a single line on the plane in the space of all possible patterns (e.g. XOR, i.e. exclusive-or, problem) [57]. It was in 1980s when more powerful learning rules have been developed and extensions of the perceptron architecture were proposed, being multilayer networks with additional layers of neurons and connections that could be trained without such limitations [65]. Multilayer networks are now widely used models in various research domains. However, the monograph of Minsky and Papert at that time discouraged not only researchers working on perceptron, but also agencies supporting their work. As a result, very little improvement was done in the area throughout the 1970s.

In 1988, Broomhead and Lowe introduced radial basis functions (RBF) to design a NN [13]. The RBF network provides an alternative approach to multilayer perceptrons. It is composed of three layers: an input layer (sensory units), a high-dimensional hidden layer, in which the input is transformed by nonlinear basis functions, which are radially symmetric, and an output layer comprising linear combinations of responses from the hidden layer nodes [33].

On the other hand, researchers who pursued to model NNs with continuous-state dynamics made some significant achievements in reproducing cognitive abilities. To describe the continuous features of neural dynamics, some models applied the concepts from linear systems theory. One such system, named Adaline (Adaptive linear element), was developed in 1960 by Widrow and Hoff [75] and later in 1962 Widrow developed it into Madaline (multiple-adaline), which is a NN with multiple adaptive elements [74]. The Adaline model was an adaptive pattern recognition machine, which employed the Least-Mean-Squares learning rule. From then on, researchers began to describe their intuitions within a mathematically familiar engineering framework and to progressively represent more nonlinear interactions, being inspired by biological findings.

Continuous-state nonlinear networks (CNN) are closer to biological NNs than their discrete-state versions. One classical continuous-state model arose from the investigations of the spiking behaviour of neurons. The original experiments and the model were produced by Hodgkin and Huxley in 1952 [38]. But the model equations were quite complicated and not easily amenable to mathematical analysis. In 1961, Fitzhugh and Nagumo simplified this model so that the dynamics of the system could be analysed [21; 58]. However, this kind of models focused on individual neurons rather than on networks of neurons.

Instead of studying the neural mechanism itself, Grossberg (1967, 1968) proposed another CNN, named additive model, which assumed the adaptive behaviour of individual neurons. Namely, this model described how an individual neuron adapts to the evolving environmental influence in real time [28; 29; 30; 31]. His model has the form of an ordinary differential equation, in which the rate of change of the neuron state is determined by passive decay, the sum of the feedback from the other neurons, and the external input. Each feedback is the product of a state-dependent nonlinear signal and a connection weight. This model is a cornerstone of NN research and has

been implemented in various applications, including computational analysis in vision, pattern recognition and associative pattern learning [32].

At the same time, mathematical analysis in order to find under what conditions the network could produce associative content-addressable memory was performed in [29; 30]. In the language of dynamical systems theory, to produce content-addressable memory in NNs, a certain stored pattern should be represented as an attractor, which should form in response to a sustained input pattern. One approach introduced to such analysis of NNs in 1970s is to construct a global Lyapunov function, also called energy function of the NN [32]. An important property of an energy function is that it always decreases (or remains constant) as the system gradually evolves towards the attractor. In 1982, Hopfield introduced an associative memory model with binary-state neurons which were learning without supervision using Hebbian learning rules inspired by a famous hypothesis of Donald Hebb that neurons which fire together wire together [39]. This model and its various generalizations are usually called Hopfield models. In addition, Hopfield used an energy function to explain the behaviour of this network. Although the origin of Hopfield's work may be traced back to the earlier work of some other researchers, Hopfield's influential paper for the first time brought them together and explicitly stated the principle of storing information as dynamical attractors. In 1984, Hopfield introduced another associative memory model with continuous-state neurons, which can be seen as one application of Grossberg's additive model [32; 41]. Through Hopfield's papers, this class of NNs attracted more attention in the 1980s. Cohen and Grossberg then described a general model for which a global energy function had been explicitly constructed in 1983 and included Hopfield's energy function as a special case [15]. In 1984, Hopfield constructed another form of energy function for a simpler version of the former continuous nonlinear model produced by Cohen and Grossberg [41]. Hopfield's energy function gained popularity and led to an important breakthrough in computational neuroscience.

Development of learning algorithms

In addition to the achievements in neural modelling, the research in learning algorithms of NNs also made a considerable progress in the 20th century. In 1949, Hebb first presented an explicit statement of a hypothesised physiological learning rule for synaptic modification [34]. This gave birth to the theory of synaptic plasticity that enables the strength of connections between neurons to change in response to inputs [66]. In Hebbian learning, the synaptic coupling between two neurons is strengthened when both of them are active. Hebb's book "The Organization of Behavior" inspired the development of computational models of learning and adaptive systems. In 1956, Rochester, Holland et al, tested Hebb's postulate of learning in a computer simulation and suggested to add inhibition to the theory [62]. In the same year, Uttley demonstrated that a NN with adjustable connections can learn to do classification of simple binary patterns [70]. In 1979, Uttley proposed a hypothesis that the strength of a connection depends on the statistical relationship between the changing states of neurons on either side of that connection [71]. Hebbian learning algorithm nowadays is still a widely used unsupervised learning rule, which requires no learning goal and no feedback from the environment.

In 1989, Zak proposed a new concept for unsupervised learning in continuous-state NNs in which each input pattern is considered as an interpolation node of the velocity vector field of the DS describing the NN [77; 79]. The connection strengths are being modified in response to stimuli in such a way that the resultant velocity field is intended to develop attractors that would represent categories of input signals. However, no demonstration of the workability of this approach has been provided with sufficiently complex realistic examples. In [78] Zak proposed to link dynamical systems in the form of a system of ODEs with an equation describing evolution of the PDF and called them self-supervised DSs. However, such systems are not associated with the traditional learning goals posed before artificial intelligent devices. In [78]

Zak introduced an idea of "smart" particles interacting with information flows in such a manner that the velocity field of the dynamical system describing their collective motion in the physical space was coupled with an equation describing the evolution of the PDF.

Another type of learning is supervised learning, in which target output and feedback are required. In 1960, based on his idea of a nonlinear adaptive filter, Gabor invented a machine which could be trained by being presented examples from various categories, and also inform about these categories, i.e. by showing how to match input patterns with the required outputs using a set of training examples [25]. In addition, in 1983, Barto, Sutton and Anderson introduced reinforcement learning and its application [8]. Reinforcement learning is a special type of supervised learning, in which the only feedback given to the system is whether the output is correct or incorrect, and it was first considered by Minsky in 1954 [56].

To date, the most widely used tool in supervised learning is backpropagation which was developed and applied in practice by Werbos in 1974 [73]. In 1985 and 1986, Parker and Le Cun discovered the backpropagation learning algorithm for multilayer perceptrons [17; 60]. This method to train a NN was widely known after Rumelhart, Hinton and Williams published their paper in the book "Parallel Distributed Processing" in 1986, which played a vital role in the development of this field [64]. In this method, the connection weights are carefully chosen to minimize a loss function calculated by subtracting of the actual output from the desired output for each input value. If the training is completed correctly, the actual output should be close to the target.

2.2 Artificial neural networks

An artificial NN is a highly simplified mathematical model of biological memory. It comprises a large number of interconnected units, called **neurons**. Artificial NNs learn from examples by adjusting the strength of the connections, called **connection weights**. In this Section, we will recall a basic model of a continuous-state neuron and a continuous-state NN, which is closer to biological NNs than their discrete-state versions. The models of learning algorithms are also illustrated at the end of the Section.

2.2.1 Biological neurons and artificial neurons

To study the collective behaviour of a NN, one first needs some basic understanding of the structure of a single neuron.

Biological neurons

There are about 10^{11} neurons of many types in a human brain [35]. Fig. 2.1 shows a schematic diagram of a typical single neuron. Connected to the neuron cell body (soma) depicted as a shaded circle, there are some tree-like structures of nerve fibers, called dendrites. A single long and thick fiber called axon extends from the cell body and eventually branches into strands, which terminate at the transmitting ends of the synaptic junctions, or synapses, to other neurons.

To transmit a signal, one cell releases specific transmitter substances from the sending side of the synaptic junction [35]. In response to this, the electrical potential inside the body of the receiving cell will either decrease or increase, depending on whether the sender neuron is inhibitory or excitatory. If this potential reaches a threshold, the cell "fires" by sending a spike or action potentials of fixed strength and duration down the axon. The spike then branches out to synaptic junctions to other

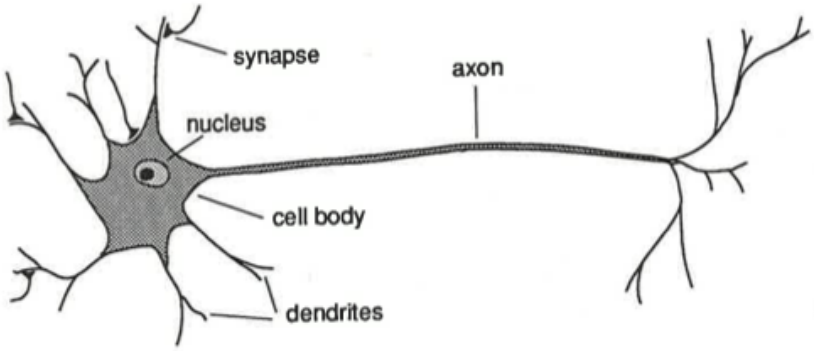


Figure 2.1: Schematic diagram of a biological neuron [35].

neurons. After firing, the cell has to wait during the time period called the refractory period before it can fire again. This signal transmission between neurons through synapses is a complex chemical process.

Artificial neurons

An artificial neuron is a highly simplified reconstruction of a biological neuron. A neuron number i is usually described as an input-output device with many inputs received from other neurons in the network (i.e. outputs of the other neurons) and one output, denoted by v_i as illustrated in Fig. 2.2. The output v_i is a function of its net input u_i, called **activation function** [35]:

$$v_i = s_i(u_i) = s_i(\sum w_{ij}v_j), \tag{2.1}$$

where w_{ij} is the strength of connection between neuron i and neuron j defined as

$$w_{ij}\begin{cases} > 0 & \text{excitatory synapse;} \\ < 0 & \text{inhibitory synapse;} \\ = 0 & \text{no synapse.} \end{cases} \tag{2.2}$$

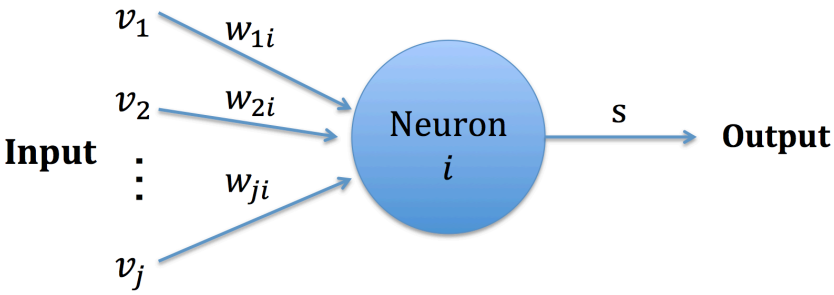


Figure 2.2: A simple artificial neuron.

The activation function s_i can be different for different neurons but in most NNs every neuron has the same activation function, denoted by s.

In a discrete-state NN, v_i can only be -1 (not firing) or 1 (firing) and the activation function is the unit step function

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0; \\ -1 & \text{otherwise.} \end{cases} \tag{2.3}$$

In a continuous-state NN, v_i can take any value in the rage $[-1, 1]$ and the activation function is a continuous and monotonically increasing function, typically a sigmoid function, such as the one given below

$$s(x) = \frac{2}{1 + \exp(-4x)} - 1, \tag{2.4}$$

which asymptotically tends to 1 or -1 as x tends to a positive infinity or a negative infinity, respectively.

Sometimes it might be desirable to have v_i and s take value either 0 or 1 in the discrete-state NN or range from 0 to 1 in the continuous-state NN.

2.2.2 Continuous-state neural network

Continuous-state nonlinear networks (CNNs) are closer to biological NNs than the discrete-state NNs. One famous and widespread model of this type is Hopfield model, although Hopfield was not the first researcher proposing such models. In a CNN all neurons update their states continuously and simultaneously. The evolution of the neuron state can be represented by a set of differential equations [16; 41]:

$$a_i \frac{\mathrm{d}u_i}{\mathrm{d}t} = -u_i + \sum_{j=1}^{N} w_{ij} s(u_j) + I_i, \tag{2.5}$$

where N is the number of the neurons in the network, a_i is the time constant, and I_i is external input to the i-th neuron.

Equation (2.5) includes a term of passive decay $-u_i$, feedback from other neurons $\sum_{j=1}^{N} w_{ij} s(u_j)$ and external input I_i. The feedback is a weighted sum of outputs of the other neurons.

2.2.3 Content-addressable memory and energy function

Hopfield [39; 40] and Hinton [37] demonstrated that a system of highly interconnected neurons is capable of classification, generalization, error correction, etc. These collective computational properties are only weakly sensitive to errors in the input pattern. A **content-addressable memory** (or **associative memory**) created by such networks enables a retrieval of a specific memory (stored pattern) when prompted by an input pattern which has some similarity, but not identical to it.

A content-addressable memory can be quite powerful. For example, suppose we store coded information about many famous writers and their literature works in a network, such as an item ”Charlotte Bronte (1847), *Jane Eyre*”. A general content-addressable memory would be capable of retrieving this entire memory based on sufficient partial information. The starting pattern ”Charlotte (1847)” might be suffi-

cient to recall this masterpiece, despite the error in the input pattern, such as "Sharlotte (1847)". Note that unless we invent a pattern that is not stored in memory, the network will always retrieve some pattern from the clue provided and the network will pick the best match from the set of stored patterns rather than retrieve a linear combination of them [35].

In the phase space of a CNN, a typical or average member of a certain class of patterns is represented as an attractor and all possible members of the same class as points in its basin of attraction. The property of a content-addressable memory can be interpreted mathematically as follows: a NN always evolves to an attractor after being presented with an arbitrary input pattern as the initial state, i.e., the solution of system (2.5) always settles down to one of its attractors. This process is known as **pattern recognition**. And therefore such NNs are usually called attractor NNs.

Hopfield showed that the system can produce content-addressable memory, with symmetric synapses w_{ij}'s, i.e. $w_{ij} = w_{ji}$ [41]. This feature is shown by constructing an **energy function** $E(v_1, v_2, ..., v_N)$, where N is the network size, also called a Lyapunov function, which always decreases (or remains constant) as the system evolves according to its dynamic rule.

One appropriate energy function is [41]

$$E(v_1, v_2, ..., v_N) = -\frac{1}{2}\sum_i \sum_j w_{ij} v_i v_j + \sum_i \int_0^{v_i} s^{-1}(v)\mathrm{d}v - \sum_i I_i v_i, \tag{2.6}$$

where the matrix of connection weights $\boldsymbol{w}$ is symmetric, i.e. $w_{ij} = w_{ji}$. Here $s^{-1}(x)$ is the inverse function of s and s is a monotonically increasing function taking values from -1 to 1, typically a sigmoid function.

The self-coupling terms w_{ii} may actually be omitted altogether from the energy function. They make no appreciable difference to the stability of the patterns when N is large [35], but they do affect the dynamics, so Kanter and Sopolinsky suggested to

omit them [47]. Therefore in the following, we define $w_{ii} = 0$ for all neurons.

One can prove that E is monotonically decreasing as the system evolves by taking its time derivative for a symmetric $\boldsymbol{w}$:

$$\begin{aligned}\frac{\mathrm{d}E}{\mathrm{d}t} &= -\sum_i \frac{\mathrm{d}v_i}{\mathrm{d}t}\Big(\sum_{i\neq j} w_{ij}v_j - u_i + I_i\Big) \\ &= -\sum_i a_i\Big(\frac{\mathrm{d}v_i}{\mathrm{d}t}\Big)\Big(\frac{\mathrm{d}u_i}{\mathrm{d}t}\Big) \\ &= -\sum_i a_i \frac{\mathrm{d}s^{-1}(v_i)}{\mathrm{d}v_i}\Big(\frac{\mathrm{d}v_i}{\mathrm{d}t}\Big)^2\end{aligned} \tag{2.7}$$

as the terms inside the parentheses in the first line represent the right-hand side of Eq. (2.5).

Since $s^{-1}(v_i)$ is a monotonically increasing function and a_i is positive, each term in this sum is nonnegative. Therefore,

$$\frac{dE}{dt} \leq 0, \quad \text{and } \frac{dE}{dt} = 0 \quad \Rightarrow \quad \frac{dv_i}{dt} = 0 \text{ for all } i. \tag{2.8}$$

Thus the change $\boldsymbol{\Delta}E$ is negative under the algorithm, i.e., E is a monotonically decreasing function. The state of the CNN keeps changing until it reaches a local minimum of E. Given the boundedness of E, evolution of the state of the CNN must lead to one of the stable states. Hence the attractors (memorized patterns) of the system correspond to the local minima of this energy landscape. Importantly, these attractors are fixed points and cannot be limit cycles or any other more complex attractors, since otherwise there would have been an oscillations of energy values, which would imply the possibility of its increase at some stages.

The concept of energy function is widely used in many fields where there is a state function that is monotonically decreasing during dynamical evolution, or that must be minimized to find a stable or an optimum state. Similarly, when the function increases or must be maximized, the convention is reversed.

2.2.4 Learning

Before the NN can start recognising patterns correctly, it should learn by receiving an appropriate training. When input patterns are presented to the NN, the network memorises this information or adjusts its connection weights according to some algorithm. Learning in algorithm-based devices can be generally divided into supervised learning and unsupervised learning. The type of learning is determined by the manner in which the connection weights change [36].

Supervised learning

In a supervised learning environment, a teacher is required to specify the desired output for every input pattern from the training set. Learning is done by comparing the actual output of the network with the known answer and adjusting the values of w_{ij} based on the error calculated [35]. This method is also known as error-correction learning [33]. Backpropagation method is another form of supervised learning [33].

Reinforcement learning is sometimes similar to supervised learning, but the only feedback from the teacher is not what the correct answer is, but whether the output is correct or incorrect. This learning method is widely used in optimal control problems [35].

Unsupervised learning

In unsupervised learning, there is no target output or feedback. The network must discover categories and regulations in the input patterns without any supervision [35]. The oldest and most famous unsupervised learning is Hebbian learning.

On-line and off-line learning

Another way to classify the learning methods is based on their ability to work on-line or off-line. In off-line learning, learning phase and operation phase are distinct.

In on-line learning, the NN learns and operates at the same time, in other words, it can categories and recognises the input patterns simultaneously. Usually, supervised learning is off-line, whereas unsupervised learning is typically on-line [51].

Here we demonstrate Hebbian learning to illustrate how to train a NN by a set of examples through a learning algorithm.

Hebbian learning

Hebbian learning is a widely used unsupervised learning method. In the Hebbian synaptic modification hypothesis, the change in the synaptic strength is proportional to the correlation between the pre- and postsynaptic signals [50]. The strength w_{ij} of a connection depends on time and is adjusted according to a set of equations which involve the neuronal activity.

Hebbian learning in a CNN can be described by [19]

$$\frac{\mathrm{d}w_{ij}}{\mathrm{d}t} = \varepsilon(-w_{ij} + s(u_i)s(u_j)), \tag{2.9}$$

where ε is the learning rate constant. A decay term $-w_{ij}$ is introduced in Eq. (2.9) to prevent the weights from diverging. Note that the Hebbian prescription (2.9) automatically yields $w_{ij} = w_{ji}$, which means that energy function can be introduced in a NN using Hebbian learning.

Hebbian learning is an interactive mechanism [33]. The rate of change of connection w_{ij} depends on an interaction between the neuron i and j on either side of the connection. Also, the synaptic modification is produced by the co-occurrence of activities of both neuron i and neuron j.

Here we show an example illustrating a CNN undergoing Hebbian learning, which was originally obtained by Dong and Hopfield [19]. The dynamics of both the modification of connection weights w_{ij} and the recall of memories are combined in the

following set of the dynamic equations

$$a_i \frac{\mathrm{d}u_i}{\mathrm{d}t} = -u_i + \beta \sum_{j=1}^{N} w_{ij} s(u_j) + \alpha I_i, \tag{2.10}$$

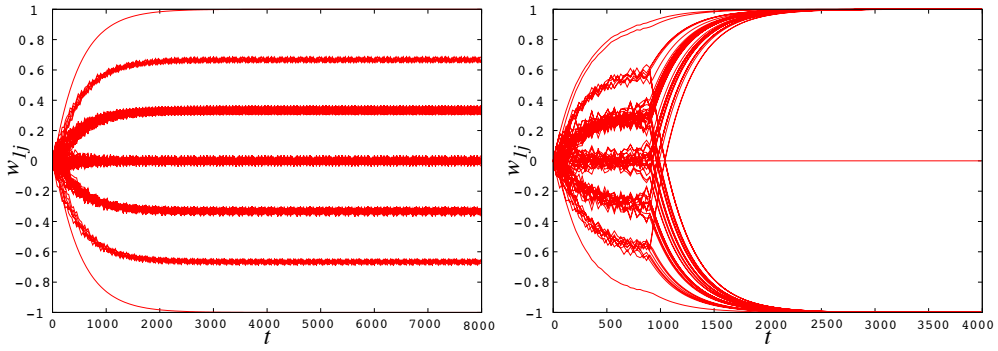
$$\frac{\mathrm{d}w_{ij}}{\mathrm{d}t} = \varepsilon(s(u_i)s(u_j) - w_{ij}), \tag{2.11}$$

where s is the sigmoid function given by Eq. (2.4). In this NN, two scale constants α and β are introduced before the external input I_i and the feedback from other neurons respectively. Therefore, when $\alpha > \beta$, the external input is stronger than the sum of signals coming through the interconnections, and the neural activities are largely determined by the current I_i.

A CNN of 81 neurons taking the form of Eq. (2.10) is trained according to Eq. (2.11) after being presented with 6 input signals $\boldsymbol{I^k} = (I_1^k, I_2^k, \cdots, I_{81}^k)$, $k = 1, \cdots, 6$. The values I_i^k are randomly chosen binary numbers. To generate the values I_i^k, we first generate 6 random vectors $\boldsymbol{x^k} = (x_1^k, x_2^k, \cdots, x_{81}^k)$, $k = 1, \cdots, 6$, whose components x_i^k are uniformly distributed random numbers, ranging from -10 to 10. Then we obtain I_i^k by taking

$$I_i^k = \begin{cases} 1 & \text{if } x_i^k \geq 0, \\ -1 & \text{if } x_i^k < 0. \end{cases} \tag{2.12}$$

A sequence of input patterns, obtained by looping these input vectors $\boldsymbol{I^k}$ for a certain times, is then presented to the network. The learning parameters are $\beta = 0.3$, $\varepsilon = 1/300$. Evolution of the strengths of connections to/from neuron 1 is illustrated in Fig. 2.3. When α is large, e.g. $\alpha = 30$, the connection weights w_{ij}'s tend to 7 different values (shown in Fig. 2.3(a)), which correspond to the values of an average over 6 patterns of the outer product of input patterns, i.e. $w_{ij} = \frac{1}{6} \sum_{k=1}^{6} I_i^k I_j^k$. All 6 of the input patterns are stored in the network [19]. But when we take the same parameters, but a smaller



(a) $\alpha = 30$ (b) $\alpha = 3$

Figure 2.3: Evolution of the connection strengths according to Eq. (2.11) in the course of Hebbian learning.

value of α, $\alpha = 3.0$, which means the connections become dominating, the network selects only one input pattern to memorise.

This example illustrates the process of learning in the NN. As shown in Fig. 2.3, the connections between neurons are very weak at the beginning, thus the internal input terms $w_{ij}s(u_j)$ contribute much less to the change of connections than the external input patterns I_i. Note that in the beginning of the learning process, the connection weights in Fig. 2.3(b) are the same as in Fig. 2.3(a), and before $t \approx 800$, there are 6 attractors in the energy landscape with shallow and small basins representing the stored patterns as required. However, at a smaller value of α, the interconnections influence the neural activities stronger than the external input, and the contribution from the input patterns I_i becomes small in comparison and can be ignored. In this case, the connections tend to either +1, or −1, or 0, and the network chooses to remember only one pattern [19]. The stability of the respective attractor makes the network ignore other incoming patterns unless they are very strong.

This example also demonstrates the restricted storage capacity in NNs. The network finds it difficult to learn any other patterns once it has selected a pattern to store, unless the input pattern is strong enough to lead the network. This will be discussed in the next Section.

2.2.5 Storage capacity of a neural network

Networks that can create associative content-addressable memories are designed to be able to recall a previously learnt pattern when given an example pattern that is similar to one of the stored patterns. This property is inherent in CNNs and is implemented through the synaptic plasticity in which connections are being changed in response to input according to certain rules. However, Amit and Fusi showed that this mechanism does not achieve long-lasting memory storage and is inefficient for many types of memory, such as memorising patterns of mean spike rates [22], memory of an uninterrupted flow of uncorrelated stimuli [3] and memory generated by Hebbian learning with low rate of presentation of patterns [2]. Therefore, CNNs possess fundamental limitations such as bounded **storage capacity** and memories that are short-lived. Storage capacity is the maximum number of patterns that can be stored in a network of the given size. It influences both the efficiency of information storage, and the time of information processing.

The main reason for the limited storage capacity is that the old memories created by a suitable choice of connections are destroyed when these connections change to form new memories [23; 24]. Connections retaining one stored pattern can be overwritten when some of them are used by the new memory. There is evidence available from neurobiology that in biological NNs old memories are destroyed by the ongoing neural activity and the acquisition of new memories, rather than simply due to their passive decay with time. Specifically, in [23; 24] evidence is provided from models, experiments and psychophysical studies. Therefore, protecting the stored memories from being corrupted by ongoing modifications of connections is the main challenge in modelling long-lasting memory with new memories being continually generated.

Storage capacities of NNs were estimated analytically in a series of papers [1; 2; 4; 5; 6]. One of such estimates is based on the mean field analysis of a stochastic neural

network, in which some function describing the given neuron depends on a random signal, as described in [35].

There are several different expressions for the memory capacity, depending on the criterion imposed on the acceptable error in memory retrieval. In a CNN of N neurons, each pattern is coded as an N-dimensional vector, i.e. each pattern comprises N tuples of real numbers. If p stored patterns are chosen randomly, the storage capacity of the network is the maximum value of p such that p original memories can be recalled with a certain level of error. If we accept that every component of the vector representing the given stored pattern is recovered correctly with probability 99%, the capacity is $p = 0.138N$ and this is known as relative capacity [6]. Another expression of capacity is based on the assumption that most of the memories are recovered perfectly, that is, if we get all N dimensions of one pattern right with 99% probability, then p can be no more than $N/(2 \ln N)$ asymptotically as N approaches infinity [54; 72]. If we add a restriction that *all* the p patterns can be recalled exactly, which requires correct recovery of Np tuples with 99% probability, the capacity p becomes $N/(4 \ln N)$ [54]. A capacity suggesting that all memories are recoverable without unacceptable errors is called absolute capacity [68].

Now we can see that the storage capacity is proportional to the number of neurons N, if a finite number exceptional memories are allowed. If we require that most of the patterns stored can be recovered exactly, the capacity is proportional to $N/(\ln N)$. This implies that in order to store more patterns, a larger network is required.

2.3 Motivation for this work

However, NNs have well-documented limitations, such as finite memory capacity and inability to retain old memories when the new ones are created. Note that the fundamental feature of a NN is the fact that the architecture of its individual units is rigid, and flexibility is present only in connections.

Low storage capacity of CNNs can be understood in terms of DS theory. If every neuron of the NN is described by one or several differential equations, i.e. as a DS, the network of coupled neurons becomes a high-dimensional continuous-time DS, in which connection strengths play the role of control parameters. By changing one or several parameters of a DS, one alters the structure of the whole velocity field. If a certain local modification of the field occurs as desired, e.g. to form an attractor, the shape of the field in other parts of the phase space can occur uncontrollably and make some other attractors disappear. Such DSs are inflexible in the sense that there is an upper limit on the number of attractors that can coexist in their phase space. When a new attractor is created by changing some parameters of the system, some of the old attractors will disappear. To have more attractors in the system, the nature of the functions describing the dynamics has to change, which means that the structure of every neuron has to change. This is impossible because each neuron is constructed with a fixed architecture, which determines the velocity vector field of the whole CNN.

On the other hand, the inflexibility of a NN leads to the problem of spurious attractors, which are attractors that do not correspond to any stored memories, or categories [35]. They tend to have rather small basins of attraction as compared to those of the stored patterns, but from suitable initial condition the network evolves towards one of these attractors which represent no meaningful category. These spurious attractors affect pattern recognition, but are created unavoidably.

To overcome these limitations of NNs, Janson and Marsden proposed an alterna-

tive approach to model memory formation [42; 43]: a DS with plastic self-organising velocity field or a self-shaping DS. For brevity, in this dissertation we will call such system plastic DS (PDS). In [42; 43] a simple model was introduced that described memory formation and spontaneous categorization of incoming patterns occurring without supervision and online. This is achieved by assuming that the whole velocity field of the DS is fully flexible and can be affected by the external stimuli directly rather than through modification of some control parameters. In a simple PDS of a gradient type, in response to the incoming stream of patterns, the vector field converges to the probability density distribution of the input process, taken with negative sign. This model will be discussed in Chapter 3.

At this stage, a PDS is a mathematical abstraction. Namely, the velocity field of even the simplest gradient PDS is a function that evolves and can develop any number of maxima or minima inside any bounded area of the phase space. Moreover, new minima are formed in such a manner that the ones formed earlier are not affected. There are no analytic functions possessing such features, and one cannot achieve the same effect by taking a DS with a velocity field described by some analytic function and modifying its parameters. Our goal is to explore the possibility to implement a conceptual idea of a PDS in practice. One possibility is electronic circuits, which can implement analytic functions. Specifically, we will try to approximate the evolving velocity field by series of evolving analytic functions in a number of ways. If our approach is successful, it will pave the way to constructing PDS based on electronic devices.

Chapter 3

Dynamical systems with plastic self-organising velocity fields

Recently, Janson and Marsden [42; 43] proposed a new mathematical approach to describe artificial learning systems by introducing a new type of DSs, which automatically modify their velocity vector fields in response to external stimuli. In a simple example of a gradient PDS subjected to a time-dependent stimulus arising from a stationary and ergodic input random process, the vector field tends to the gradient of the negative of the probability density distribution of the input process. Such systems can automatically create categories from the stimuli and recognize familiar patterns at the same time, that constitutes the online unsupervised learning. In the phase space of such a PDS, a single pattern is represented by a single point, the most probable patterns by the minima of energy landscape, and the categories of patterns by the basins of attraction. With this, attractors and basins are formed automatically at the right locations.

3.1 Simple plastic dynamical system (PDS)

The simplest PDS is derived from a conceptual analogy that our memory works like a memory foam, as found in some bed mattresses. This foam pits when you press against it but slowly returns to the initial shape after the pressure is removed.

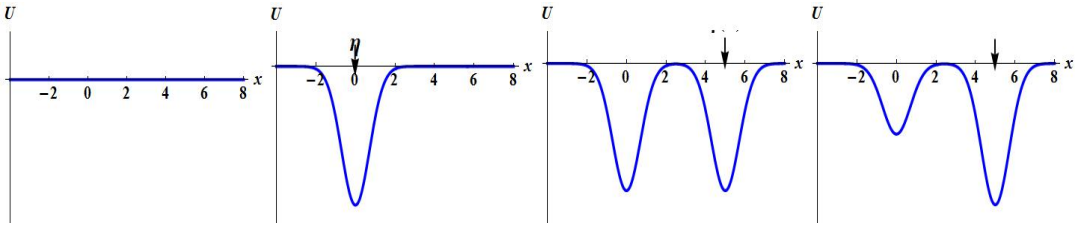
In the text of [51] evolution of the auxiliary function was analysed rather than the one of the actual landscape of the gradient PDS. Here we derive the same PDS model in a slightly different way and focus on the analysis of the evolution of the actual landscape function.

As mentioned in Section 2.2.3, the energy function of a NN provides a convenient framework that allows one to describe the formation and retrieval of memories. Similarly in this model, the shape of this memory foam is described by a certain energy function V. The simplest PDS is the gradient system, which describes the behaviour of a massless particle placed in the landscape V,

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = -\frac{\partial V(\boldsymbol{x}, t)}{\partial \boldsymbol{x}}, \tag{3.1}$$

where $\boldsymbol{x}$ represents the location in the N-dimensional space that represents a certain pattern. The particle goes to the nearest minimum at the rate which is equal to the negative of the gradient of the landscape. Note that the energy function V is being continually reshaped by the incoming stimuli, i.e. evolves in time t. Equation (3.1) can be rewritten as a set of N equations for every component of vector x as

$$\begin{aligned}
\frac{\mathrm{d}x_1}{\mathrm{d}t} &= -\frac{\partial V(x_1, x_2, \cdots, x_N, t)}{\partial x_1}, \\
\frac{\mathrm{d}x_2}{\mathrm{d}t} &= -\frac{\partial V(x_1, x_2, \cdots, x_N, t)}{\partial x_2}, \\
&\vdots \\
\frac{\mathrm{d}x_N}{\mathrm{d}t} &= -\frac{\partial V(x_1, x_2, \cdots, x_N, t)}{\partial x_N}.
\end{aligned} \tag{3.2}$$



(a) $t = 0$ (b) $t = 1$ (c) $t = 2$ (d) $t = 3$

Figure 3.1: Sketch of the idea of the one-dimensional PDS energy landscape stretched in x direction. The landscape is assumed to be initially flat, i.e. $U(x, 0) = 0$ (see (a)). A "rock" (the input) drops onto the landscape at position $x = \eta$ and generates a dent in the landscape (see (b)), which is the deepest exactly at $x = \eta$. If aother rock drops at a new position $x = \eta$ at a new time moment, another dent appears in the landscape (see (c)). With further rocks dropping at different positions (see (d)), the "foam" is shaped in response to the stimulus continuously.

This energy landscape V is a Lyapunov function, and can only decrease monotonically while the system evolves. Hence the only possible attractors of the system are fixed points.

To convey the general idea of a PDS, the energy landscape is first described by an auxiliary function $U(\boldsymbol{x}, t)$. The landscape is assumed to be initially flat, i.e. $U(\boldsymbol{x}, 0) = 0$, and infinitely elastic. At time t, for example, an external stimulus $\boldsymbol{\eta}(t)$ coded as an N-dimensional vector is applied to the landscape. This stimulus can be seen as a "rock" that drops onto the the landscape at position $\boldsymbol{x} = \boldsymbol{\eta}(t)$ and generates a dent in the landscape. This dent is deepest exactly at $\boldsymbol{x} = \boldsymbol{\eta}(t)$ and get shallower at larger distances from $\boldsymbol{\eta}$. This way, the landscape learns about the occurrence of the rock and its position. With further rocks dropping at different positions, more dents are generated and some may merge into a larger dent, which corresponds to a larger category. The forgetting capacity is introduced by assuming that the landscape is elastic with elasticity factor $k \in [0, 1)$ and that the deeper the dent at position $\boldsymbol{x}$ is, the faster the landscape tries to revert to its initial flat state. Areas not subjected to stimulus will stay at their original state. With this formulation, the landscape evolves with time

in response to a continually varying external stimulus. This process is illustrated for a one-dimensional landscape in Fig. 3.1, which is only a sketch.

The shaping of the function U is first expressed by a discretized equation

$$U(\boldsymbol{x}, t + \Delta t) = U(\boldsymbol{x}, t) - g(\boldsymbol{x} - \boldsymbol{\eta}(t))\Delta t - kU(\boldsymbol{x}, t)\Delta t, \tag{3.3}$$

which describes how the landscape changes over a small but finite time interval Δt. $g(\boldsymbol{z})$ is some non-negative bell-shaped function, such as a Gaussian function, which represents the shape of a single dent,

$$g(\boldsymbol{z}) = \left(2\pi\sigma^2\right)^{-N/2} \exp\left(-\frac{1}{2}\sum \frac{z_i^2}{\sigma^2}\right), \tag{3.4}$$

where σ determines the width of the dent and is chosen manually. In principle, a dent can have different values of widths in different dimensions. Here we assume for simplicity that the width of the dent in every dimension takes the same value σ. With a relatively large value of σ, the landscape will generate a small amount of wide dents and some initially small categories will merge and form larger categories. On the other hand, when σ is relatively small there will be many peaked dents with small basins making the landscape too detailed. This principle can be relevant to the human cognitive system in that different individuals have different capacity for categorisation and capturing the detail.

In Eq. (3.3), it is assumed that the difference between two subsequent states of the landscape is proportional to the waiting time Δt in response to a certain stimulus. That is, the longer the stimulus is applied at one location, the deeper the dent becomes. Hence the effect from applying the pressure to the landscape comes from the shape of the pressure plus its duration.

In Eq. (3.3) move $U(\boldsymbol{x}, t)$ to the left-hand side, divide both sides by Δt and take the

limit as $\Delta t \to 0$, to obtain a differential equation for U,

$$\frac{\partial U(\boldsymbol{x},t)}{\partial t} = -g(\boldsymbol{x}-\boldsymbol{\eta}) - kU(\boldsymbol{x},t). \tag{3.5}$$

One can show by numerical simulation of Eq. (3.5) with $k = 0$ that $U(\boldsymbol{x},t)$ has a linear trend and goes to $-\infty$ as time tends to ∞ [42; 43; 51]. In order to keep the shape of the landscape, but eliminate the trend, the following change of variables was applied when $t \neq 0$

$$\begin{aligned} V(\boldsymbol{x},t+\Delta t) &= \frac{U(\boldsymbol{x},t+\Delta t)}{t+\Delta t}, \\ V(\boldsymbol{x},t) &= \frac{U(\boldsymbol{x},t)}{t}. \end{aligned}$$

Substituting $U(\boldsymbol{x},t+\Delta t) = V(\boldsymbol{x},t+\Delta t)(t+\Delta t)$ and $U(\boldsymbol{x},t) = tV(\boldsymbol{x},t)$ into Eq. (3.3) one obtains

$$V(\boldsymbol{x},t+\Delta t)(t+\Delta t) = V(\boldsymbol{x},t)t - g(\boldsymbol{x}-\boldsymbol{\eta}(t))\Delta t - k\Delta t V(\boldsymbol{x},t)t, \tag{3.6}$$

$$V(\boldsymbol{x},t+\Delta t)t - V(\boldsymbol{x},t)t = -V(\boldsymbol{x},t+\Delta t)\Delta t - g(\boldsymbol{x}-\boldsymbol{\eta}(t))\Delta t - k\Delta t V(\boldsymbol{x},t)t.$$

Assuming that $t \neq 0$ and $\Delta t \neq 0$, one can divide all terms by $t \cdot \Delta t$ to obtain

$$\frac{V(\boldsymbol{x},t+\Delta t) - V(\boldsymbol{x},t)}{\Delta t} = -\frac{1}{t}\Big(V(\boldsymbol{x},t+\Delta t) + g(\boldsymbol{x}-\boldsymbol{\eta}(t))\Big) - kV(\boldsymbol{x},t). \tag{3.7}$$

Taking the limit as $\Delta t \to 0$, we obtain the continuous-time version of the system

$$\frac{\partial V(\boldsymbol{x},t)}{\partial t} = -\frac{1}{t}\Big(V(\boldsymbol{x},t) + g(\boldsymbol{x}-\boldsymbol{\eta}(t))\Big) - kV(\boldsymbol{x},t). \tag{3.8}$$

Therefore, the full description of the simplest PDS consists of Eqs. (3.1), (3.8) and the initial condition for V

$$V(\boldsymbol{x},0) = 0. \tag{3.9}$$

In [51] it was shown that with $k = 0$ all input patterns equally contribute to the formation of the landscape V regardless of their ordering in time. However, the behavior of this landscape with non-zero k was not considered. Below we analyse the evolution of the plastic landscape under the assumption that $k \in [0, 1)$.

Return to the discrete form (3.6) of equation for $V(\boldsymbol{x}, t)$ and rearrange terms in it as follows

$$V(\boldsymbol{x}, t + \Delta t) = \frac{t}{t + \Delta t}(1 - k\Delta t)V(\boldsymbol{x}, t) - \frac{\Delta t}{t + \Delta t}g(\boldsymbol{x} - \boldsymbol{\eta}(t)). \tag{3.10}$$

Write out several consecutive values of $V(\boldsymbol{x}, t)$ separated by the time step Δt

$$\begin{aligned}
V(\boldsymbol{x}, 0) &= 0, \\
V(\boldsymbol{x}, \Delta t) &= -g(\boldsymbol{x} - \boldsymbol{\eta}(0)), \\
V(\boldsymbol{x}, 2\Delta t) &= \frac{\Delta t}{2\Delta t}(1 - k\Delta t)V(\boldsymbol{x}, \Delta t) - \frac{\Delta t}{2\Delta t}g(\boldsymbol{x} - \boldsymbol{\eta}(\Delta t)) \\
&= -\frac{1}{2}(1 - k\Delta t)g(\boldsymbol{x} - \boldsymbol{\eta}(0)) - \frac{1}{2}g(\boldsymbol{x} - \boldsymbol{\eta}(\Delta t)), \\
V(\boldsymbol{x}, 3\Delta t) &= \frac{2}{3}(1 - k\Delta t)V(\boldsymbol{x}, 2\Delta t) - \frac{1}{3}g(\boldsymbol{x} - \boldsymbol{\eta}(2\Delta t)) \\
&= -\frac{1}{3}(1 - k\Delta t)^2 g(\boldsymbol{x} - \boldsymbol{\eta}(0)) - \frac{1}{3}(1 - k\Delta t)g(\boldsymbol{x} - \boldsymbol{\eta}(\Delta t)) - \frac{1}{3}g(\boldsymbol{x} - \boldsymbol{\eta}(2\Delta t)), \\
&\vdots \\
V(\boldsymbol{x}, n\Delta t) &= -\frac{1}{n}(1 - k\Delta t)^{n-1} g(\boldsymbol{x} - \boldsymbol{\eta}(0)) - \frac{1}{n}(1 - k\Delta t)^{n-2} g(\boldsymbol{x} - \boldsymbol{\eta}(\Delta t)) \\
&\quad - \cdots - \frac{1}{n}(1 - k\Delta t)g(\boldsymbol{x} - \boldsymbol{\eta}((n-2)\Delta t)) - \frac{1}{n}g(\boldsymbol{x} - \boldsymbol{\eta}((n-1)\Delta t)).
\end{aligned}$$

When $k \neq 0$, we have

$$0 < k < 1,\ 0 < \Delta t < 1 \Rightarrow 0 < (1 - k\Delta t)^i < 1, \text{ and } (1 - k\Delta t)^{i+1} < (1 - k\Delta t)^i,\ (i = 1, 2, ..., n).$$

The latter implies that the more recent stimuli make more impact on shaping the energy landscape V. The stimuli applied in the more distant past are being erased from the landscape faster than the more recent ones.

When $k = 0$, the landscape does not forget any information, namely

$$V(\boldsymbol{x}, n\Delta t) = -\frac{1}{n}g(\boldsymbol{x} - \boldsymbol{\eta}(0)) - \frac{1}{n}g(\boldsymbol{x} - \boldsymbol{\eta}(\Delta t)) - \cdots - \frac{1}{n}g(\boldsymbol{x} - \boldsymbol{\eta}((n-1)\Delta t)), \tag{3.11}$$

which coincides with the expression obtained in [51] and confirms that every single stimulus at every time moment makes the same contribution to the landscape.

Equation (3.8) describes how the landscape $V(\boldsymbol{x}, t)$ evolves in time in response to a continually varying external stimulus $\boldsymbol{\eta}(t)$. It is shown in [42; 43] that under if the PDS does not forget any information it learnt, i.e. $k = 0$, and the stimulus represents a realization of a stationary and ergodic random process, the energy landscape tends to the multi-dimensional probability density distribution $P(\boldsymbol{\eta})$ of this process taken with negative sign. This way, the system automatically creates categories from the input patterns. Since we only consider gradient systems in which the only possible attractors are fixed points, the most probable patterns from each category are represented by stable fixed points, and categories are represented by their basins of the attraction.

Online pattern recognition in the PDS (3.1), (3.8), (3.9) is achieved by using the input twice: not only to shape the landscape V, but also as an initial condition for Eq. (3.1). The state of the system will start at the location of the input and then evolve to the local minimum. Thus, after the landscape is altered by the stimulus, the same stimulus is recognised by the system as belonging to one of the existing, but slightly adjusted, categories, or to a category just created. The procedure is repeated in time, resulting in continually evolving classification rules.

In this dissertation we will use the PDS defined by Eqs. (3.1), (3.8) and (3.9) as the reference model whose performance we will be trying to imitate in models, that can potentially be implemented in practice.

3.2 Examples of learning in the plastic dynamical system (PDS)

Here we illustrate how the one-dimensional PDS (3.8) without the forgetting term (k=0) performs categorisation using random data as an input.

We produce random stimuli consisting of a sequence of numbers coming from two and three different categories. We assume that these numbers are randomly taken from a probability density distribution (PDF) with two and three peaks, respectively. The values corresponding to the maxima of the PDF would be the most probable (typical) representatives of a certain category, and the vicinities of each maximum are the members of the respective category.

The easiest way to generate such a signal is by launching an Ornslein-Uhlenbeck process [26] through the creation of a potential $L(\eta)$ of the required shape with gradient $l(\eta)$ and formulating a stochastic gradient DS with an applied random Gaussian white noise $\zeta(t)$ with zero mean and unit variance as follows:

$$\begin{aligned} \frac{\mathrm{d}\eta}{\mathrm{d}t} &= \frac{\partial L(\eta)}{\partial \eta} + \sqrt{D}\zeta(t) \\ &= l(\eta) + \sqrt{D}\zeta(t), \end{aligned} \tag{3.12}$$

where D is the variance of the random force. Equation (3.12) describes the behaviour of a massless particle on a landscape $L(\eta)$ under the influence of a stochastic force $D\zeta(t)$.

The stimulus $\eta(t)$ generated this way is correlated, which means that the current and the future values of this random process depend on its previous values. Correlated stimulus helps to illustrate how the plastic landscape evolves while the PDS learns. An example using uncorrelated stimulus is given in [42; 43].

Example: stimuli from two categories

We first generate the stimuli taking values from the PDF with two peaks, and for this purpose construct the landscape $L(\eta)$ for Eq. (3.12) with two minima, such that $\frac{\partial L(\eta)}{\partial \eta} = l(\eta)$ reads

$$l(\eta) = 0.36\eta(1 - 0.6\eta)(1 + 0.4\eta). \tag{3.13}$$

By applying to Eq. (3.12) the random noise with $D = 0.625$, we obtain a sequence of values of $\eta(t)$ shown in Fig. 3.2(a) that models a collection of patterns belonging to two categories.

One can formulate a Fokker-Planck equation for the evolution of the PDF of the random process $\eta(t)$, which would correspond to Langevin equation (stochastic differential equation) (3.12) [10]:

$$\frac{\partial}{\partial t}P(x) = -\frac{\partial}{\partial x}(l(x)P(x)) + \frac{1}{2}\frac{\partial^2}{\partial x^2}(DP(x)). \tag{3.14}$$

A stationary solution to this Fokker-Planck equation describing an equilibrium PDF of $\eta(t)$ can be obtained analytically [10] and reads

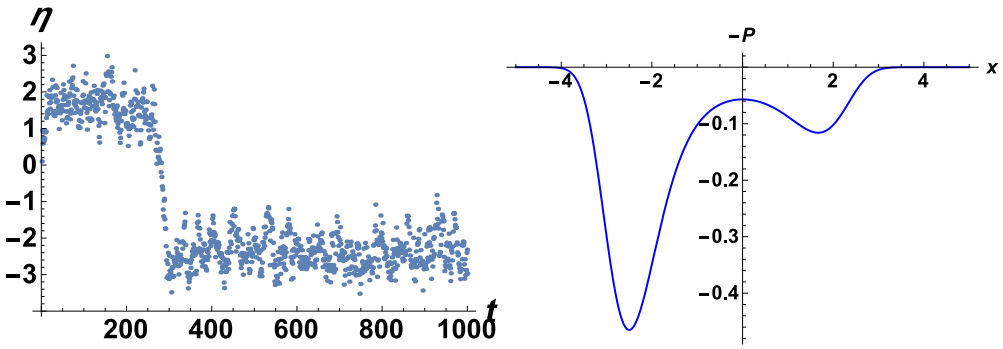
$$P(x) = \frac{1}{C}\exp\left(2\int \frac{1}{D}l(x)\mathrm{d}x\right), \tag{3.15}$$

where the integral inside the brackets is an indefinite integral. Substituting the formula of $l(x)$ expressed by Eq. (3.13) and the value of D into the integral, one obtains

$$\int \frac{1}{D}l(x)\mathrm{d}x = \frac{0.36}{0.625}(0.5x^2 - 0.067x^3 - 0.06x^4).$$

In Eq. (3.15) C is a constant such that $\int_{-\infty}^{\infty} P(x)\mathrm{d}x = 1$. Then C can be found as

$$C = \int_{-\infty}^{\infty} \exp\left(2\int \frac{1}{D}l(x)\mathrm{d}x\right)\mathrm{d}x.$$



(a) (b)

Figure 3.2: Illustration of the input patterns belonging to two categories.
(a) Stimulus applied to the PDS in which values are randomly taken from two categories.
(b) Analytically estimated PDF taken with negative sign.

Substituting the formula of $l(x)$ and the value of D, one obtains

$$P(x) = \frac{\exp\left(2\int \frac{1}{0.625} l(x)\mathrm{d}x\right)}{\int_{-\infty}^{\infty} \exp\left(2\int \frac{1}{0.625} l(x)\mathrm{d}x\right)\mathrm{d}x} = \frac{1}{17.562} \exp\left(1.152(0.5x^2 - 0.067x^3 - 0.06x^4)\right).$$

We express this PDF as a function of x rather than of η for the convenience of its comparison with the shape of the landscape $V(x)$ of the PDS. The PDF of the random process $\eta(t)$ taken with negative sign is shown in Fig. 3.2(b). With the chosen shape of $l(\eta)$ all values of η are in the range $[-4,\ 4]$.

Evolution of the landscape $V(x,t)$ being subjected to $\eta(t)$ is illustrated in Fig. 3.3(a) in 3D. Also, in Fig 3.3(b) $V(x,t)$ is shown together with input $\eta(t)$: the depth of the landscape is represented by color and $\eta(t)$ by dots. Here in $g(x)$ we use $\sigma^2 = 0.1$. When a new value of η is applied at each consecutive time moment t, the landscape adjusts itself and finally develops two dents. The landscape $V(x,t)$ at the end of learning is shown in Fig. 3.3(c). Eventually the landscape shapes into the negative of the PDF $P(x)$ shown in Fig. 3.3(d) which is estimated as a distribution histogram from the realisation of the stimulus.

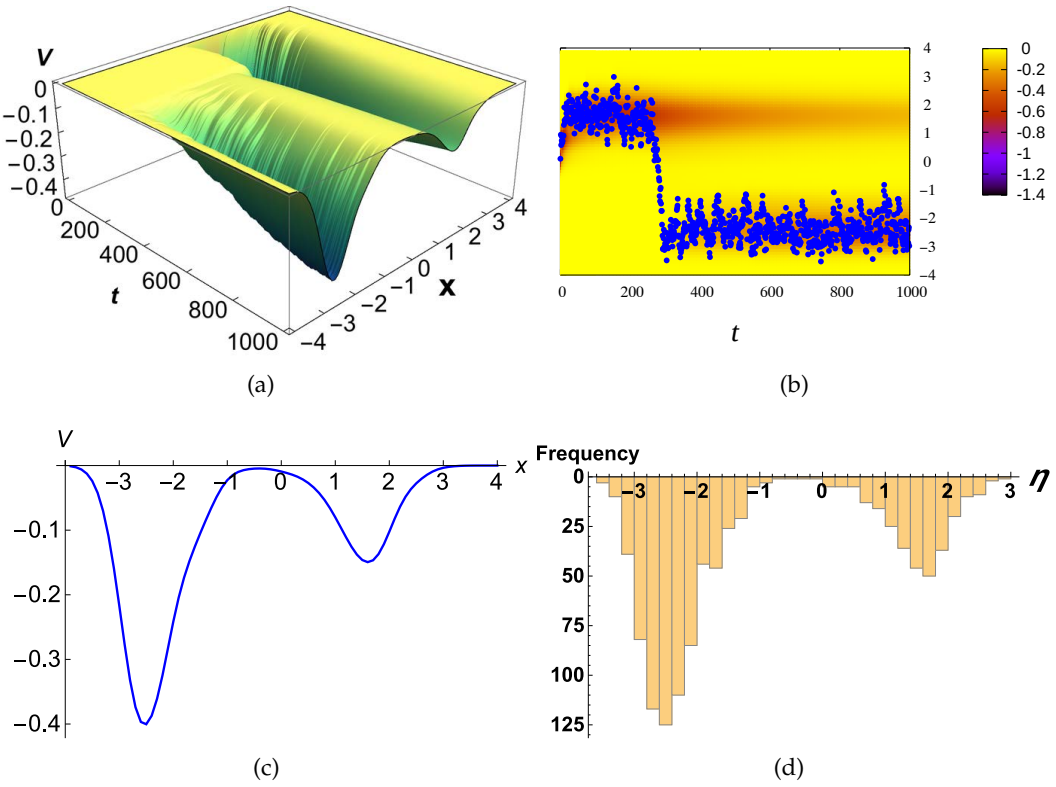
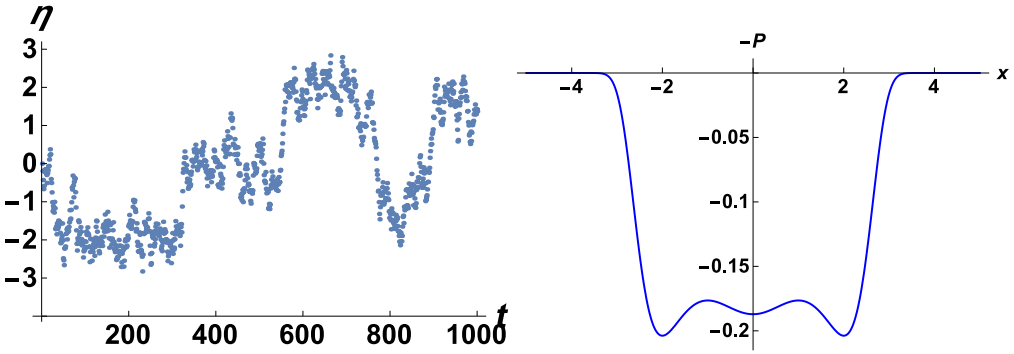


Figure 3.3: Energy landscape of the PDS (with $\sigma^2 = 0.1$) processing input patterns that come from two categories.
(a) Evolution of PDS $V(x,t)$ illustrating the development of two wells representing two categories.
(b) Evolution of $V(x,t)$ (same as in (a)) with the value of V shown by color on the (x,t) plane, and input stimuli shown by blue points.
(c) Energy landscape at the end of learning.
(d) Distribution histogram of input stimuli.



(a) (b)

Figure 3.4: Illustration of the input patterns belonging to three categories.
(a) Input to PDS: a sequence of values randomly taken from three categories.
(b) Analytically estimated PDF taken with negative sign.

Example: stimuli from three categories

In the second example, a sequence of input patterns belonging to three categories is used. The input is generated by numerically solving Eq. (3.12) but with a new form of $l(\eta)$ described by

$$l(\eta) = -0.08\eta(1 - 0.5\eta)(1 + 0.5\eta)(1 - \eta)(1 + \eta). \tag{3.16}$$

The input η and its analytically estimated PDF taken with negative sign are plotted in Fig. 3.4. The stationary PDF reads

$$P(x) = \frac{\exp\left(2\int \frac{1}{0.625} l(x)\mathrm{d}x\right)}{\int_{-\infty}^{\infty} \exp\left(2\int \frac{1}{0.625} l(x)\mathrm{d}x\right)\mathrm{d}x} = \frac{1}{5.344}\exp\Big(-0.256(0.5x^2 - 0.3125x^4 + 0.04167x^6)\Big).$$

The inputs can be categorised into three classes and the most probable patterns are -2, 0, and 2, respectively. The PDF $P(x)$ is symmetric with respect to $x = 0$.

Figure 3.5 shows that in response to the input signals shown by blue points in Fig. 3.5(b), the PDS develops the landscape with three wells representing three categories. The landscape $V(x, t)$ at the end of learning (Fig. 3.5(c)) shapes into the PDF $P(x)$ (Fig.

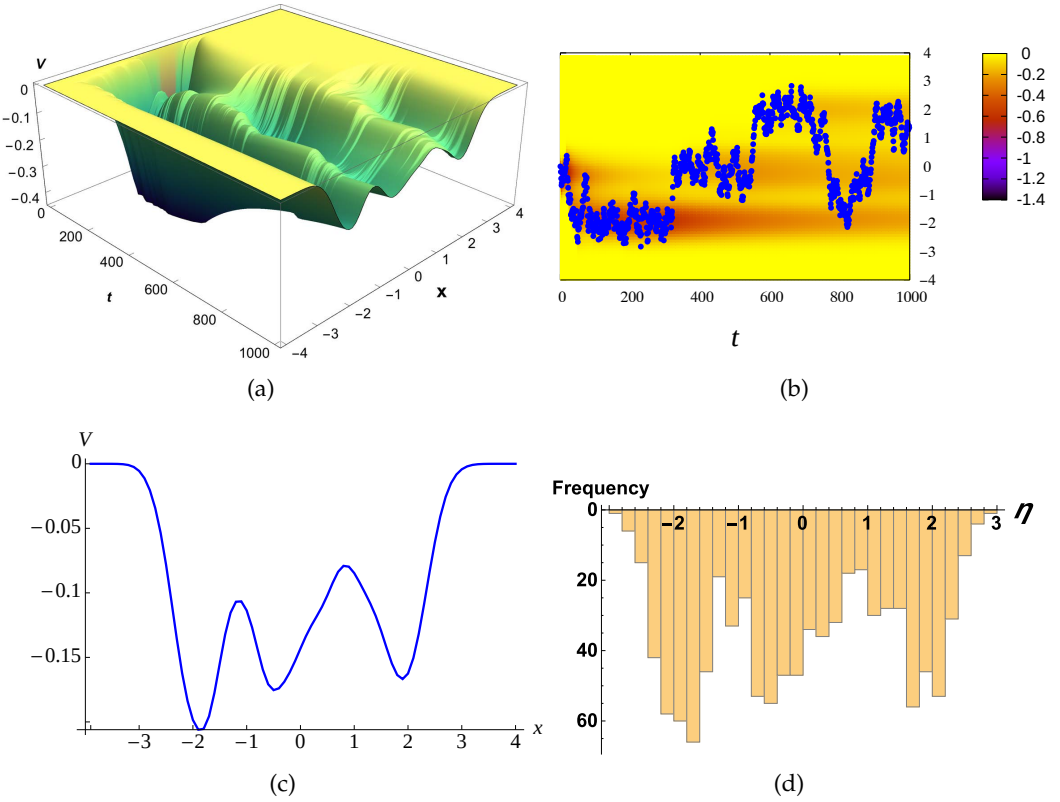


Figure 3.5: Energy landscape of the PDS (with $\sigma^2 = 0.1$) processing input patterns that come from three categories.
(a) Evolution of PDS in response to stimulus given in Fig. 3.4(a).
(b) Evolving $V(x,t)$ (same as in (a)) with the values of V shown by color on the (x,t) plane together with input signal (same as in Fig. 3.4(a)) shown by blue points.
(c) Energy landscape at the end of learning.
(d) Distribution histogram from the realization of the input $\eta(t)$ to the PDS.

3.5(d)), which is calculated from the given input data.

In Fig. 3.6, for a range of values of σ^2 from 0.05 to 0.5 the plastic landscape reproduces the general shape of the PDF (taken with negative sign) of the input signal from a relatively short sequence of input data containing only 1000 points whose distribution histogram from the realization is shown in Fig. 3.5(d). It has three minima at the correct locations compared with Fig. 3.5(d). It was checked for other values of σ^2 that the general shape of V can be reproduced. However, if compare with the

analytically estimated PDF shown in Fig. 3.4(b), the depths and the details of the shape of the individual wells are not reproduced very well, although this is not a problem from the viewpoint of the ability of the system to recognise patterns. In order to illustrate the performance of the plastic landscape with longer stimuli, we run the simulation with a sequence of $20,000$ values of input signal (Fig. 3.6). One can see from Fig. 3.6(d) that the resultant landscape is much closer to the correct analytically estimated PDF shown in Fig. 3.4(b) because it reproduces the shapes and depths of individual wells with much better accuracy.

Another demonstration using musical data is given by [42; 43]. The gradient PDS automatically discovers and memorises musical notes and phrases (collection of notes). Further illustrations of multi-dimensional PDS will be shown in Section 6.4.

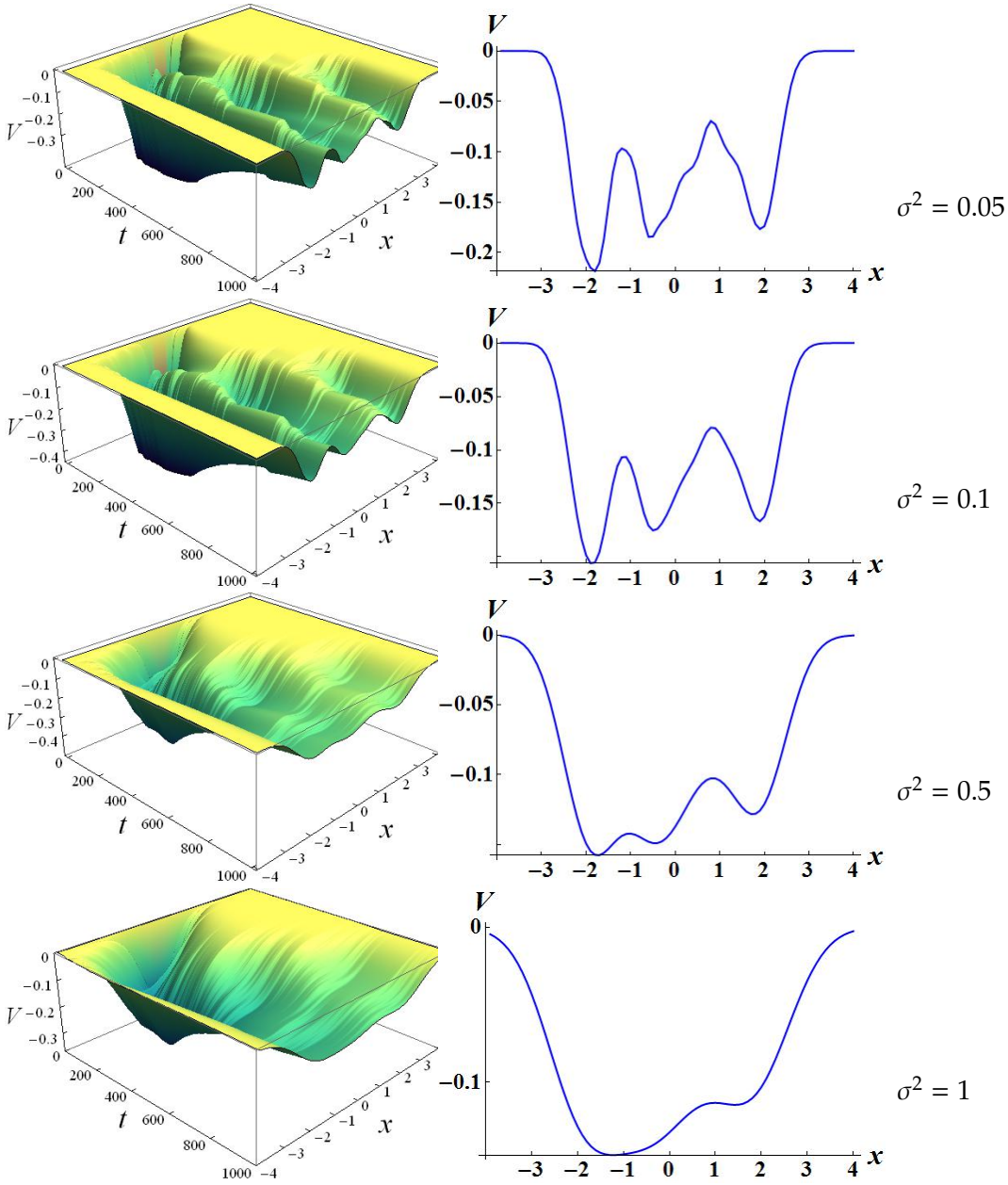


Figure 3.6: Energy landscape of the PDS with different values of σ^2 processing input patterns that come from three categories. Each row corresponds to a single value of parameter σ^2 with the values of σ^2 given in the field of the figure.
(a) Evolution of PDS in response to stimulus given in Fig. 3.4(a).
(b) Energy landscape at the end of learning (compare with Fig. 3.5(d)).

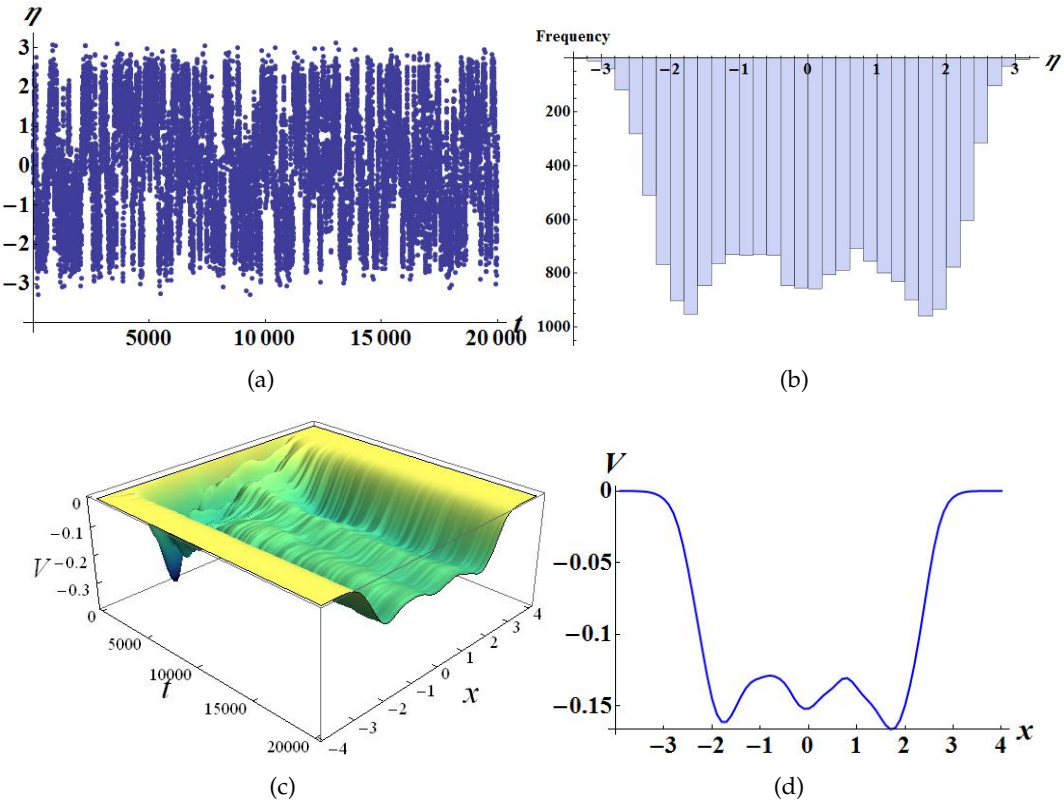


Figure 3.7: Energy landscape of the PDS (with $\sigma^2 = 0.05$) processing input patterns that come from three categories.
(a) Input to PDS: a sequence of 20,000 values randomly taken from three categories.
(b) Distribution histogram from the realization of the input given in (a).
(c) Evolution of PDS in response to stimulus given in (a).
(d) Energy landscape at the end of learning (compare with Fig. 3.4(b)).

3.3 Similarity between the plastic dynamical systems (PDSs) and artificial neural networks (NNs)

PDSs are amending their velocity fields in response to external stimuli. From this viewpoint, both biological and artificial NNs are doing the same, but indirectly through modification of synaptic connections, which evolve under the influence of sensory stimuli. Also, the PDS of the form (3.1), (3.8) can perform an online unsupervised learning. The same task can in principle be solved by NNs if they use an appropriate learning rule, such as Hebbian learning. In addition, in the PDS (3.8) and in CNNs of Hopfield type with symmetric couplings, the ability to perform categorization and pattern recognition can be explained in terms of the energy function that evolves during training.

Biological NNs are known to be able to carry out the full set of cognitive functions modelled by the PDS (3.1)-(3.6) at the very least. It is natural to suggest that the same functions could be implemented in models of NNs, such as artificial NNs of Hopfield type. In Chapter 4 we will compare the performance of NNs with that of PDS (3.1), (3.8) in order to assess their ability to carry out the same combination of tasks. We anticipate that NNs might not be able to mimic the performance of the PDS accurately for the following reason.

The PDS (3.8) is infinitely plastic but it has no prototype in the real world. Artificial NNs, on the other hand, have real-world prototype, and are plastic, but only to some extent, because both the architecture of any neuron, and the architecture of the network are usually fixed and only the connection strengths are allowed to vary. Thus, the velocity field of the PDS describing a NN with plastic connections will only have a limited amount of plasticity. Nevertheless, the NNs seem to have more features in common with PDSs than any other devises known to date.

Chapter 4

Neural approximations of plastic self-organsing vector fields

In the previous Chapter, we introduced the DSs which self-organise their velocity vector field in response to external stimuli. Neural networks, including biological NNs and artificial NNs with continuous time and state, are natural systems that are in the same class. Namely, if every neuron is described as a DS and every inter-neuron coupling can be modelled as some coupling term in the respective equation, the whole NN is also a DS with a high-dimensional phase space. This motivates us to explore the possibility to approximate the PDS by an artificial NN, so that the PDS could eventually be implemented in a device. In this Chapter, we try to approximate the vector fields of PDS by NNs and compare them to see whether a NN can do the same job as the PDS does with acceptable accuracy.

4.1 The model

We consider the NN comprising continuous-valued units, which is discussed in Section 2.2.2.

The PDS (3.1), (3.8) employs completely unsupervised learning and spontaneously creates categories from the external input. We want the NN to behave in the same way and need to choose an appropriate learning method. The most famous unsupervised learning technique is Hebbian learning which we adopt here and study how the velocity vector field of the NN is automatically modified in response to the evolution of inter-neuron connections. We will compare the performance of the NN learning in an unsupervised Hebbian manner with that of the PDS (3.8) and evaluate its potential to reproduce the cognitive functions as required.

The CNN of Hopfield type comprising N neurons undergoing Hebbian learning can be represented by Eqs. (2.10) and (2.11).

4.2 Representation of the state of the PDS by the state of the NN

There is an important problem that needs to be solved in order to make a comparison between the PDS and a NN. Namely, in the PDS (3.8) the phase space is not bounded, i.e. $\boldsymbol{x}$ can vary between $-\infty$ and ∞, so that the system can process the incoming stimuli of any magnitude. However, in any NN the states are principally bounded, and specifically in (2.10) they vary between -1 and 1. So we need to find some way to represent the value of the stimulus to PDS by a combination of states of neurons entering the NN (2.10), i.e. to code the state of the PDS by the state of a NN.

In conventional problems associated with artificial NNs, it is assumed that the stimuli take values from bounded range $[a, b]$, where a and b are some real numbers

and $b > a$. In order to enable the network to process these stimuli, they are simply rescaled, to fall within the range of values in which the states of the NN are allowed to be, e.g. to $[-1, 1]$ for Eq. (2.10). However, rescaling needs an a-priori knowledge of the range $[a, b]$ of the stimuli values, which in the on-line unsupervised learning should not be available because it can only be found after all stimuli values arrive. Also, it does not seem plausible that in a biological NN all stimuli are simply rescaled and shifted based on the fixed values of parameters a and b before reaching a specific neuron. In fact, the question about the way memories are represented in biological NNs remains open to date.

Below we introduce two possible methods, which are both based on the same idea: a single scalar number is represented by a combination of states of a chain of neurons, so that to represent numbers from broader ranges longer chains are needed. Our approach does not require rescaling of sensory signals and is quite general, because one can in principle make a chain of neurons of any length, and also such chains exist in biological NNs.

To convey the idea of the coding technique proposed, assume that the state of, or the value of stimulus to, the one-dimensional PDS (3.1), (3.8) is denoted by x^* and $x^* \in [a,\ b]$. We split the interval $[a,\ b]$ into N segments of equal size numbered from 1 to N and associate each segment with one neuron. The cell in which x^* is located has number h which can be found as $h = [\frac{x^*-a}{\Delta x}] + 1$, where $\Delta x = (b-a)/N$, which is the width of each cell, and $[x^*]$ denotes the largest integer less than or equal to x^*.

Gaussian representation method

This method is illustrated in Fig. 4.1(a). We draw a bell-shaped Gaussian curve whose peak is located at the point $(x^*, 1)$ and the width is determined by some constant σ. The value of the Gaussian curve at each grid point number i is taken to be the state u_i of the corresponding neuron, whose number is i. More specifically, u_i is coded as a

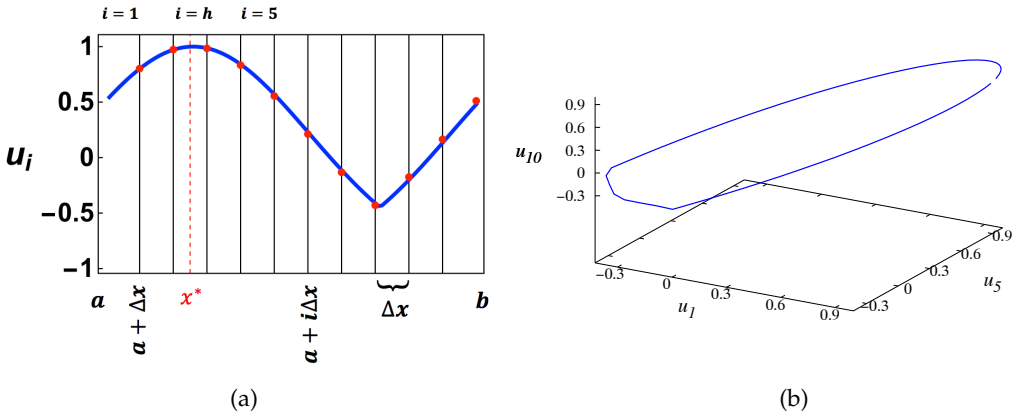
value of a shifted Gaussian function:

$$u_i = -1 + 2\exp\Big(-\frac{(i\Delta x + a - x^*)^2}{\sigma^2}\Big), \tag{4.1}$$

such that $u_i \in [-1,\ 1]$. Thus, a single number x^* is converted into a discretised Gaussian curve with a peak at x^*.

Note, that a Gaussian curve never takes zero values, so to represent the whole discretised curve, a neural chain of an infinite length would be needed. However, at a large distance from its maximum, Gaussian curve is very close to zero, so its values which are considerably different from zero are located close to x^*. Thus, the neural chain representing the value x^* can have finite length. Also, when x^* is very close to the boundary a or b, one tail of the "bell" will tend to -1 such that many neuron states will be very close to -1. To avoid this, we "wrap" the the other tail as a replacement by assigning

$$\begin{cases} u_{h+j} = -1 + 2\exp\Big(-\dfrac{((h+j)\Delta x + a - x^*)^2}{\sigma^2}\Big), \text{ for } j = 1,\ 2, ..., k \text{ and } h + j \le N; \\ u_{h-j} = -1 + 2\exp\Big(-\dfrac{((h-j)\Delta x + a - x^*)^2}{\sigma^2}\Big), \text{ for } j = 1,\ 2, ..., k \text{ and } h - j \ge 1. \end{cases} \tag{4.2}$$



(a) (b)

Figure 4.1: Illustration of Gaussian representation.
(a) Representation method after wrapping for $x^* = -1.6$. Here $N = 11$, $a = -3$, $b = 3$ and $\sigma^2 = 7$.
(b) Three dimensions, u_1, u_5, and u_{10} of the phase space of the NN. The curve shows the location of the units in the phase space after coding.

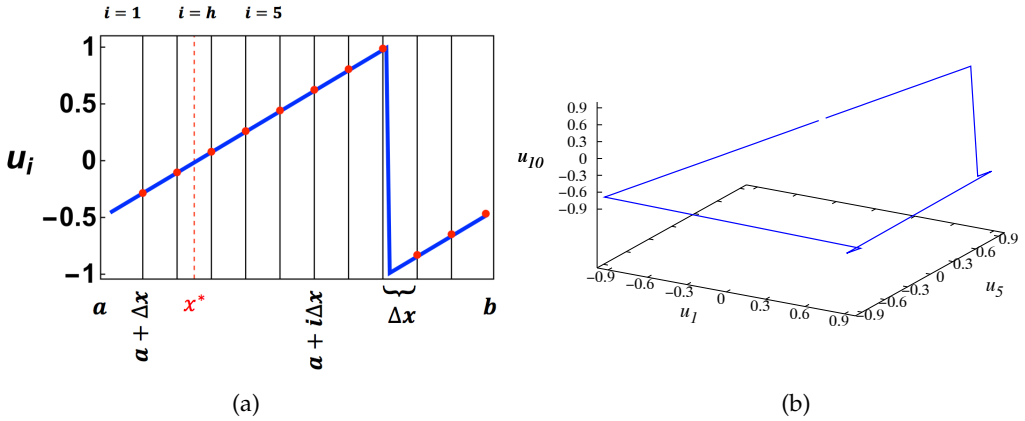
Linear representation method

Another possible coding method is illustrated by Fig 4.2(a). Here we draw a straight line through the point $(x^*, 0)$, such that the slope of the line is $\frac{2}{b-a}$,

$$u(x) = \frac{2}{b-a}(x - x^*). \tag{4.3}$$

To prevent the values of u from leaving the range $[-1, 1]$, we wrap them to fit inside this interval. Then the states u_i of neurons can be founded as follows

$$\begin{cases} u_{h+j} = \dfrac{2}{b-a}((h+j)\Delta x + a - x^*), \text{ for } j = 1,\ 2, ..., k \text{ and } h + j \leq N; \\ u_{h-j} = \dfrac{2}{b-a}((h-j)\Delta x + a - x^*)), \text{ for } j = 1,\ 2, ..., k \text{ and } h - j \geq 1. \end{cases} \tag{4.4}$$



(a) (b)

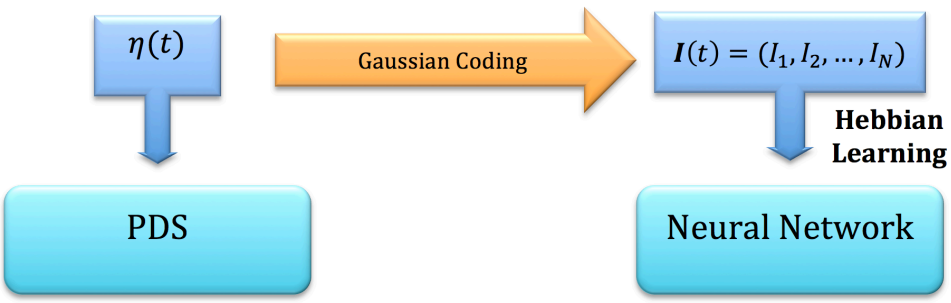
Figure 4.2: Illustration of Linear representation.
(a) Representation method after wrapping for $x^* = -1.6$.
(b) Three dimensions, u_1, u_5 and u_{10} of the phase space of the NN. Location of the units in the phase space after coding are shown by lines of points.

With both Gaussian and linear coding methods, any feasible value x^* is coded by a single point in an N-dimensional phase space of a NN (2.10) that belong to the same closed curve. The projections of the respective curves for both coding methods are shown in Figs. 4.1(b) and 4.2(b).

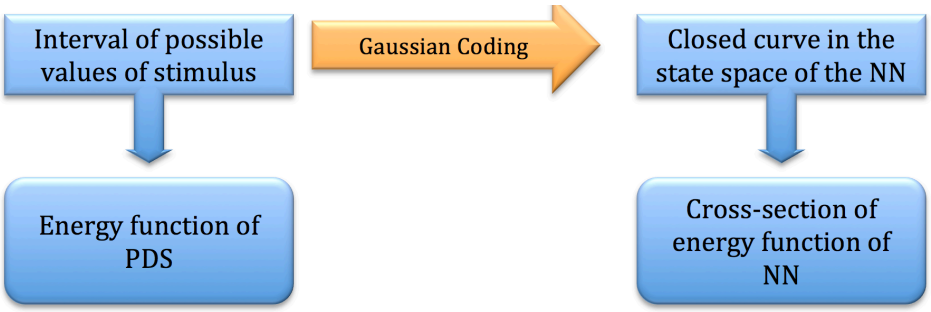
4.3 Visualisation of the energy landscape of the NN

To test the performance of a CNN (2.10), (2.11) against the one of a PDS (3.1), (3.8), we need to compare the evolution of their energy landscapes in response to the same stimulus. At any time moment t, the signal $\eta(t)$ is applied to the one-dimensional PDS as described by (3.8), and the energy landscape V changes. Since at any fixed time t, V is a function of only one variable x, its evolution can be visualised by plotting a surface $V(x,t)$ as shown in Fig. 3.3(a). As to the N-dimensional NN (2.10), its learning will be implemented through the spontaneous adjustment of connection weights w_{ij} in response to the incoming vector stimulus $\boldsymbol{I}(t) = (I_1(t), I_2(t), \cdots, I_N(t))$. Given that the weights are symmetric, i.e. $w_{ij} = w_{ji}$, energy function E can be introduced according to the rule (2.6). However, E is a function of N variables $u_1, u_2, \cdots, u_N$ and cannot be easily visualised for the full set of all possible states of the NN. Here we note that the stimulus $\boldsymbol{I}(t)$ is applied only at the points of the state space that belong to some closed curve illustrated in Fig. 4.1(b) and Fig. 4.2(b).

We appreciate that even if the initial conditions of the NN are set on this curve, the state of the NN (2.10) can in principle deviate from it during evolution. Nevertheless, for visualisation purposes, we pick the values of energy E of the NN only corresponding to this stimulus curve, and thus obtain a one-dimensional function $E(x)$ at any time moment t. Here x is the path length along the stimulus curve from the reference point representing the smallest possible value of $\eta(t)$, i.e. point a. Thus, $E(x)$ is a cross-section of the full energy function of the NN along the stimulus curve. Evolution of $E(x)$ can be visualised in the same manner as evolution of $V(x)$. Importantly, the minima of the cross-section $E(x)$ might not be the minima of the full energy function of the NN, but their location should indicate the approximate location of the minima of $E(x)$. The procedure for comparing the functions of the PDS with those of the CNN is illustrated in Fig. 4.3.



(a)



(b)

Figure 4.3: Schematic diagram of the simulation process.
(a) Coding for neural approximation.
(b) Coding for energy visualisation.

4.4 Results

In the neural approximation of the PDS by Eqs. (2.10), (2.11) used in this Chapter we use the same parameters as in [19] and Fig. 2.3, namely, the standard parameters are taking values as shown in Table 4.1.

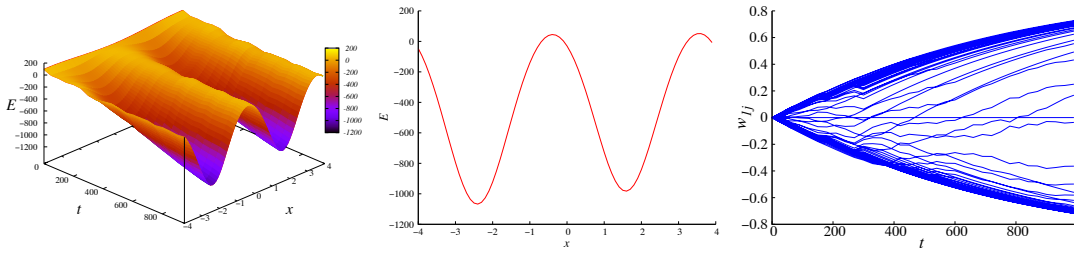
Parameter	Value
N	81
a_i	1
ε	1/300
β	0.3
α	30

Table 4.1: Standard parameter values used in neural approximation.

The inputs shown in Figs. 3.2 and 3.4 are submitted to the NN after being converted into stimulus vectors $\boldsymbol{I}(t)$. The 3-D view of evolution in time of the cross-section of the energy function and of its snapshot at one instant at the end of learning when Gaussian and linear representation methods were used to code the stimulus are shown in Figs. 4.4 and 4.5, respectively. In addition, the evolution in time of the respective connection weights is illustrated in Figs. 4.4(c) and 4.5(c), where the connections to/from neuron 1 are given.

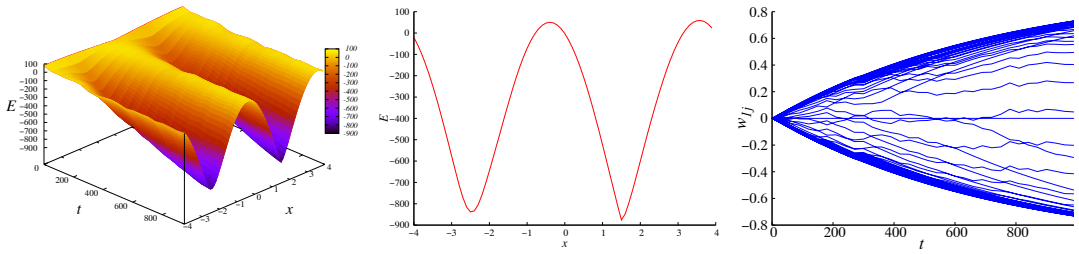
With either coding method, the NN develops the landscape with two dents representing two categories as required. Evolution of the cross-section $E(x)$ of the energy function of the NN can be compared with evolution of the landscape $V(x)$ illustrated in Fig. 3.3, when the PDS was subjected to the same input $\eta(t)$. Although in both the PDS and the NN two wells are spontaneously developed, the shape of the energy is different: in the PDS the wells are of non-equal depth. However, this distinction does not affect the ability of the NN to recognise the pattern correctly, since it is the location of the input inside one of the wells that matters rather than the depth or the shape of the well. However, as seen from Figs. 4.4(c) and 4.5(c), at the end of learning,

the connection weights are small and have not converged to any values. Therefore, it seems that longer time is needed for the connections to settle down and for the NN to memorise the patterns firmly. Given that we only have a finite sequence of data, one way to artificially extend the learning time is to repeat the sequence of input patterns as we did in illustration of Hebbian learning in Section 2.2.4.



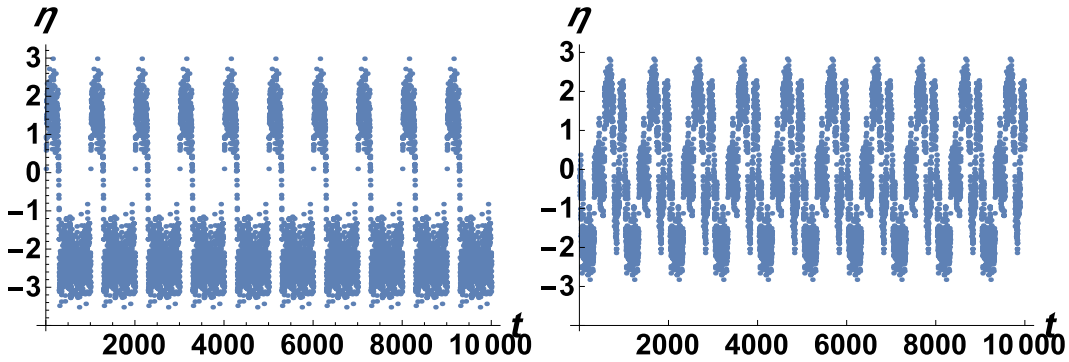
(a) (b) (c)

Figure 4.4: Neural approximation with Gaussian representation using input belonging to two categories, which is illustrated by Fig. 3.2. The parameters take values shown in Table 4.1.
(a) Evolution of the cross-section $E(x)$ of the energy landscape of the NN.
(b) Energy landscape $E(x)$ at the end of learning.
(c) Evolution of connection weights w_{ij}, $j = 1, 2, \cdots, 81$, during learning.



(a) (b) (c)

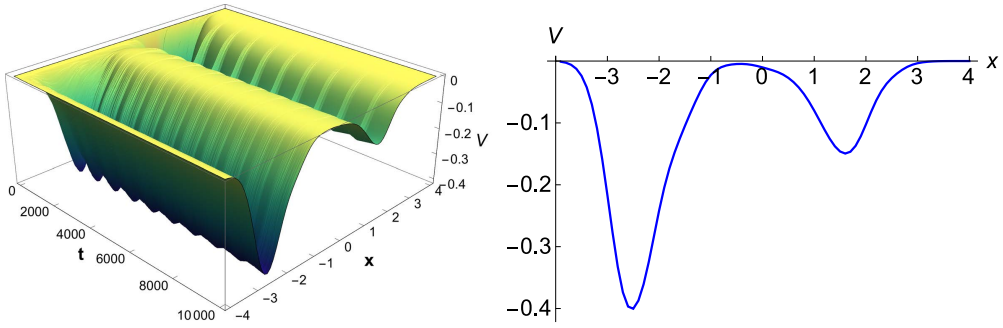
Figure 4.5: Neural approximation with linear representation using input belonging to two categories, which is illustrated by Fig. 3.2. The parameters take values shown in Table 4.1.
(a) Evolution of the cross-section $E(x)$ of the energy landscape of the NN.
(b) Energy landscape $E(x)$ at the end of learning.
(c) Evolution of connection weights w_{ij}, $j = 1, 2, \cdots, 81$, during learning.



(a) (b)

Figure 4.6: Input patterns applied in neural approximation artificially extended by repeating training dataset from Fig. 3.2(a) and Fig. 3.4(a).
(a) Input patterns belonging two categories.
(b) Input patterns belonging three categories.

To generate the stimulus formally having a larger number of input training patterns, the sequences of input shown in Fig. 3.2(a) and Fig. 3.4(a) are repeated several times, resulting in data sets shown in Fig. 4.6, which are then applied to the PDS and to the NN. The PDFs of the new sequences of input do not change as compared to the ones of the original data shown in Fig. 3.2(b) and Fig. 3.4(b).

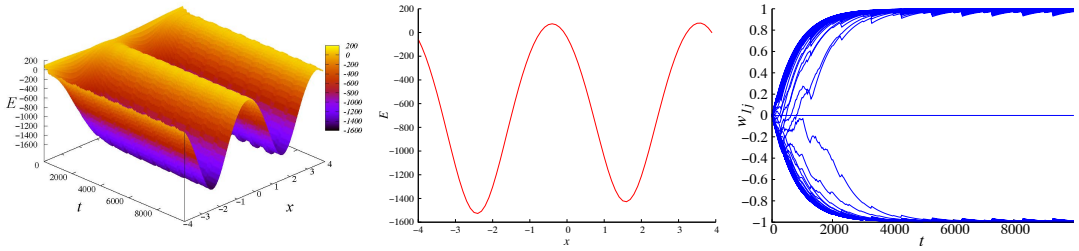


(a) (b)

Figure 4.7: PDS (with $\sigma^2 = 0.1$) processing a repeated input belonging to two categories.
(a) Evolution of PDS $V(x, t)$.
(b) Energy landscape at the end of learning.

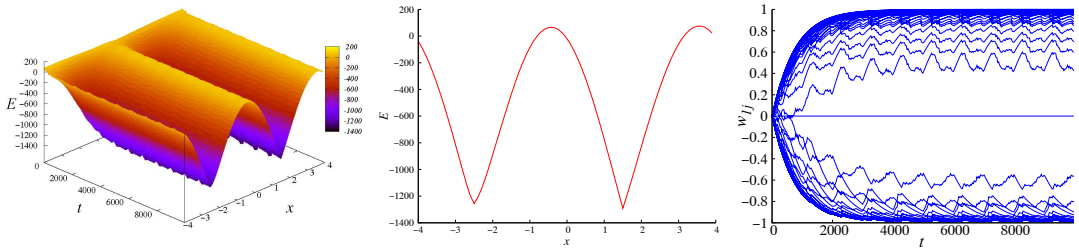
For the NN processing the cyclically repeated input, the evolution and the final shape of the cross-section $E(x)$ of the energy function and the evolution of the connection weights are shown in Figs. 4.8 and 4.9 with Gaussian and Linear codings, respectively. From Figs. 4.8(a) and 4.9(a), one can see that as the NN learns from the input signals, two dents are developed on the energy landscape, that is, the network acquires the knowledge of two classes.

As shown in Figs. 4.8(c) and 4.9(c), the connections between neurons are very weak at the beginning, implying that the internal input term $w_{ij}v_j$ contributes much less to the evolution of the network than the external input values I_i. When the connection weights become much larger, the small input values I_i can be ignored. The connection weights do not converge to any fixed value but continue to fluctuate, and the period of their fluctuations coincides with that of the duration of the original input dataset before it was repeated. This can be explained as follows: while processing new information a portion of the connection weights need to change, leading to reshaping of the energy landscape of the NN, i.e. to alterations of the depths and widths of the wells, and even possibly of their number. When the same sequence of input data is applied to the NN repeatedly, the weights undergo modifications of a similar kind again and again,



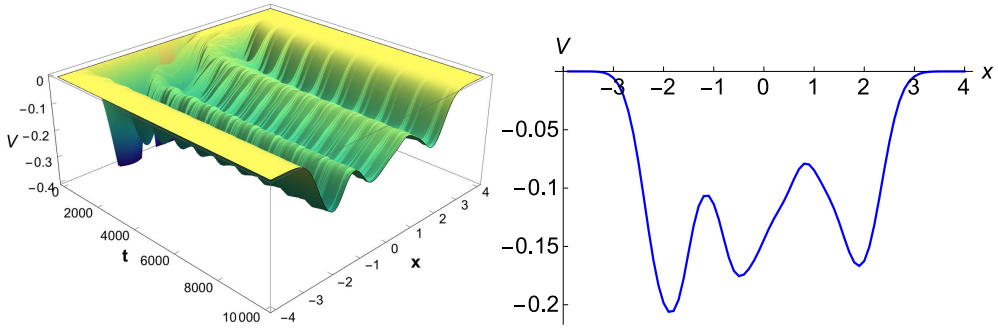
(a) (b) (c)

Figure 4.8: Neural approximation with Gaussian representation using cyclically repeated input belonging to two categories, which is illustrated in Fig. 4.6(a). The parameters take values shown in Table 4.1.
(a) Evolution of the cross-section $E(x)$ of the energy landscape of the NN.
(b) Energy landscape $E(x)$ at the end of learning.
(c) Evolution of connection weights $w_{ij},\ j = 1, 2, \cdots, 81$, during learning.



(a) (b) (c)

Figure 4.9: Neural approximation with linear representation using cyclically repeated input belonging to two categories, which is illustrated in Fig. 4.6(a). The parameters take values shown in Table 4.1.
(a) Evolution of the cross-section $E(x)$ of the energy landscape of the NN.
(b) Energy landscape $E(x)$ at the end of learning.
(c) Evolution of connection weights $w_{ij},\ j = 1, 2, \cdots, 81$, during learning.



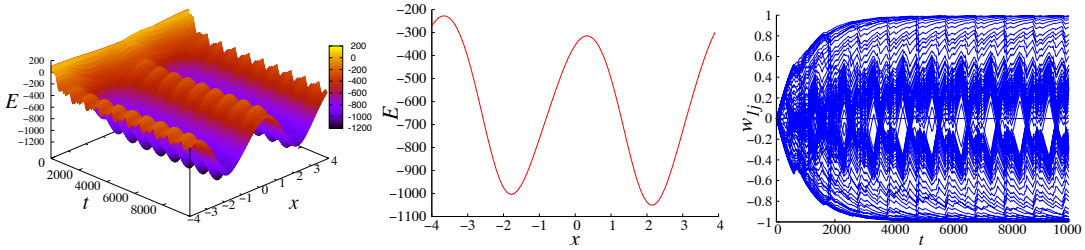
(a) (b)

Figure 4.10: PDS (with $\sigma^2 = 0.1$) processing a repeated input belonging to three categories.
(a) Evolution of PDS $V(x,t)$. (b) Energy landscape at the end of learning.

resulting in repeated changes in the landscape shape.

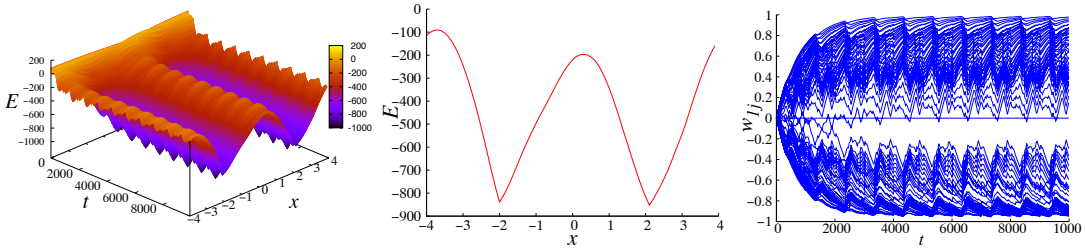
Figures 4.11 and 4.12 illustrates the neural approximation of the PDS affected by stimulus belonging to three categories shown in Fig. 4.6(b). This stimulus is obtained by a cyclic repetition of the dataset shown in Fig. 3.4(a). The PDS subjected to this stimulus behaves in a manner similar to the one when only one cycle of the input signal was applied, as can be appreciated by comparing Fig. 3.5 and 4.10. The only difference resulting from the respetitive nature of the input consists in the slight fluctuations of the energy $V(x)$, but its shape was well formed already after the first portion of data. However, the energy $E(x)$ of the NN appears unable to form the required three wells even after many repetitions of the same input, as evidenced by Figs. 4.11 and 4.12: at any time instant the landscape has only two wells.

Figures 4.13 and 4.14 illustrate the neural approximation with either Gaussian or linear coding method using stimulus belonging to three categories shown in Fig. 4.6(b), but with different values of parameter α. When α is relevant small, such as $\alpha = 3$, the connection weights converge to -1, 0 or 1. With larger values of α the external input becomes much more pronounced. However, the neural approximation only generates two wells for $\alpha \gg \beta$.



(a) (b) (c)

Figure 4.11: Neural approximation with Gaussian representation using cyclically repeated input belonging to three categories, which is illustrated in Fig. 4.6(b). The parameters take values shown in Table 4.1.
(a) Evolution of the cross-section $E(x)$ of the energy landscape of the NN.
(b) Energy landscape $E(x)$ at the end of learning.
(c) Evolution of connection weights w_{ij}, $j = 1, 2, \cdots, 81$, during learning.



(a) (b) (c)

Figure 4.12: Neural approximation with linear representation using cyclically repeated input belonging to three categories, which is illustrated in Fig. 4.6(b). The parameters take values shown in Table 4.1.
(a) Evolution of the cross-section $E(x)$ of the energy landscape of the NN.
(b) Energy landscape $E(x)$ at the end of learning.
(c) Evolution of connection weights w_{ij}, $j = 1, 2, \cdots, 81$, during learning.

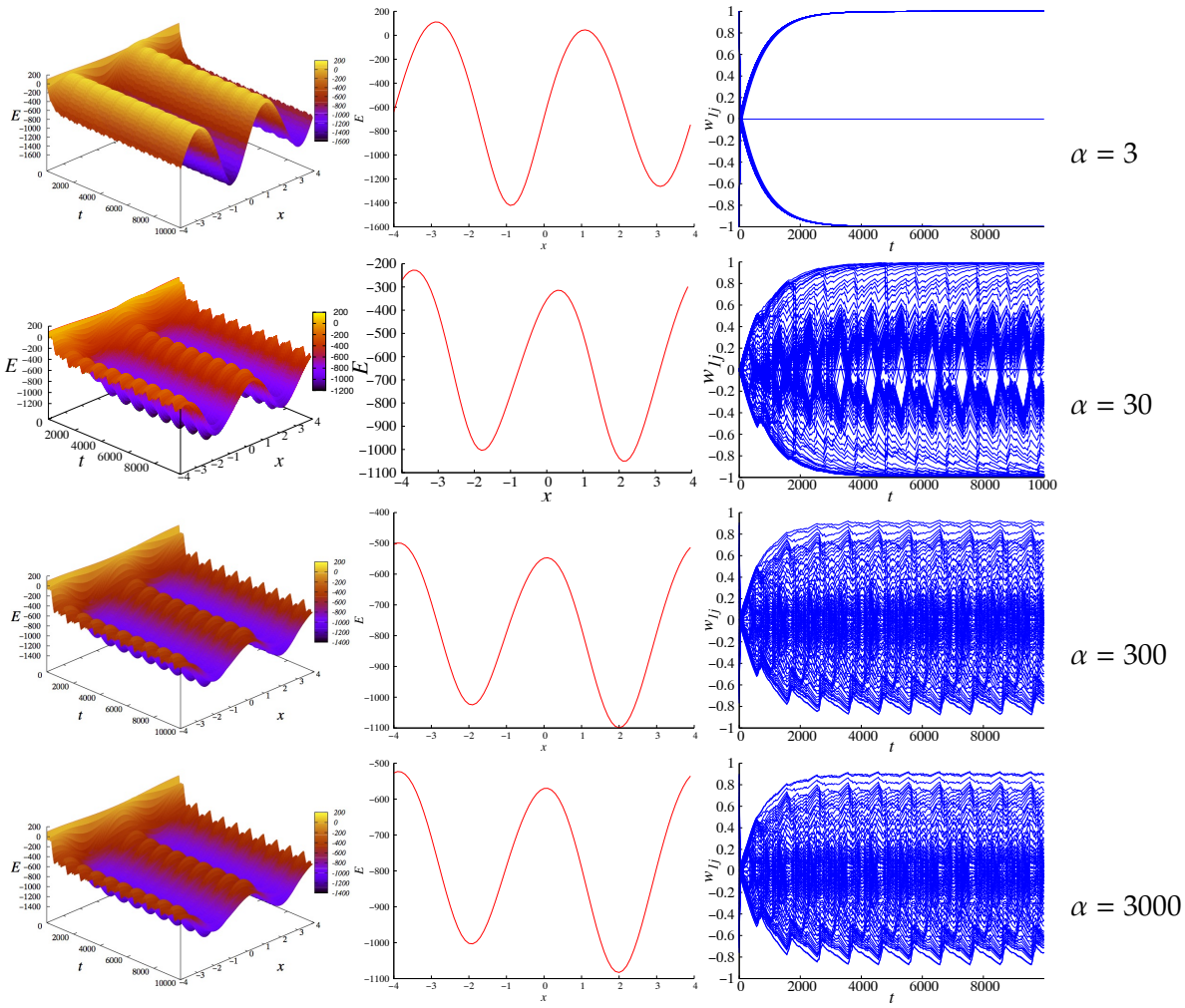


Figure 4.13: Neural approximation with Gaussian representation taking different values of parameter α using cyclically repeated input belonging to three categories, which is illustrated in Fig. 4.6(b). Each row corresponds to a single value of parameter α with the values of α given in the field of the figure. The other parameters take values shown in Table 4.1.
(a) Evolution of the cross-section $E(x)$ of the energy landscape of the NN.
(b) Energy landscape $E(x)$ at the end of learning.
(c) Evolution of connection weights w_{ij}, $j = 1, 2, \cdots, 81$, during learning.

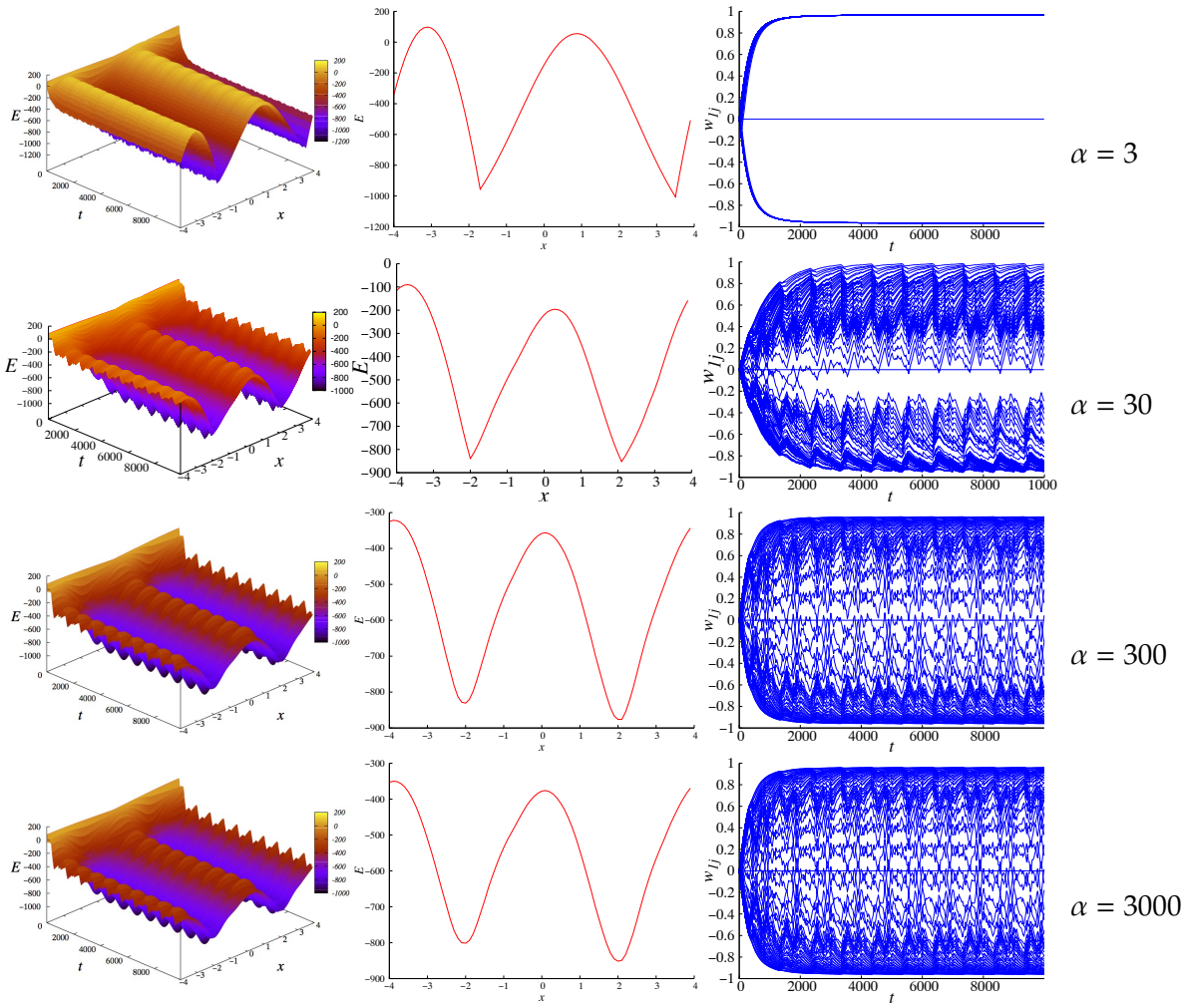


Figure 4.14: Neural approximation with linear representation taking different values of parameter α using cyclically repeated input belonging to three categories, which is illustrated in Fig. 4.6(b). Each row corresponds to a single value of parameter α with the values of α given in the field of the figure. The other parameters take values shown in Table 4.1.
(a) Evolution of the cross-section $E(x)$ of the energy landscape of the NN.
(b) Energy landscape $E(x)$ at the end of learning.
(c) Evolution of connection weights w_{ij}, $j = 1, 2, \cdots, 81$, during learning.

4.5 Summary and conclusions

We explored the possibility for the CNN to imitate the performance of a PDS while performing unsupervised online learning. The NN of Hopfield type with symmetric couplings was chosen because one could introduce energy function for such a system, which would hopefully approximate the energy function of the PDS of a gradient type described by Eqs. (3.1), (3.8). In order to compare the two systems, it was necessary to solve a problem of coding the state of the PDS in an unbounded phase space by the state of the NN, whose phase space is fundamentally bounded. We introduced and tested two coding methods, and found that with both methods the performance of the NN was largely similar.

We also needed to find a way to compare energy function in the simplest PDS depending on a single state variable with that in the NN depending on $N > 1$ state variables. We proposed to consider the energy of the NN only along the one-dimensional curve in its N-dimensional phase space being an image of a set of all possible stimuli to the system. Such a one-dimensional cross-section of the full energy function of the NN does not describe the behaviour of the NN completely, and its minima do not necessarily coincide with those of the full energy. However, the dips of this function approximately indicate where the minima of the full energy are located and give a rough idea of the expected evolution of the NN. Introduction of the cross-section of the energy of the NN allowed us to make comparisons as necessary. We discovered that while different coding methods resulted in different behaviours of connection weights during learning, the shape of the cross-section of energy of the NN was qualitatively similar.

It was found out that the NN with symmetric couplings does not generally reproduce the behaviour of the PDS during unsupervised learning in cases when the number of categories of input values is larger than two. This can be explained by

the limited memory capacity of a NN discussed in Section 2.2.5. Because the largest number of memories/categories that can be held by a network is limited, to accommodate a new category, one of the old ones should disappear. Namely, while processing new stimuli, the NN adjusts its connection weights, thus altering the energy landscape and modifying the locations, the depths, the widths and the number of its minima, i.e. the knowledge of the network about the categories of the sensory data. Since the structure of every neuron and of every connection it makes is rigidly fixed, and the only flexibility in the NN arises from the variable connection strengths, the velocity field of the NN (i.e. the derivative of the energy function in the given example) is quite inflexible and can have room for not more than a certain number of attractors (here minima of energy function). This feature distinguishes the NN from a PDS, in which the number of minima in the landscape, i.e. memories/categories, can be unlimited.

Moreover, the network is sensitive to the learning parameter in Eq. 2.10. As shown in [19], with smaller value of α, all connections ultimately go to a single magnitude and reflect that the network selects some of the input patterns to memorise and will be unable to learn any other patterns or to notice the inputs. Therefore, the observed smaller efficiency of the NN is not surprising in itself.

Chapter 5

Implementing plastic velocity field through moment-based density approximation: one-dimensional case

In Chapter 4 we have shown that a continuous-state NN cannot readily reproduce the learning abilities of a PDS and suggested that the main reason for that is the lack of plasticity of the velocity field of such a network. In this Chapter we explore an alternative idea to implement a PDS in hardware, in particular in electronic circuits based on standard elements. As demonstrated in [51], in the simplest PDS (3.1), (3.8) the energy function V tends to take the shape of the PDF of the random process generating the incoming sensory stimulus. Moreover, at every time moment the instantaneous shape of the energy V is in fact the estimation of the PDF based on the input signals processed so far. Thus, our idea is to replace the energy V in the equation describing the gradient PDS by the estimation of the PDF of the random process affecting the

learning system.

We note that one can easily construct an electronic circuit carrying out multiplications and summations of input voltages by means of analogue multipliers and adders [76], and thus implement any polynomial function. With this, it is well known that a PDF of a random variable or a process can be represented as a polynomial expansion with coefficients formed by combinations of its moments [18]. Thus, it seems possible to establish an explicit connection between the PDS of a gradient type and an electronic circuit multiplying and adding voltages. Since the expansion of a PDF generally requires an infinite number of terms, while the number of circuit components can only be finite, only an approximation of a PDF could be achieved with this approach. However, similarly to NNs, the ability of a PDS to recognize a pattern is not very sensitive to the details of the shape of its energy landscape, so an approximation might be sufficient.

An important question here is how to find the coefficients of polynomials approximating the PDF. Since the energy V of the PDS is continually evolving while processing stimuli, the PDF estimation replacing V should be evolving correspondingly, implying evolution of the coefficients of its polynomial expansion. We need to determine the laws of their evolution correctly so that the circuit could reproduce not only the final shape of the energy V, but also all stages of its development.

5.1 Evolving landscape from a one-dimensional moment-based density approximation

The density function can be represented by expansions in an infinite series involving the moments of the variate, called moment-based density approximation (MBDA). Namely, the PDF $f(x)$ of a continuous random process $X(t)$ can be represented as [18]

$$f(x) = \sum_{k=0}^{\infty} C_k \psi_k(x), \tag{5.1}$$

where $\{\psi_k,\ k \geq 0\}$ is a given family of basis functions, which are independent of f, such as Legendre, Laguerre, Hermite or Jacobi polynomials. Here we assume that $X(t)$ is stationary, i.e. its PDF $f(x)$ does not depend on time t. The coefficients in the expansion, denoted by C_k, could be linear combinations of finite numbers of moments. The jth moment of X, denoted by μ_j is defined as

$$\mu_j = E(x^j) = \int_{-\infty}^{\infty} x^j f(x) \mathrm{d}x. \tag{5.2}$$

All moments are determined by the same $f(x)$, which in our problem is unknown. The approximation (5.1) is valid only for the PDFs whose moments are finite [18]. Also, the moments determine the PDF uniquely only for such continuous random variables (or processes), whose support is a closed interval. We can safely assume both the above properties in the random processes being considered here, since they are fairly generic.

Given the first n moments of $X(t)$, $\mu_1,\ \mu_2,\ \ldots\ \mu_n$, and setting $\mu_0 = 1$, $f(x)$ can be approximated by a truncated series as

$$f(x) \approx f_n(x) = \sum_{k=0}^{n} C_k \psi_k(x). \tag{5.3}$$

Note, that the PDF approximation with a finite number of terms of the series may have

negative values particularly near the tails, which violates one of the main properties of a PDF, namely its positivity. Also, the integral of $f_n(x)$ might not be equal to 1, which violates the normalization condition of a PDF. For these reasons such finite series are not very useful in problems that require the knowledge of PDF for estimation of various statistical characteristics. However, in our problem we need to use the PDF estimation only as a substitute of an energy function, which can be either positive, or negative, and does not have to obey normalization condition.

Both the expansion of the PDF (5.1) and its approximation (5.3) are valid under the assumption that the moment μ_j of the given random process are known exactly, i.e. are estimated based on the full (usually infinitely large) ensemble of its realizations known during infinitely long observation times. However, in our problem all we know about the random process is a sequence of values at the input to the learning system, i.e. only a single realization, instead of the full ensemble of realizations. Moreover, at every time moment t we know only a segment of this realization on the interval $[0, t]$. Certainly, as time goes by, the amount of information about the realization grows, but we need to update the shape of the energy landscape based on the information we have by the given moment.

Firstly, the lack of information associated with the unavailability of the ensemble of realizations of the random process can be overcome if the random process $X(t)$ is not only stationary, but also ergodic. In that case, all of its moments can be obtained from any single realization $x^*(t)$ by averaging over time rather than over the ensemble [69], namely

$$\mu_j = \int_{-\infty}^{\infty} x^j f(x) \mathrm{d}x = \lim_{T\to\infty} \frac{1}{T} \int_0^T (x^*(t))^j \mathrm{d}t. \tag{5.4}$$

As a consequence, in ergodic random processes the PDF can also be obtained from a single realization. Although ergodicity cannot be proved if nothing is known about

the random process beyond its single realization, it is a common assumption used when processing experimental data.

Secondly, we need to determine how to update representation of the energy landscape V by means of the PDF expansion (5.3) while gradually incorporating the newly arriving values of the input. The most natural step would be to approximate the moments μ_j by their estimates during finite time intervals, i.e. to abandon the limit $T \to \infty$ in Eq. (5.4). This way, at each time instant t the moments μ_j could be replaced by their estimates

$$\mu_j(t) \approx \frac{1}{t} \int_0^t (x^*(s))^j \mathrm{d}s. \tag{5.5}$$

Thus, in Eq. (5.3) the coefficients C_k will become linear combinations of μ_j, and the function $f_n(x)$ an updated approximation of the PDF (and thus of $V(x)$) corresponding to time t.

In Sections 5.2-5.4 we investigate how the MBDA allows one to approximate the energy of the PDS at the final stage of learning with various examples of input signals. We assess the performance of two different basis functions, Hermite and Legendre polynomials, used in expansion (5.3).

Note, that although technically Eq. (5.5) should give a satisfactory approximation of the moments μ_j at each time t, it is not convenient for applications for the following reason. The use of Eq. (5.5) assumes that at each time instant t one should integrate all values of the input available by this time, with their number steadily growing with t. This implies that not only all values of input $x^*(t)$ need to be stored in the memory of the circuit, but also that the time required to find the moments is steadily growing as t increases. These two requirements are certainly impractical and need to be avoided when designing an electronic circuit. We would like to determine the rules for updating the values of μ_j such that the storage of input data is not required and the time for obtaining the next value of $\mu_j(t)$ is the same for all time instants t. With this,

in Section 5.5 we derive the rules according to which μ_j can be updated in agreement with specifications above.

In Section 5.6 we analytically estimate the error of approximation of the energy $V(x)$ by Eq. (5.3) and demonstrate the accuracy of this estimation by comparing with errors obtained numerically. In Section 5.7 we summarise the findings of this Chapter.

5.2 Density approximation based on Legendre polynomials

In what follows, to convey the idea of the approach, we will be referring to a random variable, bearing in mind that the same ideas apply to random processes which are also random variables, albeit varying in time.

5.2.1 Approximation with Legendre polynomials

Assume that X is a continuous random variable defined on the interval $[-1,\ 1]$. The density function $f(x)$ of X can be approximated by the series [18]:

$$f_L(x) = \sum_{k=0}^{n} C_k^L L_k(x), \tag{5.6}$$

where $L_k(x)$ is a Legendre polynomial of degree k in x, which is expressed as

$$\begin{aligned} L_k(x) &= \frac{1}{2^k k!}\frac{\partial^k}{\partial x^k}(x^2-1)^k \\ &= \sum_{i=0}^{[k/2]}(-1)^i 2^{-k}\binom{k}{i}\binom{2k-2i}{k}x^{k-2i} \\ &= \sum_{i=0}^{[k/2]}(-1)^i 2^{-k}\frac{(2k-2i)!}{i!(k-i)!(k-2i)!}x^{k-2i} \end{aligned} \tag{5.7}$$

where $[k/2]$ denotes the largest integer less than or equal to $k/2$. The middle expression is known as Rodrigues' formula [18]. Figure 5.1 shows the polynomials up to order 5. C_k^L are parameters involving moments of X, and their expression is derived from the orthogonality of the Legendre polynomials, which is formulated as

$$\int_{-1}^{1} L_i(x)L_j(x)\mathrm{d}x = \begin{cases} 0 & \text{for } i \neq j, \\ \frac{2}{2i+1} & \text{for } i = j. \end{cases} \tag{5.8}$$

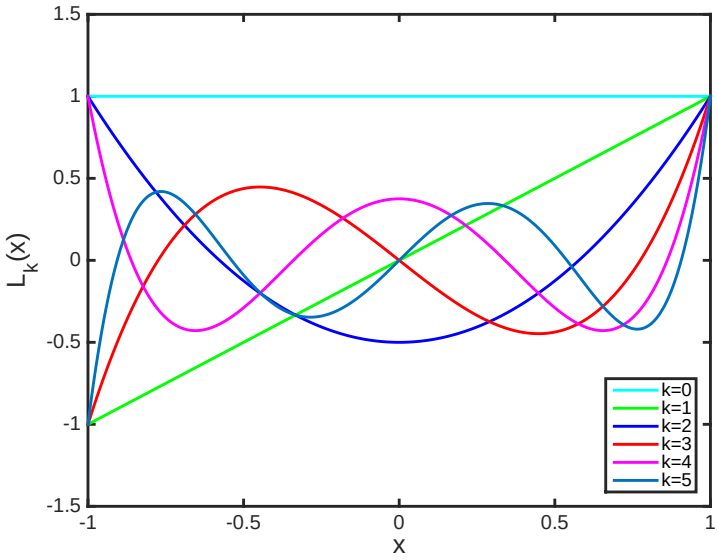


Figure 5.1: Legendre polynomials up to 5th order.

Multiplying both sides of Eq. (5.6) by $L_r(x)$ and integrating from -1 and 1, we obtain for $r = k$,

$$\int_{-1}^{1} f_L(x)L_r(x)\mathrm{d}x = \sum_{k=0}^{n} C_k^L \int_{-1}^{1} L_k(x)L_r(x)\mathrm{d}x = \frac{2}{2k+1}C_k^L,$$

so that the coefficients C_k^L can be expressed as

$$C_k^L = \frac{2k+1}{2}\int_{-1}^{1} f_L(x)L_k(x)\mathrm{d}x. \tag{5.9}$$

Substituting the third of Eqs. (5.7) into Eq. (5.9) we obtain

$$\begin{aligned} C_k^L =& \frac{2k+1}{2}\sum_{i=0}^{[k/2]}(-1)^i 2^{-k}\frac{(2k-2i)!}{i!(k-i)!(k-2i)!}\int_{-1}^{1} f_L(x)x^{k-2i}\mathrm{d}x \\ =& \frac{2k+1}{2}\sum_{i=0}^{[k/2]}(-1)^i 2^{-k}\frac{(2k-2i)!}{i!(k-i)!(k-2i)!}\mu_{k-2i}, \end{aligned}$$

where μ_{k-2i} is the $(k-2i)$th moment of X.

The polynomial expansion (5.6) is proved to be the least squares' approximation

of $f(x)$ by the polynomial of degree n that minimises the integrated squared error $\int_{-1}^{1}(f(x) - f_L(x))^2 \mathrm{d}x$ [11].

5.2.2 Generalisation of Legendre polynomial approximation

Expansion (5.6) of $f(x)$ using Legendre polynomials can be generalized to approximate the density of a continuous random variable X defined on the interval $[a,\ b]$. Applying a linear transformation

$$Y = \frac{2X - (a+b)}{b-a},$$

which maps $X \in [a,\ b]$ onto $Y \in [-1,\ 1]$, the density function of X can be approximated by an expansion as [61]:

$$f_L(x) = \frac{2}{b-a}\sum_{k=0}^{n} C_k^L L_k\Big(\frac{2x-(a+b)}{b-a}\Big). \tag{5.10}$$

Here $L_k(x)$ is defined by Eq. (5.7) and the coefficients are expressed as

$$C_k^L = \frac{2k+1}{2}\sum_{i=0}^{[k/2]}(-1)^i 2^{-k}\frac{(2k-2i)!}{i!(k-i)!(k-2i)!}\mu_{k-2i}^Y, \tag{5.11}$$

where μ_k^Y defined as

$$\mu_k^Y = \frac{1}{(b-a)^k}\sum_{j=0}^{k} 2^j(-a-b)^{k-j}\frac{k!}{j!(k-j)!}\mu_j \tag{5.12}$$

is the kth moment of Y, and μ_j is the jth moment of X. The expression (5.12) is obtained from the expected value of the binomial expansion of $\left(\frac{2X-(a+b)}{b-a}\right)^k$ [18].

5.3 Density approximation based on Hermite polynomials (Gram-Charlier series)

Now we consider another density approximation based on Hermite polynomials. Suppose X is a continuous random variable defined on the interval $(-\infty, \infty)$, and the density function $f(x)$ has tails congruent to that of a normal density function. Then $f(x)$ can be approximated by a polynomial given by [48]

$$f_H(x) = \sum_{k=0}^{n} C_k^H \frac{\mathrm{d}^k g^*(x)}{\mathrm{d}x^k}, \tag{5.13}$$

where $g^*(x)$ is the standard normal distribution

$$g^*(x) = \frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{1}{2}x^2\Big), \tag{5.14}$$

obtained by substituting $\mathrm{E}(X) = m_x = 0$, $\mathrm{Var}(X) = \sigma_x^2 = 1$ in the normal distribution

$$\hat{g}(x) = \frac{1}{\sqrt{2\pi\sigma_x^2}} \exp\Big(-\frac{1}{2}\Big(\frac{x - m_x}{\sigma_x}\Big)^2\Big). \tag{5.15}$$

The Hermite polynomial $H_k(x)$ of the order k is defined by the identity

$$(-1)^k \frac{\mathrm{d}^k g^*(x)}{\mathrm{d}x^k} = H_k(x) g^*(x). \tag{5.16}$$

From Eq. (5.16) one can express $H_k(x)$ as

$$\begin{aligned} H_k(x) &= (-1)^k \frac{1}{g^*(x)} \frac{\mathrm{d}^k g^*(x)}{\mathrm{d}x^k} \\ &= (-1)^k \exp\Big(\frac{1}{2}x^2\Big) \frac{\mathrm{d}^k}{\mathrm{d}x^k} \exp\Big(-\frac{1}{2}x^2\Big). \end{aligned} \tag{5.17}$$

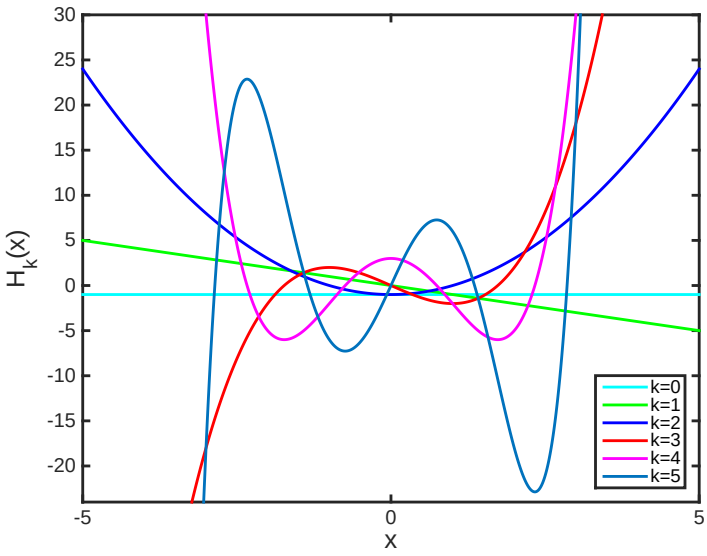


Figure 5.2: Hermite polynomials up to 5th order.

Since

$$\frac{\mathrm{d}^k}{\mathrm{d}x^k}\exp\Big(-\frac{1}{2}x^2\Big)=\sum_{i=0}^{[k/2]}(-1)^{i-k}\frac{k!}{i!(k-2i)!2^i}x^{k-2i}\exp\Big(-\frac{1}{2}x^2\Big),$$

we have

$$H_k(x)=\sum_{i=0}^{[k/2]}(-1)^i\frac{k!}{i!(k-2i)!2^i}x^{k-2i}, \tag{5.18}$$

where $[k/2]$ denotes the largest integer less than or equal to $k/2$. The first ten polynomials are listed in Appendix A and the plots of the polynomials up to 5th order are shown in Fig. (5.2). Therefore, the expansion (5.13) becomes

$$f_H(x)=\sum_{k=0}^{n}D_k^H H_k(x)g^*(x), \tag{5.19}$$

where $D_k^H=(-1)^kC_k^H$ are the coefficients.

The coefficients D_k^H can be obtained from the orthogonality property of the Hermite polynomials. Namely, these polynomials are orthogonal with respect to the weight

function $g^*(x)$,

$$\int_{\infty}^{\infty} H_i(x)H_j(x)g^*(x)\mathrm{d}x = \begin{cases} 0 & \text{for } i \neq j, \\ i! & \text{for } i = j. \end{cases} \tag{5.20}$$

The proof of the orthogonality is shown in Appendix A.

Multiplying both sides of Eq. (5.19) by $H_r(x)$ and integrating from $-\infty$ to ∞, we obtain for $r = k$

$$\int_{-\infty}^{\infty} f_H(x)H_r(x)\mathrm{d}x = \sum_{k=0}^{n} D_k^H \int_{-\infty}^{\infty} H_k(x)H_r(x)g^*(x)\mathrm{d}x = k!D_k^H,$$

so that the coefficient D_k^H can be expressed as

$$D_k^H = \frac{1}{k!}\int_{-\infty}^{\infty} f_H(x)H_k(x)\mathrm{d}x. \tag{5.21}$$

Substituting Eq. (5.18) into the above equation we obtain the expression for D_k^H

$$\begin{aligned} D_k^H =& \frac{1}{k!}\sum_{i=0}^{[k/2]}(-1)^i \frac{k!}{i!(k-2i)!2^i}\int_{-\infty}^{\infty} f_H(x)x^{k-2i}\mathrm{d}x \\ =& \sum_{i=0}^{[k/2]}(-1)^i \frac{1}{i!(k-2i)!2^i}\mu_{k-2i}, \end{aligned}$$

where μ_{k-2i} is the $(k-2i)$th moment of X.

This polynomial expansion is also known as Gram-Charlier series of type A. This type of expansions is often truncated after terms of forth order since the third order term makes allowance of skewness and kurtosis while the forth order terms enable the MBDA to approximate the density function which has more than one peak [44].

Another series, which is formally identical with Gram-Charlier series, was derived by Edgeworth [20] through a different approach. The Edgeworth series involves derivatives of standard Gaussian function $g^*(x)$ and the coefficients based on cumulants. The cumulants can be expressed through moments as shown in [48]. Therefore, it is straightforward to transform either of the two series into another.

5.4 Moment-based approximations of the landscape at the end of learning

In this Section we compare the final stage of evolution of the energy landscape of the PDS according to Eq. (3.8) at the end of learning, with the polynomial approximation of the PDF of random processes generating the training signal $\eta(t)$. The idea to approximate an unknown function by a sum of standard functions with unknown parameters is due to Galerkin [12]. However, in this Section we are testing the possibility to approximate the final shape of the landscape function by using the unknown parameters in a very specific form as combinations of estimates of moments of the input signals with the subsequent goal to implement this idea in electronic circuits. Both the landscape and the polynomial expansion of the density develop into their final shapes under the influence of the same signal $\eta(t)$. For the MBDA the moments μ_k at time t are estimated as

$$\mu_k \approx \frac{1}{t}\int_0^t (\eta(s))^k \mathrm{d}s.$$

We compare the performance of two polynomial expansions, Legendre and Hermite, of various orders.

5.4.1 Input data from two categories

Here we illustrate the formation of the MBDA in response to input data coming from two categories, i.e. when the PDF of the generating random process has two peaks. The input signal $\eta(t)$ is shown in Fig. 3.2(a). In Fig. 5.3 the energy landscape of the PDS at the end of learning is shown by blue line against the negative of the MBDA with Legendre polynomials of various orders shown by orange line. The difference between the two functions in (a) is given in (b) and is called the "Error".

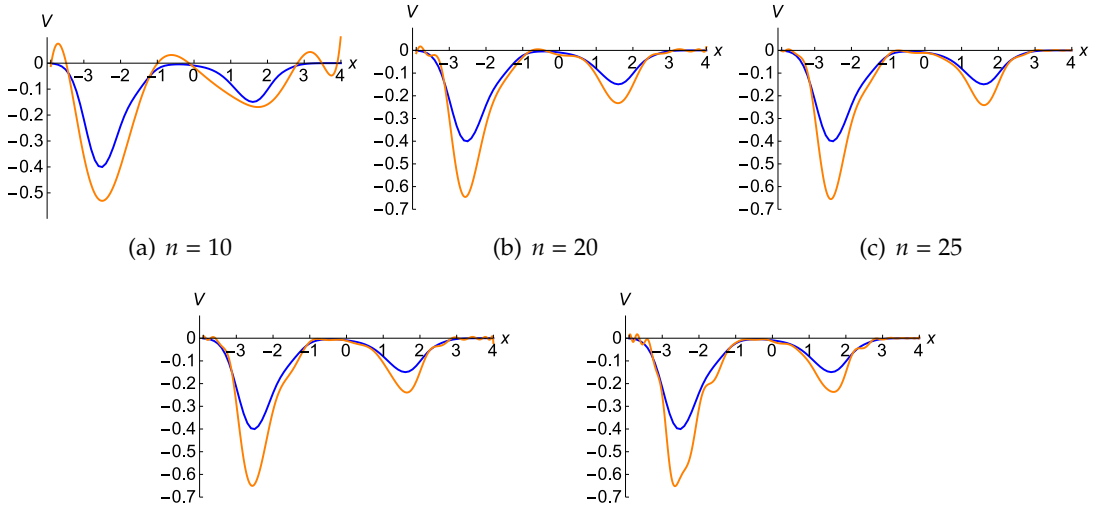
When we compare the PDS with its MBDA, we focus on not numerical quantitative coincidence between the approximation and the PDS landscape but the location of the

minima and the boundaries of the respective basin of attractions. As long as the approximation reproduce the width of the wells and their location of the minima, then we will regard the approximation as successful.

From Fig. 5.3 one can see that in all cases approximation satisfactorily produces functions with two large wells corresponding to two large categories, and the shapes of both wells are broadly similar to those of the target energy function. The most probable patterns which correspond to the local minima, and the boundaries of each category coincide with those of the PDS approximately.

When the polynomial order n is small, such as $n = 10$, on the final MBDA after learning there are several ripples outside the range of inputs, i.e. $[-3,\ 3]$, whereas the landscape of PDS here is flat. In other words, the polynomial approximations generate several spurious minima besides the two desired attractors. Spurious minima are local minima that do not correspond to any of the classes. In the approximations with higher-order polynomials, such as $n = 20,\ 25,\ 30,\ 40$, more ripples appear on both tails but with smaller amplitude. Besides, within the range $[-3,\ 3]$, some spurious states occur inside the basin of attraction.

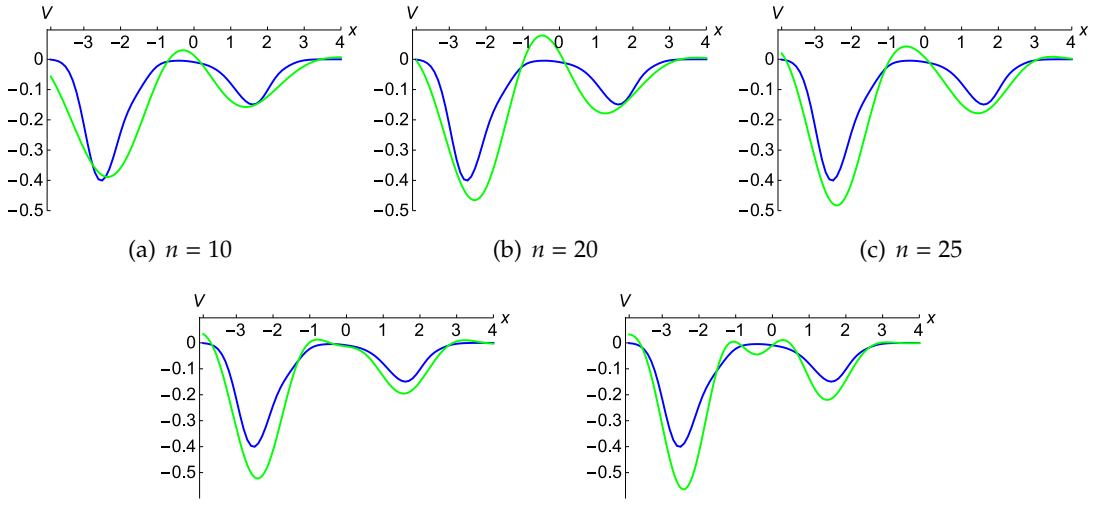
Hermite polynomial approximations, shown in Fig. 5.4, generate two categories more successfully in two respects. Firstly, the location of the attractors, i.e. of critical points, and the shapes of the basins of all desired attractors coincide with those of the PDS with smaller errors. Secondly, the number and amplitude of spurious minima outside the interval $[-3,\ 3]$ are much smaller than that of Legendre polynomial approximation. With both kinds of polynomials spurious minima appear inside $[-3, 3]$. However, when higher-order terms are included, $n > 30$, these minima are more pronounced in Hermite polynomial expansion.



(a) $n = 10$ (b) $n = 20$ (c) $n = 25$

(d) $n = 30$ (e) $n = 40$

Figure 5.3: The final landscapes of the PDS (blue line) and the final MBDA (orange line) based on Legendre polynomials of various orders n indicated in the field of each panel after all input data belonging to two categories were processed.



(a) $n = 10$ (b) $n = 20$ (c) $n = 25$

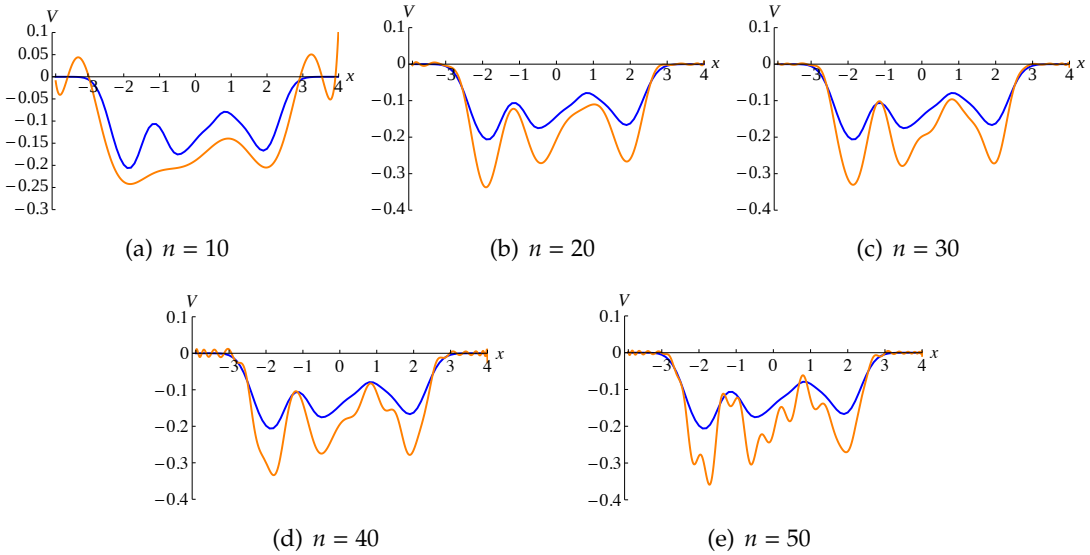
(d) $n = 30$ (e) $n = 40$

Figure 5.4: The final landscapes of the PDS (blue line) and the final MBDA (green line) based on Hermite polynomials of various orders n indicated in the field of each panel after all input data belonging to two categories were processed.

5.4.2 Input data from three categories

Here we illustrate the formation of the MBDA in response to input data coming from three categories, i.e. when the PDF of the generating random process has three peaks. The input signal $\eta(t)$ is shown in Fig. 3.4(a) and is used to shape the PDF approximation based on Legendre and Hermite polynomials, shown in Figs. 5.5 and 5.6, respectively. With this input, the effect of the order of polynomials applied in approximating the density function is more pronounced. When only lower-order polynomials are used in the MBDA, such as with $n = 10$, both methods fail to capture three categories appropriately. With higher-order terms added to the approximation, three categories are generated with acceptable error. However, spurious minima appear when the order exceeds a certain value.

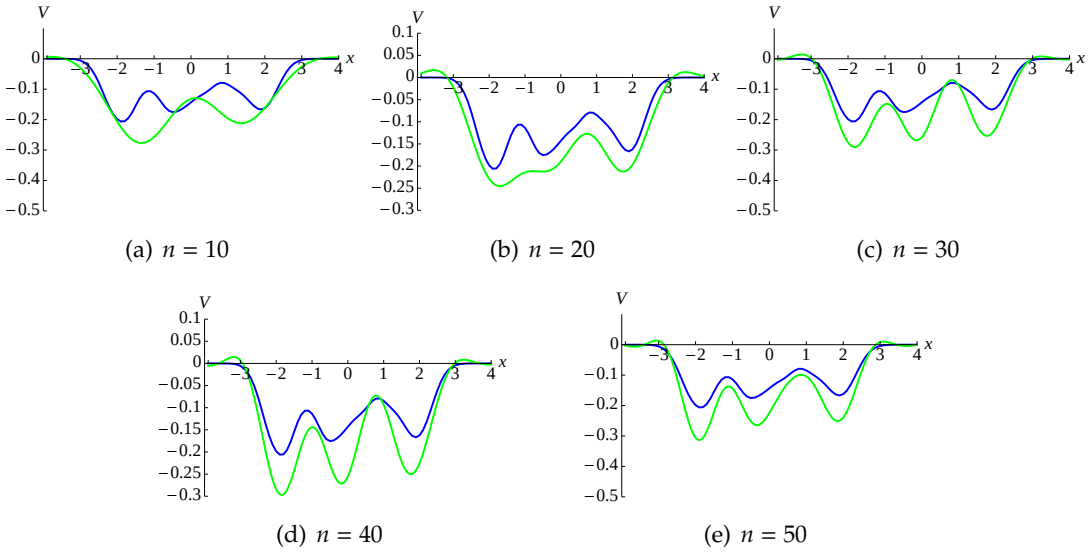
The spurious minima problem of the MBDA is caused by the property of polynomial itself. As is shown in the plots of Hermite polynomials (Fig. 5.2) and Legendre polynomials (Fig. 5.1), the number of extrema increases as the order of polynomial grows. In the two examples that we illustrated here, the MBDA using Hermite polynomials performs an approximation with a smaller number of spurious minima.



(a) $n = 10$ (b) $n = 20$ (c) $n = 30$

(d) $n = 40$ (e) $n = 50$

Figure 5.5: The final landscapes of the PDS (blue line) and the final MBDA (orange line) based on Legendre polynomials of various orders n indicated in the field of each panel after all input data belonging to three categories were processed.



(a) $n = 10$ (b) $n = 20$ (c) $n = 30$

(d) $n = 40$ (e) $n = 50$

Figure 5.6: The final landscapes of the PDS (blue line) and the final MBDA (green line) based on Hermite polynomials of various orders n indicated in the field of each panel after all input data belonging to three categories were processed.

5.5 Moment-based approximations of an evolving landscape

In Section 5.4, we have demonstrated with numerical stimulations that the MBDA can satisfactorily approximate the final energy landscape of the PDS when learning is finished. However, the main goal of this work is to find the way to approximate the energy of the PDS at all stages of its evolution in response to stimuli, and with an approach not requiring memory storage or a linearly growing computation time. Here we derive an evolution equation for the moments μ_j, which describes how these can be updated with account of the newly arriving stimuli without the need to store all values of the input.

In PDS (3.8), when there is no forgetting, i.e. $k = 0$, the energy landscape is updated according to

$$\frac{\partial V(\boldsymbol{x}, t)}{\partial t} = -\frac{1}{t}\Big(V(\boldsymbol{x}, t) + g(\boldsymbol{x} - \boldsymbol{\eta}(t))\Big). \tag{5.22}$$

In other words, the energy function is modified in response to the input based on its previous state. On the other hand, in Chapter 3 we have shown that under the condition that the initial landscape $V(\boldsymbol{x}, 0) = 0$, Eq. (5.22) can be discretized according to Eq. (3.11), from which one can see that $V(\boldsymbol{x}, n\Delta t)$ is an average of the "dents" resulting from the input.

In MBDA, the approximation is updated according to the moments estimated as the current accumulated averages of the input signal, i.e.

$$\mu_j(t) = \frac{1}{t}\int_0^t (\eta(s))^j \mathrm{d}s, \tag{5.23}$$

where $\mu_j(t)$ is the jth moment at time t. The discrete form of Eq. (5.23) is

$$\mu_j(n\Delta t) = \frac{1}{n}\sum_{i=0}^{n} (\eta(i\Delta t))^j, \tag{5.24}$$

which can be expressed as

$$\begin{aligned}\mu_j(n\Delta t) &= \frac{n-1}{n}\frac{1}{n-1}\sum_{i=0}^{n-1}(\eta(i\Delta t))^j + \frac{1}{n}(\eta(n\Delta t))^j \\ &= \frac{n-1}{n}\mu_j((n-1)\Delta t) + \frac{1}{n}(\eta(n\Delta t))^j, \\ \mu_j(n\Delta t) - \mu_j((n-1)\Delta t) &= -\frac{1}{n}\mu_j((n-1)\Delta t) + \frac{1}{n}(\eta(n\Delta t))^j.\end{aligned}$$

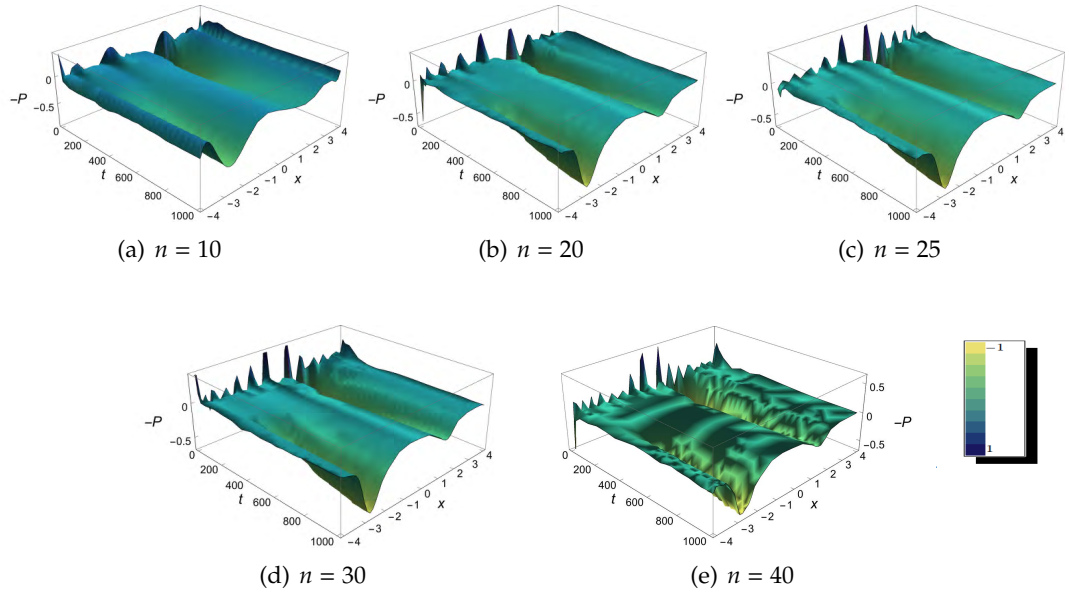
Substituting $(n-1)\Delta t = t$ and $n\Delta t = t + \Delta t$, we obtain

$$\mu_j(t+\Delta t) - \mu_j(t) = -\frac{\Delta t}{t+\Delta t}\mu_j(t) + \frac{\Delta t}{t+\Delta t}(\eta(t+\Delta t))^j.$$

Divide both sides by Δt given that $\Delta t \neq 0$, and take the limit as $\Delta t \to 0$, to obtain

$$\frac{\mathrm{d}\mu_j(t)}{\mathrm{d}t} = -\frac{1}{t}\Big(\mu_j(t) - (\eta(t))^j\Big). \tag{5.25}$$

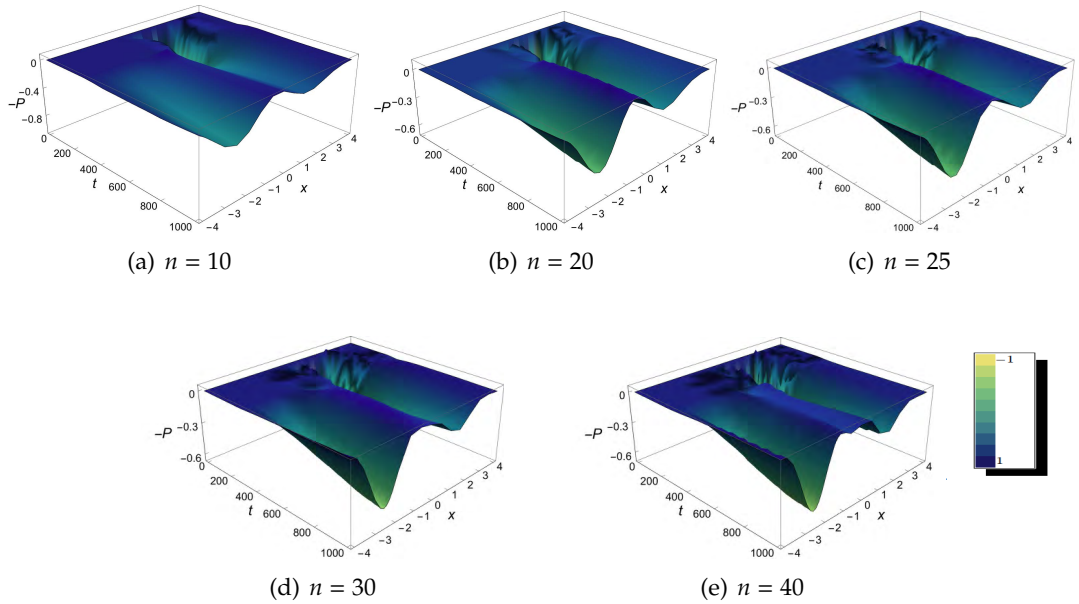
Hence the moments are updated according to this non-autonomous differential equation. Comparing the continuous-time updating equations (5.22) and (5.25) and their discrete forms (3.11) and (5.24) of PDS and MBDA, one can notice that both systems are evolving in a similar way. This is confirmed by the plots illustrating the evolution of both systems shown in Figs. 5.7-5.10. With both Hermite and Legendre polynomials, the evolution of the MBDAs broadly reproduces the evolution of the plastic self-organising energy landscape.



(a) $n = 10$ (b) $n = 20$ (c) $n = 25$

(d) $n = 30$ (e) $n = 40$

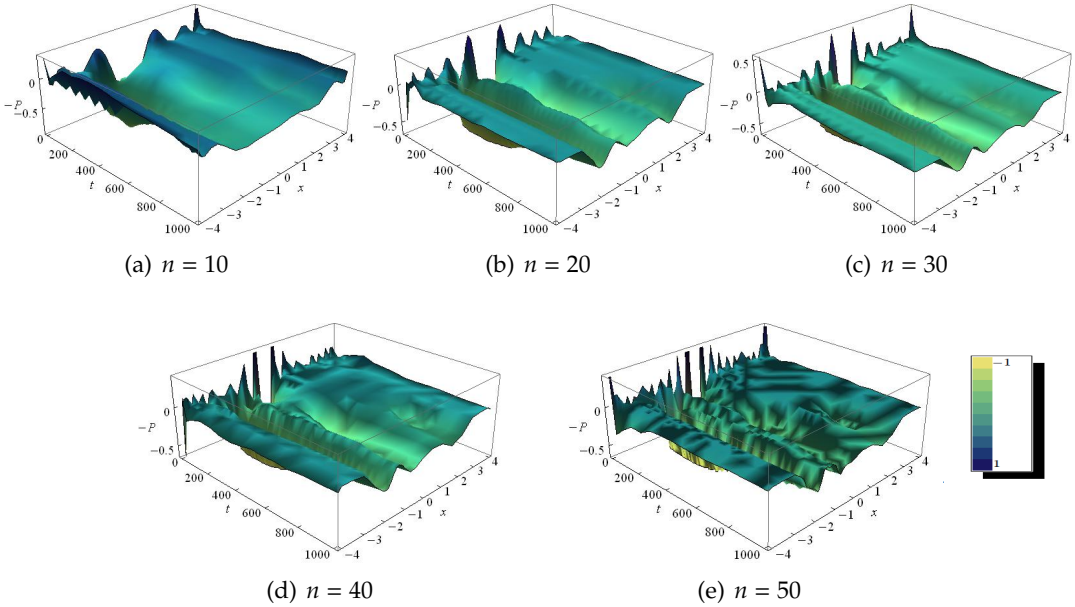
Figure 5.7: 3D plots illustrating evolution in time of the MBDA given input patterns belonging to two categories based on Legendre polynomials of various orders n indicated in the field of each panel.



(a) $n = 10$ (b) $n = 20$ (c) $n = 25$

(d) $n = 30$ (e) $n = 40$

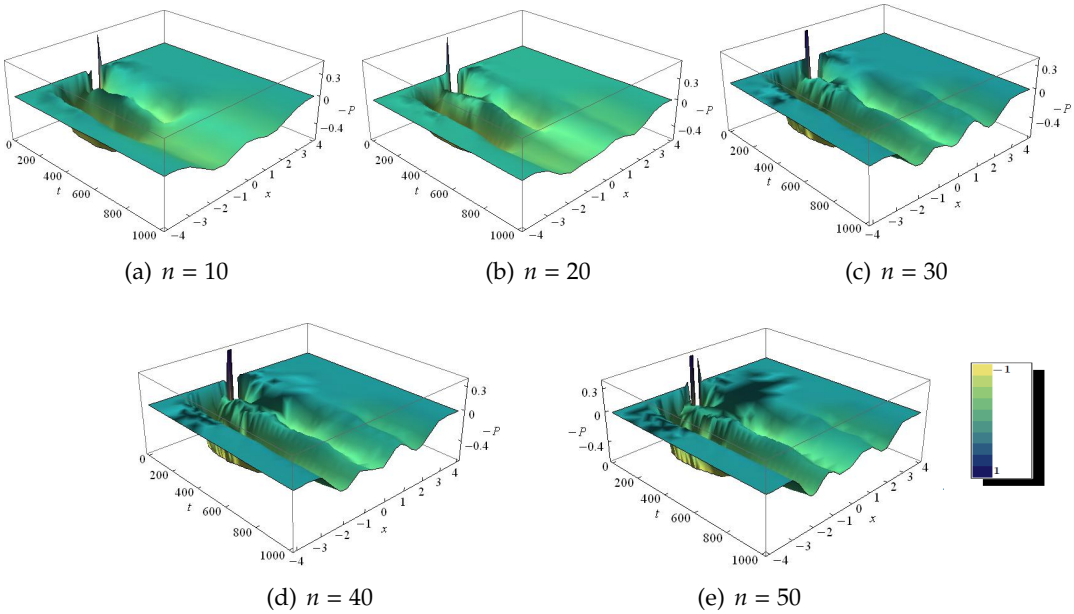
Figure 5.8: 3D plots illustrating evolution in time of the MBDA given input patterns belonging to two categories based on Hermite polynomials of various orders n indicated in the field of each panel.



(a) $n = 10$ (b) $n = 20$ (c) $n = 30$

(d) $n = 40$ (e) $n = 50$

Figure 5.9: 3D plots illustrating evolution in time of the MBDA given input patterns belonging to three categories based on Legendre polynomials of various orders n indicated in the field of each panel.



(a) $n = 10$ (b) $n = 20$ (c) $n = 30$

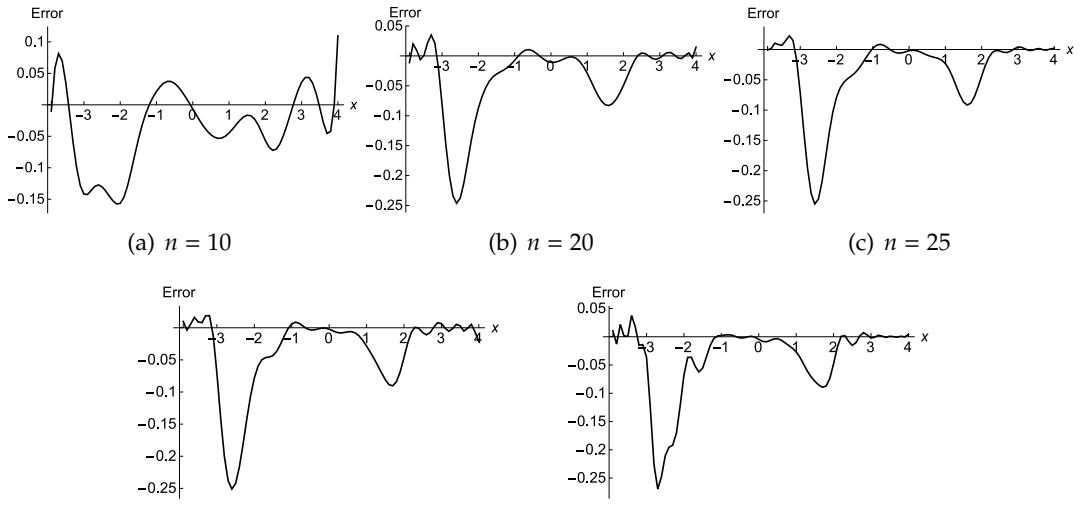
(d) $n = 40$ (e) $n = 50$

Figure 5.10: 3D plots illustrating evolution in time of the MBDA given input patterns belonging to three categories based on Hermite polynomials of various orders n indicated in the field of each panel.

5.6 Analysis of error of the approximation

In this Section, we numerically plot the difference between the MBDA using Legendre or Hermite polynomial of various orders and the PDS. We illustrate the error of the MBDA by comparing the location of the local minima of the plastic landscape and of its approximation. In addition, we analytically derive the expression of the difference between the PDS and the MBDA and hence prove that the moments of the latter are updated according to the rule (5.25).

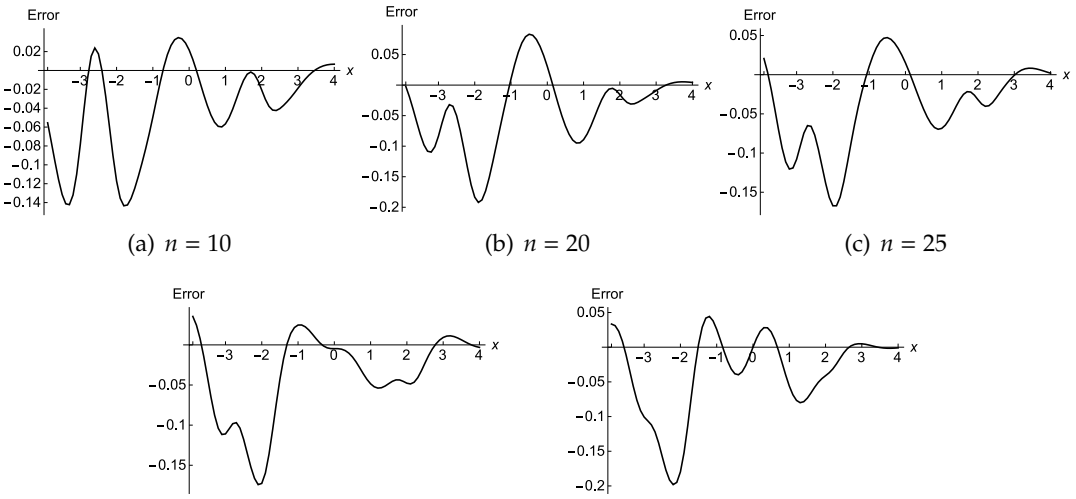
Figures 5.11-5.14 show the difference between the PDS landscape and its MBDA based on Legendre and Hermite polynomials of various orders n, i.e. the difference between the curves in Fig. 5.3-5.6, denoted by "Error". The error has a shape similar to that of the final landscapes, that is, the closer to either of the class centres, the larger the error is.



(a) $n = 10$ (b) $n = 20$ (c) $n = 25$

(d) $n = 30$ (e) $n = 40$

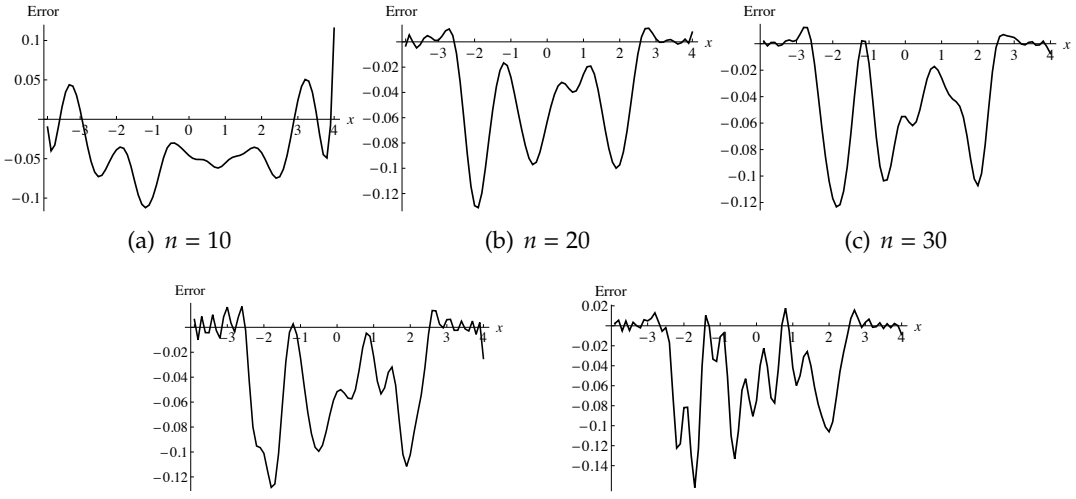
Figure 5.11: The difference between the PDS landscape and its MBDA using input data belonging to two categories based on Legendre polynomials of various orders n indicated in the field of each panel, i.e. the difference between the curves in Fig. 5.3, denoted by "Error".



(a) $n = 10$ (b) $n = 20$ (c) $n = 25$

(d) $n = 30$ (e) $n = 40$

Figure 5.12: The difference between the PDS landscape and its MBDA using input data belonging to two categories based on Hermite polynomials of various orders n indicated in the field of each panel, i.e. the difference between the curves in Fig. 5.4, denoted by "Error".



(a) $n = 10$ (b) $n = 20$ (c) $n = 30$

(d) $n = 40$ (e) $n = 50$

Figure 5.13: The difference between the PDS landscape and its MBDA using input data belonging to three categories based on Legendre polynomials of various orders n indicated in the field of each panel, i.e. the difference between the curves in Fig. 5.5, denoted by "Error".

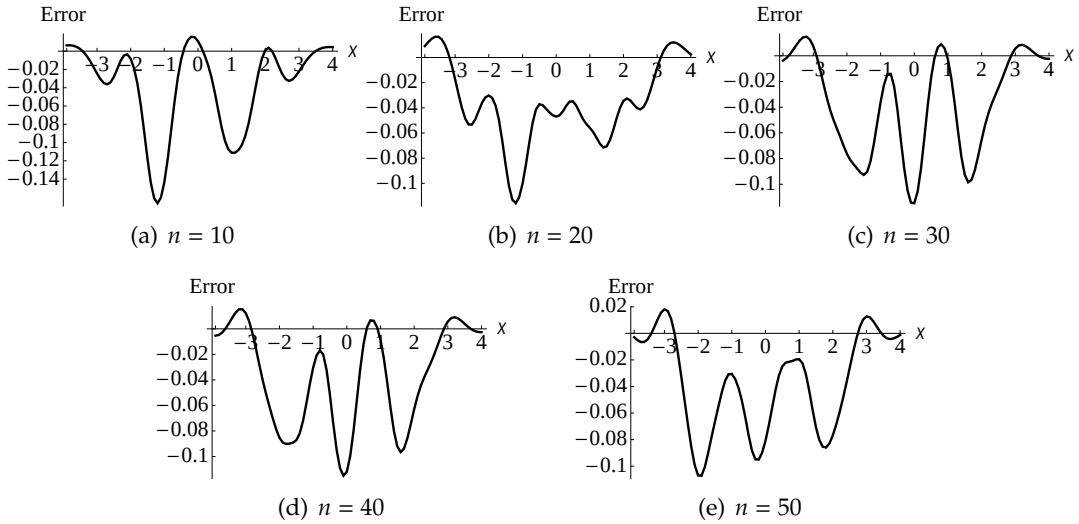


Figure 5.14: The difference between the PDS landscape and its MBDA using input data belonging to three categories based on Hermite polynomials of various orders n indicated in the field of each panel, i.e. the difference between the curves in Fig. 5.6, denoted by "Error".

Now we analytically estimate the error of MBDA of the energy landscape $V(x)$. Since in the simplest PDS (3.1), (3.8) the energy function V can be estimated by MBDA $f_H(x)$ (5.19) using Hermite polynomials, i.e. $V \approx -f_H$, we have

$$\frac{\partial V}{\partial t} \approx -\frac{\partial f_H}{\partial t}.$$

Here we use $J_{i,k} = (-1)^i \frac{k!}{i!(k-2i)!2^i}$ as abbreviated notation such that

$$H_k(x) = \sum_{i=0}^{[k/2]} J_{i,k} x^{k-2i}, \qquad D_k^H = \sum_{i=0}^{[k/2]} \frac{1}{k!} J_{i,k} \mu_{k-2i}. \tag{5.26}$$

Taking partial derivatives of V and of $-f_H$ with respect to t, we obtain

$$\begin{aligned}\frac{\partial V}{\partial t} &= -\frac{1}{t}\Big(V + g(x-\eta)\Big)\\ &\approx -\frac{1}{t}\Big(-f_H + g(x-\eta)\Big)\\ &= \frac{1}{t}\sum_{k=0}^{n} H_k(x)g^*(x)D_k^H - \frac{1}{t}g(x-\eta)\\ &= \frac{1}{t}\sum_{k=0}^{n} H_k(x)g^*(x)\sum_{i=0}^{[k/2]}\frac{1}{k!}J_{i,k}\mu_{k-2i} - \frac{1}{t}g(x-\eta),\end{aligned}$$

and

$$\begin{aligned}-\frac{\partial f_H}{\partial t} &= -\sum_{k=0}^{n} H_k(x)g^*(x)\frac{\mathrm{d}D_k^H}{\mathrm{d}t}\\ &= -\sum_{k=0}^{n} H_k(x)g^*(x)\sum_{i=0}^{[k/2]}\frac{1}{k!}J_{i,k}\frac{\mathrm{d}\mu_{k-2i}}{\mathrm{d}t}.\end{aligned}$$

Applying Eq. (5.25), we get

$$-\frac{\partial f_H}{\partial t} = -\sum_{k=0}^{n} H_k(x)g^*(x)\sum_{i=0}^{[k/2]}\frac{1}{k!}J_{i,k}\Big(-\frac{1}{t}\mu_{k-2i}\Big) - \sum_{k=0}^{n} H_k(x)g^*(x)\sum_{i=0}^{[k/2]}\frac{1}{k!}J_{i,k}\Big(-\frac{1}{t}\eta^{k-2i}\Big).$$

Then the difference between these two derivatives becomes

$$\frac{\partial Error}{\partial t} = -\frac{\partial f_H}{\partial t} - \frac{\partial V}{\partial t} = \sum_{k=0}^{n} H_k(x)g^*(x)\sum_{i=0}^{[k/2]}\frac{1}{k!}J_{i,k}\Big(\frac{1}{t}\eta^{k-2i}\Big) + \frac{1}{t}g(x-\eta). \tag{5.27}$$

Using Taylor's theorem, we can expand $g(x-\eta)$ with respect to $x-\eta$ at x and obtain

$$\begin{aligned} g(x-\eta) &= \sum_{k=0}^{n} \frac{(-1)^k}{k!}\eta^k \frac{\mathrm{d}^k g(x)}{\mathrm{d}x^k} \\ &= \sum_{k=0}^{n} \frac{(-1)^k}{k!}\eta^k \frac{\mathrm{d}^k g^*(x)}{\mathrm{d}x^k}\Big(\frac{1}{\sigma_x}\Big)^k \\ &= \sum_{k=0}^{n} \frac{(-1)^k}{k!}\eta^k (-1)^k H_k(x) g^*(x)\Big(\frac{1}{\sigma_x}\Big)^k \\ &= \sum_{k=0}^{n} \frac{1}{k!} H_k(x) g^*(x)\Big(\frac{\eta}{\sigma_x}\Big)^k. \end{aligned}$$

In the above, the following relation was used

$$\frac{\mathrm{d}^k g(x)}{\mathrm{d}x^k} = \frac{\mathrm{d}^k g^*(x)}{\mathrm{d}x^k}\Big(\frac{1}{\sigma_x}\Big)^k,$$

and the definition of the Hermite polynomial is expressed as

$$\frac{\mathrm{d}^k g^*(x)}{\mathrm{d}x^k} = (-1)^k H_k(x) g^*(x).$$

After the above expansion of $g(x-\eta)$ is substituted into Eq. (5.27), the difference becomes

$$\begin{aligned} \frac{\partial Error}{\partial t} &= -\frac{\partial f_H}{\partial t} - \frac{\partial V}{\partial t} \\ &= \sum_{k=0}^{n} H_k(x) g^*(x) \sum_{i=0}^{[k/2]} \frac{1}{k!} J_{i,k}\Big(\frac{1}{t}\eta^{k-2i}\Big) + \frac{1}{t}\sum_{k=0}^{n}\frac{1}{k!} H_k(x) g^*(x)\Big(\frac{\eta}{\sigma_x}\Big)^k \\ &= \frac{1}{t}\sum_{k=0}^{n}\frac{1}{k!} H_k(x) g^*(x)\Big(\sum_{i=0}^{[k/2]} J_{i,k}\eta^{k-2i} + \frac{\eta^k}{\sigma_x^k}\Big) \\ &= \frac{1}{t}\sum_{k=0}^{n}\frac{1}{k!} H_k(x) g^*(x)\Big(H_k(\eta) + \frac{\eta^k}{\sigma_x^k}\Big). \end{aligned} \tag{5.28}$$

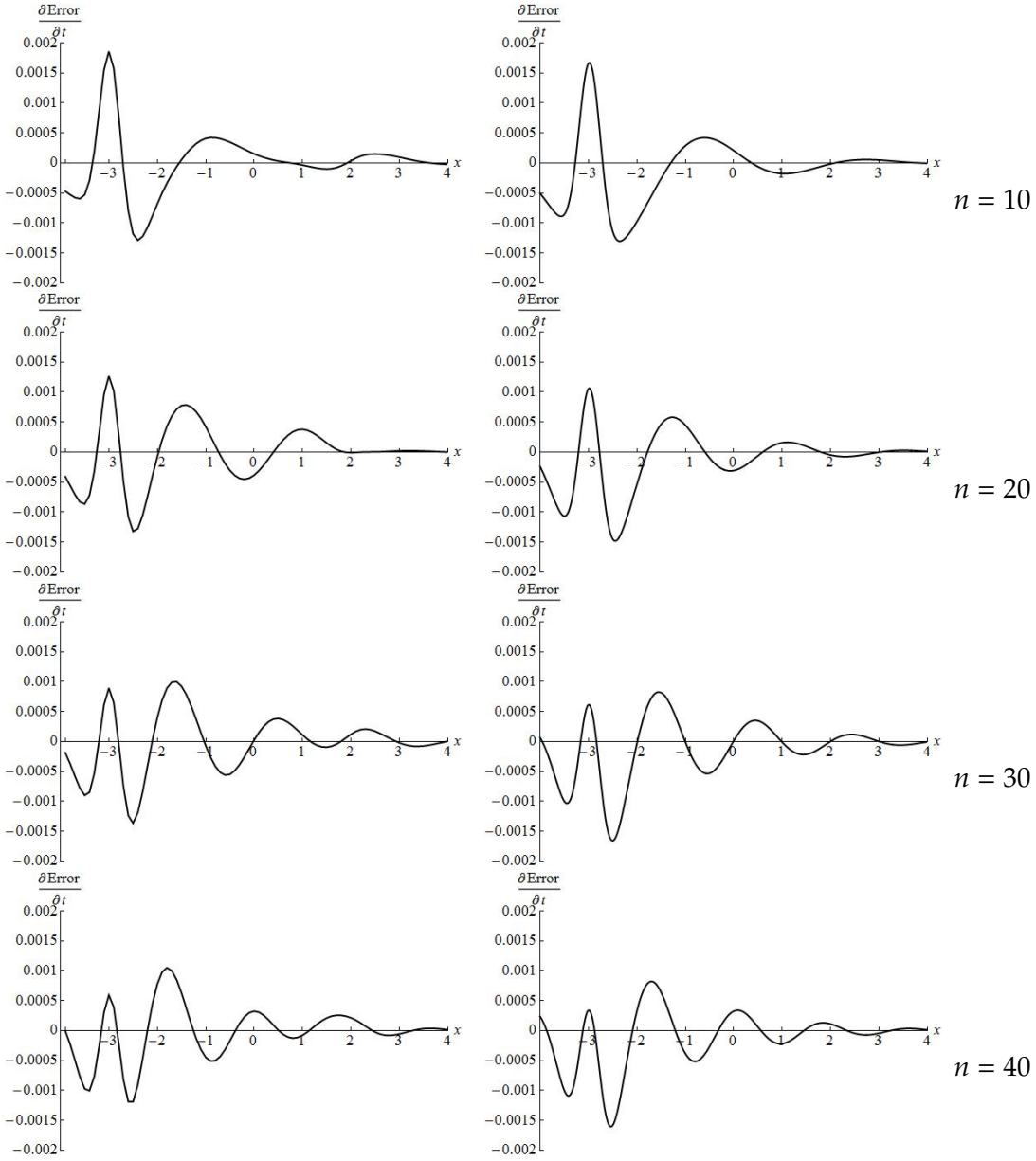
Hence we analytically derived the expression of the derivative of the error between

the MBDA using Hermite polynomial of the order n and the PDS. To demonstrate the accuracy of this estimation, the derivative of the error calculated both numerically and analytically is plotted in Fig. 5.15 for a range of values of the expansion order n and with input patterns belonging to two categories. Fig. 5.15(a) shows the difference $-\frac{\partial f_H(x)}{\partial t} - \frac{\partial V(x)}{\partial t}$ calculated numerically according to Eq. (5.19) and (5.22) and Fig. 5.15(b) is the plot of Eq. (5.28). It is shown that the derivative of the error calculated numerically coincides with the analytical one with minor difference. This confirms that the moments of the PDF are updated according to the Eq. (5.25).

Since it is the location of the patterns inside one of the categories that matters rather than the depth or the shape of the wells, here we measure the error of the polynomial approximation by comparing the location of the local minima on the PDS landscape and on its MBDA.

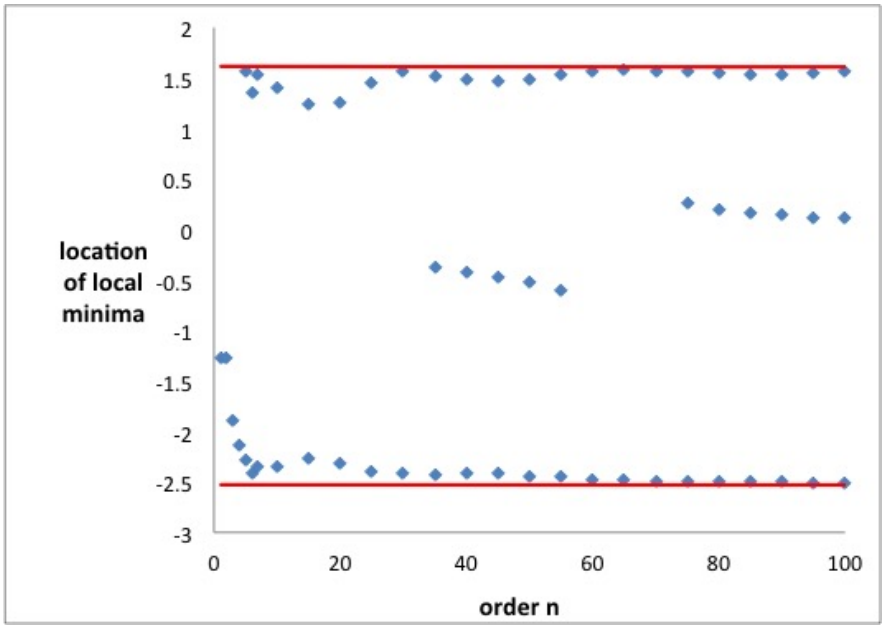
After processing the input data belonging to two categories, the PDS generates two wells and the location of the local minima are plotted by red lines in Fig. 5.16(a). The polynomial approximation using Hermite polynomials of lower order, such as $n < 5$, only generates one local minima. With higher order polynomials used in the approximation, the MBDA produces these two main local minima correctly and their locations are getting closer to that of the respective minima on the PDS landscape as the order grows. However, one spurious minima appears for some values of n, e.g. $n \in [40, 55]$ and $n \in [80, 100]$, but the amplitude of the spurious minima is relatively small compared with the depth of the main categories.

Figure 5.16(b) shows the location of the local minima of the landscapes after processing stimuli belonging to three categories. The MBDA produces only one local minimum with n taking smaller values, e.g. $n = 1$ or 5, and it generates two minima when n grows to 10. With $n >= 20$, three minima are produced and they are getting closer to the respective minima generated by the PDS as the order n grows.

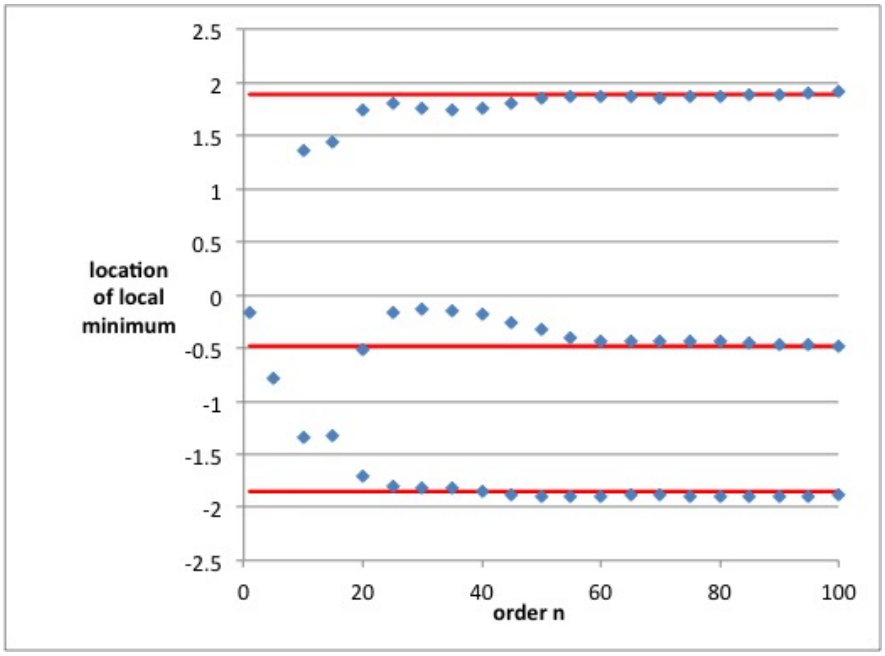


(a) (b)

Figure 5.15: The derivative of error of MBDA using Hermite polynomials with input from two categories calculated numerically and analytically. Each row corresponds to a single value of order n, which increases from top to bottom, with the values of n given in the field of the figure.
(a) Numerically calculated derivative of error, i.e. the difference $-\frac{\partial f_H(x)}{\partial t} - \frac{\partial V(x)}{\partial t}$ calculated numerically according to Eq. (5.19) and (5.22).
(b) Analytically calculated derivative of error expressed by Eq. (5.28).



(a)



(b)

Figure 5.16: The location of the local minima generated by the PDS (red lines) and by the MBDA (blue points) using Hermite polynomials of various orders n from 1 to 100. Only minima which are deeper than a certain threshold (5% of the amplitude of the deepest well) are considered.
(a) The location of minima of the landscapes after processing input data belonging to two categories.
(b) The location of minima of the landscapes after processing input data belonging to three categories.

5.7 Concluding remarks

We have considered another approach to imitate the performance of the gradient PDS. Namely, given that the energy of the latter converges to the negative of the density of the input process, the density can be approximated by polynomials, and polynomials can be implemented in electronic circuits. We approximated the density by an expansion in a series of polynomials with coefficients based on the moments of the input. Two examples, Legendre and Hermite polynomials, are used in the MBDA. The expressions for the polynomials and their coefficients are derived for each stage of data processing. The same example sets of input which were given to the PDS in Section 3.2 were also applied to the MBDA assuming that the input is a realisation of an ergodic random process, its moments can be estimated by the current updated accumulate time average, which does not require the knowledge of the density distribution.

The MBDAs with both Legendre and Hermite polynomials of various orders were analysed. Both approximations can categorise the input patterns using an appropriate order of polynomials with acceptable error. The most probable values of the input and the basins of attraction of the respective attractors are in reasonably good agreement with those generated by the PDS. However, there is a tendency of spurious noise to appear in the tails of the MBDA. In the MBDA based on Legendre polynomials, as the order grows, the number of the spurious minima increases while the magnitude of these minima decreases. When the order of the MBDA exceeds a certain value, the tails are smooth but spurious minima appear inside the basins of attraction of the desired attractors. With Hermite polynomials, the noise in the tails is avoided.

Chapter 6

Multivariate moment-based density approximation

In Chapter 5 we demonstrated the principle possibility to approximate the evolving energy function of a gradient PDS by means of a polynomial expansion based on the moments of the random process at the input of the learning system. For the first demonstration this approach was probed for a system whose input was generated by a scalar random process. However, in reality a scalar input signal will be of limited practical interest, and with the prospect of practical applications one needs to investigate the case of a multi-dimensional random input. Hence in this Chapter we attempt to apply the MBDA to describe the energy function of several variables. While the univariate MBDA is quite well developed by the present moment, it is a much less advanced area for multivariate distributions.

The density approximation using Hermite polynomial is also known as Gram-Charlier series. The Gram-Charlier series was generalised to an expansion for the PDF of two *independent* variates X_1, X_2 by Kendall and Stuart in 1948 [48]. A trivariate Gram-Charlier series was developed by Charlier to approximate the density function of three *independent* variables [14]. A more general case of three *correlated* variables

was discussed by Mihaila in 1968 [55], who also proved the convergence of this series.

In practice, it is quite unlikely that the stimuli arriving at different channels of a learning system will be statistically independent, or uncorrelated. Therefore, we need to study the cases of MBDA for the vector random processes whose components are correlated, for which the only formula available is the multi-dimensional Gram-Charlier series derived by Mihaila in [55] for a trivariate PDF. However, to visualise a PDF of three random variables even at a single time moment one needs a four-dimensional space, which is unavailable. With this, we adapt the latter expression to the case of bivariate densities, which can be depicted as surfaces.

6.1 Bivariate Gram-Charlier series

The joint PDF of two random variables $X_1,\ X_2$, which are statistically dependent, can be expressed by the bivariate Gram-Charlier series of the form

$$f(\boldsymbol{x}) = \sum_{j_1=0}^{\infty}\sum_{j_2=0}^{\infty} C_{j_1 j_2} \frac{\partial^{j_1+j_2}}{\partial x_1^{j_1}\partial x_2^{j_2}} g(\boldsymbol{x}), \tag{6.1}$$

where $\boldsymbol{x} = (x_1,\ x_2)^T$ is used for a brief notation. In the bivariate case, the Gaussian function becomes

$$g(\boldsymbol{x}) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left[-\frac{1}{2(1-\rho^2)}\left(\left(\frac{x_1-m_1}{\sigma_1}\right)^2 - 2\rho\left(\frac{x_1-m_1}{\sigma_1}\right)\left(\frac{x_2-m_2}{\sigma_2}\right) + \left(\frac{x_2-m_2}{\sigma_2}\right)^2\right)\right], \tag{6.2}$$

where $m_i = \mathbb{E}(X_i)$, and $\sigma_i^2 = \mathrm{Var}(X_i)$, $(i = 1,\ 2)$ and ρ is the correlation between X_1 and X_2 defined as

$$\rho = \frac{\mathbb{E}\Big[(X_1 - m_1)(X_2 - m_2)\Big]}{\sigma_1\sigma_2}. \tag{6.3}$$

Here we denote the correlation matrix $\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$ by

$$\boldsymbol{R} = \begin{pmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{pmatrix} \quad \text{with} \quad R_{11} = R_{22} = 1,\ R_{12} = R_{21} = \rho,$$

and its inverse matrix by

$$\boldsymbol{A} = \boldsymbol{R^{-1}} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} \frac{1}{1-\rho^2} & \frac{-\rho}{1-\rho^2} \\ \frac{-\rho}{1-\rho^2} & \frac{1}{1-\rho^2} \end{pmatrix} \text{ with } a_{11} = a_{22} = \frac{1}{1-\rho^2},\ a_{12} = a_{21} = \frac{-\rho}{1-\rho^2}.$$

To obtain Hermite polynomials, a standardisation transformation

$$y_i = \frac{x_i - m_i}{\sigma_i} \quad (i = 1, 2) \tag{6.4}$$

is applied to the variables X_1 and X_2 to produce standard random variables Y_1 and Y_2 with zero means and unit standard deviations. Hence the standardised bivariate normal distribution is

$$g(\boldsymbol{y}) = \frac{1}{2\pi\sqrt{\det(\boldsymbol{R})}} \exp\Big(-\frac{1}{2}\Big(a_{11}y_1^2 + 2a_{12}y_1y_2 + a_{22}y_2^2\Big)\Big). \tag{6.5}$$

The bivariate Hermite polynomials are defined in the same way as univariate Hermite polynomials, i.e.

$$\begin{aligned} &\frac{\partial^{j_1+j_2}}{\partial y_1^{j_1}\partial y_2^{j_2}} g(\boldsymbol{y}) = (-1)^{j_1+j_2} H_{j_1j_2}(\boldsymbol{y}) g(\boldsymbol{y}), \\ &H_{j_1j_2}(\boldsymbol{y}) = (-1)^{j_1+j_2}\Big(g(\boldsymbol{y})\Big)^{-1} \frac{\partial^{j_1+j_2}}{\partial y_1^{j_1}\partial y_2^{j_2}} g(\boldsymbol{y}). \end{aligned} \tag{6.6}$$

After standardisation of the variables according to Eq. (6.4), we have

$$\frac{\mathrm{d}y_i}{\mathrm{d}x_i} = \frac{1}{\sigma_i}, \quad (i = 1, 2),$$

and hence

$$\frac{\partial^{j_1+j_2}}{\partial x_1^{j_1}\partial x_2^{j_2}} g(\boldsymbol{x}) = \frac{1}{\sigma_1^{j_1}\sigma_2^{j_2}} \frac{\partial^{j_1+j_2}}{\partial y_1^{j_1}\partial y_2^{j_2}} g(\boldsymbol{y}) \tag{6.7}$$

Therefore the polynomial expansion in Eq. (6.1) becomes

$$f(\boldsymbol{x}) = \sum_{j_1=0}^{\infty}\sum_{j_2=0}^{\infty} D_{j_1j_2} H_{j_1j_2}(\boldsymbol{y}) g(\boldsymbol{y}) \tag{6.8}$$

where $D_{j_1j_2} = \frac{1}{\sigma_1^{j_1}\sigma_2^{j_2}}(-1)^{j_1+j_2} C_{j_1j_2}$. Note, that in the left-hand side of (6.8) the arguments y_1, y_2 need to be converted into x_1, x_2 according to (6.4) to obtain an expression for

$f(\boldsymbol{x})$.

To find the expansion coefficients $D_{j_1 j_2}$, the following notation is introduced

$$\phi(\boldsymbol{y}) = a_{11} y_1^2 + 2a_{12} y_1 y_2 + a_{22} y_2^2,$$

with a reciprocal form of it

$$\gamma(\boldsymbol{\xi}) = R_{11} \xi_1^2 + 2R_{12} \xi_1 \xi_2 + R_{22} \xi_2^2,$$

where

$$\xi_i = \frac{1}{2} \frac{\partial \phi}{\partial y_i}, \text{ i.e., } \xi_1 = a_{11} y_1 + a_{12} y_2 \text{ and } \xi_2 = a_{12} y_1 + a_{22} y_2, \tag{6.9}$$

and similarly

$$y_i = \frac{1}{2} \frac{\partial \gamma}{\partial \xi_i}, \text{ i.e., } y_1 = R_{11} \xi_1 + R_{12} \xi_2 \text{ and } y_2 = R_{12} \xi_1 + R_{22} \xi_2.$$

In other words, $\boldsymbol{y} = \boldsymbol{R} \cdot \boldsymbol{\xi}$ and $\boldsymbol{\xi} = \boldsymbol{A} \cdot \boldsymbol{y}$ where $\boldsymbol{y} = (y_1,\ y_2)^T$ and $\boldsymbol{\xi} = (\xi_1, \xi_2)^T$.
Then the expression for Hermite polynomials reads

$$H_{j_1 j_2}(\boldsymbol{y}) = (-1)^{j_1 + j_2} \exp\Big(\frac{1}{2}\phi(\boldsymbol{y})\Big) \frac{\partial^{j_1 + j_2}}{\partial y_1^{j_1} \partial y_2^{j_2}} \exp\Big(-\frac{1}{2}\phi(\boldsymbol{y})\Big).$$

The polynomial defined by

$$G_{j_1 j_2}(\boldsymbol{\xi}) = (-1)^{j_1 + j_2} \exp\Big(\frac{1}{2}\gamma(\boldsymbol{\xi})\Big) \frac{\partial^{j_1 + j_2}}{\partial \xi_1^{j_1} \partial \xi_2^{j_2}} \exp\Big(-\frac{1}{2}\gamma(\boldsymbol{\xi})\Big).$$

is also a Hermite polynomial and the polynomials H and G satisfy the orthogonality conditions

$$\iint_{-\infty}^{\infty} g(\boldsymbol{y}) G_{i_1 i_2}(\boldsymbol{y}) H_{j_1 j_2}(\boldsymbol{y}) \mathrm{d}y_1 \mathrm{d}y_2 = \begin{cases} 0 & \text{for } \boldsymbol{i} \neq \boldsymbol{j}, \\ i_1! i_2! & \text{for } \boldsymbol{i} = \boldsymbol{j}, \end{cases} \tag{6.10}$$

where $\boldsymbol{i} = (i_1,\ i_2)$ and $\boldsymbol{j} = (j_1,\ j_2)$.

Applying this property, we multiply both sides of Eq. (6.8) by $G_{i_1 i_2}(\boldsymbol{y})$ and integrate over the whole two-dimensional space of x_1, x_2 to obtain for $\boldsymbol{i} = \boldsymbol{j}$

$$\begin{aligned}
&\iint_{-\infty}^{\infty} G_{i_1 i_2}((x_1 - m_1)/\sigma_1, (x_2 - m_2)/\sigma_2) f(x_1, x_2) \mathrm{d}x_1 \mathrm{d}x_2 \\
&= \sum_{j_1=0}^{\infty} \sum_{j_2=0}^{\infty} D_{j_1 j_2} \iint_{-\infty}^{\infty} G_{i_1 i_2}(\boldsymbol{y}) H_{j_1 j_2}(\boldsymbol{y}) g(\boldsymbol{y}) \mathrm{d}y_1 \mathrm{d}y_2 \\
&= j_1! j_2! D_{j_1 j_2},
\end{aligned} \tag{6.11}$$

In the right-hand side of Eq. (6.11) all functions under the integral depend on $y_1,\ y_2$ rather than on $x_1,\ x_2$ for brevity and the ease of calculation. Obviously, the integral stays the same under the change of variables (6.4).

From (6.11) one can express $D_{j_1 j_2}$:

$$D_{j_1 j_2} = \frac{1}{j_1! j_2!} \iint_{-\infty}^{\infty} G_{i_1 i_2}((x_1 - m_1)/\sigma_1, (x_2 - m_2)/\sigma_2) f(x_1, x_2) \mathrm{d}x_1 \mathrm{d}x_2. \tag{6.12}$$

The Hermite polynomials $H_{j_1 j_2}(\boldsymbol{y})$ up to the forth order, i.e. for j_1 and j_2 such that

$j_1 + j_2 \leq 4$, calculated according to Eq. (6.6) read

$$
\begin{aligned}
&H_{00} = 1;\\
&H_{10} = \xi_1;\ \ H_{01} = \xi_2;\\
&H_{20} = \xi_1^2 - a_{11};\ \ H_{02} = \xi_2^2 - a_{22};\ \ H_{11} = \xi_1\xi_2 - a_{12};\\
&H_{30} = \xi_1^3 - 3a_{11}\xi_1;\ \ H_{03} = \xi_2^3 - 3a_{22}\xi_2;\\
&H_{21} = \xi_1^2\xi_2 - 2a_{12}\xi_1 - a_{11}\xi_2;\ \ H_{21} = \xi_2^2\xi_1 - 2a_{12}\xi_2 - a_{22}\xi_1;\\
&H_{40} = \xi_1^4 - 6a_{11}\xi_1^2 + 3a_{11}^2;\ \ H_{04} = \xi_2^4 - 6a_{22}\xi_2^2 + 3a_{22}^2;\\
&H_{31} = \xi_1^3\xi_2 - 3a_{12}\xi_1^2 - 3a_{11}\xi_1\xi_2 + 3a_{11}a_{12};\\
&H_{13} = \xi_2^3\xi_1 - 3a_{12}\xi_2^2 - 3a_{22}\xi_1\xi_2 + 3a_{22}a_{12};\\
&H_{22} = \xi_1^2\xi_2^2 - a_{22}\xi_1^2 - a_{11}\xi_2^2 - 4a_{12}\xi_1\xi_2 + a_{11}a_{22} + 2a_{12}^2.
\end{aligned}
\tag{6.13}
$$

In Eqs. (6.13) $H_{j_1 j_2}$ are expressed in terms of ξ_1, ξ_2 for brevity. To obtain their expression in terms of y_1, y_2, one needs to substitute ξ_1 and ξ_2 according to Eq. (6.9). The expressions for $G_{j_1 j_2}$ can be obtained from $H_{j_1 j_2}$ by replacing the variable ξ_i by y_i and the coefficients a_{ij} by R_{ij} [55]. For example, $G_{20} = y_1^2 - R_{11}$.

To calculate the coefficients $D_{j_1 j_2}$, we substitute the expressions of the polynomials $G_{j_1 j_2}(\boldsymbol{y})$ into Eq. (6.12) and express y_1, y_2 through x_1, x_2 according to (6.4):

$$
\begin{aligned}
D_{00} &= \int_{-\infty}^{\infty}\!\!\int f(\boldsymbol{x})\mathrm{d}x_1\mathrm{d}x_2 = 1,\\
D_{10} &= \int_{-\infty}^{\infty}\!\!\int y_1 f(\boldsymbol{x})\mathrm{d}x_1\mathrm{d}x_2 = \int_{-\infty}^{\infty}\!\!\int \frac{x_1 - m_1}{\sigma_1} f(\boldsymbol{x})\mathrm{d}x_1\mathrm{d}x_2 = \frac{\mu_{10}}{\sigma_1} = 0, \quad D_{01} = 0,\\
D_{20} &= \frac{1}{2}\int_{-\infty}^{\infty}\!\!\int (y_1^2 - R_{11}) f(\boldsymbol{x})\mathrm{d}x_1\mathrm{d}x_2 = \frac{1}{2}\left(\frac{\mu_{20}}{\sigma_1^2} - R_{11}\right), \quad D_{02} = \frac{1}{2}\left(\frac{\mu_{02}}{\sigma_2^2} - R_{22}\right).
\end{aligned}
$$

The coefficients are expressed in terms of the central moments of the variables X_1 and

X_2, which are defined as

$$\mu_{j_1 j_2} = \iint_{-\infty}^{\infty} (x_1 - m_1)^{j_1} (x_2 - m_2)^{j_2} f(\boldsymbol{x}) \mathrm{d}x_1 \mathrm{d}x_2.$$

The coefficients $D_{j_1 j_2}$ of higher orders can be obtained in a similar manner.

6.2 Generating test data

To assess the performance of a bivariate polynomial approximation of an evolving energy landscape in a learning system experiencing stimulus from two channels, some suitable test examples need to be produced. A two-dimensional stimulus is generated by the method similar to the one used in the one-dimensional input generation.

Note that it is quite easy to generate two statistically *independent* (and hence uncorrelated) random variables X_1 and X_2, given that their joint PDF $f(x_1, x_2)$ is equal to the product of the univariate PDFs $f_1(x_1)$ and $f_2(x_2)$ of the individual variables, i.e. $f(x_1, x_2) = f_1(x_1)f_2(x_2)$. Then it is sufficient to construct two individual landscapes $V_1(x_1)$ and $V_2(x_2)$ of the required shapes and launch two Ornstein-Uhlenbeck processes [26] by analogy with (3.12):

$$\frac{\mathrm{d}x_{1,2}(t)}{\mathrm{d}t} = -\frac{\partial V_{1,2}(x_{1,2})}{\partial x_{1,2}} + \zeta_{1,2}(t), \tag{6.14}$$

with $\zeta_1(t)$ and $\zeta_2(t)$ being Gaussian white noises with zero mean values and possibly differing standard deviations.

However, generation of two statistically dependent (and hence often correlated) random variables requires some additional effort, since their joint PDF $f(x_1, x_2)$ is not equal to $f_1(x_1)f_2(x_2)$. Namely, by analogy with a one-dimensional random process obtained by means of Eq. (3.12), the idea is to produce an energy landscape $V_d(x_1, x_2)$ with the required number and location of potential wells corresponding to different classes. Then, we can write down an equation for a particle moving in the given landscape under the influence of stochastic forces $\zeta_1(t)$ and $\zeta_2(t)$:

$$\begin{aligned} \frac{\mathrm{d}x_1(t)}{\mathrm{d}t} &= -\frac{\partial V_d(x_1, x_2)}{\partial x_1} + \zeta_1(t), \\ \frac{\mathrm{d}x_2(t)}{\mathrm{d}t} &= -\frac{\partial V_d(x_1, x_2)}{\partial x_2} + \zeta_2(t), \end{aligned} \tag{6.15}$$

where we assume that $\zeta_1(t)$ and $\zeta_2(t)$ are as in (6.14). The PDF $f(x_1, x_2)$ of the resultant random variables X_1 and X_2 is determined by the shape of $V_d(x_1, x_2)$, so that the location of the peaks of f coincides with that of the minima of V_d, if the noises are not too strong. Thus, the goal here is to construct the landscape V_d of an appropriate non-degenerate shape, so that its minima are not lined up in the direction of any of the axes. There could be several ways to obtain the required landscape, and we choose one of them as follows.

We introduce two auxiliary random variables $\hat{X}_1$ and $\hat{X}_2$, which are statistically independent, with one-dimensional (marginal) PDFs $V_1(\hat{x}_1)$ and $V_2(\hat{x}_2)$, respectively, obtained by means of numerically solving Eqs. (6.14) with $x_{1,2}$ replaced by $\hat{x}_{1,2}$. Their joint PDF $V(\hat{x}_1, \hat{x}_2) = V_1(\hat{x}_1)V_2(\hat{x}_2)$ has peaks lined up in the directions of axes $\hat{x}_1$ and/or $\hat{x}_2$. For example, for V_1 and V_2 shown in Fig. 6.1, their product has two peaks which lie on the line parallel to the axis $\hat{x}_1$. To avoid this degeneracy, we obtain $V_d(x_1, x_2)$ by rotating $V(\hat{x}_1, \hat{x}_2)$ about the origin, i.e. by introducing the change of variables

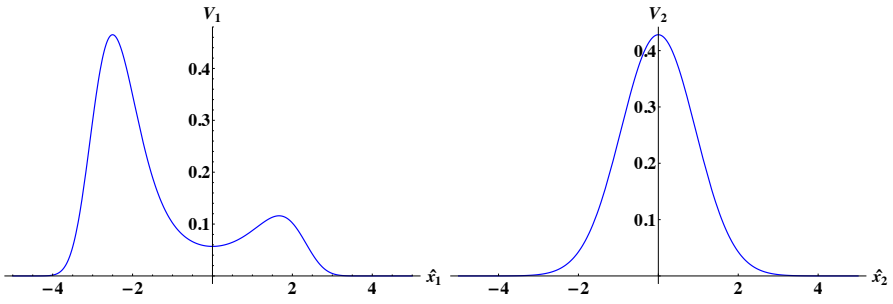
$$\begin{pmatrix} \hat{x}_1 \\ \hat{x}_2 \end{pmatrix} = \boldsymbol{T} \cdot \begin{pmatrix} x_1 \\ x_2, \end{pmatrix} \tag{6.16}$$

where $\boldsymbol{T}$ is the rotation matrix

$$\boldsymbol{T} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \quad \theta \in (0, 2\pi),$$

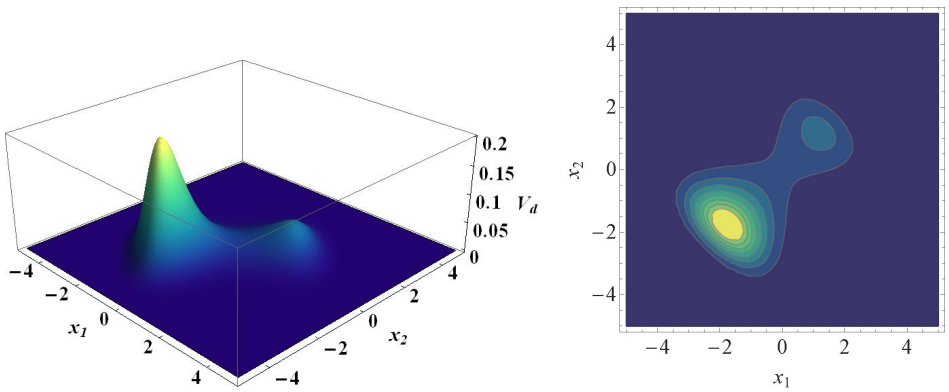
that rotates points in the (x_1, x_2)-plane counter-clockwise by an angle θ. Thus, $V_d(x_1, x_2)$ can be found as follows

$$\begin{aligned} V_d(x_1, x_2) &= V_1(\hat{x}_1)V_2(\hat{x}_2) \\ &= V_1(x_1\cos\theta - x_2\sin\theta, x_1\sin\theta + x_2\cos\theta)V_2(x_1\cos\theta - x_2\sin\theta, x_1\sin\theta + x_2\cos\theta). \end{aligned} \tag{6.17}$$



(a) (b)

Figure 6.1: Plot of $V_1(\hat{x}_1)$ and $V_2(\hat{x}_2)$ used to generate a two-peak density $V_d(x_1, x_2)$, see text. (a) $V_1(\hat{x}_1)$. (b) $V_2(\hat{x}_2)$.



(a) (b)

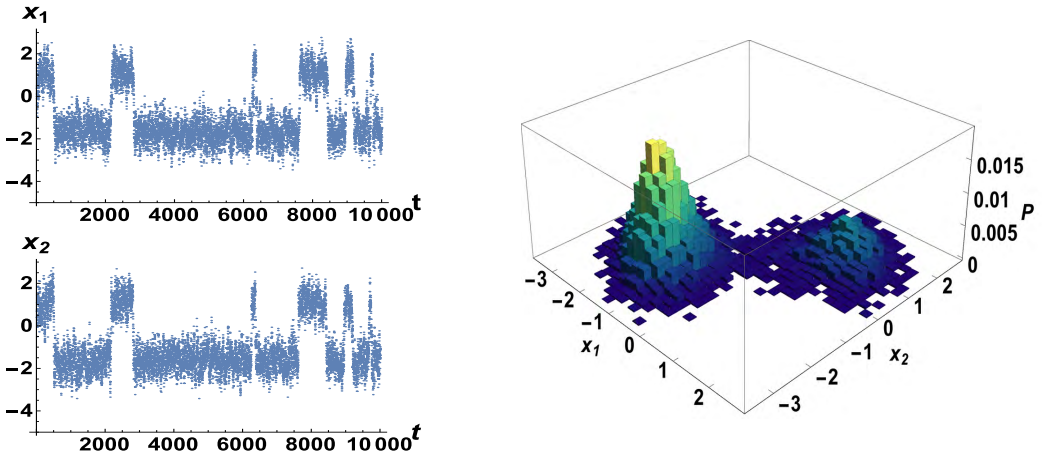
Figure 6.2: Plot of $V_d(x_1, x_2)$. (a) $V_d(x_1, x_2)$ as a surface. (b) Contour graph of $V_d(x_1, x_2)$.

6.2.1 Correlated random variables with a two-peak joint density

Here we take $\theta = \pi/4$, $\mathrm{Var}(\zeta_1) = \mathrm{Var}(\zeta_2) = 0.625$ and

$$\begin{aligned} V_1(\hat{x}_1) &= 0.057 \exp\Big(-0.069\hat{x}_1^4 - 0.077\hat{x}_1^3 + 0.576\hat{x}_1^2\Big), \\ V_2(\hat{x}_2) &= 0.428 \exp\Big(-0.576\hat{x}_2^2\Big), \end{aligned} \tag{6.18}$$

so that the landscape $V_d(x_1, x_2)$ has two peaks of different heights along the diagonal $x_1 = x_2$. The plots of $V_1(\hat{x}_1)$ and $V_2(\hat{x}_2)$ are shown in Fig. 6.1 and the landscape $V_d(x_1, x_2)$ shown as a surface and as its contour graph, is given in Fig. 6.2.



(a) (b)

Figure 6.3: Plot of test data belonging to two categories and their distribution histogram.
(a) The series of x_1 and x_2. (b) Histogram of data points (x_1, x_2).

Ten thousand of the two-dimensional data points are produced by numerically solving Eq. (6.15) with this form of $V_d(x_1, x_2)$. The series of x_1 and x_2 and their distribution histogram are shown in Fig. 6.3. The correlation between X_1 and X_2 calculated based on this sample is $\rho = 0.779$.

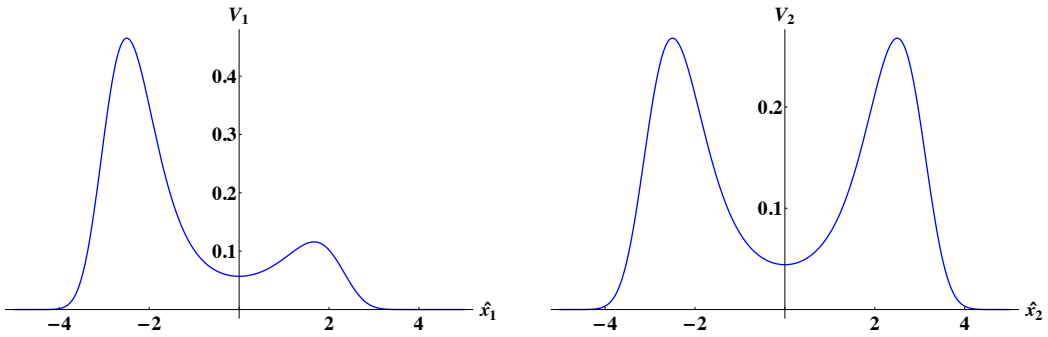
6.2.2 Correlated random variables with a four-peak joint density

A series of test data originating from (6.15) with a more complicated landscape is generated by taking $\theta = \pi/6$, $\mathrm{Var}(\zeta_1) = \mathrm{Var}(\zeta_2) = 0.625$ and

$$\begin{aligned} V_1(\hat{x}_1) &= 0.057 \exp\Big(-0.069\hat{x}_1^4 - 0.077\hat{x}_1^3 + 0.576\hat{x}_1^2 \Big), \\ V_2(\hat{x}_2) &= 0.044 \exp\Big(-0.046\hat{x}_2^4 + 0.576\hat{x}_2^2 \Big), \end{aligned} \tag{6.19}$$

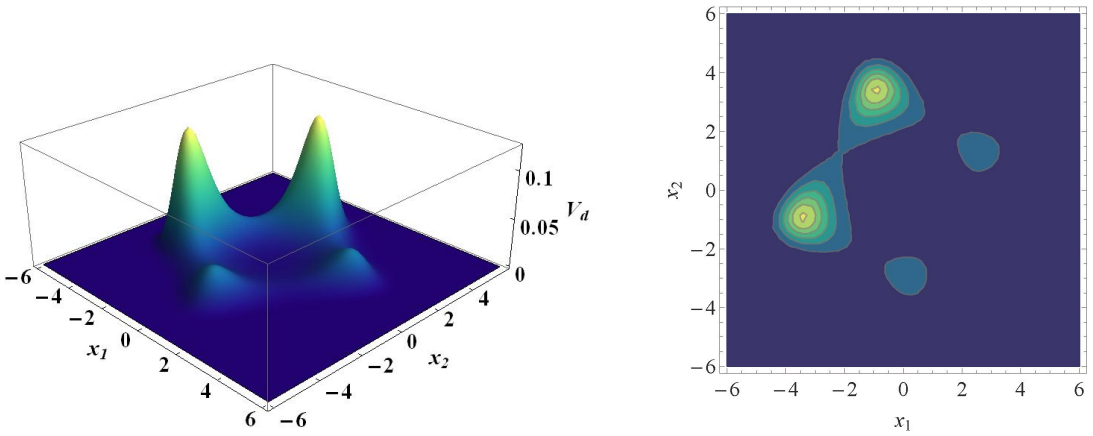
which are plotted in Fig. 6.4. $V_d(x_1, x_2)$ has four peaks as shown in Fig. 6.5.

Ten thousand of the two-dimensional data points are produced by numerically solving Eq. (6.15) with this expression of $V_d(x_1, x_2)$ having four peaks. The series of x_1 and x_2 and their distribution histogram of the data point (x_1, x_2) are shown in Fig. 6.6.



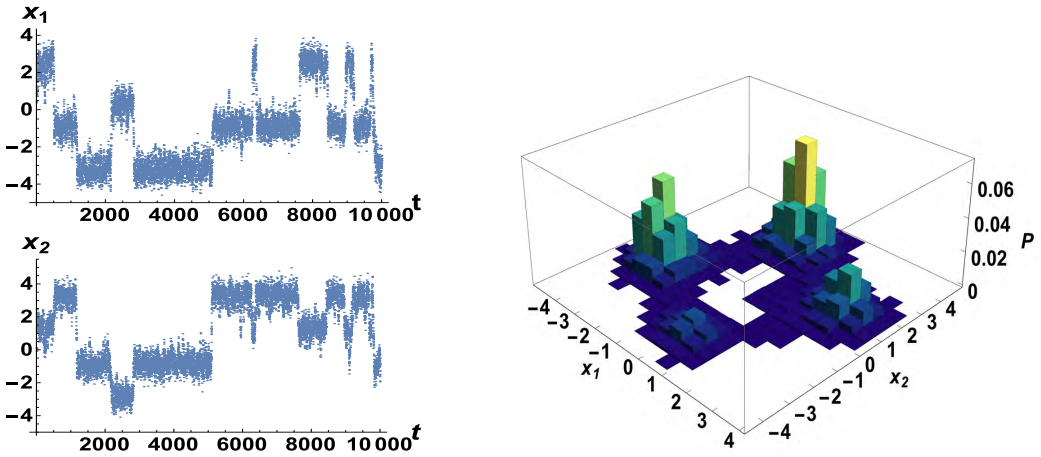
(a) (b)

Figure 6.4: Plot of $V_1(\hat{x}_1)$ and $V_2(\hat{x}_2)$ used to generate a four-peak density $V(x_1, x_2)$, see text. (a) $V_1(\hat{x}_1)$. (b) $V_2(\hat{x}_2)$.



(a) (b)

Figure 6.5: Plot of $V_d(x_1, x_2)$. (a) $V_d(x_1, x_2)$ as a surface. (b) Contour graph of $V_d(x_1, x_2)$



(a) (b)

Figure 6.6: Plot of test data belonging to four categories and the distribution histogram estimated from this data.
(a) The series of x_1 and x_2. (b) Histogram of data points (x_1, x_2).

The correlation between X_1 and X_2 calculated based on this sample is $\rho = 0.326$.

6.3 Results

Here we will demonstrate the performance of the bivariate MBDA based on Gram-Charlier series using the two samples of input data generated in Section 6.2 as compared to the vector field of the PDS. In practice, the Gram-Charlier series (6.8) is truncated to a polynomial of some order n, that is

$$f(\boldsymbol{x}) = \sum_{j_1+j_2=0}^{n} D_{j_1 j_2} H_{j_1 j_2}(\boldsymbol{y}) g(\boldsymbol{y}) \tag{6.20}$$

where $y_i = (x_i - m_i)/\sigma_i, \quad (i = 1, 2)$ are the values of the standardised variables Y_1 and Y_2.

To calculate the coefficients $D_{j_1 j_2}$, which are linear combinations of finite numbers of moments, the central moments are obtained from realisations $x_1^*(t)$ and $x_2^*(t)$ of the random processes $X_1(t)$ and $X_2(t)$ using the assumption of their ergodicity

$$\begin{aligned} \mu_{j_1 j_2} &= \iint_{-\infty}^{\infty} (x_1 - m_1)^{j_1} (x_2 - m_2)^{j_2} f(\boldsymbol{x}) \mathrm{d}x_1 \mathrm{d}x_2 \\ &= \lim_{T \to \infty} \frac{1}{T} \int_0^T (x_1^*(t) - m_1)^{j_1} (x_2^*(t) - m_2)^{j_2} \mathrm{d}t. \end{aligned} \tag{6.21}$$

Here m_1 and m_2 are the mean values of X_1 and X_2, respectively, which are constants because we assume stationarity of these random processes.

By analogy with a one-dimensional case described by Eq. (5.23), these averages over time can be estimated as the current accumulate averages, i.e.

$$\mu_{j_1 j_2} \approx \frac{1}{t} \int_0^t (x_1^*(t) - m_1)^{j_1} (x_2^*(t) - m_2)^{j_2} \mathrm{d}t. \tag{6.22}$$

At each time moment t, one input signal $(x_1(t),\ x_2(t))$, which is generated in Section

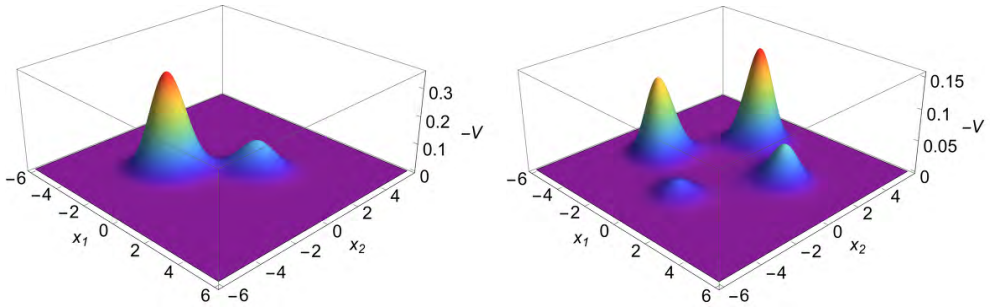
6.2, is given to the density approximation. The mean value m_i, $(i = 1, 2)$ is calculated based on the input values arrived before time $t + \Delta t$. Then the moment $\mu_{j_1 j_2}$ is updated thus updating the density approximation. In what follows we choose $\Delta t = 1$ in the numerical scheme.

6.3.1 Approximations of a two-dimensional landscape at the end of learning

In earlier literature, the Gram-Charlier series (6.8) was often truncated after the fourth-order term since this allows the density function to be flatter than Gaussian [45]. In the following, we approximate the density function, and hence the landscape of the PDS, by series of various orders, evaluate the suitability of such an expansion and its optimal order for two different cases of input data.

Figure 6.7 shows the negative of the energy landscape of the gradient PDS, formed as a result of processing of two-dimensional input data generated by random processes with a two-peak and a four-peak PDFs. The distribution histograms estimated from this data (Figs. 6.3(b) and 6.6(b)) have very similar shapes and confirm that the landscapes indeed converge to the negatives of the densities of the respective input processes. With this the locations of maxima and the shapes of the respective peaks are reproduced well, thus confirming that the PDS can successfully identify combinations of input values corresponding to the most typical representatives from each category, and the categories themselves.

The MBDAs of the final landscapes using various orders of polynomials are shown in Figs. 6.8 and 6.9 for two different kinds of input data. Namely, in Fig. 6.8 the approximations are illustrated for the two-peak landscape of Fig. 6.7(a) for a range of polynomial orders from 4 to 20. When the Gram-Charlier series is truncated after terms of 4th order, the polynomial density approximation only generates one peak, which represents one category, almost covering the whole range of the input data. With terms of higher order added to the approximation (such as $n = 6, 8, ...$), this MBDA correctly

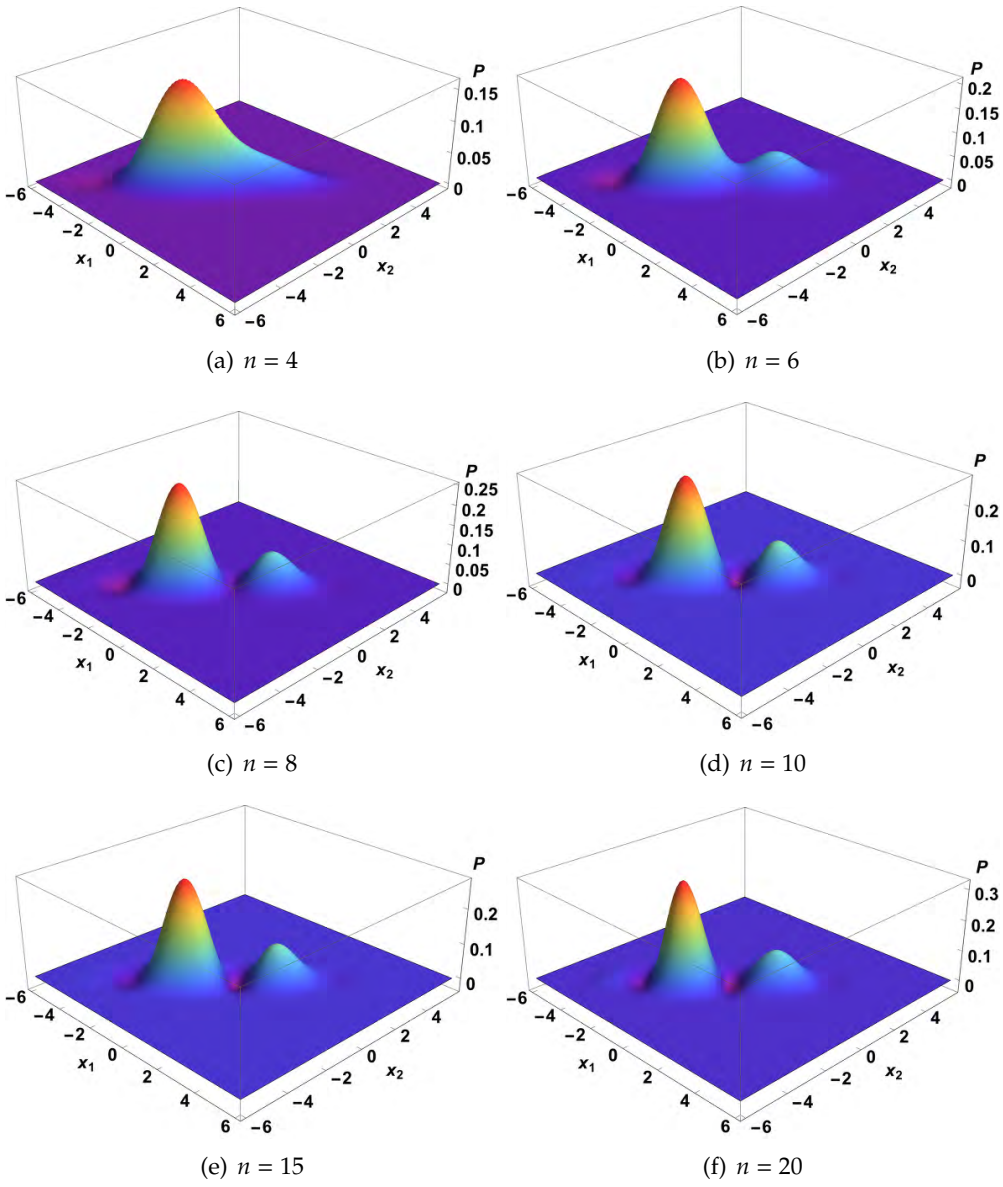


(a) (b)

Figure 6.7: Negative of the energy landscape $V(x_1, x_2)$ of the PDS at the final stage of processing the two-dimensional input signal originating from random processes with (a) two, and (b) four, peaks in the PDF.

reproduces two peaks which coincide with those in the histogram given in Fig. 6.3(b) and in the energy landscape of the PDS shown in Fig. 6.7(a). The most probable input vectors, which correspond to the top of the peaks, also coincide.

With input data which led to formation of the four-peak landscape shown in Fig. 6.7(b), the MBDA of various orders is illustrated in Fig. 6.9. Just like in the previous example, when only terms up to 4th order are used, only one peak is generated covering a broad area. Approximations obtained by series of higher order polynomials (such as $n = 6, 8$ and 10) generate the required number of peaks, each corresponding to one category. However, the shape of the landscape is not reproduced correctly, and the peaks are not clearly separated. With an increased order of polynomial, such as $n = 15$ or 20, the target peaks become separated clearly, and the expansion correctly captures their location, spread and height (compare with Fig. 6.7(b)). At the same time, new ripples appear that correspond to spurious categories, however, the basins of the respective attractors are small.



(a) $n = 4$ (b) $n = 6$

(c) $n = 8$ (d) $n = 10$

(e) $n = 15$ (f) $n = 20$

Figure 6.8: MBDA of the energy landscape of the PDS at the final stage of processing of the two-dimensional input signal originating from a random process with two peaks in its PDF. Approximation order n is given in the field of each panel. Compare with Fig. 6.7(a).

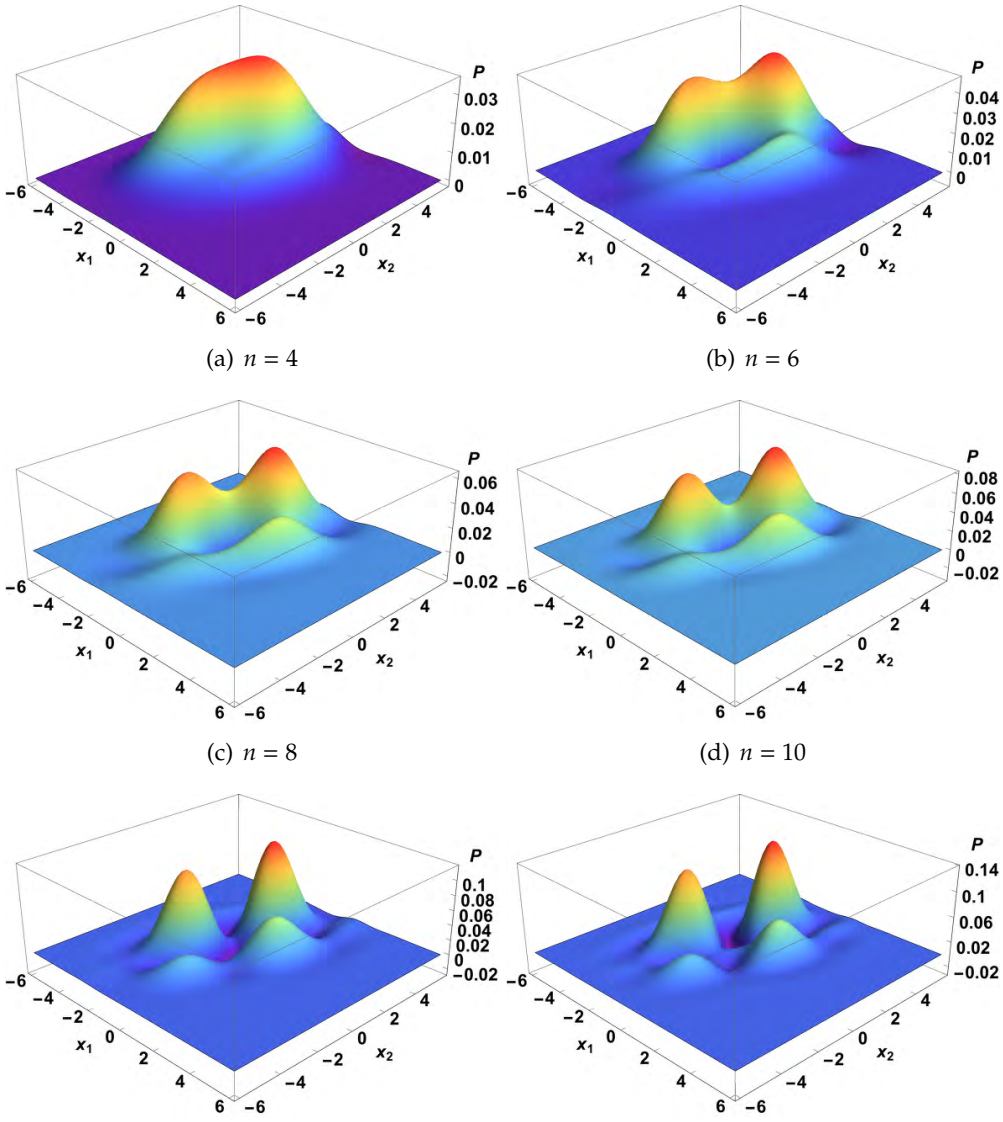


Figure 6.9: MBDA of the energy landscape of the PDS at the final stage of processing of the two-dimensional input signal originating from a random process with four peaks in its PDF. Approximation order n is given in the field of each panel. Compare with Fig. 6.7(b).

6.3.2 Approximations of the evolving landscape

In Section 6.3.1, we demonstrated that the bivariate Gram-Charlier series can satisfactorily approximate the energy landscape of the gradient PDS after all input patterns are given to the system. Here we show that the bivariate Gram-Charlier series also reproduces the evolution of the energy function while the PDS processes input data.

In Section 5.5, we derived the rules for updating the expansion coefficients in the MBDA used for approximations of the one-dimensional energy landscapes. In the two-dimensional case, one can prove that the moments are updated in the same way using the same approach.

The correlation coefficient between variables X_1 and X_2 is defined as Eq. (6.3). An important difference from the one-dimensional case becomes apparent if one compares Eqs. (5.19) and (6.20), and more specifically the expressions for Gaussian functions entering these equations described by (5.14) and (6.5), respectively. In the two-dimensional case, the function $g(\boldsymbol{y})$ in Eq. (6.20) is found using (6.5), in which coefficients a_{ij} are determined by the current value of the correlation coefficient ρ calculated according to Eq. (6.3). Below we express ρ through the moments of the two-dimensional random process (X_1, X_2). Namely

$$m_1 = \mathbb{E}(X_1) = \iint_{-\infty}^{\infty} x_1 f(\boldsymbol{x}) \mathrm{d}x_1 \mathrm{d}x_2 = \mu_{10}, \quad m_2 = \mu_{01},$$

$$\sigma_1^2 = \mathbb{E}[(X_1 - m_1)^2] = \iint_{-\infty}^{\infty} (x_1 - \mu_{10})^2 f(\boldsymbol{x}) \mathrm{d}x_1 \mathrm{d}x_2 = \mu_{20}, \quad \sigma_2^2 = \mu_{02},$$

$$\mathbb{E}[(X_1 - m_1)(X_2 - m_2)] = \iint_{-\infty}^{\infty} (x_1 - \mu_{10})(x_2 - \mu_{01}) f(\boldsymbol{x}) \mathrm{d}x_1 \mathrm{d}x_2 = \mu_{11}.$$

Then ρ can be expressed as

$$\rho = \frac{\mu_{11}}{\sqrt{\mu_{20}\mu_{02}}}. \tag{6.23}$$

Since with the new stimulus arriving at every new time moment, all moments entering (6.23) evolve, the coefficient ρ evolves, too. Therefore, the Gaussian functions entering the Gram-Charlier expansion (6.20) are updated with every new pair of values (x_1, x_2) entering the learning system. This constitutes the main difference from the one-dimensional case in which the Gaussian functions in the polynomial expansion do not change with time.

For illustrations of the evolution of this approximation, we fix the order of polynomials at which the Gram-Charlier series is truncated, so that it can achieve an acceptable accuracy of approximation and save the calculation time. With input coming from two categories, only the terms lower than 6th order are used in the MBDA. A number of snapshots of the distribution histogram, the energy function of the PDS and of its MBDA corresponding to different time moments are shown in Figs. 6.10 and 6.11. It is shown that the number, location and magnitude of peaks in these three functions evolve in the same way.

This property is more clearly seen in the example where input arrives from four categories illustrated by Figs. 6.12 and 6.13. In approximating the four-peak density function, the MBDA is truncated after the 10th order terms. The plots of the three functions at different time moments show how various categories are gradually identified by the PDS and its approximation as the input arrives. The changes in the number, magnitude and location of the peaks in the landscapes agree well with the values of the stimulus. These results confirm that the evolution of the MBDA approximates that of the PDS with a satisfactory accuracy.

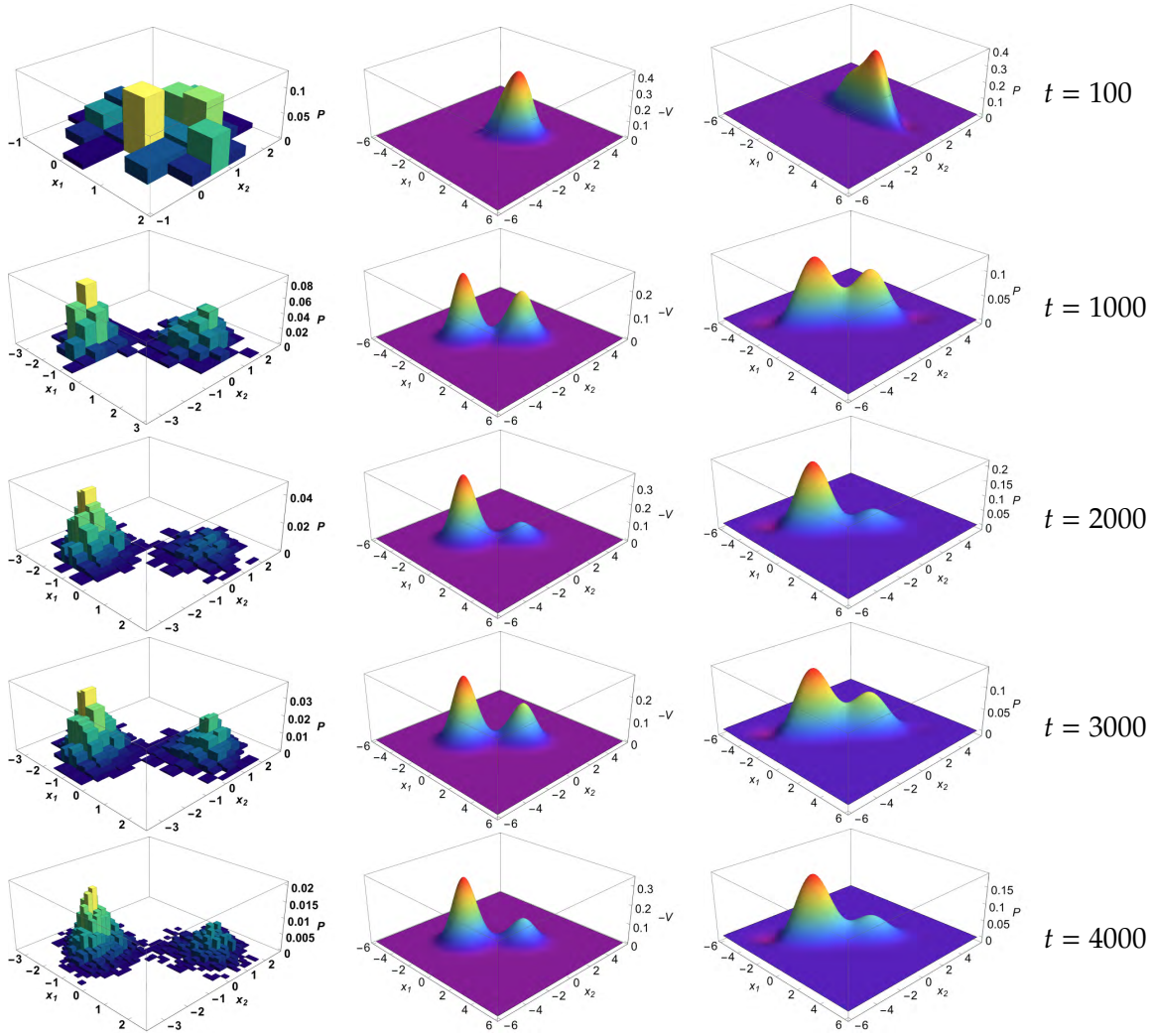
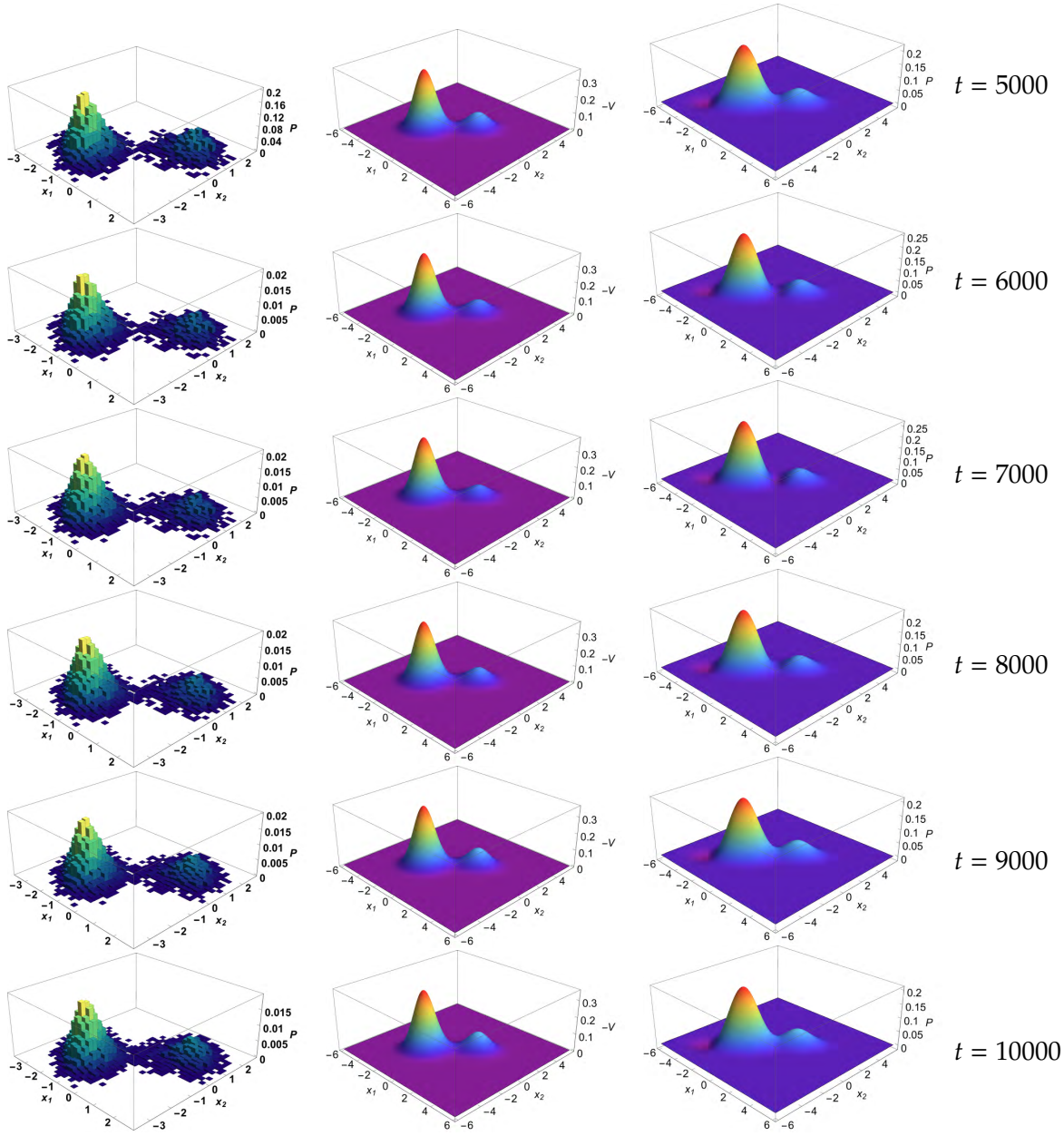


Figure 6.10: Evolution of (a) distribution histogram, (b) negative of the self-shaping landscape with $\sigma_x{}^2 = 0.1$ and (c) MBDA of the landscape with $n = 6$ while processing a two-dimensional input signal generated by a random process with a two-peak PDF. Each row corresponds to a single time moment t, which increases from top to bottom, with the values of t given in the field of the figure.



(a) (b) (c)

Figure 6.11: Evolution of (a) distribution histogram, (b) negative of the self-shaping landscape with $\sigma_x{}^2 = 0.1$ and (c) MBDA of the landscape with $n = 6$ while processing a two-dimensional input signal generated by a random process with a two-peak PDF. Each row corresponds to a single time moment t, which increases from top to bottom, with the values of t given in the field of the figure.

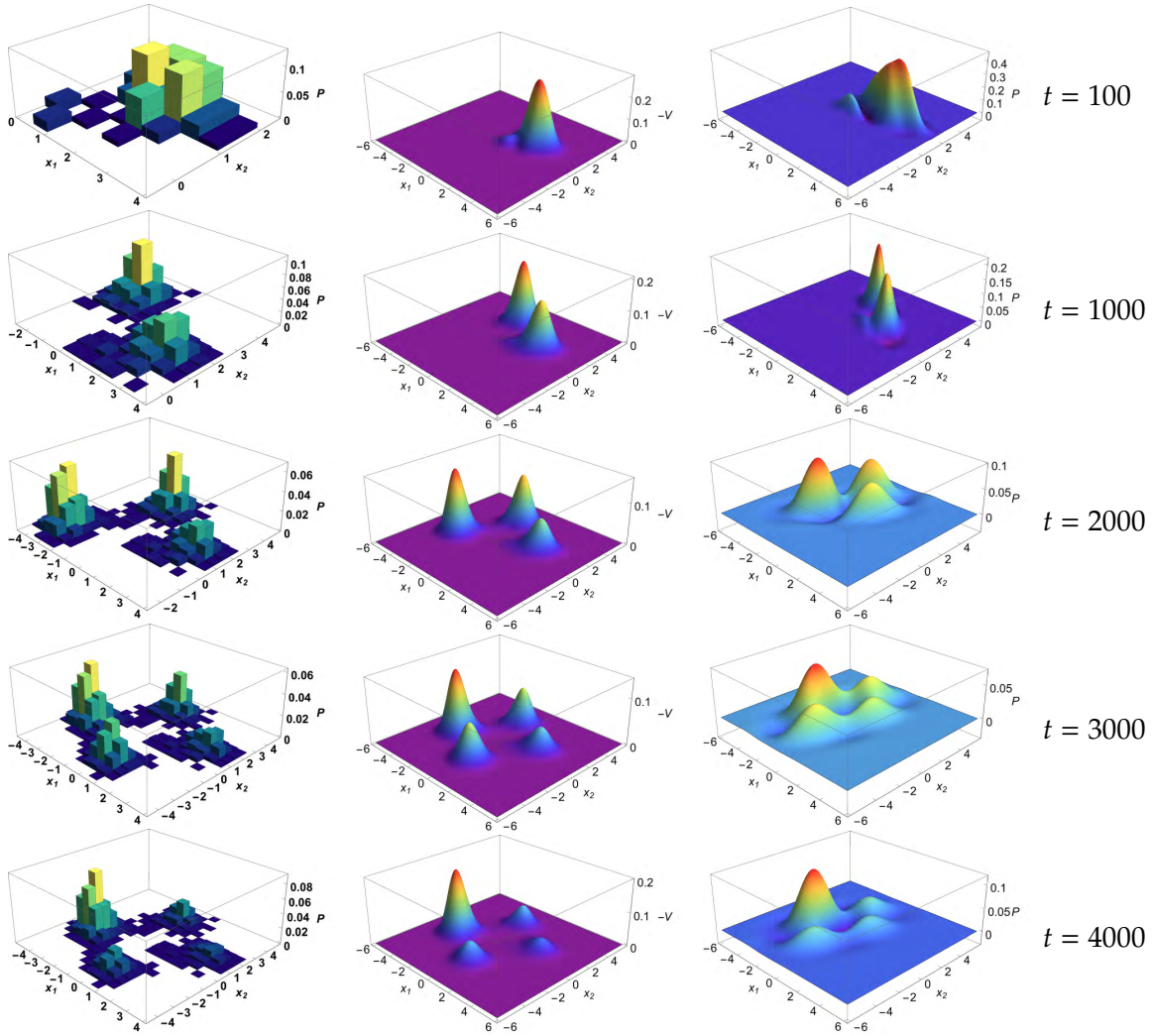


Figure 6.12: Evolution of (a) distribution histogram, (b) negative of the self-shaping landscape with $\sigma_x{}^2 = 0.2$ and (c) MBDA of the landscape with $n = 10$ while processing a two-dimensional input signal generated by a random process with a four-peak PDF. Each row corresponds to a single time moment t, which increases from top to bottom, with the values of t given in the field of the figure.

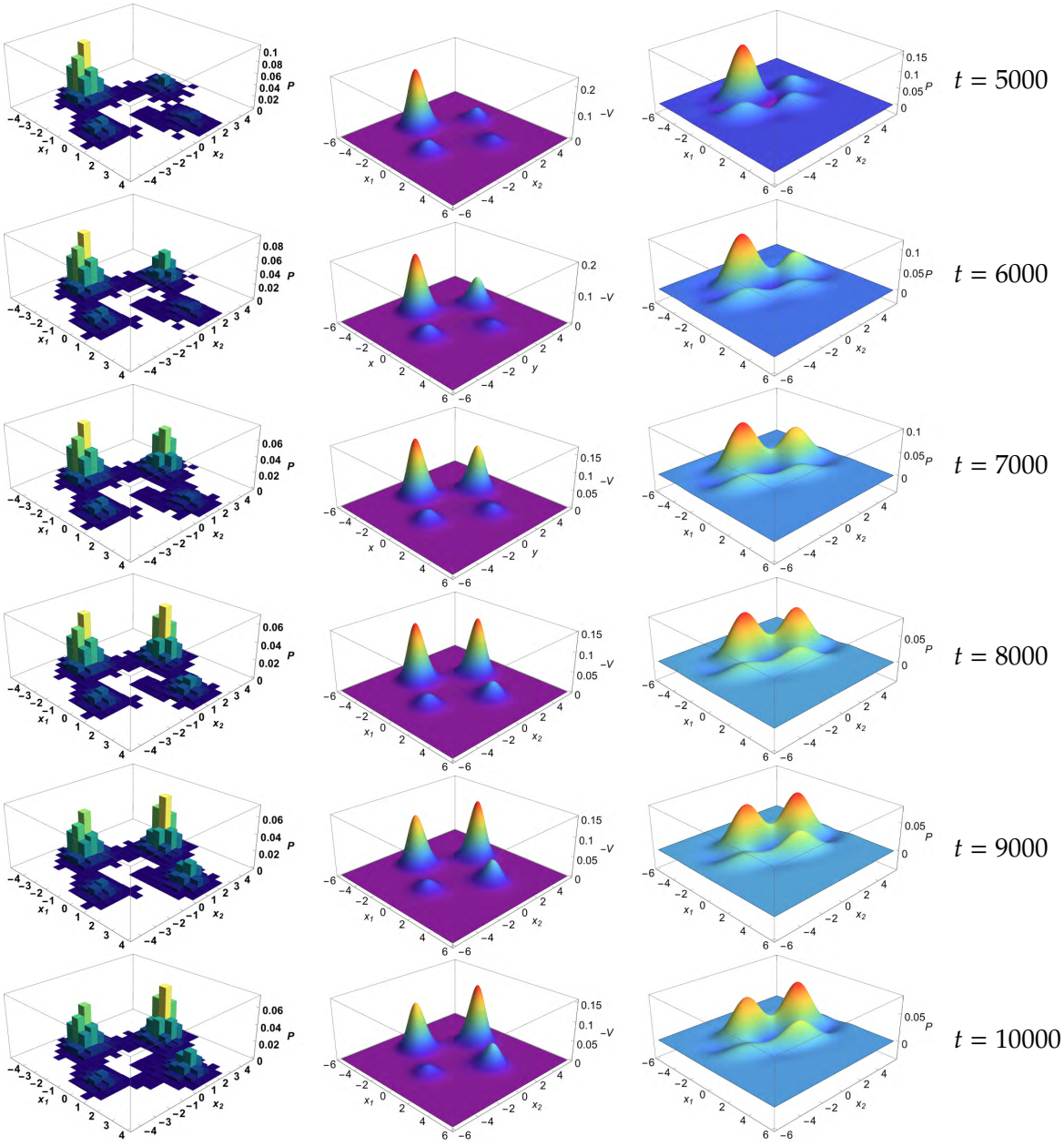


Figure 6.13: Evolution of (a) distribution histogram, (b) negative of the self-shaping landscape with $\sigma_x{}^2 = 0.2$ and (c) MBDA of the landscape with $n = 10$ while processing a two-dimensional input signal generated by a random process with a four-peak PDF. Each row corresponds to a single time moment t, which increases from top to bottom, with the values of t given in the field of the figure.

6.4 Concluding remarks

In this Chapter, encouraged by the success of polynomial approximations of an evolving energy landscape of the one-dimensional gradient PDS, we investigated the possibility to extend this method to a multi-dimensional PDS under the same conditions. It appeared that the MBDA used to describe a one-dimensional system can be extended for this purpose, and we demonstrated how this could be done using a two-dimensional system to aid visualisation of the results obtained. We revealed how the quality of the approximation depended on the number of terms involved and on the complexity of the landscape to imitate.

The significance of the results obtained consists in reducing the DS (3.8), which does not seem to be directly implementable in practice, to a collection of DSs of a well-known non-autonomous type describing evolution of the moments participating in the approximation. Unlike (3.8), non-autonomous DSs permit implementation, and our results pave the way to create the devices self-organising their velocity fields in a manner pre-determined by their purpose. This possibility comes at a price of an error, which is unavoidable when approximating the desired velocity field, but it can be made relatively small by choosing the expansion parameters appropriately.

In addition, the reduction above allows one to avoid storing all the incoming data in order to find a suitable approximation for the landscape function at each stage of information processing. The evolution rules for the moments require knowledge of only the most recent input to the system, while the history of this input could be safely forgotten. This means that the future devices with parameters evolving according to these principles should be able to process a never-ending stream of information, just like neural networks.

We detected an important difference between the one- and multi-dimensional cases, to which MBDA was applied. Namely, in the one-dimensional case throughout the

process of learning from the incoming data the basis functions are fixed, and only the expansion coefficients are evolving in time according to some rules. However, in the multi-dimensional case not only the coefficients, but also the basis functions (Gaussian functions in Hermite polynomials) are evolving as the system processes information.

The purpose of the research in the given Chapter has been to provide a theoretical basis for the future implementations of the PDS in electronic circuits. We chose polynomial approximations of the plastic velocity field because polynomials can in principle be realised in a circuit. Of course, the multi-dimensional cases are more relevant to practical applications than the one-dimensional ones. Thus, when planning the design of the suitable circuits, one will need to find the ways to implement the adjustment of not only the amplification factors for the outcome currents/voltages of the individual sub-circuits modelling the basis functions, but also of the internal parameters of these sub-circuits.

Chapter 7

Summary, conclusions and outlook

One of the two most popular paradigms of an artificial learning system is that of a neural network (NN); the other being a computer, which has not been discussed in this thesis. The attraction of NNs consists in the relative ease of their implementation, which requires building a collection of units with rigid architecture that need to be coupled together in a rigid manner, such that only the connection strengths are allowed to vary. The NN paradigm has been very promising for the future understanding of cognition because the architecture of the artificial NN was inspired by the one of the brain, which is the only truly cognitive system known so far.

However, despite much hope associated with NNs and several decades of effort in the studies of their properties with mathematical tools, it would be fair to say that the NNs did not quite fulfill expectations and did not lead to creation of artificial cognitive systems with advanced functions close to those of the brain. Moreover, even in carrying out most basic cognitive functions, such as memorization and categorization, the NNs are known to possess flaws. Namely, their memories are limited and short-lived, and contain errors caused by spontaneous and inevitable creation of spurious attractors.

The recently proposed way to model basic cognition in an alternative manner, i.e. by means of a dynamical system (DS) with plastic spontaneously evolving velocity field, can be regarded as a generalization of the concept of a NN. Namely, if every

neuron entering the network is modeled as a continuous-state DS, the whole NN becomes a high-dimensional DS with a certain velocity field. The NN learning from the incoming data in an unsupervised manner is spontaneously adjusting its connections, and hence its velocity field, in a manner typically leading to the birth of new and disappearance of the old attractors, and simultaneously reshaping their basins of attraction. Thus, such a NN would be a special case of a PDS, whose plasticity is limited due to the rigidity of its architecture. The latter can explain the limitations of the NNs mentioned above.

Despite the issues with the NNs, these might form the only class of systems know to date which have the self-organised plasticity of their velocity fields. Since the brain as a biological NN is phenomenally good at performing complex cognitive tasks, we hoped that the models of NNs might be able to reproduce the very basic cognitive tasks performed by the simple PDS (3.1), (3.8). With this, in Chapter 4 we compared the performance of the PDS with that of a simple NN, both learning from the same stream of input data without supervision. In order to be able to do that, we had to find a way to code the state of the PDS by the state of the NN. A fundamental issue had to be resolved: while the phase space of the NN is principally bounded, the phase space of the PDS is not. Even assuming that any input signal and hence the usable values of the PDS variables could in practice take values only within a certain bounded interval, we wanted to avoid the traditional simple rescaling of the input to fit it inside the range of values permitted for the NN. The reason is that this method is not convenient and does not look biologically plausible. So we proposed a coding method that essentially converted the value of the input (e.g. the voltage) into the displacement along the neural chain, which seems more biologically realistic, albeit without any biological evidence to support this idea. We tried two variations of this method and demonstrated that against all hopes the given NN does not match the abilities of the PDS. One reason for that could be a limited amount of plasticity in

its velocity field. Another reason could be the non-optimal coding technique used. Indeed, with the chosen coding method, only a small portion of the phase space of the NN is used effectively, which belongs to a closed curve.

Unlike the NN, the simplest gradient PDS (3.1), (3.8) is infinitely plastic in the sense that it can in principle form any number of attractors in such a way, that the birth of the new attractors does not lead to the death of the old ones. However, at the moment such a PDS remains a mathematical abstraction, which might not be implementable in practice. In Chapters 5 and 6 we explored the possibility to create a device, whose velocity field would be more plastic than in a NN, and whose performance would better reproduce the one of the PDS. The proposed solution is based on the knowledge that polynomials can be implemented in electronic circuits and also be used to approximate other functions, including those describing the velocity fields. The choice of appropriate polynomial approximations was inspired by the fact that in the PDS (3.1), (3.8) the energy landscape is converging to a very specific function, namely, to an approximation of the probability density of the random process generating the incoming data. Thus, at each stage of learning from the data the energy of the PDS is essentially an interim estimate of such a PDF, based on the data processed so far. At every time instant, these PDF estimates were approximated by a polynomial expansion based on the moments of the input random process obtained from the data available before that.

Importantly, we were able to reduce the equations for the evolution of the energy of the PDS to a set of non-autonomous equations for the evolution of the moments that can be realized in hardware in principle. We derived the rules according to which these moments need to evolve to reproduce the evolution of the PDS, such that there is no requirement to remember the whole history of the applied stimuli. After establishing the principal possibility to mimic the learning abilities of a one-dimensional PDS with this approach, we examined a more complex and more practically relevant case

of multi-dimensional systems. Interestingly, it appears that multi-dimensional cases present an additional challenge for the design of the appropriate circuits as compared to a one-dimensional case. Namely, in the one-dimensional case one needs to construct a collection of circuits implementing basis functions (Hermite polynomials) of a fixed form, whose outputs need to be multiplied by certain numbers evolving in time as the system learns. However, already in the two-dimensional case the basis functions need to evolve in time, too, which implies a more complex design for the respective circuits.

A number of problems were identified for the future work. Firstly, the method of coding the state of the PDS by the state of the NN, which was proposed in this thesis, might not be optimal and might have affected the performance of the NN against the one of the PDS. Therefore, the problem remains to find a better coding method, preferably using a much larger volume of the phase space of the NN. Secondly, it might be possible to construct a NN consisting of more appropriate neurons, which would lead to greater plasticity of the velocity field and result in better cognitive abilities. Thirdly, while implementation of even the simplest gradient PDS presents considerable difficulties, implementation of non-gradient PDSs not considered here would be an even larger challenge for the future.

Appendix A

Hermite polynomials and their properties

In Chapter 5, we applied Hermite Polynomials in the MBDA. Here we provide brief proof of orthogonal property of such polynomials.

The explicit expression of Hermite polynomial is

$$H_k(x) = \sum_{i=0}^{[k/2]} (-1)^i \frac{k!}{i!(k-2i)!2^i} x^{k-2i}, \tag{A.1}$$

The first ten polynomials are

$$H_0(x) = 1$$

$$H_1(x) = x$$

$$H_2(x) = x^2 - 1$$

$$H_3(x) = x^3 - 3x$$

$$H_4(x) = x^4 - 6x^2 + 3$$

$$H_5(x) = x^5 - 10x^3 + 15x$$

$$H_6(x) = x^6 - 15x^4 + 45x^2 - 15$$
$$H_7(x) = x^7 - 21x^5 + 105x^3 - 105x$$
$$H_8(x) = x^8 - 28x^6 + 210x^4 - 420x^2 + 105$$
$$H_9(x) = x^9 - 36x^7 + 378x^5 - 1260x^3 + 945x$$
$$H_{10}(x) = x^{10} - 45x^8 + 630x^6 - 3150x^4 + 4725x^2 - 945.$$

The polynomials have some interesting properties. The most important one is its orthogonality, which is used for the derivation of the coefficients in MBDA. The orthogonal property reads [48]

$$\int_{\infty}^{\infty} H_i(x)H_j(x)g(x)\mathrm{d}x = \begin{cases} 0 & \text{for } i \neq j, \\ i! & \text{for } i = j; \end{cases} \tag{A.2}$$

where $g(x)$ is the standardised gaussian function

$$g(x) = \frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{1}{2}x^2\Big). \tag{A.3}$$

Integrating by parts, we get, if $i \ll j$,

$$\begin{aligned} \int_{\infty}^{\infty} H_i(x)H_j(x)g(x)\mathrm{d}x &= \int_{\infty}^{\infty} H_i(x)(-1)^j \frac{\mathrm{d}^j g(x)}{\mathrm{d}x^j}\mathrm{d}x \\ &= (-1)^j \Big[H_i(x)\frac{\mathrm{d}^{j-1}g(x)}{\mathrm{d}x^{j-1}}\Big]_{-\infty}^{\infty} + (-1)^{j-1}\int_{\infty}^{\infty} \frac{\mathrm{d}H_i(x)}{\mathrm{d}x}\frac{\mathrm{d}^{j-1}g(x)}{\mathrm{d}x^{j-1}}\mathrm{d}x. \end{aligned} \tag{A.4}$$

The term in square brackets is equal to 0. Due to the property that

$$\frac{\mathrm{d}H_i(x)}{\mathrm{d}x} = iH_{i-1}(x), \tag{A.5}$$

which is proved in [48], the integral becomes

$$\int_{\infty}^{\infty} H_i(x)H_j(x)g(x)\mathrm{d}x = i(-1)^{j-1}\int_{\infty}^{\infty} H_{i-1}(x)\frac{\mathrm{d}^{\mathrm{j}-1}g(x)}{\mathrm{d}x^{j-1}}\mathrm{d}x. \tag{A.6}$$

Repeating integrating by parts and this process, we get that if $i \neq j$, the integral equals 0 and if $i = j$, it equals to $i!$.

References

[1] D. Amit. *Modelling Brain Function*. Cambridge University Press, 1989. 24

[2] D. Amit and S. Fusi. Constraints on learning in dynamical synapses. *Networks*, 3:443–464, 1992. 24

[3] D. Amit and S. Fusi. Learning in neural networks with material synapses. *Neural Comp.*, 6:957–982, 1994. 24

[4] D. Amit, H. Gutfreund, and H. Somopolinsky. Spin-glass models of neural networks. *Physical Review*, A 32:1007–1018, 1985. 24

[5] D. Amit, H. Gutfreund, and H. Somopolinsky. Information storage in neural networks with low levels of activity. *Physical Review*, A 35:2293–2303, 1987. 24

[6] D. Amit, H. Gutfreund, and H. Somopolinsky. Statistical mechanics of neural networks near saturation. *Annals of Physics*, 173(1):30–67, 1987. 24, 25

[7] W. Aspray and A. Burks. Papers of john von neumann on computing and computer theory. *Charles Babbage Institute Reprint Series for the History of Computing*, Vol. 12:645–666, 1986. 9

[8] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13:834–846, 1983. 13

[9] E. G. Boring. *A History of Experimental Psychology (2nd ed.)*. New York: Appleton-Century-Crofts, 1950. 7

[10] L. Borland. Ito-langevin equations within generalized thermostatistics. *Physics Letters*, A 245:67–72, 1998. 36

[11] R. L. Bourden and J. D. Faires. *Numerical Analysis*. Boston: PWS-Kent, 1989. 72

[12] S. Brenner and R. L. Scott. *The Mathematical Theory of Finite Element Methods, 2nd edition*. Springer, 2005. 76

[13] D. S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988. 9

[14] C. V. L. Charlier. *Application de la theorie des probabilities a lastronomie*. Gauthier-Villars, Paris, 1931. 94

[15] M. A. Cohen and S. Grossberg. Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Transactions*, SMC-13:815–826, 1983. 11

[16] M. A. Cohen and S. Grossberg. Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13:815–826, 1983. 17

[17] Y. Le Cun. Learning process in an asymmetric threshold network. *Disordered Systems and Biological Organization*, pages 233–240, 1986. 13

[18] L. Devroye. On random variate generation when only moments or fourier coefficients are known. *Mathematics and Computers in Simulation*, 31:71–89, 1989. 65, 66, 70, 72

[19] D. W. Dong and J. J. Hopfield. Dynamic properties of neural networks with adapting synapses. *Computation in Neural Systems*, 3(3):267–283, 1992. 21, 22, 23, 53, 63

[20] F. Y. Edgeworth. The law of error. *Trans. Camb. Phil. Soc.*, 20:36 and 113, 1904. 75

[21] R. Fitzhugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical Journal*, 1(6):445–466, 1961. 10

[22] S. Fusi. Hebbian spike-driven synaptic plasticity for learning patterns of mean firing rates. *Biol. Cybern.*, 87:459–470, 2002. 24

[23] S. Fusi and L. F. Abbott. Limits on the memory storage capacity of bounded synapses. *Nature Neuroscience*, 10(4):485–493, 2007. 24

[24] S. Fusi, P. J. Drew, and L. F. Abbott. Cascade models of synaptically stored memories. *Neuron*, 45:599–611, 2005. 24

[25] D. Gabor, W. P. L. Wilby, and R. Woodcock. A universal nonlinear filter, predictor and simulator which optimizes itself by a learning process. *Proceedings of The Institution of Electrical Engineers (London)*, 108:422–435, 1956. 13

[26] C. W. Gardiner. *Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences*. Springer, 1985. 35, 102

[27] C. G. Gross. *Brain, Vision, Memory: Tales in the History of Neuroscience*. MIT Press, 1999. 6, 7

[28] S. Grossberg. *The Theory of Embedding Fields with Applications to Psychology and Neurophyiology*. New York: Rockefeller Institute for Medical Research, 1964. 10

[29] S. Grossberg. Nonlinear diference-differential equations in prediction and learning theory. *Proceedings of the National Academy of Sciences*, 58:1329–1334, 1967. 10, 11

[30] S. Grossberg. Some nonlinear networks capable of learning a spatial pattern of arbitrary complexity. *Proceedings of the National Academy of Sciences*, 59:368–372, 1968. 10, 11

[31] S. Grossberg. Some physiological and biochemical consequences of psychological postulates. *Proceedings of the National Academy of Sciences*, 60:758–765, 1968. 10

[32] S. Grossberg. Nonlinear neural networks: Principles, mechanisms, and architectures. *Neural Networks*, 1:17–61, 1988. 7, 11

[33] S. Haykin. *Neural Networks A Comprehensive Foundation*. Macmillan College Publishing Company, 1994. 9, 20, 21

[34] D. O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. New York: Wiley, 1949. 12

[35] J. A. Hertz, R. G. Palmer, and A. S. Krogh. *Introduction To The Theory Of neural Computation*. Redwood City, Calif. ; Wokingham: Addison-Wesley, 1991. 2, 9, 14, 15, 18, 20, 25, 26

[36] G. E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40:185–234, 1989. 20

[37] G. E. Hinton and T. J. Sejnowski. *Proceedings of the IEEE Computer Science Conference on Computer Vision and Pattern Recognition*. Wanshington, DC, 1983. 17

[38] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117:500–544, 1952. 10

[39] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA*, 79:2554–2558, 1982. 11, 17

[40] J. J. Hopfield. *Modeling and Analysis in Biomedicine*. New York: World Scientific Publishing, 1984. 17

[41] J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Natl. Acad. Sci. USA*, 81:3088–3092, 1984. 11, 17, 18

[42] N. B. Janson and C. J. Marsden. Memory foam approach to unsupervised learning. *ArXiv: 1107.0674*, 2011. 2, 27, 28, 32, 34, 35, 41

[43] N. B. Janson and C. J. Marsden. Self-shaping dynamical systems and learning. *ArXiv:1111.4443*, 2011. 2, 27, 28, 32, 34, 35, 41

[44] N. J. Johnson and S. Kotz. *Continuous Univariate Distributions-2*. Houghton Mifflin Company, Boston, 1970. 75

[45] N. J. Johnson and S. Kotz. *Distributions in statistics - continuous multivariate distributions*. Wiley, 1985. 109

[46] E. R. Jones. Golgi, cajal and the neuron doctrine. *Journal of the History of the Neuroscience*, 8(2):170–178, 1999. 6

[47] I. Kanter and H. Sompolinsky. Associative recall of memory without errors. *Physical Review*, A 35:380–392, 1987. 19

[48] M. Kendall and A. Stuart. *The Advanced Theory of Statistics Volume 1 (4th edition)*. Charles Griffin and Company Limited, London and High Wycombe, 1948. 73, 75, 94, 126, 127

[49] G. E. R. Lloyd. Alcmaeon and the early history of dissection. *Sudhoffs Arch. Geschichte Med.*, 59:113–147, 1975. 6

[50] B. J. Maclennan. *Continuous Computation and the Emergence of the Discrete. In* Origins: Brain and Self-Organization. Hillsdale, NJ, 1994. 21

[51] C. Marsden. *Nonlinear Dynamics of Pattern Recognition and Optimization*. Ph.D dissertation, Loughborough University, 2012. 21, 29, 32, 33, 34, 64

[52] S. F. Mason. *A History of the Sciences*. Collier Books New York, 1962. 6

[53] W. A. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5(4):115–133, 1943. 8

[54] R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. S. Venkatesh. Information storage in neural networks with low levels of activity. *IEEE Transaction on Information Theory*, IT-33(4):461–482, 1987. 25

[55] I. M. Mihaila. Development of the trivariate frequency function in gram-charlier series. *Rev. Roum. Math. Pures et Appl. Tome*, 13(6):803–813, 1968. 95, 100

[56] M. L. Mindky. *Theory of Neural-analog Reinforcement Systems and its Application to the Brain-model Problem*. Ph. D. Thesis, Princeton University, Princeton, NJ, 1954. 13

[57] M. L. Minsky and S. A. Papert. *Perceptrons*. Cambridge, MA: MIT Press, 1969. 9

[58] J. Nagumo, S. Arimoto, and S. Yoshizawa. An active pulse transmission line simulating nerve axon. *Proceedings of the IRE*, 50(10):2061–2070, 2000. 10

[59] S. Ochs. *A History of Nerve Functions: from Animal Spirits to Molecular Mechanisms*. Cambridge Univ. Pr., 2004. 6

[60] D. B. Parker. Learning-logic. *M. I. T. Cen. ComputationalRes. Economics Management Sci.*, TR-47, 1985. 13

[61] S. B. Provost. Moment-based density approximants. *The Mathematica Journal*, 9:4:727–756, 2005. 72

[62] N. Rochester, L. H. Haibt J. H. Holland, and W. L. Duda. Tests on a cell assembly theory of the action of the brain, using a large digital computer. *IRE Transactions on Information Theory*, IT-2:80–93, 1956. 12

[63] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychologibal review*, 65(6):386–408, 1958. 9

[64] D. Rumelhart, D. Hinton, and G. Williams. Learning internal representations by error propagation. *Parallel Distributed Processing*, 1:318–362, 1986. 13

[65] D. E. Rumelhart, G. E. Hinton, and J. L. McClelland. A general framework for parallel distributed processing. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1:45–76, 1986. 9

[66] C. A. Shaw and J. C. McEachern. *Toward a Theory of Neuroplasticity*. Psychology Pr, 2001. 12

[67] G. M. Shepherd. *Foundations of the neuron doctrine*, volume 352. Oxford Univ Press, 1991. 7

[68] A. Stokey. Increasing the capacity of a hopfield network without sacrificing functionality. *Artificial Neural Networks - ICANN'97: Lecture Notes in Computer Science*, 1327:451–456, 1997. 25

[69] R. L. Stratonovich. *Topics in The Theory of Random Noise*. Gordon and Breach, 1967. 67

[70] A. M. Uttley. A theory of the mechanism of learning based on the computation of conditional probabilities. *Proceedings of The 1st International Conference on Cybernetics*, 1956. 12

[71] A. M. Uttley. *Information Transmission in the Nervous System*. London: Academic Press, 1979. 12

[72] G. Weisbuch and F. Fogelman-Soulie. Scaling laws for the attractors of hopfield networks. *Journal de Physique Lettres (Paris)*, 46:623–630, 1985. 25

[73] P. Werbos. *Beyond Regression:New Tools for Prediction and Analysis in the Behavioral Sciences*. Ph.D dissertation, Comittee on Appl. Math, Havard Univ., Cambridge, MA, 1974. 13

[74] B. Widrow. *Generalization and Information Storage in Networks of Adaline Neurons*. Wshington, DC: Spartan Books, 1962. 10

[75] B. Widrow and M. E. Hoff. Adaptive switching circuits. *IRE WESCON Convection Record*, pages 96–104, 1960. 10

[76] Wikibooks. Electronics/analog multipliers. http://en.wikibooks.org/wiki/Electronics/Analog_multipliers, 2014. 65

[77] M. Zak. Unsupervised learning in neurodynamics using example-interaction approach. *Appl. Math. Lett.*, 2(3):281–286, 1989. 12

[78] M. Zak. Self-supervised dynamical systems. *Chaos, Solitons and Fractals*, 19(3):645–666, 2004. 12

[79] M. Zak and N. Toomarian. Unsupervised learning in neurodynamics using the phase velocity field approach. *In Advanced In Neural Information Processing System 2*, pages 583–589, 1990. 12